ENHANCING VEHICLE SENSING FOR TRAFFIC SAFETY AND MOBILITY PERFORMANCE IMPROVEMENTS USING ROADSIDE LIDAR SENSOR DATA

FINAL PROJECT REPORT

by

Guohui Zhang, Ph.D., P.E., Shanglian Zhou, Ph.D.
Department of Civil and Environmental Engineering
University of Hawaii at Manoa

for

Center for Safety Equity in Transportation (CSET)
USDOT Tier 1 University Transportation Center
University of Alaska Fairbanks
ELIF Suite 240, 1764 Tanana Drive
Fairbanks, AK 99775-5910

In cooperation with U.S. Department of Transportation,
Research and Innovative Technology Administration (RITA)

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The Center for Safety Equity in Transportation, the U.S. Government and matching sponsor assume no liability for the contents or use thereof.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
Enhancing Vehicle Sensing for Traffic Safety and Mobility Performance Improvements Using Roadside LiDAR Sensor Data	June 30, 2024
	6. Performing Organization Code

7. Author(s) and Affiliations	8. Performing Organization Report No.
Guohui Zhang, Ph.D., P.E., Shanglian Zhou, Ph.D. Department of Civil and Environmental Engineering University of Hawaii at Manoa	INE/CSET 24.06

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Center for Safety Equity in Transportation ELIF Building Room 240, 1760 Tanana Drive Fairbanks, AK 99775-5910	**11. Contract or Grant No.**

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
United States Department of Transportation Research and Innovative Technology Administration 1200 New Jersey Avenue, SE Washington, DC 20590	Research Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Report uploaded to:

16. Abstract

Recent technological advancements in computer vision algorithms and data acquisition devices have greatly facilitated research activities towards enhancing traffic sensing for traffic safety performance improvements. Significant research efforts have been devoted to developing and deploying more effective technologies to detect, sense, and monitor traffic dynamics and rapidly identify crashes in in Rural, Isolated, Tribal, or Indigenous (RITI) communities. As a new modality for 3D scene perception, Light Detection and Ranging (LiDAR) data have gained increasing popularity for traffic perception, due to its advantages over conventional RGB data, such as being insensitive to varying lighting conditions. In the past decade, researchers and professionals have extensively adopted LiDAR data to promote traffic perception for transportation research and applications. Nevertheless, a series of challenges and research gaps are yet to be fully addressed in LiDAR-based transportation research, such as the disturbance of adverse weather conditions, lack of roadside LiDAR data for deep learning analysis, and roadside LiDAR-based vehicle trajectory prediction. In this technical report, we focus on addressing these research gaps and proposing a series of methodologies to optimize deep learning-based feature recognition for roadside LiDAR-based traffic object recognition tasks. The proposed methodologies will help transportation agencies monitor traffic flow, identify crashes, and develop timely countermeasures with improved accuracy, efficiency, and robustness, and thus enhance traffic safety in RITI communities in the States of Alaska, Washington, Idaho, and Hawaii.

17. Key Words	18. Distribution Statement
Computer Vision, Deep Learning, Roadside LiDAR, Traffic Object Recognition, Traffic Safety	

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified.	Unclassified.		N/A

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized.**

SI* (MODERN METRIC) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS				
Symbol	**When You Know**	**Multiply By**	**To Find**	**Symbol**
		LENGTH		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		AREA		
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
		VOLUME		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
	NOTE: volumes greater than 1000 L shall be shown in m^3			
		MASS		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
		TEMPERATURE (exact degrees)		
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
		ILLUMINATION		
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	$candela/m^2$	cd/m^2
		FORCE and PRESSURE or STRESS		
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS FROM SI UNITS				
Symbol	**When You Know**	**Multiply By**	**To Find**	**Symbol**
		LENGTH		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		AREA		
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
		VOLUME		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3
		MASS		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2000 lb)	T
		TEMPERATURE (exact degrees)		
°C	Celsius	1.8C+32	Fahrenheit	°F
		ILLUMINATION		
lx	lux	0.0929	foot-candles	fc
cd/m^2	$candela/m^2$	0.2919	foot-Lamberts	fl
		FORCE and PRESSURE or STRESS		
N	newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380.
(Revised March 2003)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The Federal Motor Carrier Safety Administration (FMCSA) developed its 2015-2018 strategic plans to identify four strategic focus areas, including “Safety 1st” culture and comprehensive data utilization and leveraging technology. Compared to crashes occurring in urban areas, traffic crashes in Rural, Isolated, Tribal, or Indigenous (RITI) communities are associated with a series of significant attributes, such as high speed, low seatbelt usage rate, poor weather and pavement conditions, inferior lighting conditions, considerable distractions, etc.

Recent technological advancements in computer vision algorithms and data acquisition devices have greatly facilitated research activities towards enhancing traffic sensing for traffic safety performance improvements. Significant research efforts have been devoted to developing and deploying more effective technologies to detect, sense, and monitor traffic dynamics and rapidly identify crashes in RITI communities.

As a new modality for 3D scene perception, Light Detection and Ranging (LiDAR) data have gained increasing popularity for traffic perception, due to its advantages over conventional RGB data, such as being insensitive to varying lighting conditions. In the past decade, researchers and professionals have extensively explored LiDAR data to promote traffic perception for transportation research and applications, especially in autonomous driving industry.

Nevertheless, a series of challenges and research gaps continue to exist in the application of LiDAR technology, including: 1) the disturbance of adverse weather conditions, where the existence of snowflakes and fog particles will significantly deteriorate the quality of LiDAR data and cause data outliers and noises in the captured LiDAR point clouds; and 2) the scarcity of roadside LiDAR data for deep learning analysis, where few research efforts were devoted to leveraging infrastructure-based LiDAR data (e.g., roadside LiDAR) for traffic object recognition tasks; and 3) roadside LiDAR-based vehicle trajectory prediction, where it remains a challenge to develop deep learning models for steady vehicle trajectory prediction and risk assessment using information extracted from roadside LiDAR data and traffic control devices; and 4) performance optimization for vehicle trajectory prediction based on roadside LiDAR data and deep learning, which lacks a systematic approach to guide the search for the optimal hyperparameter configurations to boost the model performance.

In this technical report, we focus on addressing the aforementioned research gaps and challenges, by proposing a series of methodologies to optimize deep learning-based feature recognition for roadside LiDAR-based traffic object recognition tasks. The proposed methodologies will help transportation agencies monitor traffic flow, identify crashes, and develop timely countermeasures with improved accuracy, efficiency, and robustness, and thus enhance traffic safety in the States of Alaska, Washington, Idaho, and Hawaii.

CHAPTER 1. INTRODUCTION

1.1. Problem Statement

Although Light Detection and Ranging (LiDAR) data have been extensively explored and adopted by researchers and professionals for traffic perception and object recognition tasks, a series of challenges and research gaps continue to exist, which hinder the further development of LiDAR technology for future transportation research and applications (Gargoum and El-Basyouny, 2017, Wu et al., 2020d, Sun et al., 2022). The major challenges are summarized as follows:

- The impact of adverse weather conditions, such as severe snowy weather, could negatively affect the quality of the captured LiDAR point clouds, and cause performance deterioration in the subsequent feature extraction stage. Existing methods, such as general-purpose 3D noise removal methods, are unequipped to process LiDAR data because of the unique characteristics of LiDAR data, such as non-uniformity (Charron et al., 2018, Zhou et al., 2024). Therefore, additional research efforts need to be devoted to exploring effective methodologies for noise removal from LiDAR point clouds.
- The scarcity of roadside LiDAR data for deep learning analysis may have partially caused the lack of research attempts in deep learning-based traffic object recognition using roadside LiDAR data. Existing methods are primarily focused on leveraging publicly available autonomous driving LiDAR datasets for deep learning network training and testing (Li et al., 2020b), but few have explored the feasibility of adopting infrastructure-based LiDAR data (e.g., roadside LiDAR) for traffic object recognition tasks (Zhou et al., 2022).
- It remains a challenge to achieve steady and accurate vehicle trajectory prediction and risk assessment, which is due to multiple factors, such as the heterogeneity of pedestrian behavior patterns, uncertainties in vehicle-pedestrian interactions, impacts from traffic signals, subjectivities in the judgments of “right of way”, and diversity of urban intersections and environment (Vizzari et al., 2015, Utriainen, 2020, Tan et al., 2023). In fact, few deep learning-based methodologies have explored incorporating positional information extracted from roadside LiDAR data and signal phasing information extracted from signal control devices to promote network performance on both vehicle trajectory prediction and risk assessment.
- For deep learning-based vehicle trajectory prediction tasks, user expertise or prior knowledge is often required upon selecting the network hyperparameters. In fact, there is no exact guideline on how to properly configure the network hyperparameters to achieve the optimal performance on roadside LiDAR-based vehicle trajectory prediction. Therefore, a systematic framework needs to be developed to guide the search for the optimal hyperparameter settings on the network architecture and training scheme, thus promoting the network performance on vehicle trajectory prediction.

1.2. Project Overview

This project is well-aligned with the CSET Year 6 project themes on enhancing vehicle sensing for traffic safety and mobility performance improvements using roadside LiDAR sensor data. As part of the research efforts, the project team performed sensor instrumentation and acquired roadside LiDAR data from multiple signalized road intersections in different cities of the U.S., obtaining 3D point cloud data from complex urban traffic scenarios under different weather conditions. Meanwhile, traffic monitoring

systems based on LiDAR sensors and regular RGB cameras have been installed at several locations to serve future research purposes.

This research project directly contributes to the collection, retrieval, management, visualization, and processing of roadside LiDAR data for transportation research, and it fulfills the demands by CSET on enhancing traffic safety in Rural, Isolated, Tribal, or Indigenous (RITI) communities.

1.3. Research Objectives

This project aimed to enhance vehicle sensing for traffic safety and mobility performance improvements using roadside LiDAR sensor data. To fulfill this overarching goal, the following research topics have been investigated through this technical report:

- Develop an effective analytical method to eliminate data outliers and noises from roadside LiDAR data captured under adverse weather conditions.
- Explore the feasibility of leveraging the domain knowledge of deep learning models trained on autonomous driving data for vehicle detection from roadside LiDAR data.
- Propose a novel deep learning model for vehicle trajectory prediction using positional information extracted from roadside LiDAR data and signal timing information.
- Explore the optimal hyperparameter configuration of deep learning models to boost the performance on vehicle trajectory prediction through Bayesian optimization.

1.4. Report Organization

The rest of this report is organized as follows.

Chapter 2 addresses the issue of data outliers and noises existing in roadside LiDAR data caused by adverse weather conditions. To account for the non-uniform distribution of LiDAR data, an analytical method called Dynamic Channel-wise Outlier Removal (DCOR) is proposed to filter the roadside LiDAR data to eliminate noises, such as snowflakes, and preserve critical foreground features. Unlike existing methods, the proposed DCOR filter processes the captured roadside LiDAR data in a channel-to-channel manner, to decouple the LiDAR data along the vertical axis and reduce data complexity. Meanwhile, during outlier removal, the DCOR filter employs a dynamically adjusted search radius upon searching for neighboring points.

Chapter 3 proposes a deep learning-based approach for vehicle object detection from roadside LiDAR data. To tackle the issue of scarcity of roadside LiDAR data, we develop a novel framework based on CNNs and LiDAR data for automated vehicle detection. It leverages the domain knowledge of CNNs trained on large-scale autonomous driving datasets for vehicle detection from roadside LiDAR data.

In Chapter 4, we propose a deep learning-based method for pedestrian trajectory prediction and risk assessment, using trajectory data extracted from roadside LiDAR data and corresponding signal phasing information. Meanwhile, a set of criteria referred to as the risk factor is established to quantitatively evaluate the risk of the pedestrian crossing behavior, which also serves as a learnable feature. A Long Short-Term Memory (LSTM) network is proposed, which takes the following data as the input: the pedestrian trajectory data, signal phasing data, and risk factors from the past steps. Meanwhile, the network predicts the pedestrian trajectory and risk factor at the future time step.

Chapter 5 proposes a LiDAR-based deep learning framework for vehicle trajectory prediction, which leverages LSTM networks to predict vehicle trajectories and Bayesian optimization to determine the optimal hyperparameter configuration. The optimization scheme is designed such that both the deep learning model architecture and its associated training scheme are updated through Bayesian optimization. In the experimental study, a vehicle trajectory dataset extracted from roadside LiDAR data was utilized for network training and testing. The optimal LSTM network obtained through Bayesian optimization was compared against a benchmark LSTM network with handpicked hyperparameters.

Finally, the research findings from each chapter are summarized in Chapter 6, “Conclusion and Recommendations”. Based on the experimental results, we have summarized several research outcomes and challenges, to provide insights and prior knowledge for future similar research.

CHAPTER 2. DYNAMIC CHANNEL-WISE OUTLIER REMOVAL FILTER TO DE-NOISE LIDAR DATA UNDER ADVERSE WEATHER CONDITIONS

Existing methodologies often assume LiDAR data are acquired under normal weather conditions (Zhou et al., 2024). Nevertheless, many researchers have observed that the LiDAR data captured under inclement weather are often contaminated with noises such as fog and snow, which may deteriorate the data quality and lead to false detections in traffic object recognition (Jokela et al., 2019, Wu et al., 2020a).

In this chapter, we propose a neighborhood-based noise removal methodology to eliminate snow noises from LiDAR data. It identifies a point of interest from a specific laser channel as an outlier, if the number of neighboring points in the same channel within a dynamic search radius is fewer than a threshold. Unlike existing methods that filter the entire LiDAR point cloud (Charron et al., 2018, Kurup and Bos, 2021), the proposed methodology processes LiDAR data channel-by-channel, which helps reduce the data dimensionality and decouple the snow effects along the vertical axis of the 3D point cloud, leading to more effective and efficient outlier detection. Furthermore, by dynamically changing the search radius based on the point-to-sensor distance rather than adopting a fixed search radius, the proposed methodology can account for the reduced point density at far distances caused by the non-uniformity of LiDAR data. In the experimental study, the proposed methodology is compared against some existing LiDAR de-noising approaches, including two state-of-the-art methods, and demonstrates superior performance in both accuracy (i.e., F1 score = 98.3%) and efficiency.

2.1. Background

Recent years have witnessed groundbreaking developments in the acquisition and processing of light Detection and Ranging (LiDAR) data for traffic monitoring, remote sensing, and object recognition tasks (Gargoum and El-Basyouny, 2017, Royo and Ballesta-Garcia, 2019, Wu et al., 2020d). For example, Autonomous Vehicles (AVs) often rely on LiDAR data for environment perception and object recognition (Li et al., 2020a). Despite the exuberant growth of LiDAR sensing technologies and data processing techniques, several challenges related to this type of data are yet to be fully addressed.

One of the pressing concerns is the noise contamination in LiDAR data under severe weather conditions. Many researchers have observed that the LiDAR data acquired in adverse weather, such as wind, rain, snow, and fog, often suffer from deteriorated quality (Yamauchi, 2010, Michaud et al., 2015, Kutila et al., 2018, Charron et al., 2018, Jokela et al., 2019, Wu et al., 2020a, Kurup and Bos, 2021). In a LiDAR point cloud, snow particles are often shown as diffuse, solid objects dispersing around the LiDAR sensor. Depending on the rate of precipitation and density of particles, it can be particularly challenging for LiDAR devices to operate normally under such weather situations because laser beams can be easily backscattered from these tiny particles (Kutila et al., 2018).

To address the issue of noise contamination in LiDAR data, a popular approach in AVs is to leverage multi-modal data fusion, by integrating information from other sources such as radar data and RGB images to reduce the data uncertainty in each individual data source through cross-domain feature correlation (Caesar et al., 2020, Sun et al., 2020). Nevertheless, the development and maintenance of multi-sensor platforms are often time-consuming and expensive. Alternatively, many researchers have developed and applied data pre-processing approaches to de-noise the LiDAR data captured under adverse weather conditions prior to feature extraction. Although two-dimensional (2D) or three-

dimensional (3D) general-purpose de-noising methods such as median filtering (Castleman, 1996) have been adopted to eliminate snow noises from LiDAR data, they often fail to deliver satisfactory performance in snow removal because they do not account for the non-uniformity of LiDAR data. More recently, a few methodologies specifically designed to address the issue of snow noises in LiDAR data have been reported in the literature, which showed promising results on snow removal by taking into consideration the unique characteristics of LiDAR point clouds and snow noises. Nevertheless, in the existing literature, research findings on de-noising LiDAR data corrupted by snow are still insufficient, and thus further efforts need to be devoted to developing and improving LiDAR de-noising methodologies in terms of their accuracy and efficiency in processing real-world LiDAR data captured under severe snowy weather conditions.

In this chapter, we propose Dynamic Channel-wise Outlier Removal (DCOR), a neighborhood-based noise removal approach, to eliminate snow noises from LiDAR data. The proposed methodology processes LiDAR data based on each laser channel and identifies a point of interest as an outlier if the number of neighboring points in the same laser channel within a search radius is fewer than a threshold. The search radius for each point is dynamically adjusted based on the point-to-sensor distance, such that it can account for the varying point density that decreases as the distance to sensor increases. The technical contributions can be summarized as follows:

- The Dynamic Channel-wise Outlier Removal (DCOR) filter is proposed to de-noise LiDAR data corrupted by snow. Different than the existing state-of-the-art approaches, the proposed method processes LiDAR data channel-by-channel rather than filtering the entire point cloud, which helps reduce the data dimensionality and decouple the snow effects along the vertical axis. In the experimental study, the cross comparisons between DCOR and two state-of-the-art methods have demonstrated that the proposed methodology yields superior performance in both accuracy and efficiency.
- Upon elaborating the technical details of the proposed methodology, the physical meanings and impacts of the two parameters, namely the distance factor f and the minimum number of neighboring points *MinPts*, are discussed through theoretical derivations, thus providing insights and prior knowledge for future similar studies and applications.

2.2. Related Work

It has been widely observed and reported by researchers that severe weather conditions such as snow and fog can often deteriorate the quality of LiDAR data captured under such scenarios (Yamauchi, 2010, Michaud et al., 2015, Kutila et al., 2018, Charron et al., 2018, Jokela et al., 2019, Wu et al., 2020a, Kurup and Bos, 2021). Yamauchi (2010) observed that LiDAR devices can provide good accuracy and resolution in normal weather but have difficulties functioning under precipitation and low visibility conditions. Michaud et al. (2015) performed an experimental study to characterize the behavior of LiDAR devices in snowy conditions. They discovered that the distribution of snowflakes is very close to a log-normal distribution, and the noises caused by snowflakes backscattering laser beams are often detectable within the distance of 10 meters to the sensor location. Charron et al. (2018) reported that the amount of snow decreases when the distance to the sensor increases, with a maximum detectable range between 10 meters to 20 meters. Kutila et al. (2018) tested the performance of two LiDAR devices using different laser wavelengths under stabilized foggy and rainy conditions in a weather chamber; they reported the LiDAR sensors suffered deteriorated performance when the adverse weather condition

became more severe and decreased the visibility range. They also recommended a higher (i.e., 1550 nm) operating wavelength for LiDAR sensors, leading to less scattering on fog particles. Bijelic et al. (2018) tested the performance of four LiDAR systems in controlled conditions in a fog chamber. They investigated the optimal configurations of the internal parameters of the LiDAR devices to improve their performance under adverse weather. Jokela et al. (2019) performed a benchmark testing on some popular LiDAR sensors to compare their performance in adverse weather. They observed that all the tested LiDAR sensors performed worse in fog and snow than in clear weather conditions. As revealed by the study in (Xu et al., 2021), rainy weather can severely deteriorate the quality of LiDAR point clouds and lead to drastic performance degradation in LiDAR-based object detection methods.

From the perspective of feature representation and extraction, existing methodologies to de-noise LiDAR point clouds can be mainly categorized as 2D or 3D-based methods. By transforming the information stored in 3D LiDAR data into a 2D representation such as a depth map or an intensity map, traditional image processing methods such as 2D median filtering (Castleman, 1996) and Gaussian low-pass filtering (Castleman, 1996) can be readily adapted. However, as pointed out by Charron et al. (2018), these methods primarily focus on smoothing noisy surface points and are not specifically designed to remove data outliers in an open 3D space, as is the case with LiDAR data corrupted by snow. As a result, 2D-based filtering methods are often unable to remove snow noises effectively due to the sparsity of LiDAR point clouds, and they have the negative effect of smoothing edges in key point features (Charron et al., 2018).

Compared to 2D-based methods for snow removal, 3D-based methods can be directly applied to processing 3D data and thus are better suited to the nature of LiDAR point clouds. The majority of existing 3D-based methods for noise removal are neighborhood-based. This type of methodology usually determines whether a point is an outlier based on the geometric and statistical properties (e.g., distance and distribution) of all similar points in the vicinity. Popular neighborhood-based methods include the Statistical Outlier Removal (SOR) filter (Rusu and Cousins, 2011), Radius Outlier Removal (ROR) filter (Rusu and Cousins, 2011), Density-based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996), etc. Nevertheless, these general-purpose noise removal methods often fail to deliver satisfactory performance on snow removal because they do not account for the non-uniformity of LiDAR point clouds. More recently, a few 3D de-noising methodologies specifically designed to process LiDAR data corrupted by snow have been reported in the literature and demonstrated promising results. Charron et al. (2018) developed the Dynamic Radius Outlier Removal (DROR) filter for snow removal by adjusting the search radius based on the point-to-sensor distance, which takes into consideration the non-uniform distribution of LiDAR point clouds. Wu et al. (2020c) proposed 3D-SDBSCAN by revising the conventional DBSCAN algorithm to divide a 3D LiDAR space into two subareas based on the distribution of snowflakes and employ different DBSCAN parameters in different subareas. Park et al. (2020) proposed the Low-intensity Outlier Removal (LIOR) filter, an intensity-based approach, to eliminate snow noises by deleting points with intensity values below a specified intensity threshold rather than detecting outliers based on the distance between points. By using LiLaNet (Piewak et al., 2018) as the backbone, Heinzler et al. (2020) proposed WeatherNet, a deep learning-based LiDAR de-noising method to reduce noises caused by severe weather conditions, such as rain, snow, and fog. Kurup and Bos (2021) extended the SOR filter to account for the non-uniformity in LiDAR data by introducing a distance multiplication factor. Roriz et al. (2021) developed a LiDAR de-noising method called Dynamic light-Intensity Outlier Removal (DIOR), which combines DROR and LIOR for snow removal and leverages the

Field-Programmable Gate Array (FPGA) technology for real-time performance. Wang et al. (2022b) developed the Dynamic Distance–Intensity Outlier Removal (DDIOR) filter, which is an extension of DSOR by considering the distance and intensity values of snow noises on the basis of DSOR.

In the subsequent sections, four different approaches that have been adopted for LiDAR noise removal, including two general-purpose methods (i.e., ROR and SOR) and two state-of-the-art methods specifically designed to process LiDAR data (i.e., DROR and DSOR), are described in detail. It is worth noting that all the four methods are implemented in section 2.4, "Experimental Study and Results", for cross comparison purposes.

- *Statistical Outlier Removal (SOR)*

The SOR filter (Rusu and Cousins, 2011) is a general-purpose filter that has been widely adopted to remove noises from point cloud data. The SOR filter first performs a *k*-nearest neighbor search for each point in the point cloud and then calculates the average distance to its neighboring points. The mean and standard deviation of the average distances are calculated to determine a global threshold defined as (2-1). Points whose distances to their *k*-nearest neighbors are larger than the threshold T_g will be classified as outliers. Because the SOR filter employs a global threshold for outlier detection, it does not consider the non-uniform distribution of LiDAR point cloud data.

$$T_g = \mu + (\sigma \times \beta) \tag{2-1}$$

where μ and σ denote the mean and standard deviation of the average distances from all points to their *k*-nearest neighbors; β is a multiplication factor.

- *Radius Outlier Removal (ROR)*

The ROR filter (Rusu and Cousins, 2011) is also a general-purpose filter designed to remove noises from point cloud data, which iterates over each point in a point cloud and calculates the number of neighboring points within a fixed search radius. If the total number of neighboring points is smaller than a threshold, then the point of interest is identified as an outlier. As mentioned by (Charron et al., 2018, Kurup and Bos, 2021), the ROR filter showed deteriorated performance on snow removal because it does not consider the non-uniformity of the LiDAR data.

- *Dynamic Radius Outlier Removal (DROR)*

Proposed by Charron et al. (2018), the DROR filter was extended from the ROR filter, which addresses the non-uniformity of LiDAR point cloud by dynamically adjusting the search radius in the ROR filter based on the point-to-sensor distance, as expressed in (2-2) (Charron et al., 2018). In (Charron et al., 2018), the DROR filter was compared against the ROR filter on real-world LiDAR data corrupted by snowflakes and showed better performance on snow removal.

$$SR_p = \begin{cases} SR_{min} & r_p < SR_{min} \\ \beta \times r_p \times \alpha & otherwise \end{cases} \tag{2-2}$$

where SR_p denotes the search radius for the point *p*; r_p refers to the distance between *p* and the LiDAR sensor; SR_{min} is a minimum threshold for the search radius; α denotes the horizontal angular resolution of the LiDAR sensor; β is a multiplication factor.

- *Dynamic Statistical Outlier Removal (DSOR)*

The DSOR filter was extended from the SOR filter by employing a dynamic threshold on the average distances instead of a fixed threshold. The modification on the threshold can be expressed as (2-3) (Kurup and Bos, 2021). If the average distance between a point and its *k*-nearest neighbors is larger than T_d, it will be classified as an outlier. A comparative study was performed in [11] using real-world LiDAR data, which showed that the DSOR filter outperformed the SOR filter and DROR filter on snow removal.

$$T_d = T_g \times r \times d \tag{2-3}$$

where T_d denotes the dynamic threshold adjusted for each point; T_g denotes the global threshold defined in (2-1); r is a multiplication factor; d refers to the point-to-sensor distance.

2.3. Methodology

This section first introduces the LiDAR working mechanism and the characteristics of LiDAR data and snow noises, and then describes the technical details of the proposed LiDAR de-noising methodology. Subsequently, the physical meanings of the parameters involved in this methodology are elaborated through theoretical derivations, and their impacts on the de-noising performance are also discussed and assessed qualitatively.

2.3.1. LiDAR Working Mechanism

3D LiDAR data usually have a very high resolution along the horizontal direction but a relatively low resolution along the vertical direction. For example, for the Velodyne VLP-32C LiDAR sensor employed in this chapter, the horizontal angular resolution is 0.2°; in contrast, the angles between laser channels along the vertical direction range from 0.33° to 9.36°. Figure 2-1 illustrates a schematic diagram of the sensor laser pattern for VLP-32C. This type of sensor consists of 32 laser channels, and each channel is a single 902-nm Infrared (IR) laser emitter and detector pair. Note that each channel has a unique and fixed elevation angle relative to the horizontal plane of the sensor, and its horizontal angle relative to the other laser channels is also fixed, as shown in Figure 2-1 (a). This figure also indicates that the laser channels are arranged into four groups such that the channels in each group are inline vertically and each group has a horizontal angular offset (i.e., 2.8°) to the other adjacent groups. This sensor pattern keeps a widely separated angle between each two adjacent laser channels that are fired in pairs (e.g., channel 0 and channel 1, channel 2 and channel 3, etc.) to minimize IR crosswalk. When the IR laser emits a laser pulse, its time-of-shooting and direction are recorded. As the laser pulse travels through air and gets reflected by an object, the detector receives a portion of the reflected laser energy and records the time-of-acquisition. Thus, the distance between the laser and the object can be readily calculated based on the speed of laser pulse and the time difference.

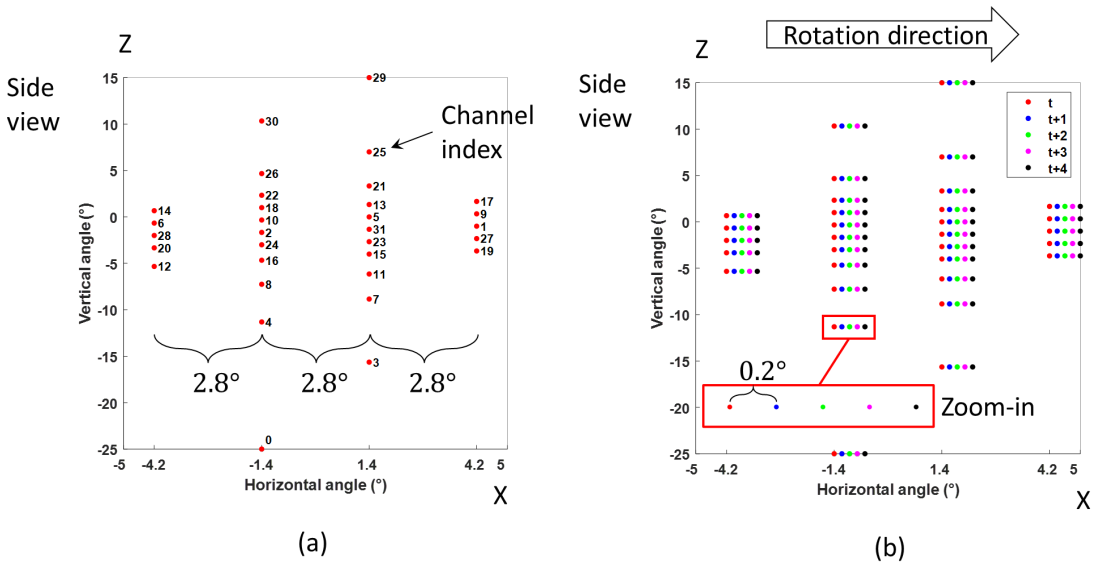


Figure 2-1. Schematic diagram showing the sensor laser pattern: (a) a single firing; and (b) five consecutive firings.

During data acquisition, the LiDAR sensor rotates about its vertical axis with a constant spin rate (i.e., 600 RPM) to obtain a 360° horizontal field-of-view (FOV). Figure 2-1 (b) shows the sensor laser pattern after five consecutive firings, as the sensor is self-rotating. Thus, the horizontal angular resolution can be calculated as (2-4).

$$\alpha = 360 \cdot \frac{\varphi}{60} \cdot t_0 = 0.2° \tag{2-4}$$

where φ is the spin rate of the LiDAR sensor, equal to 600 RPM; t_0 denotes the firing timing of the sensor, equal to 55.296 μs per firing cycle.

2.3.2. LiDAR Data Characteristics

Because laser lights travel straight, the density of the captured 3D point cloud reduces as the point-to-sensor distance increases. An example in Figure 2-2 shows the scenario when two laser channels capture four points after two consecutive firings. It is obvious that the area of the rectangular region enclosed by the four laser pulses is dependent on the distance between the point and the laser. At a farther distance, with a larger region covered (i.e., Region B > Region A), the point density in that region will reduce accordingly.

The other contributing factors to the non-uniformity of LiDAR data include the irregular shapes of the captured objects, the existence of noises, the diversity of background, etc. Especially, for LiDAR data captured in snowy weather, because the tiny snow particles can easily backscatter laser beams, these snowflakes often present themselves in the point cloud as diffuse point clusters spreading around the sensor. As observed by (Michaud et al., 2015), snow noises in LiDAR data usually follow a log-normal distribution, typically detectable within a close range (e.g., 10~15 meters) to the sensor.

The unique characteristics of LiDAR data and snow noises determine that conventional methods for general-purpose noise removal are unequipped to effectively de-noise LiDAR data while preserving important edges and foreground features (Charron et al., 2018).

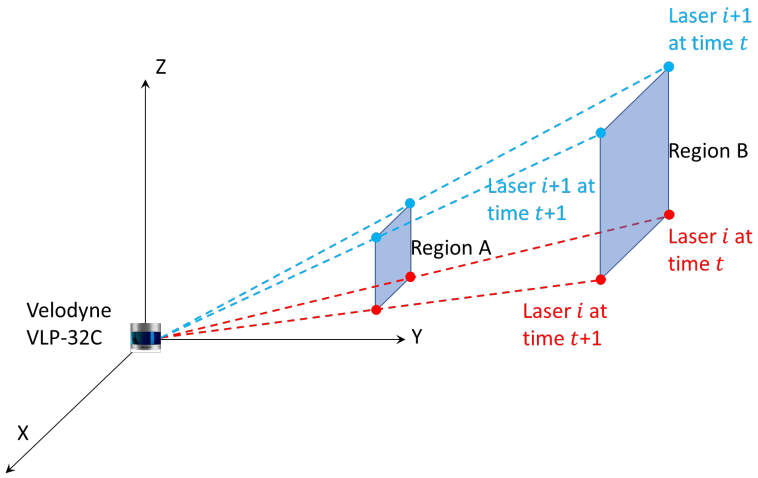


Figure 2-2. The non-uniformity of LiDAR point cloud.

2.3.3. Proposed DCOR Filter

(1) Dynamic search radius

In a 3D LiDAR point cloud, foreground objects such as vehicles are often shown as massive point clusters, with each point in close proximity to its neighboring points in the same cluster; in contrast, noises caused by snowflakes exist in the form of isolated points that are dispersing around the center of the point cloud (i.e., sensor location). Based on this observation, the proposed methodology, Dynamic Channel-wise Outlier Removal (DCOR) filter, identifies data outliers based on the distances from each data outlier to all the points in its vicinity: if the number of neighboring points within a specific distance (i.e., search radius) to the point of interest is smaller than a threshold, then that point is considered isolated and labeled as an outlier.

The term “dynamic” indicates that, different than existing general-purpose methods, the proposed methodology dynamically adjusts the search radius for each point based on its distance to the sensor upon calculating the number of neighboring points, to address the non-uniformity of LiDAR data.

(2) Channel-wise outlier removal

As can be inferred from the term “channel-wise”, another unique feature of the proposed methodology is that it processes LiDAR data channel-by-channel, unlike the state-of-the-art methods (i.e., DSOR, DROR) which choose to filter the entire point cloud. In 3D LiDAR data, the captured point clouds are organized based on their laser channels, and the point cloud data in a specific laser channel share many common attributes, such as elevation angle. Processing LiDAR data in a channel-wise manner can reduce the data dimensionality and decouple the effects of snow noises along the vertical axis. With the channel-wise data, the noise removal task has been simplified, and the proposed methodology can now focus on distinguishing between the snow noises and foreground points only in that channel, based on their distinct geometric features. In addition to adopting the channel-wise data processing strategy, the proposed methodology differs from DROR and DSOR in the following aspects: *i*) the DROR filter requires to specify a minimum search radius for LiDAR points with a very small distance to sensor, as expressed in (2-2); in contrast, the proposed DCOR filter does not employ such a piecewise-linear design for the dynamic search radius; and *ii*) the DSOR filter removes snow outliers by comparing the mean distance

from a point of interest to its *k* nearest neighbors against a distance threshold; on the contrary, the proposed DCOR filter determines if a point is an outlier by assigning a minimum threshold on the number of neighboring points within a predefined dynamic search radius.

The pseudocode for the proposed DCOR filter is provided as Algorithm 1. The distance factor *f* and the minimum number of neighboring points (i.e., a threshold) *MinPts* are the two control parameters involved in the proposed methodology. The physical meanings and impacts of these two parameters are described in the subsequent section.

It is worth noting that, in section 2.4, "Experimental Study and Results", a variant of the proposed methodology, referred to as DCOR-variant1, is also implemented for cross comparison purposes. The only difference between DCOR and DCOR-variant1 is that the latter adopts a fixed search radius. Thus, by comparing the de-noising performance between the proposed methodology and its variant, the impact from the use of a dynamic search radius can be demonstrated.

Algorithm 1 Dynamic Channel-wise Outlier Removal

Input:
LiDAR point cloud (**P**) = $\{P_1, P_2, \ldots, P_n\}$; n = number of laser channels; *f* = distance factor
MinPts = minimum number of neighboring points

Output:
Outliers (**O**)
Filtered point cloud (**F**)

for $P_i \in$ **P do** // Iterate over point clouds
    for $p \in P_i$ **do** // Iterate over laser channels
        $(x, y, z) \leftarrow$ coordinate of p
        distance $\leftarrow \sqrt{x^2 + y^2 + z^2}$ // Calculate the dynamic search radius
        SearchRadius $\leftarrow$ *distance* $\times f$ // Search for the neighboring points
        k $\leftarrow$ NeighboringPoints(*p*, *SearchRadius*) // Remove the snow noises
        if *k* < *MinPts* **then**
            add *p* to **O**
        else
            add *p* to **F**
        end if
    end for
end for
return O and **F**

(3) Physical meanings of the parameters f and MinPts

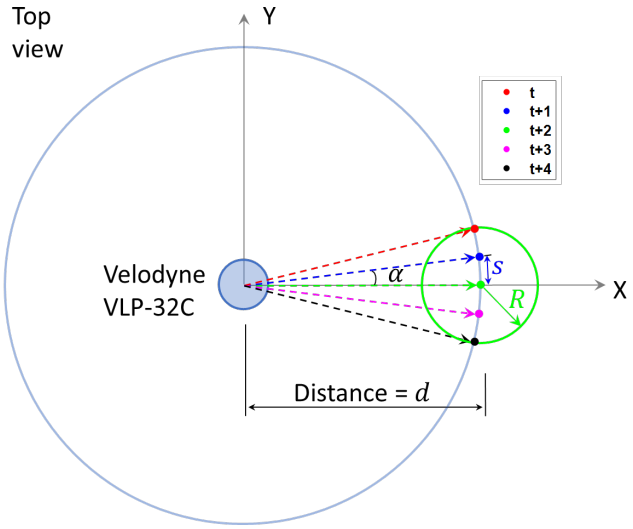


Figure 2-3. Schematic diagram showing the captured points by a single laser channel through five consecutive firings.

Figure 2-3 illustrates a schematic diagram of five points captured by a laser channel through consecutive firings. The distance between the point of interest, shown as a green dot in Figure 2-3, and its nearest neighbor (shown as a blue dot) can be calculated as (2-5). In the meantime, it is worth noting that the subsequent derivation processes are based upon single-channel LiDAR data.

$$s = 2d \cdot \sin\left(\frac{\alpha}{2}\right) \tag{2-5}$$

where s is the Euclidean distance between the point of interest to its nearest neighboring point; d denotes the distance from the point of interest to the sensor; α is the horizontal angular resolution, as expressed in (2-4).

If we employ s in (2-5) as the search radius for the point of interest, the number of neighboring points (including itself) within this radius is three, under the scenario depicted by Figure 2-3. Similarly, we expand the range of the search radius by letting it cover at least NP neighboring points, and its expression can be formulated as (2-6).

$$R = 2d \cdot \sin\left(\frac{\mathrm{NP}-1}{2} \cdot \frac{\alpha}{2}\right) \tag{2-6}$$

where R refers to the minimum search radius to enclose NP points; NP denotes the number of neighboring points within the search radius.

From (2-6), we define the distance factor as the search radius divided by the distance from the point of interest to the sensor location, expressed as (2-7). This expression provides a theoretical basis for designing a search space for the optimal *f* value in section 2.4, “Experimental Study and Results”. For example, in order to capture at least three points, the lower bound for the corresponding distance factor is calculated as 0.0035, based on (2-7). Meanwhile, (2-6) and (2-7) both show that the search radius specified for 3D LiDAR data is proportional to the point-to-sensor distance, thus proving the

necessity of employing a dynamically adjusted search radius instead of a fixed one for neighborhood-based outlier removal.

$$f = \frac{R}{d} = 2\sin\left(\frac{(\mathrm{NP}-1)\cdot\alpha}{4}\right) \tag{2-7}$$

Note that the relationships in (2-6) and (2-7) only hold under "perfect" conditions without any snow noises. As illustrated in Figure 2-4, with the existence of snowflakes and other objects, the number of points within the search radius is often much fwer than the theoretical value, which can be described by (2-8).

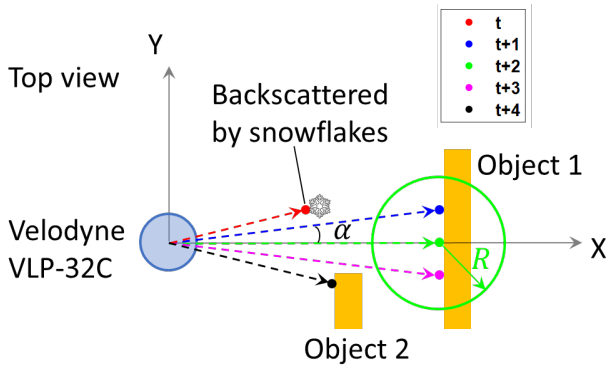


Figure 2-4. Schematic diagram showing the captured points by a single laser channel through five consecutive firings, under snow noises and occlusion.

$$\mathrm{NP}_{\mathrm{actual}} \ll \frac{4\arcsin\left(\frac{R}{2d}\right)}{\alpha} + 1 \tag{2-8}$$

where $\mathrm{NP}_{\mathrm{actual}}$ denotes the actual number of neighboring points within a specific search radius when real-world challenges such as snow noises and occlusion exist.

The expression in (2-8) implies that, for neighborhood-based outlier removal as with the proposed methodology, the minimum threshold for the number of neighboring points (denoted as *MinPts*) is rather critical, and it remains a challenge to determine a proper threshold value to effectively distinguish between foreground points and snow outliers within a specific search radius. The *MinPts* value needs to be smaller than $\mathrm{NP}_{\mathrm{actual}}$, otherwise all the foreground points will be misidentified as outliers; on the other hand, if *MinPts* is much smaller than $\mathrm{NP}_{\mathrm{actual}}$, the algorithm then becomes less aggressive in detecting snow outliers, leading to more false negative detections.

The two parameters *f* and *MinPts* jointly control the performance of the proposed DCOR filter through coupled influences. That is, increasing the value of one parameter will essentially require to increase the value of the other accordingly, in order for the proposed methodology to reduce false positive/negative detections and operate with effectiveness and consistency. There is no exact guideline on the design procedures for these two parameters, and their optimal values often need to be determined through extensive experiments. In Case I of the experimental study, we employ grid search, a straightforward yet effective design approach, to perform an exhaustive search for the joint optimal configuration on *f* and *MinPts* from a large search space.

2.4. Experimental Study and Results

Section 2.4.1 describes the data acquisition and ground truth generation processes. Then, section 2.4.2 briefly introduces the performance metrics adopted for quantitative evaluation. Subsequently, in section 2.4.3, two experimental cases are presented: in Case I, a parametric study is performed to obtain the optimal parameter values for the proposed methodology to achieve the highest accuracy in outlier detection; furthermore, in Case II, the proposed methodology is compared against some existing methodologies for cross comparison and performance demonstration purposes.

2.4.1. Data Acquisition

The LiDAR data adopted in the experimental study were acquired from a road intersection in Reno, Nevada, U.S. As illustrated in Figure 2-5, a Velodyne VLP-32C sensor was installed on a traffic signal lighting pole at a corner of the road intersection for data collection. The LiDAR sensor features 32 channels of laser beams with a 40° FOV along the vertical direction and a 360° FOV along the horizontal direction. A data cabinet which houses the other equipment is also deployed at the site, containing a data processing computer, network communication devices, and external hard drives. Data acquisition was performed under severe snowy weather and resulted in 150 frames of LiDAR data for analysis. Figure 2-6 provides an illustrative example of the captured LiDAR data corrupted by snow. As shown in Figure 2-6, the LiDAR point cloud is contaminated with a massive amount of snow noises concentrated near the sensor location. Another observation from the captured LiDAR data is that the snow noises are predominantly scattered within 15 meters.

Figure 2-5. A LiDAR sensor installed on a traffic signal lighting pole at a road intersection.

The captured LiDAR data were processed by using the DBSCAN-based LiDAR de-noising algorithm proposed by Wu et al. (2020c). The obtained outlier detection results were further carefully examined and adjusted by a group of trained personnel to generate ground truth labels for the snow outliers.

Although the experimental study is performed using the Velodyne VLP-32C sensor, the proposed methodology can adapt to LiDAR data captured by different types of sensors (e.g., VLP-32C vs. VLP-16) by tuning the *f* and *MinPts* values accordingly, because the underlying mechanism of the proposed methodology stays unchanged, assuming the differences between the captured LiDAR data lie in their

resolution, number of laser channels, elevation angle, etc. Meanwhile, the observations and research findings in this section based on the 32-channel LiDAR data can provide prior knowledge and insights for future similar studies using LiDAR data captured by other types of sensors as well.

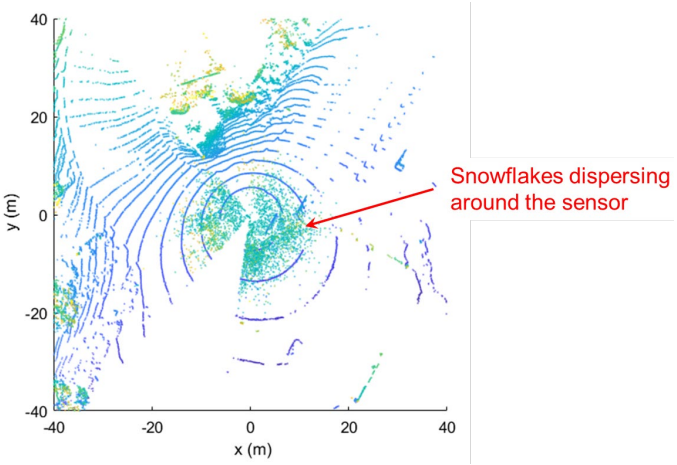


Figure 2-6. An example of the acquired roadside LiDAR data corrupted by snow (bird-eye view).

2.4.2. Performance Evaluation Metrics

The precision-recall analysis (Fawcett, 2006) is adopted to provide a quantitative measure of the de-noising performance of the implemented methodologies. This analysis is comprised of three metrics, including the Precision, Recall, and F1 score, as defined in (2-9), (2-10), and (2-11), respectively. In these equations, *TP*, *FP*, and *FN* denote the number of true-positive detections, false-positive detections, and false-negative detections, respectively. Precision is defined as the number of true-positive detections divided by the total number of positive detections (i.e., *TP*+*FP*); Recall is defined as the number of true-positive detections divided by the total number of true objects (i.e., *TP*+*FN*). Precision reflects the ability of an algorithm to make the prediction results relevant to the true objects, while Recall evaluates the ability to classify all the true objects correctly. By taking the harmonic mean of Precision and Recall, the F1 score provides a comprehensive evaluation of the classification performance of an algorithm. In the experimental study, the F1 score is employed as the primary metric to quantitatively measure the accuracy of the adopted methodologies for snow removal.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2-9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2-10}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2-11}$$

2.4.3. Experimental Results

Two experimental cases are performed. In Case I, a parametric study is conducted using the captured LiDAR data to investigate the optimal configuration of the distance factor *f* and the minimum number of neighboring points *MinPts*. Subsequently, in Case II, the proposed methodology is compared against some existing methodologies regarding their de-noising performance. The experiments are performed in

MATLAB 2022a with the following computer specifications: CPU: Intel Core i7-8750H @2.20 GHz; RAM: 16 GB.

- *Case I: A parametric study to determine the optimal parameter configuration on f and MinPts*

In this subsection, a parametric study is performed through grid search to explore the optimal choice on the *f* and *MinPts* values, such that the proposed DCOR filter can achieve the highest F1 score on snow removal. The values for the distance factor *f* range from 0.005 to 0.1 with an incremental step of 0.005, as expressed in (2-12). Meanwhile, the values for the minimum number of neighboring points *MinPts* are odd numbers ranging from 3 to 21, as shown in (2-13). Thus, 200 combinations of *f* and *MinPts* values are considered in the experimental study. In Case I, each set of the 200 parameter combinations is assigned to the proposed methodology to evaluate the corresponding performance on filtering 150 frames of LiDAR data. For concision, the detailed statistics of the precision-recall metrics are not presented herein; instead, the Precision, Recall, and F1 score values (averaged over the 150 frames of LiDAR data) for the 200 subcases are illustrated in Figure 2-7, Figure 2-8, and Figure 2-9, respectively, to provide a graphical representation on the performance with different parameter values.

$$f = 0.005i \quad i \in \{1, 2, 3, \dots, 20\} \tag{2-12}$$

$$MinPts \in \{3, 5, 7, \dots, 21\} \tag{2-13}$$

In Figure 2-7, Figure 2-8, and Figure 2-9, the horizontal *x* and *y* axes refer to the *f* and *MinPts* values, respectively, while the vertical axis refers to the percentage value of each metric. In these figures, the 200 data samples are shown as blue dots; meanwhile, a surface is fitted to each of the calculated metrics through cubic interpolation to help interpret the changing trend incurred by different parameter values.

Figure 2-7 illustrates the average Precision value evaluated on the captured LiDAR point clouds under different parameter configurations. As can be observed, when the distance factor *f* is relatively small (e.g., 0.01), employing a large value for the *MinPts* threshold leads to more aggressive detection of outliers. The resulted Precision values by the proposed DCOR filter approach the ratio between the number of snow points and the total number of points in the investigated area (i.e., ≈40%) because the algorithm tends to predict every point as an outlier, as illustrated by the deep blue region in Figure 2-7. Meanwhile, increasing the search radius through enlarging the distance factor *f* makes the algorithm less aggressive, resulting in fewer false-positive detections and improved Precision values, as represented by the deep red region in the figure. Overall, it can be observed as a clear trend that a smaller *MinPts* value with a larger *f* value leads to a higher Precision value.

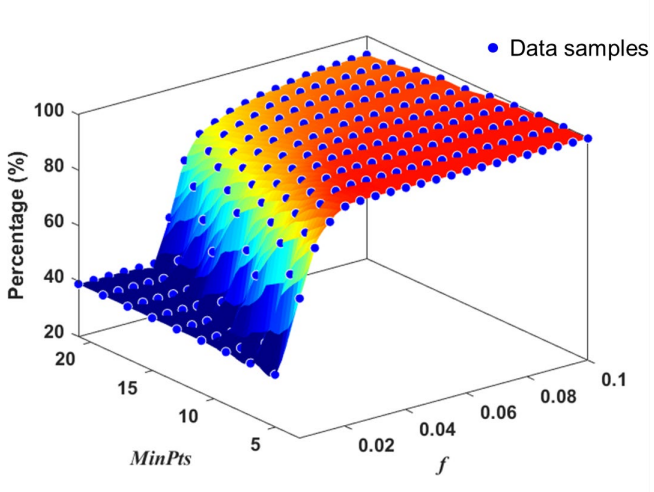


Figure 2-7. Case I: Fit a surface to the Precision values.

Similarly, the average Recall values and the fitted surface are plotted in Figure 2-8. It can be observed that the distance factor *f* governs the performance of the Recall value. Under a fixed threshold on the minimum number of neighboring points (e.g., *MinPts* = 3), when the distance factor *f* increases, the proposed DCOR filter becomes less aggressive in classifying snow points as outliers due to a larger search radius, thus producing more false-negative detections (indicated by the reduced Recall value). Meanwhile, by increasing the value for the *MinPts* parameter, the proposed methodology tends to classify all the points as snow noises, leading to a 100% Recall value at the cost of misidentifying foreground points as noises, as shown by the deep red region in Figure 2-8. Overall, the Recall value will increase with a smaller *f* value and a larger *MinPts* value.

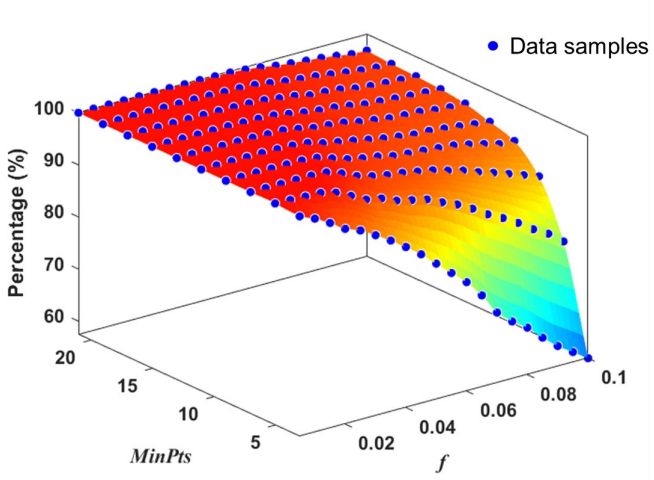


Figure 2-8. Case I: Fit a surface to the Recall values.

Figure 2-9 illustrates the average F1 score and the fitted surface. The F1 score is the harmonic mean of the Precision and Recall value, which provides an overall measure of the performance of the proposed methodology on snow removal. It can be clearly observed that the F1 score reaches the peak value when *MinPts* = 5 and *f* = 0.025, as indicated by the green dot in Figure 2-9. Thus, through Case I, the

optimal parameter set for the proposed methodology to achieve the highest accuracy in outlier detection has been determined.

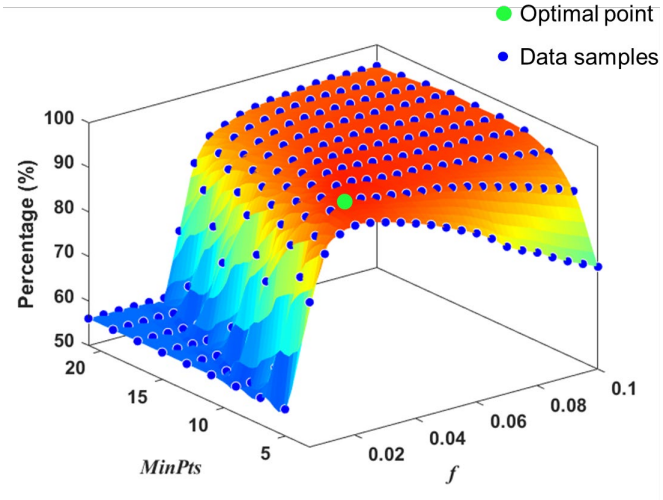


Figure 2-9. Case I: Fit a surface to the F1 score values.

- *Case II: Cross comparisons on the outlier detection performance by the implemented methodologies*

The proposed methodology is further compared against the following methodologies: DCOR-variant1, ROR, DROR, SOR, and DSOR. Note that the DCOR-variant1 filter is the same as the proposed methodology except that it does not adjust the search radius based on different point-to-sensor distances. The intention of implementing the DCOR-variant1 filter for cross comparisons is to reveal the impact of using a dynamic search radius to account for the non-uniformity of LiDAR data.

In total, six methodologies are implemented in Case II and applied to the captured LiDAR data for snow removal. The following parameters are adopted for these methodologies: *i*) DCOR: *MinPts* = 5, f = 0.025; *ii*) DCOR-variant1: *MinPts* = 5, search radius = 0.5 meters; *iii*) ROR: *MinPts* = 5, search radius = 0.5 meters; *iv*) DROR: *SRmin* = 0.05 meters, β = 0.1, α = 0.333°, *MinPts* = 5; *v*) SOR: β = 0.1, *MinPts* = 5; and *vi*) DSOR: β = 0.1, *MinPts* = 5, r = 0.05. Note that for each of these methodologies except the proposed one, the parameter configuration has been validated through prior knowledge and preliminary studies, and the optimal choice on their values is not discussed herein for concision.

The following performance metrics, including the average Precision, Recall, F1 score, and execution time per frame, are illustrated in Figure 2-10, with detailed performance metrics tabulated in Table 2-1. Several observations can be made from Figure 2-10 and Table 2-1 as follows:

Proposed DCOR vs. DCOR-variant1: When the point-to-sensor distance is less than 20 meters, the fixed search radius (0.5 meters) adopted in the DCOR-variant1 filter is much larger than the dynamic search radius (distance factor = 0.025) employed by the proposed DCOR filter. As a result, the differences between DCOR-variant1 and DCOR in the Precision and Recall values are 2.5% and -6.4%, respectively. Such observations are consistent with the findings from Figure 2-7 and Figure 2-8. Meanwhile, judging from the F1 score, the proposed DCOR filter with an adaptive search radius shows better performance than DCOR-variant1, where the improvement in the F1 score is 2.1%.

Proposed DCOR vs. DROR/DSOR: According to Table 2-1, the F1 score values for DCOR, DROR, and DSOR are 98.3%, 96.4%, and 95.4%, respectively. Compared to the state-of-the-art methodologies, the proposed DCOR filter yields the highest F1 score on snow removal, with an improvement of 1.9% and 2.9%, respectively. Thus, through Case II, the accuracy of the proposed methodology on snow removal has been validated through cross comparisons with the two state-of-the-art methodologies.

DCOR-variant1 vs. ROR: The significant difference between DCOR-variant1 and ROR lies in that the former processes the LiDAR data in a channel-wise manner to decouple the snow noises along the vertical axis of the 3D space for a more effective outlier detection. In contrast, the latter directly filters the entire point cloud. As shown in Figure 2-10 and Table 2-1, DCOR-variant1 outperforms ROR in every metric, where the improvements in the Precision, Recall, and F1 score values are 0.3%, 34.1%, and 22.4%, respectively. This result suggests that processing LiDAR data channel-by-channel will significantly boost the performance in correctly identifying all the snow noises, as indicated by the higher Recall value.

DROR vs. ROR: Using a dynamic search radius for outlier detection, the DROR filter shows improved results compared to ROR. From Table 2-1, the differences between DROR and ROR in the Precision, Recall, and F1 score values are -0.2%, 34.9%, and 22.6%.

DSOR vs. SOR: The comparison between DSOR and SOR shows the differences in the Precision, Recall, and F1 score values are -6.8%, 20.4%, and 8.0%, respectively. By employing a dynamic threshold that changes according to the point-to-sensor distance rather than a fixed threshold [see (2-3)], the DSOR filter yields higher values on the Recall and F1 score, showing better overall performance on snow removal.

Comparison on the efficiency of the implemented methodologies: from Figure 2-10, the proposed DCOR filter and its variant both yield drastically reduced execution time compared to the other implemented methodologies. For example, the execution time per frame for DCOR, DROR, and DSOR are 0.242, 2.221, and 3.652 secs, respectively, indicating that the proposed methodology has much higher efficiency in snow removal. The reason behind such a significant improvement in the execution efficiency is that the proposed methodology chooses to decouple the effect of snow noises and reduce the data dimensionality by processing data channel-by-channel. Thus, with a much smaller search space in each channel, the computational efforts needed for nearest-neighbor searches are effectively reduced.

Table 2-1 Case II: Performance metrics of the implemented methodologies (averaged over 150 frames of lidar data).

Algorithm	Precision (%)	Recall (%)	F1 score (%)	Execution time per frame (sec)
DCOR	97.4	99.2	98.3	0.242
DCOR-variant1	99.9	92.8	96.2	0.188
ROR	99.6	58.7	73.8	3.266
DROR	99.4	93.6	96.4	2.221
SOR	99.6	77.8	87.4	3.678
DSOR	92.8	98.2	95.4	3.652

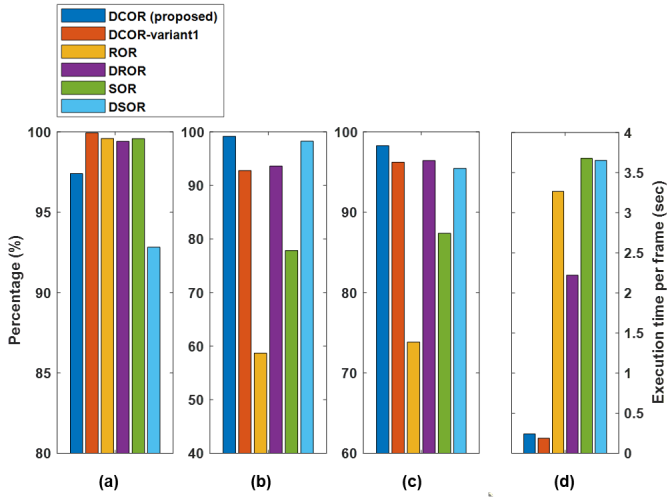


Figure 2-10. Case II: Performance metrics by the implemented methodologies: (a) Precision; (b) Recall; (c) F1 score; and (d) execution time per frame.

Furthermore, an illustrative example is provided in Figure 2-11 to qualitatively evaluate the performance of the implemented methodologies on snow removal. Figure 2-11 (a) shows the raw LiDAR point cloud corrupted by heavy snow, while Figure 2-11 (b-g) demonstrates the filtered point cloud by DCOR, DCOR-variant1, ROR, DROR, SOR, and DSOR, respectively. Judging from the graphical results in Figure 2-11 (b), (e), and (g), DCOR, DROR, and DSOR yield better performance on snow removal than the other methodologies. On the other hand, some remnants of the snow noises can be seen in the filtered results, especially by ROR [Figure 2-11 (d)] and SOR [Figure 2-11 (f)], because these two types of filters are general-purpose filters that do not consider the non-uniformity of LiDAR data.

Figure 2-12 further evaluates the performance of DCOR, DROR, and DSOR on eliminating snow noises at difference distances. In Figure 2-12, the horizontal axis represents the distance between the snow noises and the sensor, and the vertical axis refers to the percentage of removed snow noises, which is the same as Recall. It can be seen that the proposed DCOR filter captures more snow noises than the other two in the range of 0~1 meters. As the distance to the sensor keeps increasing, DCOR shows similar performance to DSOR, and they both outperform DROR in the range of 7~13 meters. Such observations are consistent with the results on the Recall metric (see Table 2-1).

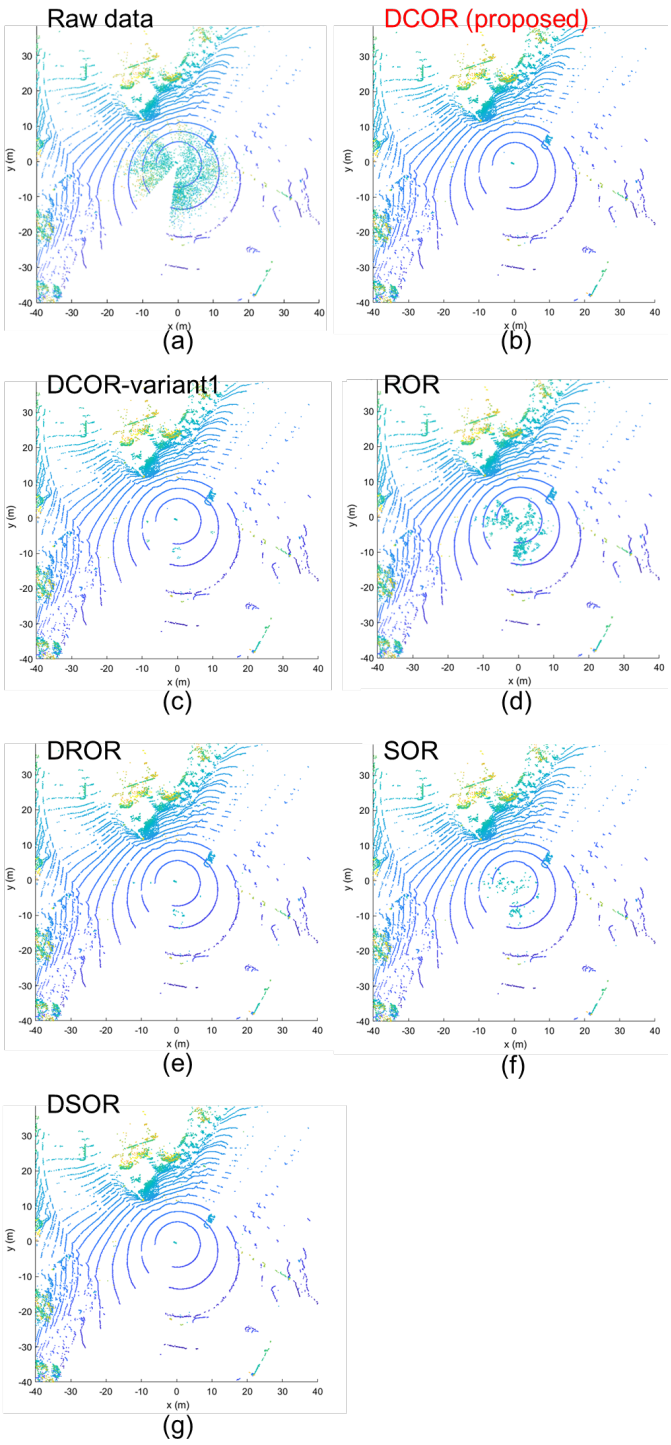


Figure 2-11. Case II: An example of the de-noising result: (a) raw point cloud with snow noises; (b) filtered point cloud by DCOR; (c) DCOR-variant1; (d) ROR; (e) DROR; (f) SOR; and (g) DSOR.

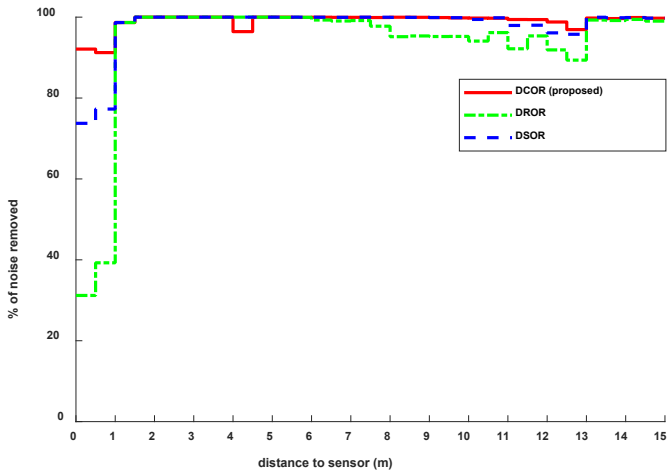


Figure 2-12**.** Case II: Percentage of removed snow noises as a function of the distance to sensor.

2.5. Summary

Severe weather conditions such as heavy fog and snow can often reduce the quality of LiDAR data because laser beams can be easily backscattered by fog and snow particles. Tiny snow particles are usually shown as diffuse, solid objects scattered around the sensor location in a LiDAR point cloud. Such disturbances must be effectively eliminated before the LiDAR data captured under severe weather can be adopted for the subsequent traffic object recognition, safety assessment, and decision-making tasks.

This study proposed a novel methodology called Dynamic Channel-wise Outlier Removal (DCOR) to de-noise LiDAR data corrupted by snow. The proposed methodology iterates over the LiDAR data based on different laser channels and marks a point of interest in a channel as an outlier if the number of neighboring points in the same channel (within a dynamic search radius) is fewer than a threshold. Unlike the existing LiDAR de-noising methodologies, the proposed methodology processes LiDAR data in a channel-wise manner, to reduce the data dimensionality and decouple the snow effects along the vertical axis; furthermore, upon searching for neighboring points, the proposed methodology adopts a dynamically adjusted search radius which is proportional to the point-to-sensor distance, to account for the varying point density that decreases as the distance to sensor increases.

Two control parameters, namely the distance factor *f* and the minimum number of neighboring points *MinPts*, are involved in the proposed methodology. The first parameter *f* is utilized to modify the search radius for each point of interest, and the second parameter *MinPts* sets a minimum threshold to identify outliers based on the number of nearest neighbors. In section 2.3, “Methodology”, their physical meanings and impacts to the de-noising performance of the proposed methodology are elaborated and assessed qualitatively. These two parameters jointly control the performance of the proposed DCOR filter through coupled influences: increasing the value of one parameter will essentially require to increase the value of the other accordingly, in order for the proposed methodology to reduce false positive/negative detections and operate with effectiveness and consistency.

Data acquisition was performed at a road intersection in Reno, Nevada, U.S., resulting in 150 frames of LiDAR data with snow noises for analysis. In Case I of the experimental study, a parametric study is conducted to investigate the optimal choice of the distance factor *f* and the minimum number of

neighboring points *MinPts*, such that the proposed methodology can yield the highest accuracy in snow removal. A large search space consisting of 200 combinations of *f* and *MinPts* values is constructed to explore the joint optimal parameter configuration through grid search. It has been observed as a general trend that a larger value for *f* expands the search radius, and thus, the DCOR filter tends to detect more snow noises as false negatives (i.e., reduced Recall value); a larger value for *MinPts* makes the DCOR filter more aggressive in outlier detection, leading to more false positives (i.e., reduced Precision value). Through Case I, the optimal values for *f* and *MinPts* that lead to the highest F1 score are 0.025 and 5, respectively.

In Case II, the proposed DCOR filter is compared against some other methodologies, including DCOR-variant1, ROR, DROR, SOR, and DSOR, regarding their performance on snow removal. Note that the only difference between DCOR and DCOR-variant1 is that the former employs a dynamic search radius while the latter utilizes a fixed search radius. The experimental results show that the proposed DCOR filter outperforms its variant with a 2% increase in the F1 score by leveraging a dynamic search radius that considers the non-uniformity of LiDAR data. Meanwhile, the comparison between DCOR-variant1 and ROR demonstrates that the strategy of processing LiDAR data in a channel-wise manner can significantly boost the performance with an increase of 4.7% in the F1 score, where ROR processes the entire point cloud as opposed to DCOR-variant1. Moreover, through the cross comparisons in Case II, it has been demonstrated that the proposed DCOR filter outperforms the state-of-the-art methodologies, including DROR and DSOR, in both accuracy and efficiency, where the average F1 score is 98.3%, 96.4%, and 95.4%, and the execution time per frame is 0.242 secs, 2.221 secs, and 3.652 secs, respectively.

In the experimental study, the optimal values for the two parameters are obtained through grid search, which is a straightforward but time-consuming approach. Thus, further research efforts need to be devoted to developing more effective parameter tuning strategies to improve the efficiency and robustness of the proposed methodology.

CHAPTER 3. LEVERAGING DEEP CONVOLUTIONAL NEURAL NETWORKS PRE-TRAINED ON AUTONOMOUS DRIVING LIDAR DATA FOR VEHICLE DETECTION FROM ROADSIDE LIDAR DATA

The majority of existing methodologies applied deep learning (DL)-based techniques, especially Convolutional Neural Networks (CNNs), for vehicle detection and tracking on autonomous driving datasets (Royo and Ballesta-Garcia, 2019, Li et al., 2020a, Alaba and Ball, 2022). Nevertheless, fewer studies were focused on DL-based vehicle detection using roadside LiDAR data, partially due to the lack of publicly available roadside LiDAR datasets for network training and testing.

In this chapter, we develop a novel framework based on CNNs and LiDAR data for automated vehicle detection. It leverages the domain knowledge of CNNs trained on large-scale autonomous driving datasets for vehicle detection from roadside LiDAR data. In the experimental study, roadside LiDAR data were collected at a road intersection in Reno, Nevada, U.S. Meanwhile, a CNN architecture was proposed to detect vehicles from LiDAR data through 3D bounding boxes. The proposed CNN was modified from the established PointPillars network by adding dense connections to the convolutional layers to achieve more complete feature extraction. Three CNNs, including the proposed CNN, PointPillars, and YOLOv4, were trained and tested on PandaSet, a publicly available large-scale autonomous driving LiDAR dataset. Subsequently, the trained CNNs were reused for vehicle detection from the captured roadside LiDAR data. The experimental results demonstrated that the proposed CNN outperformed the others in the testing metrics. All three networks showed good performance on vehicle detection from the captured roadside LiDAR data.

3.1. Background

Recent years have witnessed a steady trend of growth in the investment and development of the U.S. transportation infrastructure systems (ASCE, 2021). Nevertheless, a series of challenges and issues facing the transportation infrastructure systems continue to exist, such as aging of materials, natural hazards, severe weather events, heavy traffic loads, and deteriorated traffic conditions due to congestion and accidents, which may cause economic losses and even threaten the public safety. In the recently published 2021 Infrastructure Report Card (ASCE, 2021) issued by the American Society of Civil Engineers (ASCE), the overall condition of the U.S. road infrastructures was rated as "poor and at-risk". From the U.S. Interstate Highway System Report (TRIP, 2021), pavements on 11% of the U.S. Interstate Highways were rated in poor or mediocre condition, requiring immediate maintenance and rehabilitation. In response to these pressing concerns, the Moving Ahead for Progress in the 21st Century Act (MAP-21) (FHWA, 2012) has required governments and state transportation agencies to establish performance-based programming and planning strategies and activities to facilitate transportation performance evaluation and decision making. It has thus become a rising demand for researchers and professionals to devote long-term research efforts to implementing accurate and efficient traffic condition monitoring and assessment methodologies to promote the safety, efficiency, and serviceability of the transportation infrastructure systems.

Vehicle detection is essential among the research topics on traffic safety analysis. It leads to information such as vehicle speed, accident rate, and traffic congestion level, which is a prerequisite to the transportation safety assessment and decision-making. With the rapid advancements in both computer vision algorithms and data acquisition devices, computer vision-based methodologies have emerged in

the recent decade as the most popular methodologies for automated vehicle detection (Buch et al., 2011). Researchers often adopted digital video data obtained from road surveillance cameras for traffic data collection and vehicle detection (Graettinger et al., 2005, Zhang et al., 2007, Wang et al., 2008, Chintalacheruvu and Muthukumar, 2012) in the early years. However, the performance of video-based vehicle detection can be deteriorated under severe weather events and varying illumination conditions (Zhang et al., 2007, Zhang et al., 2020a).

Since the past decade, three-dimensional (3D) Light Detection and Ranging (LiDAR) technology has been extensively explored and adopted by transportation engineers and researchers for computer vision-based vehicle detection (Zhang et al., 2019a, Zhao et al., 2019b, Song et al., 2021). One of the significant advantages of LiDAR data over traditional video or image data is that it is much less sensitive to weather and lighting conditions variations. Therefore it has broader applicability (Wu et al., 2018a). LiDAR sensors are often deployed on mobile platforms such as Autonomous Vehicles (A.V.s) and Unmanned Aerial Vehicles (UAVs) to capture much spatial and temporal information for traffic monitoring and detection. Besides, many studies and applications using roadside LiDAR can also be found (Zhang et al., 2020c, Wu et al., 2020a, Song et al., 2021), where the LiDAR sensors were deployed at stationary locations such as light poles and road intersections. Driven by the technological advancements in the A.V.s industry, many publicly available large-scale LiDAR datasets have been acquired for autonomous driving research purposes, such as SemanticKitti (Behley et al., 2019), nuScenes (Caesar et al., 2020), and PandaSet (2021).

More recently, there has been a dramatic increase in the use of deep learning (DL)-based techniques, more specifically, Convolutional Neural Networks (CNNs), for vehicle detection from LIDAR data, by exploiting the readily available large-scale autonomous driving datasets for network training (Miglani and Kumar, 2019, Li et al., 2020a). Compared to non-DL-based methods, which often require handcrafted feature generation and subjective parameter selection processes, DL-based techniques such as CNNs can directly learn from data and self-adaptation to perform pattern recognition with the less human intervention (Goodfellow et al., 2016).

In general, non-DL-based methodologies usually employ data processing and feature extraction procedures, including background removal, LiDAR point cloud clustering, vehicle object classification, and vehicle tracking. Although certain successes were reported in these studies and applications, some challenges and issues from the handcrafted feature generation procedures may exist and remain to be fully addressed. Several common issues include the subjectivity of threshold selection for background filtering, prior user knowledge and expertise requirements upon designing the feature generation procedures, and performance deterioration under severe environmental conditions (Wu et al., 2020a). Such issues and challenges may limit the applicability of most of the aforementioned non-DL-based methodologies under real-world scenarios.

Instead, using DL-based techniques for vehicle detection from roadside LiDAR data may alleviate such challenges and yield more consistent and robust performance, owing to the advantages of DL-based techniques in directly learning from data and adapting to the real-world complexities through multi-level feature extractions. Nevertheless, very few studies (Zhang et al., 2020b) applied DL-based techniques for vehicle detection from roadside LiDAR data, partially due to the lack of publicly available large-scale roadside LiDAR datasets. Moreover, although a significant amount of 3D LiDAR datasets for autonomous driving have been published in the literature (Caesar et al., 2020, Sun et al., 2020, Pham et al., 2020,

Geyer et al., 2020, Behley et al., 2019, 2021), the feasibility and challenges of training CNNs on autonomous driving datasets and reusing them for vehicle detection from roadside LiDAR data have not yet been explored and investigated. One of the concerns is that the roadside LiDAR data have different point cloud features and characteristics than autonomous driving data, such as point cloud density, vehicle-to-sensor distance, data occlusion, and background.

In this chapter, we propose a novel DL-based framework to detect vehicles from roadside LiDAR data by leveraging large-scale autonomous driving datasets for network training and testing. In the experimental study, roadside LiDAR data were collected from a road intersection in Reno, Nevada, U.S. Meanwhile, a CNN architecture was developed by modifying the established PointPillars object detection network (Lang et al., 2019) by adding dense connections between convolutional layers to promote feature fusion and extraction. Then, the proposed CNN was trained and tested on PandaSet (2021), a publicly available large-scale autonomous driving dataset, for vehicle object detection. The trained network was reused to detect vehicles from the captured roadside LiDAR data. The technical merits can be summarized as follows:

- To promote effective deep learning-based strategies for vehicle detection in roadside LiDAR data while publicly available large-scale roadside LiDAR dataset is lacking, this study proposes a methodology of reusing CNNs pre-trained on autonomous driving data to detect vehicles from roadside LiDAR data. In model reuse, the general learned features of vehicles from the autonomous driving data are leveraged by the CNNs to help detect similar vehicle objects in the roadside LiDAR data. The experiment has validated the efficacy and feasibility of the proposed methodology under real-world scenarios, providing insights for future similar applications using roadside LiDAR data.
- A CNN architecture is proposed to detect vehicle objects from LiDAR data. That of PointPillars inspires the basic architecture; meanwhile, dense connections are established between convolutional layers to enable maximum information flow and achieve more effective feature extraction and fusion. In the experimental study, the proposed CNN was compared to PointPillars and YOLOv4 (Bochkovskiy et al., 2020) and demonstrated better detection accuracy on autonomous driving and roadside LiDAR data.

3.2. Related Work

In the literature, a series of studies and applications adopting DL-based techniques with autonomous driving datasets have reported successes in traffic object recognition tasks (Li et al., 2020a, Feng et al., 2020). Feng et al. (2020) reviewed DL-based object detection and semantic segmentation for autonomous driving, introducing recent developments in autonomous driving datasets and methodologies for multi-modal sensor fusion. They also discussed several challenges, such as data diversity and proper data fusion strategies. Gao et al. (2021) surveyed to study the data hunger problem for DL-based semantic segmentation tasks for autonomous driving from three perspectives. Firstly, popular 3D LiDAR datasets were reviewed, followed by statistical analysis on three representative datasets to reveal the characteristics of LiDAR data; secondly, existing DL-based 3D semantic segmentation methods were introduced; finally, related problems and challenges were discussed. Vaquero et al. (2020) developed a dual-branch CNN for vehicle detection and tracking using 3D LiDAR data. Two CNNs were trained separately using the point clouds' front and bird-eye view representations, respectively. The fused outputs of the CNNs were then processed for feature clustering and bounding

box extraction. Theodose et al. (2021) proposed a 3D object detection CNN for vehicle detection, which is resilient to variations in point cloud resolution. A two-step approach was developed: firstly, point cloud layers were randomly discarded during training to increase the variability of the training data; then, in the network output, multi-variate Gaussian functions were adopted to represent the bounding box parameters of each vehicle object, allowing to reduce the number of critical hyperparameters such as the size of anchors for bounding box regression (Theodose et al., 2021).

Meanwhile, some other studies adopted non-DL-based methodologies to detect vehicles from roadside LiDAR data. In Sun et al. (2018b), roadside LiDAR point clouds were first pre-processed for ground point removal and then utilized for vehicle data clouds clustering; then, each vehicle in the extracted vehicle cluster was further identified through a 3D bounding box fitting process. Zhang et al. (2019a) proposed a vehicle detection method based on adjacent frame fusion of LiDAR point clouds. The vehicle trajectories were successfully extracted through frame fusion without any bounding box fitting procedures. In another vehicle detection study using roadside LiDAR data (Zhang et al., 2019b), a background construction procedure was proposed to distinguish the foreground objects such as vehicles and pedestrians from the background data based on their different distance features. Then, a density-based spatial clustering method was employed to group the extracted foreground points into clusters and detect vehicles and pedestrians subsequently. Zhao et al. (2019b) adopted a backpropagation artificial neural network (BP-ANN) to classify the vehicle and pedestrian objects in pre-processed LiDAR point clouds after background filtering and clustering. Three features, including the total number of points in a cluster, distance from each cluster to the LiDAR sensor, and direction of the clustered points' distribution, were taken as the inputs for the BP-ANN to help distinguish vehicles from pedestrian objects. Zhang et al. (2021b) developed an algorithm based on a probabilistic neural network (PNN) to classify the road objects into cars, trucks, pedestrians, and bicycles, using their distinct point cloud features (e.g., number of points, distance) extracted from the roadside LiDAR data as the inputs.

3.3. Methodology

3.3.1. Flow Chart of the Proposed DL-based Framework

As shown in Figure 3-1, this DL-based framework is comprised of three phases. In Phase I, a LiDAR sensor is installed at a road intersection to capture 3D point cloud data of the surrounding objects, including vehicles, pedestrians, roadways, buildings, and background.

In Phase II of the proposed framework, a publicly available large-scale autonomous driving dataset (i.e., PandaSet) is adopted for network training and testing. Data pre-processing and augmentation techniques are applied to the training data to remove data outliers and improve the network training performance.

The trained CNN is then utilized in Phase III for prediction to generate 3D bounding boxes for vehicle objects in the captured roadside LiDAR data. The core concept of the proposed methodology is to take advantage of the readily available autonomous driving dataset for network training and testing and further transfer the domain knowledge of the trained CNN to detect vehicle objects from the captured roadside LiDAR data.

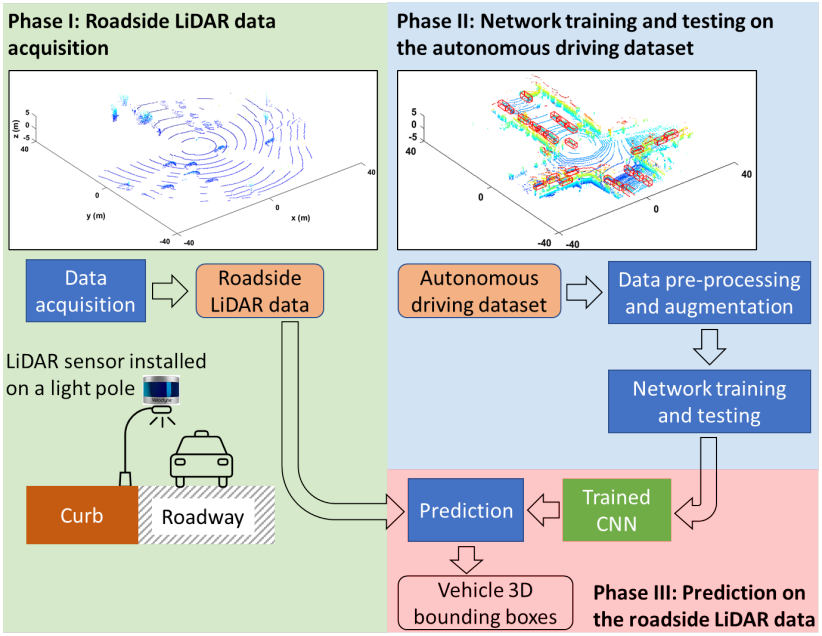


Figure 3-1. Flow chart of the proposed DL-based framework for vehicle object detection.

3.3.2. Data Pre-processing

The LiDAR data from PandaSet contain 3D point cloud information captured from autonomous driving scenarios, ranging from [-200, 200] meters along both horizontal axes. Commonly, point clusters closer to the origin (i.e., LiDAR sensor) are reflected from foreground objects such as cars with a relatively higher point cloud density. In comparison, those scattered at farther distances usually belong to background objects. Besides, the vertical distances between the onboard LiDAR sensor and nearby vehicles are relatively small. Thus, the first step of data pre-processing is to crop the point clouds based on the criteria in (2-1) and (3-2), focus on detecting vehicle objects at closer distances, and improve the training efficiency by reducing the data size.

- *Thresholding based on the distance from each point to the origin:*

$$\sqrt{x^2 + y^2 + z^2} \leq tol_1 \tag{3-1}$$

- *Thresholding based on the vertical location of each point:*

$$|z| \leq tol_2 \tag{3-2}$$

where x, y, and z are the coordinates of the points to be preserved; tol_1 refers to the distance threshold, selected as 40 meters; tol_2 is the maximum vertical distance between the origin and each point, which is equal to 5 meters. Note that these threshold values are determined based on prior knowledge, and discussions on the optimal choices of these values are out of the scope of this study.

The LiDAR point clouds are further processed through a median filter in the second step of data pre-processing. The median filter is a simple yet efficient spatial domain non-linear filtering technique known to have good performance in preserving edges, suppressing noises, and removing data outliers (Gonzales and Woods, 2002). As defined in (3-3), the median filter is applied by convolving a filter kernel w with the point cloud data.

- *Removing noises and data outliers through median filtering:*

$$\tilde{z}(x_k, y_k) = \text{median}_{(x_j, y_j) \in w}\{z(x_j, y_j)\} \tag{3-3}$$

where the $\text{median}\{\cdot\}$ operator calculates the median of a set of data; $z(x_j, y_j)$ denotes the vertical coordinate of the point at (x_j, y_j) location; w refers to the neighborhood centered on the point at (x_k, y_k). In the experimental study, the radius of the neighborhood w is set as 0.05 meters; the output $\tilde{z}(x_k, y_k)$ is assigned as the new vertical coordinate of the point at (x_k, y_k).

3.3.3. Data Augmentation

Data augmentation techniques such as rotation and flipping can effectively reduce model overfitting and improve generalization by increasing the number of training data through label-preserving transformations (Shorten and Khoshgoftaar, 2019). In this study, two widely used data augmentation techniques, including rotation [defined in (3-4)] and random translation [defined in (3-5)], are employed to expand the LiDAR dataset and avoid positional bias in the data.

- *Rotation along the vertical axis:*

$$[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}]_{m\times 3} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]_{m\times 3} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-4}$$

- *Random translation along all three axes:*

$$[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}]_{m\times 3} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]_{m\times 3} + \mathbf{T} \qquad \mathbf{T} \sim \mathbf{U}(a, b) \tag{3-5}$$

where m refers to the total number of points in the point cloud data, and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ denote their coordinates along the three axes, respectively; θ denotes the rotation angle, measured counterclockwise along the vertical axis, where $\theta \in \{45°, 90°, 135°, 180°, 225°, 270°, 315°\}$; each entry in the $\mathbf{T}$ matrix is drawn from a uniform distribution $\mathbf{U}(a, b)$ with a = -0.2 meters and b = 0.2 meters, respectively.

3.3.4. Proposed CNN architecture

- *Basic layout*

PointPillars was developed by Lang et al. (2019) for 3D object detection from LiDAR point clouds. The PointPillars network leverages PointNet (Qi et al., 2017) to learn a representation of point clouds organized in vertical columns, referred to as pillars. Features are extracted from each point in the pillar based on the expressions in (3-6). The pillar features are encoded into a pseudo image to be processed through a 2D CNN encoder, followed by a Single Shot Detector (SSD) module (Liu et al., 2016) to predict a 3D bounding box for each object instance. Compared to CNNs using 3D convolutional layers for feature extraction from point clouds, PointPillars can achieve both higher efficiency and accuracy through a less expensive feature extraction process using 2D CNN encoders [30]. Interested readers can refer to (Lang et al., 2019) for more detailed descriptions of this deep network.

$$\mathbf{p}_i = \left[x_i, y_i, z_i, r_i, c_{x_i}, c_{y_i}, c_{z_i}, p_{x_i}, p_{y_i}\right] \tag{3-6a}$$

$$c_{x_i} = x_i - \frac{1}{N}\sum_{j \in P} x_j \tag{3-6b}$$

$$c_{y_i} = y_i - \frac{1}{N}\sum_{j \in P} y_j \tag{3-6c}$$

$$c_{z_i} = z_i - \frac{1}{N}\sum_{j \in P} z_j \tag{3-6d}$$

$$p_{x_i} = x_i - x_{pc} \tag{3-6e}$$

$$p_{y_i} = y_i - y_{pc} \tag{3-6f}$$

where $\mathbf{p}_i$ is the extracted pillar feature vector from the i^{th} point in the pillar; (x_i, y_i, z_i) are the coordinates of the i^{th} point, and r_i denotes the laser beam intensity at that point; $(c_{x_i}, c_{y_i}, c_{z_i})$ refer to the distances from the i^{th} point to the arithmetic mean of all points in the pillar; P denotes the pillar region, centered at (x_{pc}, y_{pc}), and N is the total number of points in the pillar; p_{x_i} and p_{y_i} are the offsets from the pillar center.

As illustrated in Figure 3-2, the proposed CNN shares a similar architecture layout as PointPillars. Nonetheless, the critical difference between PointPillars and the proposed CNN lies in that the proposed CNN adopts dense blocks by adding dense connections to the 2D convolutional layers in PointPillars, to maximize the information flow in the CNN encoder and thus achieve a more comprehensive and efficient feature extraction.

- *Dense Connectivity*

Developed in (Huang et al., 2017), the “dense connection” (Figure 3-2) represents a straightforward yet very efficient connectivity pattern in which each layer obtains information from all preceding layers and passes on its feature map outputs to all subsequent layers. Such a connectivity pattern can be expressed as (7). According to (Huang et al., 2017), the use of dense connections in a CNN architecture has several advantages, such as alleviating the vanishing gradient problem (Goodfellow et al., 2016, Maas et al., 2013) and encouraging feature reuse by maximizing the information flow.(3-7)

$$\mathrm{x}_i = H_i([\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_{i-1}]) \tag{3-7}$$

where x_i denotes the feature map output from layer i; $[\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_{i-1}]$ refers to the concatenation of the feature maps produced by all preceding layers; $H_i(\cdot)$ is defined as a composition function of three consecutive operations: a 3'3 convolution for feature extraction (Goodfellow et al., 2016), a batch normalization for input regularization (Ioffe and Szegedy, 2015), and a Leaky Rectified Linear Unit (Leaky ReLU) to add nonlinearity to the network (Maas et al., 2013).

Three dense blocks (shown as "Denseblock" in Figure 3-2) are employed for feature extraction and down-sampling in the proposed CNN architecture. Two factors can describe each dense block as n and k, where n refers to the number of composite functions [defined in (3-7)], and k denotes the number of convolutional filters in each composite function. For feature expansion, the outputs from each dense block are then fed into corresponding transposed convolutional blocks (shown as "TransConv" in Figure 3-2). Each transposed convolutional block consists of three operations: a 3'3 transposed convolutional layer with 128 filters for feature expansion, a batch normalization, and a Leaky ReLU. Outputs from all

the transposed convolutional blocks are then combined through a depth concatenation layer, serving as the inputs for the SSD module for bounding box detection.

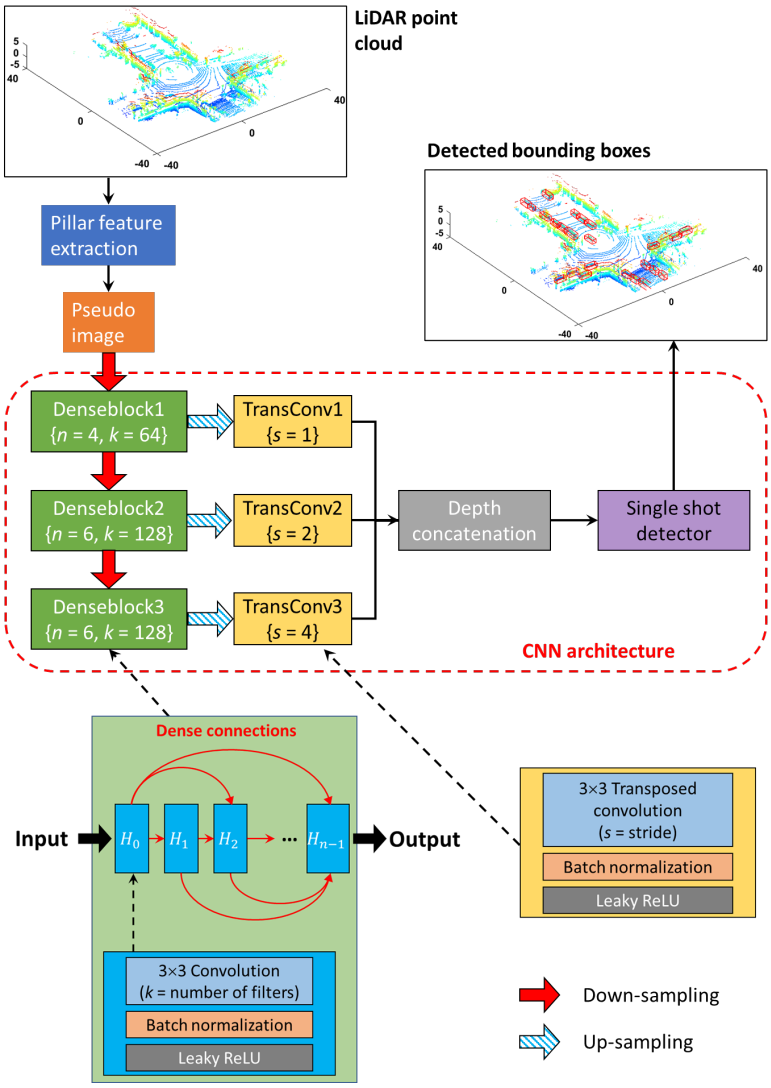


Figure 3-2. Network overview.

3.4. Experimental Study and Results

This section presents an experimental study to validate the efficacy of the proposed methodology for vehicle detection. Firstly, in section 3.4.1, the data acquisition and dataset generation procedures are introduced in detail; then, in section 3.4.2, the adopted CNNs which are to be trained and tested for cross-comparison are introduced; subsequently, in section 3.4.3, the quantitative metrics employed for performance evaluation are described; in section 3.4.4, the experimental setups for network training and testing are explained, including the computing environment, training scheme and hyperparameter configuration, and network parameter initialization; section 3.4.5 presents the results and discussions on two experimental cases, where Case I applied the autonomous driving dataset for network training and testing, and Case II further adopted the trained networks for vehicle detection from the acquired roadside LiDAR data.

3.4.1. Data Acquisition and Dataset Generation

- *Autonomous Driving Data for Network Training and Testing*

PandaSet (2021) was selected as the autonomous driving dataset for network training and testing. This dataset features more than 8,000 3D LiDAR point clouds of various urban scenes, captured through a Panda64 LiDAR sensor with 64 channels of laser beams and a 40° vertical field of view (FOV). The dataset provides 3D bounding box labels for 18 different object classes such as cars, trucks, and pedestrians. An example of the autonomous driving data is illustrated in Figure 3-3, in which 3D bounding boxes are overlaid with the vehicle objects (i.e., cars and trucks) as the ground truth.

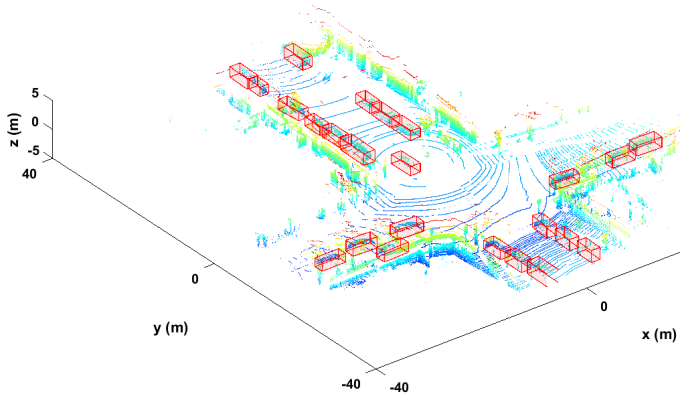


Figure 3-3. An example of the point clouds in PandaSet.

A subset of PandaSet was randomly selected and processed through the aforementioned data pre-processing and data augmentation procedures. As a result, 11,200 LiDAR point cloud frames, together with their ground truth label data, were prepared for network training and 2,400 frames for testing, respectively.

- *Roadside LiDAR Data for Prediction*

The LiDAR sensor is part of the roadside LiDAR sensing system that has been deployed at the site since 2019 for long-term data acquisition and monitoring. The roadside LiDAR data were acquired at the intersection of Evans Ave & N McCarran Blvd in Reno, Nevada, U.S., using a Velodyne VLP-32C sensor with 32 channels of laser beams and a 40° vertical FOV. The LiDAR sensor and a data cabinet were mounted on the traffic signal pole at the northeast corner of the intersection, as shown in Figure 3-4. The cabinet was installed to house the equipment of the roadside LiDAR sensing system, including a data processing computer, network devices, and external hard drives. A substantial amount of roadside LiDAR data was captured while the system functioned under challenging environments such as high temperature (110°F) and ambient vibration due to traffic, dust, wind, snow, and rain. Figure 3-5 demonstrates an example of the captured roadside LiDAR data.

In Case II of the experimental study, a subset of the captured roadside LiDAR data containing 1,000 frames was randomly selected as the Testing Dataset 2 for prediction and performance evaluation purposes. The LiDAR datasets used for network training and testing are tabulated in Table 3-1.

Figure 3-4. The roadside LiDAR sensor for data acquisition.

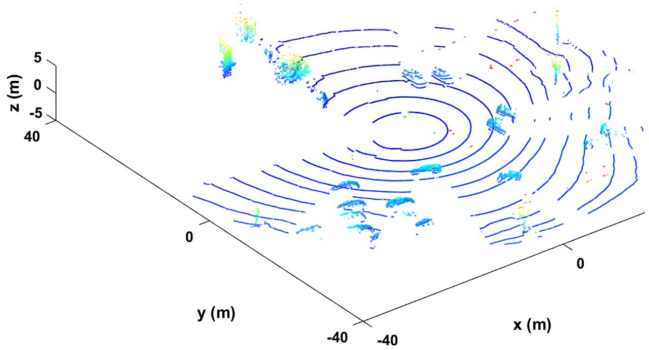


Figure 3-5. An example of the acquired roadside LiDAR data.

Table 3-1 A summary of the datasets utilized for network training and testing.

Dataset name	**Data type**	**Number of point clouds**	**Used in**
Training dataset	Autonomous driving data	11,200	Case I
Testing dataset 1	Autonomous driving data	2,400	Case I
Testing dataset 2	Roadside LiDAR data	1,000	Case II

- *Comparisons between the Autonomous Driving Data and Roadside LiDAR Data*

The domain knowledge learned from the autonomous driving data by the CNNs consists of available features such as the dimensions and shapes of vehicles, roadways, and light poles, which also exist in the roadside LiDAR data. Thus, it is feasible to reuse the pre-trained networks to detect vehicles from the roadside LiDAR data, based on the similar vehicle shapes and dimensions features in these two types of data.

Nevertheless, there are still several differences between the autonomous driving data and roadside LiDAR data, explained as follows:

(1) Point Cloud Density

The roadside LiDAR data were captured by a 32-channel LiDAR sensor, while the autonomous driving data in PandaSet were obtained through a 64-channel sensor. More data channels in the autonomous driving datasets result in higher point cloud densities of the vehicle clusters.

(2) Average Distance from the Vehicle Cluster to the Sensor

Autonomous driving data are often acquired by mounting a LiDAR sensor on a survey vehicle to capture surrounding vehicle objects. Roadside LiDAR sensors, however, are installed at stationary locations such as light poles to acquire LiDAR data at a relatively farther distance. For example, the average distances from the vehicle point clusters to the sensor in the adopted autonomous driving data and roadside LiDAR data are around 18 m and 22 m, respectively. The difference in the vehicle-to-sensor distance also affects the number of points reflected from each vehicle object.

(3) Average Height of the Vehicle Point Cluster

Due to the different approaches to sensor installation, the average height of the vehicle clusters in the autonomous driving dataset is around 2.5 m lower than that in the captured roadside LiDAR data.

(4) Vehicle Occlusion

Under the influences of multiple impact factors, including the point cloud density, vehicle-to-sensor distance, and height of the vehicle point cluster, vehicle occlusion in these two types of LiDAR data occurs differently. In the autonomous driving dataset, both the side and top of some vehicle objects at far distances are often occluded; in the roadside LiDAR data, because the sensor is typically installed at a relatively high location, the point cloud information from the top of each vehicle object can be readily obtained without occlusion.

When reusing the pre-trained CNNs for vehicle detection from the roadside LiDAR data, the different data characteristics may cause performance degradation, as shown in Case II of the experimental study.

3.4.2. CNNs for Cross Comparison

In addition to the proposed CNN and PointPillars (Lang et al., 2019), a state-of-the-art object detection network, namely YOLOv4 (Bochkovskiy et al., 2020), was adopted in the experimental study for cross-comparison. The architecture of YOLOv4 consists of CSPDarknet53 (Wang et al., 2020a) as the backbone, spatial pyramid pooling (SPP) module (He et al., 2015) as additional blocks, PANet (Liu et al., 2018) for parameter aggregation, and YOLOv3 (Redmon and Farhadi, 2018) object detection head. The benchmark study in (Bochkovskiy et al., 2020) showed that YOLOv4 outperformed other state-of-the-art object detectors with higher accuracy and efficiency. More technical details on YOLOv4 can be found in (Bochkovskiy et al., 2020). To apply YOLOv4 to detect vehicles in the 3D LiDAR data, the LiDAR input was transformed into 2D bird-eye view images with the dimension of 1024×1024 pixels. Each RGB image has 3 channels that contain the information on point density, height, and intensity at each pixel location.

To sum up, three CNNs were employed for training and testing in the experimental study to help validate the proposed methodology through cross-comparisons.

3.4.3. Performance Evaluation Metrics

The precision-recall analysis [46] was adopted in the experimental study to evaluate network performance on vehicle object detection(Fawcett, 2006) quantitatively. Three performance metrics are included, which are the Precision, Recall, and F1 scores, as defined in (2-9), (2-10), and (2-11), respectively. Precision is the number of accurate positive detections divided by the total number of positive detections by the network. The recall is the number of true positive detections divided by the number of true vehicle objects. And as a harmonic mean of the Precision and Recall, the F1 score provides an overall measure of the detection performance of the network on each LiDAR point cloud. Note that in the experimental study, the mean value for each metric is calculated, but also its histogram is illustrated to comprehensively measure the network performance on the testing datasets.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3-8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3-9}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3-10}$$

where *TP* and *FP* refer to the number of true positive and false positive detections, respectively; note that a bounding box detection is considered a true positive if it has the largest overlap (more than 50%, measured by intersection over union (Rezatofighi et al., 2019)) with the ground truth bounding box (Geiger et al., 2012), and multiple detections of the same object are considered as false positives; and, *FN* refers to the number of true vehicle objects that are not detected by the network.

3.4.4. Experimental Setup

- *Computing Environment*

The experimental study was performed using MATLAB (MATHWORKS, 2019) deep learning toolbox and LiDAR toolbox, with the computer specifications: CPU: Intel Xeon Gold 6254 @ 3.10 GHz; GPU: Nvidia RTX Quadro 8000 with 48 GB RAM.

- *Training Scheme and Hyperparameter Configuration*

The loss function introduced in (Yan et al., 2018) was adopted for optimization. In training the adopted three networks, the Adam optimizer (Kingma and Ba, 2014) was employed to minimize the loss function. Adam is a straightforward yet computationally efficient algorithm for gradient-based optimization of stochastic objective functions (Kingma and Ba, 2014); thus, it has been extensively adopted for network training. A series of hyperparameters associated with the training process is as follows: mini-batch size (10), number of epochs (90), initial learning rate (0.0002), learning rate drop period (15) and drop factor (0.8), gradient decay factor (0.9), squared gradient decay factor (0.999), and leaky ReLU factor (0.01), with their values provided in parentheses. The values for these hyperparameters are determined based on prior knowledge and other successful applications such as (Lang et al., 2019), and discussions on the optimal choices of these values are out of the scope of this study.

- *Network Parameter Initialization*

The weights in each convolutional layer are initialized using the Glorot initializer (Glorot and Bengio, 2010), which randomly draws samples from a Gaussian distribution with zero mean and variance based on the dimension of the weights. The initial biases are all set as zeros. In each batch normalization layer, the initial scale and shift factor are equal to one and zero, respectively.

3.4.5. Experimental Results

Two experimental cases were performed. In Case I, the adopted CNNs was trained and tested using the autonomous driving data (see Table 3-1). After training was completed, the trained CNNs were then reused for vehicle detection from the roadside LiDAR data in Case II.

- *Case I: Network Training and Testing on the Autonomous Driving Dataset*

The trained CNNs were tested on the autonomous driving dataset, namely the Testing Dataset 1 in Table 3-1. Three metrics, including the Precision, Recall, and F1 score, are calculated from Testing Dataset 1, and their average values are illustrated in Figure 3-6. Meanwhile, each CNN's training time and testing time are also illustrated in Table 3-1 as a measure of the network efficiency. Detailed statistics are tabulated in Table 3-2. From Figure 3-6, Table 3-1, and Table 3-2, some observations can be made as follows:

(1) Testing Accuracy

As an overall measure, the F1 scores of the proposed CNN, PointPillars, and YOLOv4 are 76.5%, 69.5%, and 73.8%, respectively. Compared to PointPillars and YOLOv4, the proposed CNN achieves the highest F1 score with an improvement of 7.0% and 2.7%, respectively, indicating the best performance in detection accuracy. It can be observed that the proposed CNN outperforms PointPillars in all three metrics, including the Precision, Recall, and F1 score values, by an increase of 5.3%, 8.9%, and 7.0%, respectively. Such a significant improvement in the network performance is induced by using dense connections to promote feature fusion and extraction. It is also worth noting that YOLOv4 yields the highest Precision value, which indicates the capability of reducing false-positive detections. In Case I, all the three CNNs have achieved good accuracy when testing on the autonomous driving dataset, measured by the precision-recall metrics.

(2) Training and Testing Efficiency

Judging from the training and testing time by the proposed CNN and PointPillars, it can be concluded that the proposed CNN has similar efficiency as PointPillars. Although the proposed CNN employs a much more complicated connectivity pattern between convolutional blocks through dense connections, its training and testing efficiency is not severely deteriorated, where the relative differences between the proposed CNN and PointPillars in the training and testing time are only 3.8% and 3.2%, respectively. Meanwhile, YOLOv4 achieves the best efficiency in network training and testing, mainly caused by the fact that the input data for YOLOv4 are 2D images generated by transforming the 3D LiDAR data into 2D bird-eye review representation.

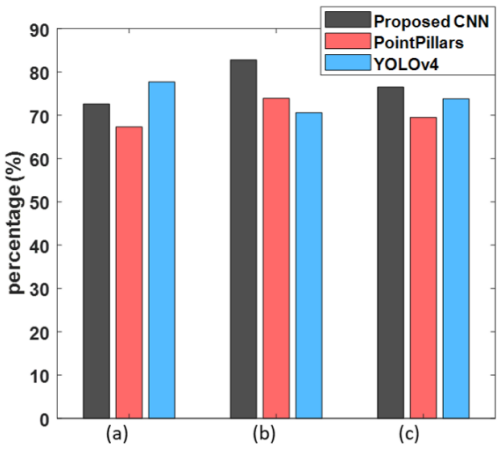


Figure 3-6. Case I: Bar plots of the average metrics values evaluated on the Testing Dataset 1: (a) Precision; (b) Recall; (c) F1 score.

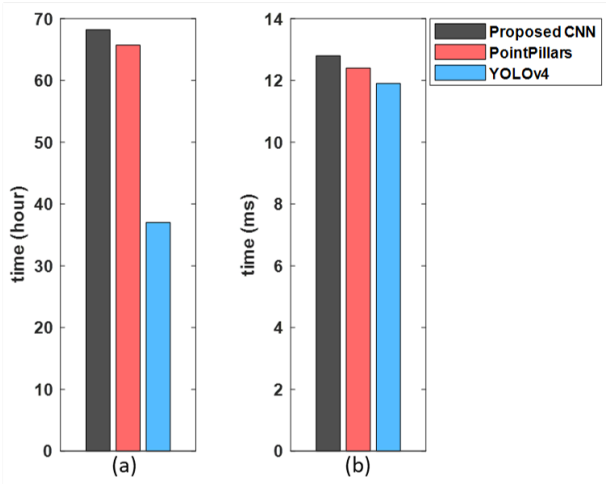


Figure 3-7. Case I: Bar plots of (a) training time and (b) testing time per frame.

Table 3-2 Case I: Performance metrics evaluated on the autonomous driving data.

Network	**Average Precision (%)**	**Average Recall (%)**	**Average F1 score (%)**	**Training time (hour)**	**Testing time per frame (ms)**
Proposed CNN	72.6	82.8	76.5	68.2	12.8
PointPillars	67.3	73.9	69.5	65.7	12.4
YOLOv4	77.7	70.6	73.8	37.0	11.9

In addition to measuring the detection performance using the average values of the metrics, histograms of each metric evaluated from the Testing Dataset 1 are illustrated in Figure 3-8 to provide a graphical representation of the differences between the proposed CNN and the other two CNNs in terms of their detection accuracy. In Figure 3-8 (a-c), the histograms of the Precision, Recall, and F1 score by the proposed CNN (shown in gray color in Figure 3-8) and PointPillars (red color) are overlaid with each other. Similarly, in Figure 3-8 (d-f), the histograms of the three metrics by the proposed CNN (gray color)

are overlaid with those by YOLOv4 (blue color). The horizontal axis of each plot indicates the percentage value of each metric, and the vertical axis refers to the number of bins. By comparing the results between the proposed CNN and PointPillars in Figure 3-8 (a-c), the histograms of all three metrics by the proposed CNN are more skewed to the right-hand side of the plot, indicating higher metric values. Thus, it can be concluded that the proposed CNN has better performance than PointPillars in detecting vehicle objects from the autonomous driving dataset. Meanwhile, as shown in Figure 3-8 (d-f), YOLOv4 outperforms the proposed CNN in the histogram of the Precision value. Still, the histogram of the F1 score by the proposed CNN is more skewed to the right-hand side of the plot, indicating better overall performance.

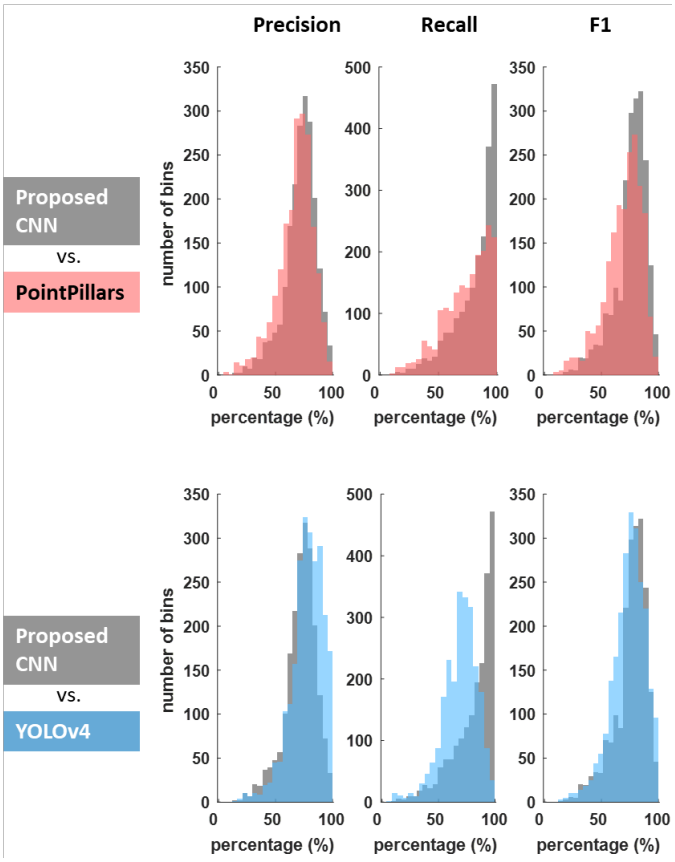


Figure 3-8. Case I: Histograms of the metrics evaluated on the Testing Dataset 1: (a-c) the Precision, Recall, and F1 score by the Proposed CNN vs. PointPillars; and (d-f) the Precision, Recall, and F1 score by the Proposed CNN vs. YOLOv4.

- *Case II: Reusing the Trained CNNs to Detect Vehicle Objects from the Roadside LiDAR Data*

The trained networks, including the proposed CNN, PointPillars, and YOLOv4 from Case I, were further utilized to make predictions on the Testing Dataset 2 (see Table 3-1), a dataset comprised of the captured roadside LiDAR data.

The three metrics, including the Precision, Recall, and F1 score, are calculated from the Testing Dataset 2 to measure the detection performance on the roadside LiDAR data. The average values of these metrics are illustrated in Figure 3-9 and tabulated in Table 3-3. Similar to Case I, the proposed CNN yields the highest F1 score of 72.9%, with an increase of 5.2% and 2.4% compared to those by

PointPillars and YOLOv4, respectively, indicating the best detection accuracy on the roadside LiDAR data. It has been observed as a consistent trend that by directly reusing the pre-trained CNNs to detect vehicles from the roadside LiDAR data, all the three CNNs still yield good detection accuracy, where the differences in the average F1 scores by the three CNNs between Case I vs. Case II are only 3.6%, 1.8%, and 3.3%, respectively. In model reuse, the general learned features of vehicles from the autonomous driving dataset are leveraged by the CNNs to help detect similar vehicle objects from the roadside LiDAR data. The slight performance deterioration may be caused by some differences in the characteristics of autonomous driving and roadside LiDAR data. Thus, through the comparisons between Case I and Case II, the feasibility of reusing pre-trained CNNs on autonomous driving data for vehicle detection from roadside LiDAR data has been validated with real-world data.

Table 3-3 Case II: Performance metrics evaluated on the roadside LiDAR data.

Network	**Average Precision (%)**	**Average Recall (%)**	**Average F1 score (%)**
Proposed CNN	69.9	79.7	72.9
PointPillars	65.6	72.4	67.7
YOLOv4	74.2	69.2	70.5

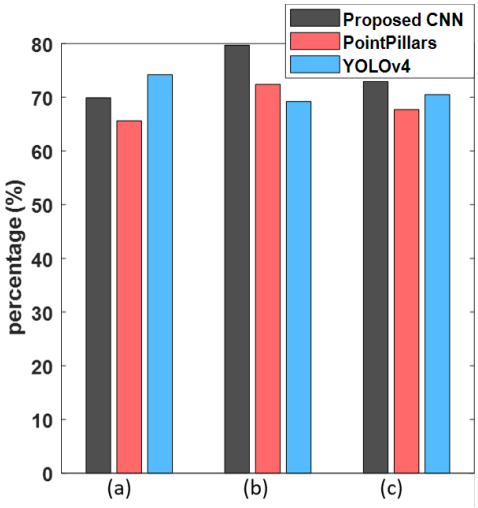


Figure 3-9. Case II: Bar plots of the average metrics values evaluated on the Testing Dataset 2: (a) Precision; (b) Recall; (c) F1 score.

An example of the detection results on the roadside LiDAR data by PointPillars, YOLOv4, and the proposed CNN is illustrated in Figure 3-10 to demonstrate their performance. Figure 3-10 (a), (b), and (c) display the detection results on a LiDAR point cloud by PointPillars, YOLOv4, and the proposed CNN, respectively, in which the detected bounding boxes (shown in red color) are overlaid with the point cloud data. An effective post-processing step is applied to remove most false positive detections by setting a minimum threshold on the number of points enclosed by each bounding box. As can be seen, all the three CNNs can detect the majority of the vehicle objects as true positives. Nevertheless, in Figure 3-10 (a), a false negative detection by PointPillars is observed, as indicated by the green circle. Another observation from Figure 3-10 (a) is that the yaw angle for one of the predicted bounding boxes by PointPillars is not accurate. The detection results by YOLOv4 in Figure 3-10 (b) also exhibit two false negatives (shown in green circles) and a false positive detection which is caused by the disturbance of

the background. Note that the bounding boxes produced by YOLOv4 are initially 2D, which are then transformed into 3D boxes by assigning a fixed vertical box dimension and calculating their vertical coordinates based on ground heights. Therefore, the 3D bounding boxes predicted by YOLOv4 are less accurate in their vertical dimensions than those by the proposed CNN. Finally, Figure 3-10 (c) shows that the proposed CNN yields the best detection results in this example without producing any false negative detections.

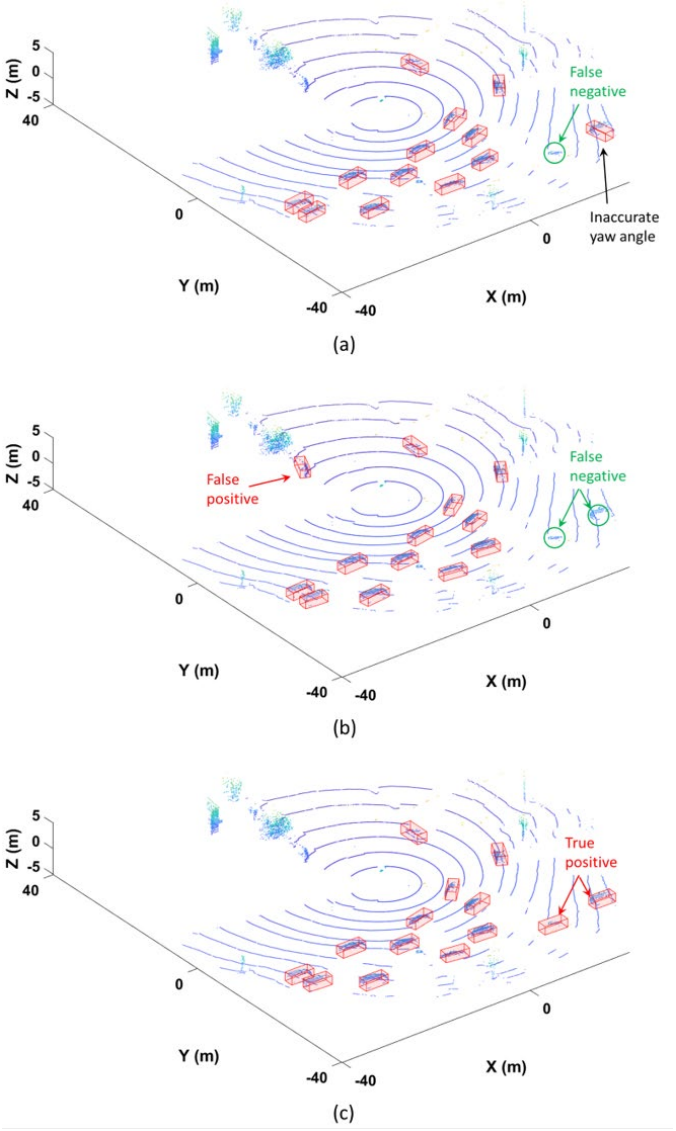


Figure 3-10. Case II: Demonstration of the detection results: (a) PointPillars; (b) YOLOv4; and (c) the Proposed CNN.

3.5. Summary

In the AVs industry, many autonomous driving LiDAR datasets have been made available to the public for deep learning-based vehicle object detection research. On the contrary, there is a lack of large-scale publicly available LiDAR datasets captured from the roadside to train and test deep learning models. This study proposed a methodology based on CNNs, to explore the feasibility and challenges of reusing

CNNs trained on autonomous driving datasets for vehicle detection from roadside LiDAR data. The proposed CNN architecture is modified from the established PointPillars object detection network by adding dense connections between convolutional layers to promote feature fusion and extraction.

In Case I of the experimental study, PandaSet, a publicly available large-scale autonomous driving dataset, was adopted for network training and testing. Three performance metrics, including the Precision, Recall, and F1 score, are employed to quantitatively evaluate the network performance on vehicle object detection. In addition to the proposed CNN, two extensively adopted object detection networks, PointPillars and YOLOv4, are also implemented in the experimental study for cross-comparison. The training and testing results show that the proposed CNN, PointPillars, and YOLOv4 are capable of achieving good detection performance on the autonomous driving dataset; moreover, the proposed CNN outperforms the other two CNNs on the autonomous driving dataset with an improvement of 7.0% and 2.7%, respectively, measured by the F1 score. In Case II, roadside LiDAR data were collected at an intersection in Reno, Nevada, the U.S, for performance evaluation. The trained CNNs from Case I were further utilized to make predictions on the captured roadside LiDAR data. It is shown that the proposed CNN yields higher F1 scores on the roadside LiDAR data than PointPillars and YOLOv4, indicating the best detection performance. Another observation from Case II is that the detection performance of all the three CNNs is slightly deteriorated compared to that in Case I, caused by the impact of different data features and characteristics between the autonomous driving data and roadside LiDAR data. By comparing Case I and Case II, the feasibility of reusing pre-trained CNNs on autonomous driving data for vehicle detection from roadside LiDAR data has been validated with real-world data.

The proposed methodology may have difficulties detecting vehicles under data occlusion. For example, when making predictions on consecutive frames, some vehicle objects occluded by other objects may not be continuously detected by the proposed CNN due to the issue of data occlusion.

Some future research efforts can be devoted to 1) improving the detection performance by refining the proposed CNN architecture to achieve more effective and efficient feature fusion and extraction; 2) developing effective strategies through data fusion to reduce false-negative detections due to data occlusion.

CHAPTER 4. DEEP LEARNING-BASED PEDESTRIAN TRAJECTORY PREDICTION AND RISK ASSESSMENT AT SIGNALIZED INTERSECTIONS USING TRAJECTORY DATA CAPTURED THROUGH ROADSIDE LIDAR

In recent years, rapid advancements in the Autonomous Vehicles (AVs) industry have greatly motivated the research and development in pedestrian trajectory prediction and risk assessment. One of the critical requirements for AVs is to predict the future trajectories of pedestrians and provide collision warnings in an accurate and prompt manner. Nevertheless, accurate prediction of pedestrian trajectories remains a technical challenge, mainly caused by the heterogeneity of pedestrian crossing behavior and uncertainties in vehicle-pedestrian interactions.

In this chapter, we propose a deep learning-based method for pedestrian trajectory prediction and risk assessment, using trajectory data extracted from roadside LiDAR data and corresponding signal phasing information at MLK and Georgia Avenue in Chattanooga, TN. Meanwhile, a set of criteria referred to as the risk factor is established to quantitatively evaluate the risk of the pedestrian crossing behavior, which also serves as a learnable feature. A Long Short-Term Memory (LSTM) network is proposed, which takes the following data as the input: the pedestrian trajectory data, signal phasing data, and risk factors from the past 10 steps. Meanwhile, the network predicts the pedestrian trajectory and risk factor at the future time step. In the experimental study, the root-mean-square errors between the predicted and ground truth x and y coordinates are 0.225 meters and 0.377 meters, respectively, and the F1 score value for the risk factor is 99.6%, demonstrating the efficacy of the proposed LSTM-based methodology on pedestrian trajectory prediction and risk assessment.

4.1. Background

In the past decade, technological advancements in artificial intelligence, smart sensing, digital signal processing, and high-performance computing have extensively promoted the development of the Autonomous Vehicles (AVs) industry (Ridel et al., 2018, Sighencea et al., 2021a). As more AVs operate in urban traffic scenarios, safety concerns have arisen, especially for vulnerable road users (VRUs) such as pedestrians and cyclists. The ability to timely and precisely detect, classify, and predict pedestrian crossing behavior has become a crucial factor and prerequisite for AVs to manage vehicle-pedestrian interactions and avoid/mitigate collision risks, thus improving road safety (Ahmed et al., 2019). However, pedestrian trajectory prediction remains a technical challenge in the existing literature due to multiple factors such as the heterogeneity of pedestrian behavior patterns, uncertainties in vehicle-pedestrian interactions, impacts from traffic signals, subjectivities in the judgments of “right of way”, and diversity of urban intersections and environment (Vizzari et al., 2015, Utriainen, 2020, Tan et al., 2023).

In response to this challenge, researchers and professionals have devoted extensive research efforts to developing pedestrian trajectory prediction methodologies using various types of data, especially vision data (e.g., image, video) from cameras and point cloud data from Light Detection and Ranging (LiDAR) scanners (Wang et al., 2017, Ridel et al., 2018, Sighencea et al., 2021a). Although vision-based and LiDAR-based methodologies have both demonstrated success in pedestrian trajectory prediction tasks, researchers have observed that LiDAR data are advantageous over vision data in the following aspects: *i*) requiring less processing power and computational cost; *ii*) providing more accurate range information

and a larger field of view; and *iii*) being less sensitive to the changing illumination condition (Wang et al., 2017, Zhao et al., 2019b).

Deep Learning (DL)-based methodologies have rapidly emerged for pedestrian trajectory prediction in recent years. Compared to other types of methods, DL-based methodologies can directly learn from data and make self-adaptations, owing to their ability to achieve a high level of feature abstraction through a hierarchical layout, thus mitigating the uncertainties from human intervention or prior user input.

Although existing DL-based methodologies have reported certain successes, further research efforts are needed to fully explore the potential and feasibility of applying DL-based methodologies such as LSTM networks for pedestrian trajectory prediction and risk assessment tasks, especially on roadside LiDAR data. One observation is that the majority of the aforementioned methodologies adopted LiDAR data from autonomous driving scenarios for pedestrian prediction, but very few utilized roadside LiDAR Data for analysis. This may be partly owing to the lack of roadside LiDAR datasets with well-labeled pedestrian and vehicle trajectory data. Roadside LiDAR data have several different features and data characteristics than autonomous driving data, such as sensor-to-object distance and object occlusion (Zhou et al., 2022). Such discrepancies may lead to different feature representations of vehicle-pedestrian interactions and their corresponding motion patterns. Besides, most of the existing methodologies based on LSTM only adopted spatial information such as the location, velocity, and yaw angle of pedestrian or vehicle objects as the input, but they barely took advantage of temporal information such as the Signal Phasing and Timing (SPaT) data which could also be utilized to infer the pedestrian behavior. Moreover, the aforementioned DL-based methodologies did not explore the feasibility of incorporating risk assessment into the formulation of their LSTM network architectures.

In this chapter, we propose a DL-based methodology based on LSTM to perform pedestrian trajectory prediction and risk assessment, using pedestrian and vehicle trajectory data extracted from roadside LiDAR data and the associated SPaT information for sequence-based learning. The technical merits can be summarized as follows:

In this chapter, the feasibility of applying LSTM networks for pedestrian trajectory prediction has been validated using pedestrian and vehicle trajectory data obtained from signalized road intersections. Upon formulating the network input and output, critical features from the trajectory data and the corresponding SPaT data are utilized.

- A deep LSTM network is proposed for pedestrian trajectory prediction and risk assessment. The proposed network is comprised of three branches of feature extraction modules, each consisting of a different number of LSTM layers to extract temporal dependencies from the trajectory and SPaT data. The extracted features are then integrated through feature-level fusion by the network. The experimental study demonstrates that the proposed network architecture can adequately adapt to the trajectory data and yield high accuracy in pedestrian trajectory prediction and risk assessment.
- A set of criteria is established to quantitatively evaluate the risk of pedestrian crossing behavior at each time step, referred to as the risk factor. The risk factor is also incorporated into the network input and output sequence data. Thus, the proposed methodology can provide quantitative risk assessment for each pedestrian at the future time step and meanwhile predict their future trajectories.

4.2. Related Work

In the recent decade, the vigorous development of LiDAR sensing technology has led to a drastic increase in pedestrian trajectory prediction and risk assessment using LiDAR data or multi-sensor fusion data by incorporating LiDAR point cloud features with additional data sources such as video and radar data (Ferguson et al., 2014, Wang et al., 2017, Habibi et al., 2018, Xue et al., 2019, Zhao et al., 2019b, Zhao et al., 2019c). Early methodologies utilized hand-crafted dynamic functions and models such as Kalman filters (KFs) for pedestrian trajectory prediction (Ahmed et al., 2019, Sighencea et al., 2021a). Moosmann and Fraichard (2010) developed an approach to track moving objects (e.g., pedestrians) through multiple algorithms, including feature matching, Iterative Closest Point (ICP), Kalman filtering, and dynamic mapping. Azim and Aycard (2012) proposed a framework based on three-dimensional (3D) LiDAR data to detect, classify, and track moving objects, including pedestrians, cyclists, and vehicles, by using Kalman filtering for object tracking and Global Nearest Neighborhood (GNN) for data association. Zhao et al. (2019c) adopted a discrete Kalman filter to track pedestrian and vehicle trajectories from roadside LiDAR data and then trained a deep autoencoder-artificial neural network (DA-ANN) model using the extracted trajectory data for pedestrian crossing intention prediction. Wu et al. (2018b) proposed a threshold-based method for vehicle-pedestrian near-crash identification based on the trajectories of VRUs extracted from roadside LiDAR data. The position, velocity, and timestamp data from pedestrians and vehicles were considered in determining the thresholds to identify collision risks. Zhao et al. (2019a) proposed a probabilistic model based on the modified Naïve Bayes method to predict pedestrian crossing intention using pedestrian trajectory data extracted from roadside LiDAR data. By developing an infrastructure-based LiDAR sensing system, Zhao (2019) proposed an approach to extract pedestrian and vehicle trajectories through a series of feature generation procedures, including background filtering, object clustering, vehicle/pedestrian classification, and tracking. Wu et al. (2020b) proposed a pedestrian-vehicle near-crash identification method based on roadside LiDAR data. The trajectories of VRUs were extracted through background filtering, lane identification, object clustering, and object tracking. In another study on pedestrian trajectory prediction, Zhou et al. (2021) developed an approach based on multi-sensor fusion to extract pedestrian position and attribute properties by integrating the information collected from roadside LiDAR and smartphone sensors. Li et al. (2022) proposed a probabilistic framework to estimate the risk of pedestrian-vehicle interaction at road intersections using a Gaussian Process Regression (GPR) model for pedestrian trajectory prediction and a Random Forest model to account for different driver maneuvers and behavior. The aforementioned methodologies often rely on hand-crafted feature descriptors, which are expertise-intensive and may require prior user input (Ahmed et al., 2019). Thus, they may have difficulties achieving robust and consistent performance in vehicle-pedestrian-mixed scenarios with real-world complexities.

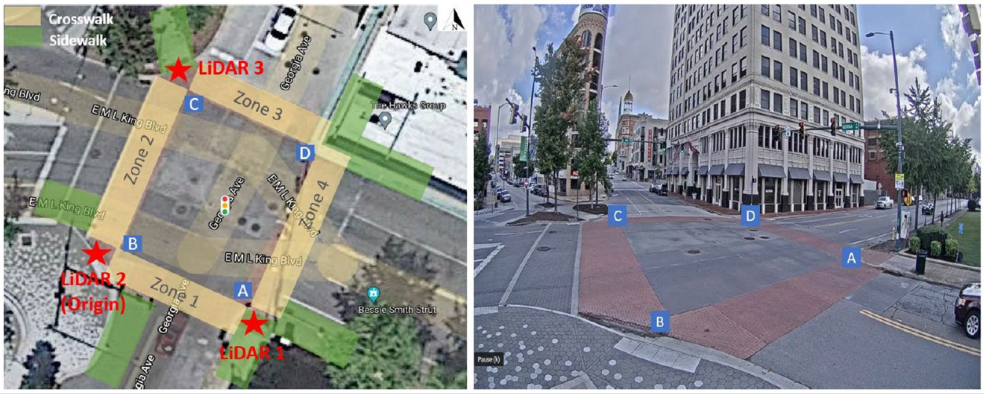
Methodologies using the vanilla Recurrent Neural Network (RNN) and its variants such as Long Short-Term Memory (LSTM) network and Gated Recurrent Unit (GRU) for pedestrian trajectory prediction have been reported in the literature (Ridel et al., 2018, Li et al., 2019, Ahmed et al., 2019). Sun et al. (2018a) developed a DL-based approach for pedestrian trajectory prediction from LiDAR data collected by mobile service robots, in which they trained a shared-triple-layer LSTM network to learn from long-term temporal information and short-term pedestrian pose observations. Xue et al. (2019) proposed a methodology based on an encoder-decoder LSTM network to predict pedestrian trajectories using data captured by a vehicle-mounted LiDAR sensor. The encoder with a two-stream layout was designed to extract features from vehicle and pedestrian trajectories, which were then fused through the decoder

for pedestrian trajectory prediction. Besides, Zhang et al. (2021a) proposed a deep architecture based on Convolutional Neural Networks (CNNs) to predict pedestrian trajectories using LiDAR data captured in autonomous driving scenes. Zhang et al. (2022) proposed a joint detection and tracking scheme to predict the trajectories of moving objects, which involves two parallel procedures: the PointVoxel-RCNN network was employed to detect vehicle and pedestrian objects, while the Unscented Kalman Filter (UKF) and Joint Probabilistic Data Association Filter (JPDAF) were utilized for trajectory prediction. Wang et al. (2022a) developed a deep learning-based multi-sensor fusion approach to detect and track traffic objects. In Wang et al. (2022a), two-dimensional (2D) trajectories were predicted by the improved YOLOv5 network from image data and then fused with 3D trajectories from roadside LiDAR data by applying the improved PointRCNN network.

4.3. Methodology

4.3.1. Trajectory Data and Signal Phasing Data

The roadside LiDAR datasets were collected at MLK and Georgia Avenue in Chattanooga, TN. Three Ouster OS1-128 LiDAR sensors with 10 Hz rotational frequency were installed on the light poles at the three corners of the intersection, as shown in Figure 4-1, to capture the movement of road users, including vehicles, pedestrians, cyclists, etc.



(a) (b)

Figure 4-1. Data collection site: (a) bird-eye view from Google Map; and (b) actual interaction scene from a surveillance camera.

The captured point cloud data were processed by Seoul Robotics to generate trajectory data for pedestrian object recognition and tracking tasks. The x and y coordinates and speed of the pedestrian trajectories are employed as the input data for the proposed LSTM network.

In addition to the point cloud data, Signal Phasing and Timing (SPaT) data were collected by the traffic control device and utilized. From the SPaT data, the timestamps associated with two signal phases, namely the Pedestrian Begin Walk Time (Phase 2) and Pedestrian Begin Walk Time (Phase 4), are utilized as the input to the LSTM network. Signal phases 2 and 4 correspond to the pedestrian "green light" signal status for the crosswalks along the horizontal and vertical directions. By incorporating the elapsed pedestrian "green light" time at each crosswalk into the input to the LSTM network, the network can establish a correlation between the pedestrian motion patterns and the crosswalk signal, which helps predict their future crossing behavior.

4.3.2. Problem Formulation

Let $\boldsymbol{X} = \left\{\left\{x_k, y_k, u_k, v_k, t_{2,k}, t_{4,k}, r_k\right\}_{k=k_o-N+1}^{k_o}\right\}$ be the observed trajectory data from step $k_o - N + 1$ to k_o, where k_o denotes the current step, N denotes the sequence length, equal to 10; x_k, y_k denote the x and y coordinates of the pedestrian at step k, respectively; u_k, v_k denote the x and y speed of the pedestrian at step k, respectively; $t_{2,k}, t_{4,k}$ denote the elapsed time from the most recent Pedestrian Begin Walk Time in Phase 2 and Phase 4 to step k, respectively; r_k denotes the risk factor at step k; and, let $\boldsymbol{Y} = \left\{x_{k_o+1}, y_{k_o+1}, r_{k_o+1}\right\}$ be the trajectory data and risk factor at step $k_0 + 1$. The proposed DL-based methodology aims to learn a regression function that maps $\boldsymbol{X}$ to $\boldsymbol{Y}$, as expressed in (4-1):

$$\text{Regression: } \boldsymbol{X} \rightarrow \boldsymbol{Y} \tag{4-1}$$

4.3.3. Risk Assessment

A binary prediction approach for risk assessment is proposed. Two criteria are established to quantitatively evaluate the risk of pedestrian crossing behavior at each time step, as expressed in (4-2) through (4-9). The first criterion in (4-6) is based on "Colliding Past/Future Trajectories", where a pedestrian object is considered at risk if its distance to any vehicle object during the past or future 2 secs is smaller than a tolerance value and meanwhile its head-to-head speed towards the vehicle is larger than a tolerance value. The second criterion in (4-7) is established based on "Unexpected Pedestrian Behavior" to assess the collision risk of pedestrian objects using their location information; that is, a pedestrian object is considered at risk if it enters a pre-defined unsafe region (i.e., roadway) as illustrated in Figure 4-2. It is worth noting that the process to define the unsafe region is dependent on the position and shape of the roadways, crosswalks, and sidewalks at each road intersection, following a case-by-case manner, and thus it is a manual procedure.

- *Criterion 1: Collision Risk based on Colliding Past/Future Trajectories:*

The relative position of a pedestrian object (denoted as p) with respect to a vehicle object (denoted as h) during the past or future 2 secs can be expressed as (4-2c):

$$\tilde{x}_k = x_{p,k} - x_{h,k\pm2} \tag{4-2a}$$

$$\tilde{y}_k = y_{p,k} - y_{h,k\pm2} \tag{4-2b}$$

$$\vec{a} = (\tilde{x}_k, \tilde{y}_k) \tag{4-2c}$$

where $x_{p,k}, y_{p,k}$ denote the x and y coordinates of pedestrian p at step k, respectively; $x_{h,k\pm2}$ and $y_{h,k\pm2}$ are the x and y coordinates of vehicle h at step $k \pm 2$ secs.

Now, define a distance threshold based on the relative position as (4-3):

$$d_k = \begin{cases} 1 & \|\vec{a}\| < tol_1 \\ 0 & otherwise \end{cases} \tag{4-3}$$

In this equation, $\|\cdot\|$ denotes a vector norm; tol_1 denotes the "safe" distance threshold between the pedestrian object and nearby vehicle objects, determined as 1 meter.

Meanwhile, the relative velocity of the pedestrian object with respect to the vehicle object is expressed as (4-4c).

$$\tilde{u}_k = u_{p,k} - u_{h,k\pm2} \tag{4-4a}$$

$$\tilde{v}_k = v_{p,k} - v_{h,k\pm2} \tag{4-4b}$$

$$\vec{b} = (\tilde{u}_k, \tilde{v}_k) \tag{4-4c}$$

where $u_{p,k}, v_{p,k}$ refer to the x and y speed of pedestrian p at step k, respectively; $u_{h,k\pm2}$ and $v_{h,k\pm2}$ denote the x and y speed of vehicle h at step $k \pm 2$ secs, respectively.

Similarly, a velocity threshold based on the relative position and velocity of the pedestrian object with respect to the vehicle object can be defined as follows:

$$s_k = \begin{cases} 1 & \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|} > tol_2 \\ 0 & otherwise \end{cases} \tag{4-5}$$

where $\vec{a}$ and $\vec{b}$ are the relative position and velocity vector of the pedestrian with respect to the vehicle, defined in (4-2c) and (4-4c), respectively; tol_2 denotes the "safe" relative speed threshold between the pedestrian object and nearby vehicle objects, determined as 5 m/s.

By jointly considering the thresholding results from the relative position and velocity, the collision risk based on Colliding Past/Future Trajectories is formulated as (4-6). Note that, for each pedestrian, (4-6) is evaluated for all the nearby vehicle objects, to jointly determine the collision risk based on a logical "or" operation.

$$r_{1,k} = d_k \cdot s_k \tag{4-6}$$

where $r_{1,k}$ refers to the risk factor at step k, calculated based on Criterion 1.

- *Criterion 2: Collision Risk based on Unexpected Pedestrian Behavior:*

$$r_{2,k} = \begin{cases} 1 & (x_k, y_k) \in \boldsymbol{R} \\ 0 & otherwise \end{cases} \tag{4-7}$$

where $r_{2,k}$ denotes the risk factor at step k, calculated based on Criterion 2; $\boldsymbol{R}$ refers to a pre-defined "unsafe" region (i.e., roadway) for the pedestrian object, as illustrated in Figure 4-2. In this figure, the pedestrian trajectory history is overlaid on the road intersection. The majority of the "green" dots fall into the sidewalk and crosswalk locations and therefore are considered "safe".

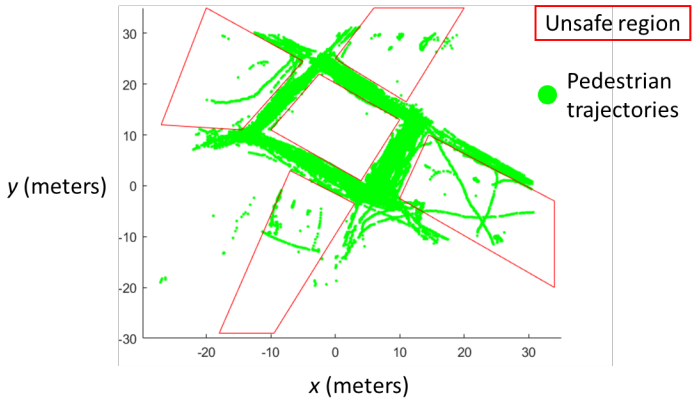


Figure 4-2. Illustration of the pre-defined "unsafe" region for pedestrians.

Thus, based on the above two criteria, the risk factor for each pedestrian at step k can be expressed as (4-8). The operator "V" represents a logical "or" operation in this equation. $r_k = 1$ (i.e., true) indicates "high-risk" pedestrian crossing behavior at step k, while $r_k = 0$ (i.e., false) indicates "safe" crossing behavior.

$$r_k = r_{1,k} \ \mathrm{V} \ r_{2,k} \tag{4-8}$$

Furthermore, a scale factor and an offset factor are employed to transform the risk factor into non-zero input to the LSTM network, thus enhancing the training performance, as expressed in (4-9).

$$\overline{r}_k = f_1 \cdot r_k + f_2 \tag{4-9}$$

where f_1 and f_2 are the scale factor and offset factor, respectively; in the experimental study, their values are selected as 10 and 0.5, respectively; $\overline{r}_k$ denotes the modified risk factor that participates in the network training and testing process.

4.3.4. Data Pre-processing

(1) Data Normalization

Prior to network training and testing, data normalization, as expressed in (4-10), is performed to regularize the input and improve network generalization over a wide range of input data. It is worth noting that the input data, except the risk factor, are normalized through (4-10).

$$\bar{x} = \frac{x - \mu}{\sigma} \tag{4-10}$$

where x denotes the raw feature input; μ and σ are the mean and standard deviation of feature x, respectively, calculated on the training dataset; $\bar{x}$ refers to the normalized sequence data ready for network training and testing.

(2) Data Transformation

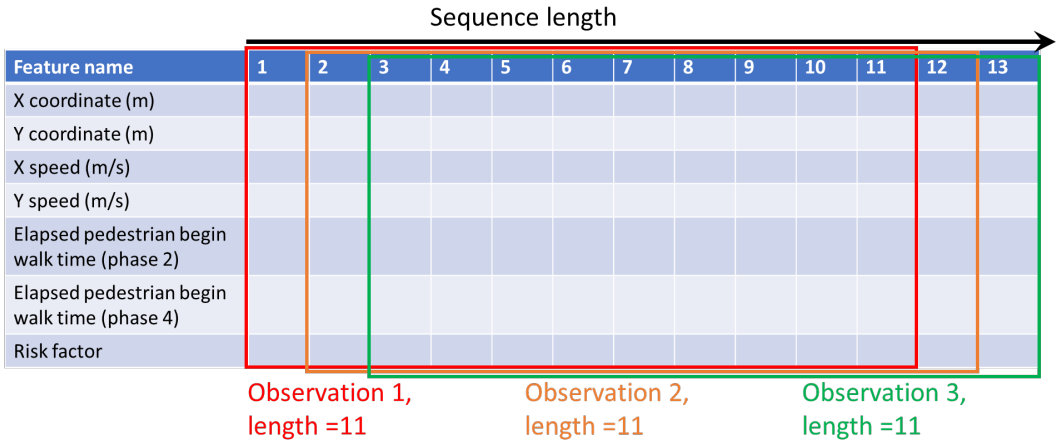


Figure 4-3. An example of the data transformation process.

The obtained pedestrian trajectory data have varying sequence lengths. Note that the time step in each sequence is 0.1 secs. In this study, only the pedestrian trajectory data whose sequence lengths are longer than 10 are utilized for network training and testing. The selected pedestrian trajectory data are randomly split into training and testing datasets. Then, a data transformation procedure is applied to

convert the training and testing data with varying lengths into fixed-length sequence data. An illustrative example is provided in Figure 4-3. In Figure 4-3, the sequence length of the original data is 13. Through a sliding window procedure which generates consecutive sequences with length = 11 and stride = 1, the original sequence is transformed into 3 new observations with the same sequence length.

(3) Generating Predictor and Response Data from Each Fixed-length Sequence

The final step of data pre-processing is to separate the normalized fixed-length sequence data into predictor and response data, which serve as the network input and output, respectively. As shown in Figure 4-4, the predictor data are comprised of all the features from the first 10 steps of the fixed-length sequence data. In contrast, the response data are constructed by selecting only the *x* coordinate, *y* coordinate, and risk factor at the 11th step. Note that the selection of sequence length is based on prior knowledge. The network does not need to make predictions on the Elapsed Pedestrian Begin Walk Time (Phases 2 and 4) since they can be directly calculated using information from previous observations.

Feature name	1	2	3	4	5	6	7	8	9	10	11
X coordinate (m)											
Y coordinate (m)											
X speed (m/s)											
Y speed (m/s)											
Elapsed pedestrian begin walk time (phase 2)											
Elapsed pedestrian begin walk time (phase 4)											
Risk factor											

Predictor Response

Figure 4-4. Generating predictor and response data from each fixed-length sequence.

4.3.5. Long Short-Term Memory (LSTM)

The vanilla RNN may yield deteriorated performance when processing long-sequence data because of the "vanishing or exploding gradient" problem (Hochreiter and Schmidhuber, 1997). As a variant of RNN, LSTM was proposed by Hochreiter and Schmidhuber (1997), which can effectively address the problem by adding special units such as memory cells and gate control to model the long-term dependencies in the long-sequence data. The selective remember-forget mechanism in LSTM has made it well-suited for sequence-based prediction. An LSTM unit is comprised of a memory cell, an input gate (i_t) [(4-11)], a forget gate (f_t) [(4-12)], and an output gate (o_t) [(4-13)], which collaboratively control the information flow in the LSTM unit. At time step t, the LSTM unit takes c_{t-1} (i.e., cell state from the previous time step), x_t (i.e., input vector from the current step), and h_{t-1} (i.e., hidden state output from the previous time step) as the input. It generates c_t and h_t [(4-14) and (4-15)] which are to be used by the future time step. Interested readers can refer to (Hochreiter and Schmidhuber, 1997) for more detailed explanations and formulations on LSTM.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (4\text{-}11)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (4\text{-}12)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (4\text{-}13)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (4\text{-}14)$$

$$h_t = o_t \odot \tanh(c_t) \quad (4\text{-}15)$$

where σ denotes the sigmoid function; W and b are the weights and biases, respectively; x_t, c_t and h_t denote the input vector, cell state, and hidden state output at time step t, respectively; the operator $\odot$ denotes element-wise multiplication.

4.3.6. Proposed LSTM Network Architecture

Figure 4-5 shows the layout of the proposed three-branch LSTM network. The proposed network architecture is comprised of three branches of feature extraction modules, each consisting of a different number of LSTM layers. A batch normalization layer (Ioffe and Szegedy, 2015) is utilized to regularize the input mini-batch sequence data. A dropout layer and a fully connected layer (Goodfellow et al., 2016) are employed at the end of each branch. The purpose of the dropout layer is to avoid model overfitting by randomly deactivating some neuron input by a probability (i.e., dropout factor) (Srivastava et al., 2014). Then, the fully connected layer condenses the input features into a 3x1 vector. At the end of the fully connected layers, the extracted features from each branch are integrated through feature-level fusion by an "addition" operation. The proposed three-branch architecture design allows the network to adapt to the training data and adequately reflect the data complexity by tuning the weights in each branch through training. Finally, the regression layer calculates the half-mean-square-error loss between the ground truth and predicted response data. In Figure 4-5, the number inside the parentheses in each LSTM layer indicates the dimension of the hidden state output in that LSTM layer.

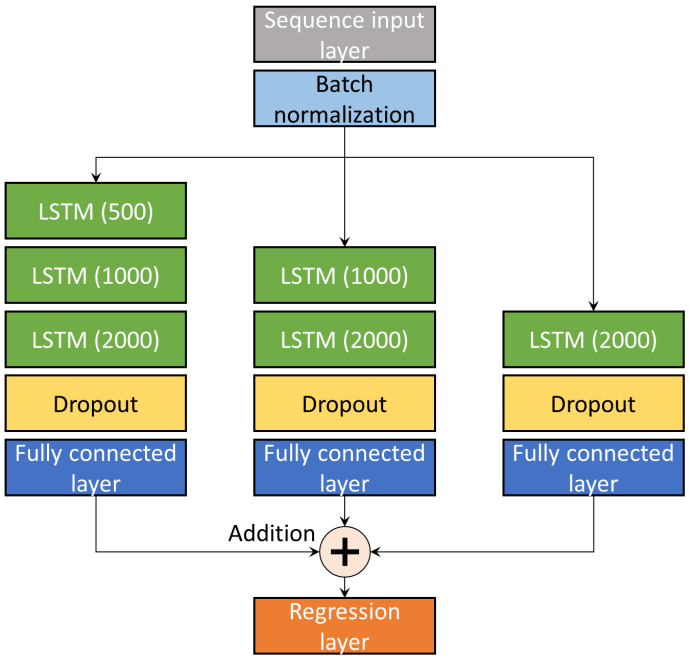


Figure 4-5. Proposed three-branch LSTM network for pedestrian trajectory prediction and risk assessment.

4.4. Experimental Study and Results

4.4.1. Dataset Generation

The pedestrian trajectory data are normalized and then randomly split into training and testing datasets, following a ratio of 90%:10%. Then, by applying the aforementioned data transformation procedure to the training and testing datasets, the pedestrian trajectory data with varying sequence lengths in each dataset are transformed into more sequence data with a fixed sequence length. Finally, the fixed-length sequence data are separated into the predictor and response data (see Figure 4-4) for network training and testing purposes. Table 4-1 tabulates the total number of training and testing data.

Table 4-1. Training and testing datasets.

Dataset name	**Number of predictor data (dimension: 5×10)**	**Number of response data (dimension: 3×1)**
Training	52,000	52,000
Testing	5,635	5,635

4.4.2. Experimental Setup

(1) Computing Environment

The network implementation, training, and testing are accomplished using MATLAB (MATHWORKS, 2022) deep learning toolbox and LiDAR toolbox, with the computer specifications as follows: CPU: Intel Xeon Gold 6254 @ 3.10 GHz; GPU: Nvidia RTX Quadro 8000 with 48 GB RAM.

(2) Hyperparameter Configuration

On training the LSTM network, the stochastic gradient descent with momentum algorithm (Bengio, 2012) is adopted as the optimization algorithm. The values for the training hyperparameters are as follows: mini-batch size (1000), number of epochs (100), initial learning rate (0.02), learning rate drop period (10 epochs), learning rate drop factor (0.2), weight decay factor (0.0001), momentum (0.9), dropout factor (0.5).

(3) Network Parameter Initialization

The weights are initialized using the Glorot initializer (Glorot and Bengio, 2010), which randomly draws samples from a Gaussian distribution with zero mean and a variance based on the dimension of the weights. The initial biases are all set as zeros. In the batch normalization layer, the initial scale factor and shift factor are equal to one and zero, respectively.

4.4.3. Performance Metrics

(1) Root-Mean-Square-Error (RMSE)

This study uses the Root-Mean-Square-Error (RMSE) to measure the discrepancy between the predicted and ground truth coordinate data evaluated on the testing dataset. The expression of RMSE is provided in (4-16).

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(g_i - p_i)^2}{n}} \tag{4-16}$$

where n denotes the total number of responses in the testing dataset, equal to 5,635; g_i and p_i refer to the ground truth and predicted coordinate data, respectively.

4.4.4. Precision-Recall Analysis

The precision-recall analysis (Fawcett, 2006) is adopted to evaluate the prediction accuracy in the risk factor quantitatively. The precision-recall analysis consists of three metrics, namely the Precision, Recall, and F1 score, expressed as (4-17), (4-18), and (4-19), respectively. The Precision is calculated as the number of true-positive detections (i.e., risk factor = 1) over the total number of positive detections; the Recall is equal to the number of true-positive detections divided by the total number of at-risk behavior (i.e., risk factor = 1); and, as a harmonic mean of the Precision and Recall, the F1 score provides a comprehensive measure on the accuracy in risk assessment.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4-17}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4-18}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4-19}$$

where TP (i.e., true positive) denotes the number of correctly identified at-risk behavior (i.e., risk factor = 1) by the network; FP (i.e., false positive) denotes the number of safe behavior (i.e., risk factor = 0) misidentified as at-risk, and FN (i.e., false negative) denotes the number of at-risk behavior misidentified as safe.

4.4.5. Experimental Results

The training process has gone through 100 epochs and reached convergence. The trained LSTM network is then utilized to predict pedestrian trajectories and their risk factors on the testing dataset. Subsequently, a reverse data normalization operation is performed on the predicted trajectory data to restore the *x* and *y* coordinates to their original scale.

Figure 4-6 and Figure 4-7 illustrate the prediction results on the *x* and *y* coordinates of future pedestrian trajectories, respectively. In Figure 4-6 (a) and Figure 4-7 (a), the ground truth data are plotted against the predicted data; additionally, the "*y* = *x*" line is overlaid with the scatter plot; that is, if the prediction is "perfect", then the data will fall on the diagonal line in the plot. Besides, the difference between the predicted and ground truth response data is calculated, with its histogram provided in Figure 4-6 (b) and Figure 4-7 (b). Meanwhile, the RMSE between the predicted and ground truth response data are calculated from the testing dataset and tabulated in Figure 4-6. Table 4-4 shows the confusion matrix for the prediction results on the risk factor, and the corresponding Precision, Recall, and F1 score values are provided in Table 4-3. Several observations and conclusions can be made from Figure 4-6, Figure 4-7, Table 4-2, Table 4-3, and Table 4-4 as follows:

- As can be observed from Figure 4-6 (a) and Figure 4-7 (a), the majority of the data points are very close to the "$y = x$" line, indicating very accurate prediction performance on the x and y coordinates. Additionally, from the histogram plots Figure 4-6 (b) and Figure 4-7 (b), it can be observed that the discrepancies between the predicted and ground truth coordinate data are quite small, where the mean (μ) and standard deviation (σ) values are as follows: μ = -0.030 meters, σ = 0.223 meters for the x coordinate, and μ = 0.017 meters, σ = 0.377 meters for the y coordinate, respectively. The RMSE between the predicted and ground truth x and y coordinates are 0.225 meters and 0.377 meters, respectively.
- Based on the confusion matrix in Table 4-4, the Precision, Recall, and F1 score values for the risk factor are 99.6%, 99.6%, and 99.6%, respectively, demonstrating high accuracy in risk assessment.
- Overall, based on the experimental results, the efficacy of the proposed methodology on pedestrian trajectory prediction and risk assessment has been validated through real-world data.

Table 4-2. RMSE between the predicted and ground truth x and y coordinates.

	***x* coordinate (meters)**	***y* coordinate (meters)**
RMSE	0.225	0.377

Table 4-3. Precision-recall analysis of the prediction results on the risk factor.

	Precision (%)	**Recall (%)**	**F1 score (%)**
Risk factor	99.6	99.6	99.6

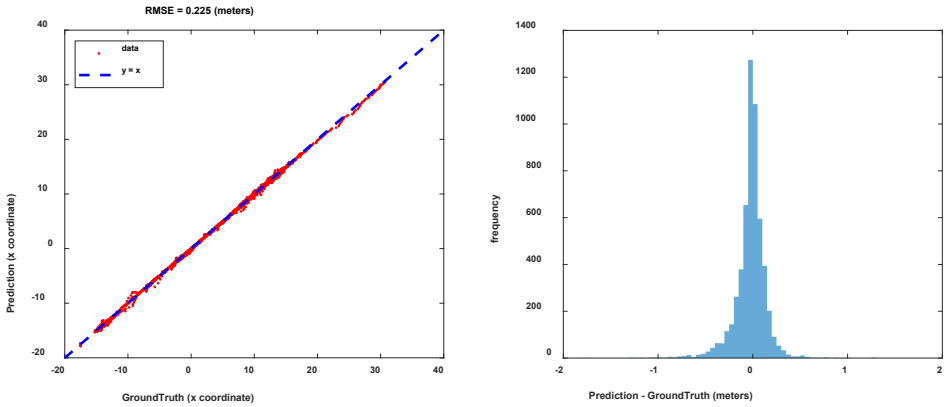


Figure 4-6. Prediction results on the x coordinate: (a) GroundTruth vs. Prediction; and (b) histogram of (Prediction - GroundTruth).

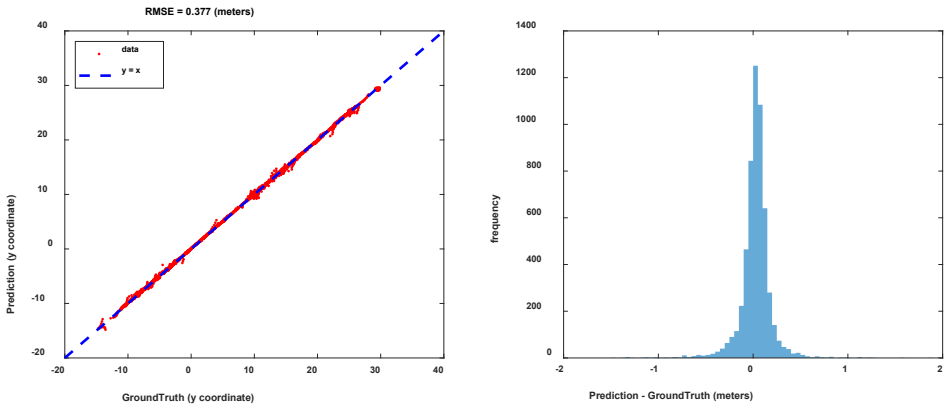


Figure 4-7. Prediction results on the *y* coordinate: (a) GroundTruth vs. Prediction; and (b) histogram of (Prediction - GroundTruth).

Table 4-4. Confusion matrix for the prediction results on the risk factor.

		Ground Truth	
		Positive	**Negative**
Prediction	**Positive**	True Positive = 690	False Positive = 3
	Negative	False Negative = 3	True Negative = 4939

In addition to showing the scatter plots and histograms of the predicted response data over the entire testing dataset, Figure 4-8 selects the first 100 sequence data from the testing dataset to further demonstrate the network prediction performance. For the bar plots in Figure 4-8 (a), (b), and (c), the horizontal axis corresponds to the observation index, ranging from 1 to 100; the vertical axes refer to the *x*, *y* coordinates and risk factor of the pedestrian trajectory at the future time step, respectively. Ground truth data (blue color) are plotted together with the predicted data (orange color). A close match between the ground truth and predicted responses is observed from Figure 4-8, demonstrating very accurate prediction performance by the proposed methodology.

Figure 4-9 illustrates an example of applying the trained LSTM network for pedestrian trajectory prediction and risk assessment on long-sequence data (length = 454). The prediction is performed iteratively by cropping the original data into fixed-length sequences, which serve as the input for the network. Figure 4-9 (a) plots the *x* coordinate, *y* coordinate, and risk factor of the pedestrian trajectory, respectively. Note that the predicted risk factor illustrated in the plot represents a probability value between 0 and 1 rather than a binary number (i.e., 0 or 1) used in the quantitative performance evaluation. Furthermore, a bird-eye view of the pedestrian trajectory is displayed in Figure 4-9 (b). Figure 4-9 shows the ground truth and predicted response data in black and red colors, respectively. It can be observed from the time history plots in Figure 4-9 (a) and the bird-eye-view plot in Figure 4-9 (b) that the trained LSTM network yields very accurate prediction on the *x* and *y* coordinates, where the predicted coordinates match very well with the ground truth data. Meanwhile, from Figure 4-9 (a), it can be seen that the proposed network can accurately predict the risk factor as well.

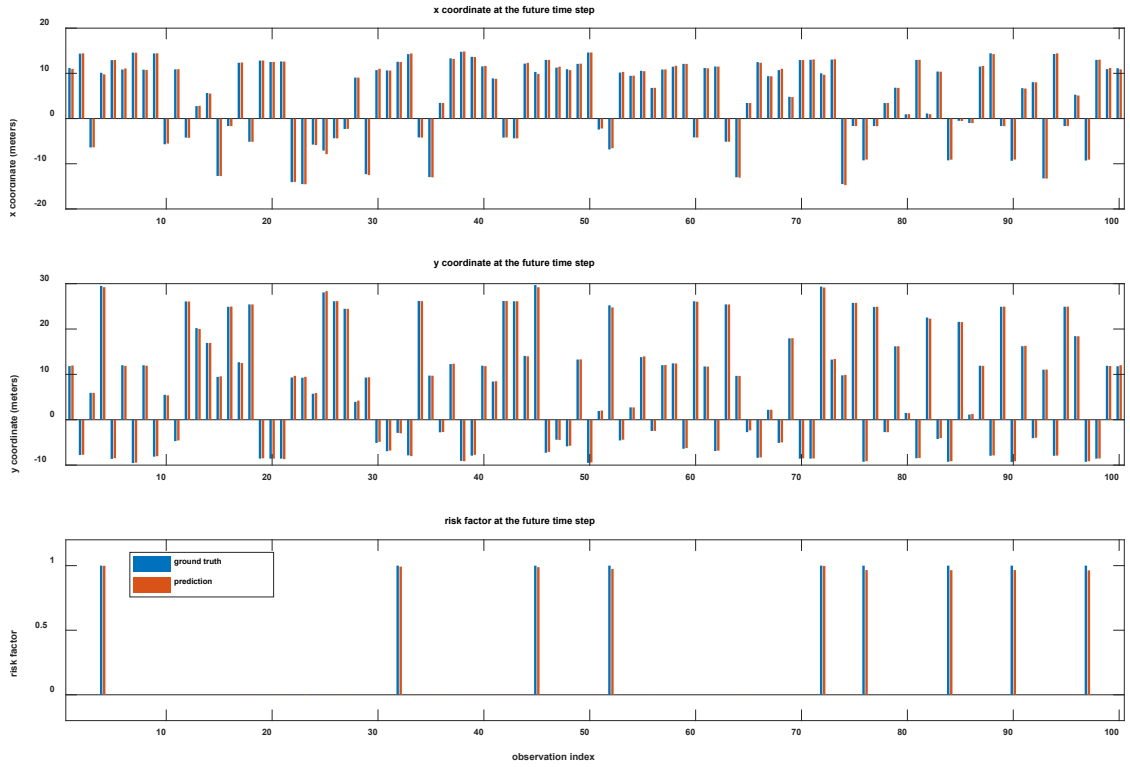
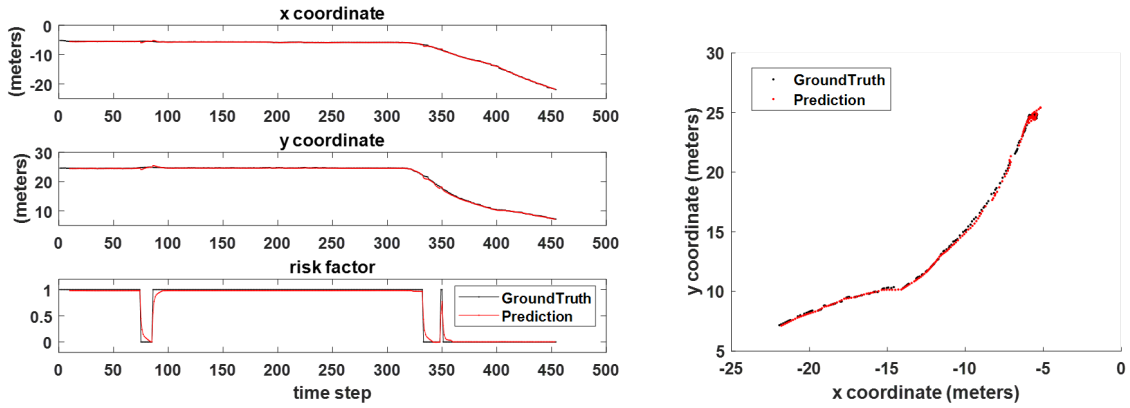


Figure 4-8. Demonstration of the prediction results on the first 100 sequences: (a) *x* coordinate; (b) *y* coordinate; and (c) risk factor.



(a) (b)

Figure 4-9. Demonstration of the prediction result on long-sequence data: (a) *x* coordinate, *y* coordinate, and risk factor; and (b) bird-eye view of the trajectory.

4.5. Summary

In this chapter, we proposed a Deep Learning (DL)-based methodology utilizing LSTM for pedestrian trajectory prediction and risk assessment. The proposed methodology leverages the trajectory data extracted from roadside LiDAR data and the corresponding signal phasing information to learn temporal dependencies between sequence data. In addition, risk assessment of pedestrian crossing behavior is

incorporated into the proposed methodology by establishing two criteria, namely the collision risks based on “Colliding Past/Future Trajectories” and based on “Unexpected Pedestrian Behavior”, respectively. The risk factor at each time step is evaluated from the two criteria and further incorporated into the input and output of the LSTM network.

The proposed LSTM network contains three branches of feature extraction modules, each consisting of a different number of LSTM layers to extract temporal dependencies from the input data. The extracted features are then integrated through feature-level fusion by the network. The proposed LSTM network takes the following features from the past 10 steps as the input: *i*) the coordinate and speed data from pedestrian trajectories, *ii*) Signal Phasing and Timing (SPaT) data, and *iii*) risk factors. Meanwhile, the network makes predictions on the *x*, *y* coordinates and the risk factor at the future time step.

In the experimental study, roadside LiDAR data and SPaT data were collected at MLK and Georgia Ave in Chattanooga, TN, under complex traffic scenarios. Pedestrian trajectory data, SPaT data, and risk factors were generated and pre-processed through a series of procedures, including data normalization, data transformation, and dataset generation. In total, 52,000 fixed-sequence data participated in network training and 5,635 in testing, respectively. The RMSE values between the predicted and ground truth *x* and *y* coordinates calculated from the testing dataset are 0.225 meters and 0.377 meters, respectively, indicating high accuracy in trajectory prediction. Meanwhile, the F1 score value for the risk factor is 99.6%, which shows that the proposed methodology can achieve very accurate risk assessment as well. Thus, the experimental study has validated the efficacy of the proposed LSTM-based methodology for pedestrian trajectory prediction and risk assessment under complex traffic scenarios using real-world data.

Future research efforts need to be devoted to 1) improving the network prediction performance through more robust and efficient data pre-processing procedures and training schemes, and 2) developing LSTM encoder-decoder networks to perform long-term (output sequence > 2.5 secs) pedestrian trajectory prediction and risk assessment on roadside LiDAR data.

CHAPTER 5. OPTIMIZED LONG SHORT-TERM MEMORY NETWORK FOR LIDAR-BASED VEHICLE TRAJECTORY PREDICTION THROUGH BAYESIAN OPTIMIZATION

Conventional approaches for vehicle trajectory prediction, such as Kalman filtering, usually require extensive user expertise or prior knowledge for parameter tuning and selection. Researchers have recently developed deep learning-based methodologies such as Long Short-Term Memory (LSTM) networks for vehicle trajectory prediction on LiDAR data. Nevertheless, challenges associated with the learning-based methodologies, such as human intervention and subjectivity in hyperparameter selection, may still exist and thus impact the prediction performance.

This study proposes a LiDAR-based deep learning framework for vehicle trajectory prediction, which leverages LSTM networks to predict vehicle trajectories and Bayesian optimization to determine the optimal hyperparameter configuration. In the experimental study, a vehicle trajectory dataset extracted from roadside LiDAR data was utilized for network training and testing; the optimal LSTM network obtained through Bayesian optimization was compared against a benchmark LSTM network with handpicked hyperparameters. The results show that the proposed deep learning-based framework with robust hyperparameter selection through Bayesian optimization yields more accurate and consistent prediction performance than the benchmark network.

5.1. Background

In the past decade, Light Detection and Ranging (LiDAR) technology has rapidly emerged as one of the most popular and advanced sensing technologies for transportation research and applications, including Traffic Big Data (TBD) analytics, surveying, transportation system modeling and optimization, Connected Autonomous Vehicles (CAV), traffic object recognition and tracking, sensor fusion technologies, etc. (Sun et al., 2022, Mekala et al., 2021, Sighencea et al., 2021b). Compared to traditional data acquisition devices such as cameras, LiDAR sensors have demonstrated several significant advantages, such as insensitivity to changing illumination conditions, high-resolution distance measurement, wide-range detection, and high-speed data acquisition (Wang et al., 2017, Zhao et al., 2019b, Sun et al., 2022, Barad, 2021). With such advantages, LiDAR sensors have been frequently deployed on mobile platforms or transportation infrastructures, to provide distance measurement in traffic scenarios where high-speed acquisition of high-quality three-dimensional (3D) data during daytime and nighttime is required, such as autonomous driving.

The vigorous developments of the Autonomous Vehicles (AVs) industry in recent years have given rise to an ever-increasing demand for accurate and prompt perception and recognition of traffic objects, such as vehicles, traffic signs, obstacles, bicyclists, pedestrians, and other Vulnerable Road Users (VRUs), to promote transportation safety and efficiency in autonomous driving (Sun et al., 2022, Leon and Gavrilescu, 2021). In traffic safety analysis, vehicle detection and tracking involves obtaining the precise location, velocity, identity, and other critical attributes of vehicle objects from sequence data captured in a traffic scene (Leon and Gavrilescu, 2021). Meanwhile, vehicle trajectory prediction is focused on estimating vehicle objects' future motion and behavior patterns (Leon and Gavrilescu, 2021). Providing accurate, real-time prediction on vehicle trajectories in highly interactive traffic environments is of vital importance to support prompt decision-making in traffic safety analysis, including motion estimation, obstacle avoidance, vehicle-pedestrian interaction prediction, collision risk assessment, near-crash identification, etc. As the emerging LiDAR technology has proven its advantages over traditional data

acquisition approaches and demonstrated wide applicability in traffic scene perception, researchers have devoted substantial efforts to developing state-of-the-art methodologies leveraging LiDAR data for vehicle trajectory prediction.

Commonly used methods for vehicle trajectory prediction, such as Kalman filters (KFs) (Xie et al., 2017, Lefkopoulos et al., 2020, Baek et al., 2020), Particle filters (PFs) (Hoermann et al., 2017, Fünfgeld et al., 2017), Monte Carlo (MC) sampling (Wang et al., 2019), hidden Markov models (HMMs) (Deo et al., 2018, Liu et al., 2020), and Dynamic Bayesian Networks (DBNs) (Schulz et al., 2018a, Schulz et al., 2018b, Jiang et al., 2022), usually require establishing a motion model with handcrafted parameters to predict vehicle trajectories by considering the law of physics (i.e., physics-based), maneuver patterns (i.e., maneuver-based), or the interaction between neighboring traffic objects (i.e., interaction-aware) (Lefèvre et al., 2014, Leon and Gavrilescu, 2021). Although these classic methods have been extensively adopted for motion prediction tasks, they are subject to several disadvantages, as mentioned in (Leon and Gavrilescu, 2021). For example, their performance is sensitive to the choice of motion models and initial conditions, which rely heavily on a careful design and selection procedure through prior knowledge or trial-and-error, and thus are often expertise-intensive and subjective.

Recent years have witnessed an ever-growing popularity of deep learning-based methods, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for traffic object recognition and motion prediction tasks, owing to the rapid advancements in high-performance computing devices to facilitate compute-intensive programming. By adapting to the domain-specific data through self-learning, deep learning-based methods can achieve hierarchical feature extraction and pattern recognition without the need to manually design features (Goodfellow et al., 2016).

Long Short-Term Memory (LSTM) network is a popular deep learning-based method that has been extensively adopted for vehicle trajectory prediction (Leon and Gavrilescu, 2021, Huang et al., 2022). As a variant of RNNs, the LSTM network is designed to address the “vanishing or exploding gradient” problem when processing long-sequence data through the selective remember-forget mechanism in each LSTM unit (Hochreiter and Schmidhuber, 1997, Hochreiter, 1998). Various studies on traffic object recognition and motion prediction have demonstrated that LSTM networks are very suited to handling data with long-term dependencies (Leon and Gavrilescu, 2021).

Nevertheless, several challenges related to LSTM-based vehicle trajectory prediction are yet to be fully addressed. One of the pressing challenges lies in the proper configuration of hyperparameters. The hyperparameters in a deep learning model refer to the parameters that determine the model architecture (e.g., network depth) or control the training process (e.g., learning rate). Different than the learnable parameters such as weights and biases, which are to be updated through gradient-based optimization, the hyperparameters are configured prior to network training and play a vital role in determining the model performance. However, in the literature, a wide variety of hyperparameter configurations for LSTM-based vehicle trajectory prediction have been reported, because there is currently no well-established strategy to determine which LSTM architecture is best suited to a particular task or scenario. Thus, selecting hyperparameters is often an empirical process (Leon and Gavrilescu, 2021). As pointed out by (Leon and Gavrilescu, 2021), in addition to a significant amount of appropriate data for training, deep learning-based approaches for vehicle trajectory prediction tasks require a careful selection of hyperparameters, which is often performed through a substantial amount of experimentation (e.g., grid search, random search) or trial-and-error (e.g., manual search). For

example, in (Kim et al., 2017, Lin et al., 2021), the hyperparameters related to the LSTM architectures were tuned through grid search. Overall, very few studies have explored designing a systematic approach to search for the optimal hyperparameter configuration to enhance the performance of deep learning models for vehicle trajectory prediction tasks.

To address the challenge, in this chapter, we propose an automated and systematic framework to explore the optimal LSTM network performance through Bayesian optimization (Brochu et al., 2010, Snoek et al., 2012, Shahriari et al., 2015, Frazier, 2018) rather than relying on empirical knowledge or trial-and-error. The technical merit can be summarized as follows:

- A deep learning framework based on LSTM and Bayesian optimization is proposed for vehicle trajectory prediction, in which the architecture- and training-related hyperparameters are updated during optimization.
- The feasibility and efficacy of the proposed framework are successfully validated by comparing the performance between the optimal LSTM network vs. a handcrafted one, using vehicle trajectory data extracted from roadside LiDAR data under complex urban traffic scenarios.
- A sensitivity analysis is performed to investigate the robustness of the proposed methodology subject to different levels of perturbation around the optimal hyperparameter values.

5.2. Related Work

In this section, popular methodologies on vehicle trajectory prediction are introduced in detail. Firstly, the developments and limitations of classic methods that are not deep learning-based are reviewed. Subsequently, the recent works of deep learning-based methods, particularly LSTM networks, are discussed.

5.2.1. Classic Methods for Vehicle Trajectory Prediction

On vehicle trajectory prediction, classic methods, such as Kalman filters (KFs) (Xie et al., 2017, Lefkopoulos et al., 2020, Baek et al., 2020), particle filters (PFs) (Hoermann et al., 2017, Fünfgeld et al., 2017), Monte Carlo (MC) sampling (Wang et al., 2019), hidden Markov models (HMMs) (Deo et al., 2018, Liu et al., 2020), and Dynamic Bayesian Networks (DBNs) (Schulz et al., 2018a, Schulz et al., 2018b, Jiang et al., 2022), have been extensively adopted.

Xie et al. (2017) developed interactive multiple model trajectory prediction using multi-source data, which integrates the trajectory prediction from a physics-based motion model through the Unscented Kalman Filter (UKF) and the prediction from a maneuver-based motion model considering uncertainty. Lefkopoulos et al. (2020) designed a scheme for single/multiple vehicle motion prediction based on an Interacting Multiple Model Kalman Filter (IMM-KF), by combining ideas from physics-based, maneuver-based, and interaction-aware approaches. Baek et al. (2020) proposed a Kalman filter-based approach for vehicle trajectory prediction and collision risk assessment through multi-modal data fusion. The measurements from each sensor were processed through a Kalman filter and then fused to obtain the target state estimates. Then, trajectory prediction for the detected vehicle targets was performed by employing the constant turn rate and velocity (CTRV) motion model.

Hoermann et al. (2017) designed a probabilistic motion model based on a particle filter that continuously estimates the driving style parameters of the Intelligent Driver Model (IDM) and the relational motion between traffic objects. In Hoermann et al. (2017), the particle filter was adopted to

handle continuous behavior changes inter and intra driver with arbitrarily shaped parameter distribution. Fünfgeld et al. (2017) developed a stochastic forecasting framework based on an explanatory model and stochastic processes. They proposed a hierarchic structure of random processes consisting of three components, including the route process, driver dynamics process, and target velocity process. Then, the distributions of the future vehicle dynamics were approximated through sequential Monte Carlo (SMC) simulation (also referred to as particle filtering) by sampling from those random processes.

In a study on trajectory planning for autonomous vehicles, Wang et al. (2019) handled motion prediction of other traffic participants (e.g., vehicles, pedestrians, and cyclists) by adopting Monte Carlo simulation to predict the probability of occupancy of the other traffic participants, to let the autonomous vehicle avoid the area with a high probability of occupancy. They established a physics-based motion model for the traffic participants and leveraged Monte Carlo simulation to estimate the probability distribution of the moving distance of the traffic participants along the longitudinal and transverse directions. The probability distribution results were estimated and stored through offline processing, but they can be retrieved for real-time trajectory planning for autonomous vehicles.

In another vehicle trajectory prediction study, Deo et al. (2018) leveraged an HMM for maneuver recognition by assigning confidence scores for surrounding vehicles' maneuver patterns. Specifically, the HMM-based maneuver recognition module was trained to classify 10 different maneuver classes. Liu et al. (2020) proposed a driving intention prediction method based on HMM for autonomous vehicles. Three HMMs were trained to take the mobility features of the target vehicle and surrounding vehicles as the input and categorize the driving intention into three types, namely changing to the left lane, changing to the right lane, and keeping on the original lane.

Schulz et al. (2018a) proposed a behavior prediction framework to estimate possible routes and maneuvers of vehicles, in which they modeled the development of a traffic situation as a stochastic process consisting of multiple interacting agents and leveraged a DBN model to describe the stochastic process. Later, Schulz et al. (2018b) developed a multi-model UKF-based (MM-UKF) inference method with the DBN proposed in Schulz et al. (2018a) for driver intention estimation and multi-agent trajectory prediction. In Schulz et al. (2018b), one UKF is defined per possible combination of route and maneuver intentions of all agents. Jiang et al. (2022) proposed a probabilistic approach for driver intention estimation and vehicle trajectory prediction based on DBN, where the driver intention, maneuvering behavior, and vehicle dynamics were modeled and integrated probabilistically. Then, a particle filter was introduced in Jiang et al. (2022) to predict the vehicle trajectory and driver's lane-changing intention.

Although these classic methods have reported successes for handling vehicle motion prediction tasks, they may be subject to several disadvantages (Leon and Gavrilescu, 2021). For example, their performance is sensitive to the choice of motion models and initial conditions, which rely heavily on a careful design and selection procedure through prior knowledge or trial-and-error, and thus are often expertise-intensive and subjective.

5.2.2. Deep Learning-based Methods for Vehicle Trajectory Prediction

Kim et al. (2017) developed a framework consisting of LSTM and occupancy grid mapping to predict the future trajectories of vehicles using coordinate data from fused sensor measurements. Ma et al. (2019) proposed an LSTM-based algorithm for trajectory prediction for heterogeneous traffic agents, including

vehicles, pedestrians, and bicyclists. The proposed algorithm by Ma et al. (2019) consists of two components: an instance layer to capture the instances' movements and interactions; and a category layer to learn the behavior similarities of the instances belonging to the same category. Choi et al. (2021) proposed a deep learning-based method using the Random Forest (RF) algorithm for lane change prediction and an LSTM encoder-decoder architecture for vehicle trajectory prediction. In Choi et al. (2021), information from Vehicle-to-Vehicle (V2V) communication, LiDAR sensor, and camera sensor was fused as the input data of the prediction model. Wang et al. (2021) proposed a fusion network architecture for vehicle trajectory prediction. The fusion network is comprised of three modules, including a feature extractor with Gated Recurrent Units (GRUs) and CNNs to extract information from fused sensor data, an attention model with two attention mechanisms to boost the temporal and spatial salient features during feature extraction, and an LSTM encoder-decoder architecture for trajectory prediction. Buhet et al. (2021) proposed a deep learning architecture to produce a probabilistic representation of vehicle trajectories from image and LiDAR point cloud data. For each vehicle, the method (Buhet et al., 2021) outputs a multivariate Gaussian mixture for a fixed number of possible trajectories, taking a concatenation of multi-modal features as input, which include the LSTM encoding on past vehicles' trajectories, Bird-Eye-View (BEV) encoding on LiDAR point clouds, and image encoding. Besides, some LSTM-based studies on vehicle trajectory prediction using other modalities (e.g., image, radar) instead of LiDAR data have also been reported in the literature (Deo and Trivedi, 2018, Dai et al., 2019, Xiao et al., 2020, Wang et al., 2020b, Chandra et al., 2020, Mo et al., 2020).

In these LSTM-based studies, a wide range of hyperparameter values were selected. One of the challenges facing these methods is that there is no well-established strategy in the literature to determine which LSTM architecture is best suited to a particular task or scenario, and selecting proper hyperparameters is often an empirical process (Leon and Gavrilescu, 2021). For example, in Kim et al. (2017) and Lin et al. (2021), the hyperparameters related to the LSTM architectures were tuned through grid search. Overall, very few studies have explored designing a systematic approach to search for the optimal hyperparameter configuration to enhance the performance of deep learning models for vehicle trajectory prediction tasks.

5.3. Methodology

This section describes the proposed LSTM-based deep learning framework with Bayesian optimization for vehicle trajectory prediction. Firstly, the research scope and assumptions of this study are clearly stated. Then, the concept and formulation of Bayesian optimization are described in detail. Subsequently, the LSTM formulation, several candidates of the proposed LSTM architectures, and the hyperparameters to be updated through optimization are introduced. Lastly, the proposed objective function for hyperparameter tuning is formulated.

5.3.1. Research Scope and Assumptions

The proposed methodology leverages Bayesian optimization to explore the joint optimal configurations on architecture- and training-related hyperparameters for LSTM-based vehicle trajectory prediction. In this study, the processes to extract information from low-level features (i.e., raw LiDAR data captured from road intersections), including background filtering, object clustering, vehicle identification, and data association, are handled by the approach described in Zhao et al. (2019b). Meanwhile, the high-level features, including the coordinate and speed data of each vehicle trajectory for network training

and testing, are considered to be given within this work. Thus, the scope of this study is within the task of vehicle trajectory prediction through deep learning, barring issues such as data occlusion during object detection and data association. It is worth noting that similar assumptions have been adopted in many other related research and successful applications (Asvadi et al., 2016, Kim et al., 2017, Schulz et al., 2018a, Ma et al., 2019, Chandra et al., 2020, Choi et al., 2021).

5.3.2. Bayesian Optimization

Instead of applying a brute-force method such as grid search to thoroughly explore the high-dimensional input space of the hyperparameters, which is extremely time-consuming and compute-intensive, this study chooses to leverage Bayesian optimization (Brochu et al., 2010, Snoek et al., 2012, Shahriari et al., 2015, Frazier, 2018) to determine the optimal LSTM network architecture as well as the corresponding training configuration.

Bayesian optimization is a powerful and efficient approach to finding the global optimum of black-box derivative-free functions that are expensive to evaluate. It is applicable in situations where one does not have a closed-form expression (i.e., black box) for the objective function but can obtain observations of this function at sampled values (Brochu et al., 2010). The term "expensive-to-evaluate" refers to a situation where the number of function evaluations is limited because each evaluation typically takes a substantial amount of time (Frazier, 2018). Bayesian optimization is efficient in terms of the number of function evaluations required, largely due to its ability to incorporate all of the information available from previous evaluations to help direct the sampling and to trade off exploration and exploitation of the search space (Brochu et al., 2010, Snoek et al., 2012). Instead of relying on local gradient and Hessian approximations, Bayesian optimization builds a surrogate for the objective function. It quantifies the uncertainty in the surrogate using a Bayesian statistical model (e.g., Gaussian Process regression (Rasmussen and Williams, 2006)) and then uses an acquisition function defined from the surrogate to determine the next sample point (Frazier, 2018). The ability and versatility of Bayesian optimization to optimize expensive functions have made it well suited for machine learning- and deep learning-based applications in interdisciplinary domains, such as tuning hyperparameters for deep neural networks (Snoek et al., 2012).

Let $\mathbf{D}_{1:t} = \{v_{1:t}, \mathbf{f}_{1:t}\}$ be the previous observations of independent and identically distributed examples from the sample space of the objective function. Here, the following compact notation is used for functions sampled at collections of input points: $v_{1:t}$ indicate the sample points $v_1, v_2, \cdots, v_t$, each of which represents a set of hyperparameters; $\mathbf{f}_{1:t} = [f(v_1), f(v_2), \cdots, f(v_t)]$ are the observed objective function values. As a principled design approach in Bayesian optimization, the Gaussian Process (Rasmussen and Williams, 2006) is employed as a surrogate for the objective function, as expressed in Equation (5-1). As extensively discussed in the literature (Rasmussen and Williams, 2006, Brochu et al., 2010, Snoek et al., 2012, Lei et al., 2021), Gaussian Process is a typical choice of the probability surrogate model used to fit the original objective function. It is a powerful stochastic modeling technique which distinguishes itself from others by its mathematical explicitness, computational flexibility, and tractability. Therefore, Gaussian Process has proven to be useful and well-suited to common optimization tasks (Mockus, 1994, Brochu et al., 2010).

$$f(v_i) = g(v_i) + \varepsilon_i, \quad i = 1, 2, \cdots, m \tag{5-1}$$

where $g(v)$ is a Gaussian Process prior, defined in Equation (5-2); $\varepsilon_i \sim \mathcal{N}\left(0, \sigma_{\text{noise}}^2\right)$ are independent and identically distributed "noise" variables with independent normal distributions.

$$g(v) \sim \mathcal{GP}\left(m(v), k(v, v')\right) \tag{5-2}$$

where $\mathcal{GP}(\cdot)$ denotes a Gaussian Process. A Gaussian Process is analogous to a function. But, instead of returning a scalar $f(\tilde{v})$ for an arbitrary $\tilde{v}$, it returns the mean and variance of a normal distribution over the possible values of $f(\tilde{v})$ at $\tilde{v}$. Here, we assume a Gaussian Process prior over $g(v)$; $m(v)$ and $k(v, v')$ are the mean and covariance functions, respectively. Without loss of generality, we let the prior mean be the zero function $m(v) = 0$; meanwhile, the automatic relevance determination (ARD) Matérn 5/2 kernel (Snoek et al., 2012) is adopted as the covariance function $k(v, v')$. More technical details regarding this kernel and the choice of covariance functions can be found in (Snoek et al., 2012).

Now, let $\{v_{t+1}, f_{t+1}\}$ be the new observation drawn from the same sample space, where v_{t+1} is the next sample point and $f_{t+1} = g_{t+1} + \varepsilon_{t+1}$. Based on the properties of Gaussian Processes, $\mathbf{g}_{1:t}$ and g_{t+1} follow a joint multivariate Gaussian distribution, expressed as:

$$\begin{bmatrix} \mathbf{g}_{1:t} \\ g_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(v_{t+1}, v_{t+1}) \end{bmatrix}\right) \tag{5-3}$$

where $\mathbf{g}_{1:t} = [g(v_1), g(v_2), \cdots, g(v_t)]$; the formulation of the covariance matrix $\mathbf{K}$ is provided in Equation (5-4); $\mathbf{k}$ is expressed as Equation (2-1).

$$\mathbf{K} = \begin{bmatrix} k(v_1, v_1) & \cdots & k(v_1, v_t) \\ \vdots & \ddots & \vdots \\ k(v_t, v_1) & \cdots & k(v_t, v_t) \end{bmatrix} \tag{5-4}$$

$$\mathbf{k} = [k(v_{t+1}, v_1), k(v_{t+1}, v_2), \cdots, k(v_{t+1}, v_t)\ ]^T \tag{5-5}$$

Similarly, from the assumption on the noise variables, the following relationship can be established:

$$\begin{bmatrix} \boldsymbol{\varepsilon}_{1:t} \\ \varepsilon_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \sigma_{\text{noise}}^2 I\right) \tag{5-6}$$

where $\boldsymbol{\varepsilon}_{1:t} = [\varepsilon(v_1), \varepsilon(v_2), \cdots, \varepsilon(v_t)]$; the noise variables ε are defined in Equation (5-1).

The sum of independent Gaussian random processes is also Gaussian. Thus, $\mathbf{f}_{1:t}$ and f_{t+1} are jointly Gaussian:

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{1:t} \\ g_{t+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon}_{1:t} \\ \varepsilon_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(v_{t+1}, v_{t+1}) \end{bmatrix} + \sigma_{\text{noise}}^2 I\right) \tag{5-7}$$

From Equation (5-7), the posterior predictive distribution of f_{t+1} [Equation (5-8)] can be characterized by the predictive mean [Equation (5-9)] and predictive variance [Equation (5-10)], using Bayes' rule (Rasmussen and Williams, 2006).

$$P(f_{t+1} | \mathbf{D}_{1:t}, v_{t+1}) = \mathcal{N}\left(\mu(v_{t+1}), \sigma^2(v_{t+1}) + \sigma_{\text{noise}}^2\right) \tag{5-8}$$

where the predictive mean and variance are given as:

$$\mu(v_{t+1}) = \mathbf{k}^T \left(\mathbf{K} + \sigma_{\text{noise}}^2 I\right)^{-1} \mathbf{f}_{1:t} \tag{5-9}$$

$$\sigma^2(v_{t+1}) = k(v_{t+1}, v_{t+1}) - \mathbf{k}^T \left(\mathbf{K} + \sigma_{\text{noise}}^2 I\right)^{-1} \mathbf{k} \tag{5-10}$$

To guide the search for the optimum, Bayesian optimization locates the next point to evaluate by maximizing the acquisition function (denoted as α), which is constructed from the posterior predictive distribution. Under the Gaussian Process prior, the acquisition function depends on the posterior predictive distribution sorely through its predictive mean and predictive variance (Snoek et al., 2012). In this study, the Expected Improvement (EI) over the best-expected value (Brochu et al., 2010, Frazier, 2018) is utilized as the acquisition function:

$$\alpha_{EI}(v_{t+1}) = \sigma(v_{t+1})\big(\gamma_{t+1}\Phi(\gamma_{t+1}) + \phi(\gamma_{t+1})\big) \tag{5-11}$$

where γ_{t+1} is defined by Equation (5-12); Φ and ϕ are the cumulative distribution function (CDF) and probability distribution function (PDF) of the standard normal distribution, respectively.

$$\gamma_{t+1} = \frac{\mu_{\text{best}} - \mu(v_{t+1})}{\sigma(v_{t+1})} \tag{5-12}$$

$$\mu_{\text{best}} = \max_{v_i \in v_{1:t}} \mu(v_i) \tag{5-13}$$

Thus, the task of finding the next query point to evaluate can now be accomplished via a proxy optimization to maximize the acquisition function, expressed as Equation (5-14):

$$v_{t+1} = \text{argmax}_{v_{t+1}} \alpha_{EI}(v_{t+1}) \tag{5-14}$$

Unlike the original black-box objective function, $\alpha_{EI}(\cdot)$ has a closed-form expression and can be cheaply sampled (Brochu et al., 2010).

5.3.3. Long Short-Term Memory (LSTM)

In the literature, various studies have successfully applied LSTM for motion prediction tasks (Leon and Gavrilescu, 2021). Proposed by Hochreiter and Schmidhuber (1997), LSTM is a variant of the vanilla RNN designed to address the "vanishing or exploding gradient" problem from backpropagation when processing long-sequence data. As discussed in Kawakami (2008), LSTM has several key advantages over traditional methods (e.g., HMMs): 1) the constant error backpropagation within memory cells results in LSTM's ability to bridge very long time lags; 2) For long time lag problems, LSTM can handle noise, distributed representations, and continuous values; besides, it does not require a priori choice of a finite number of states; and 3) LSTM is very efficient, with an excellent update complexity per time step and weight. Various studies on traffic object recognition and motion prediction have demonstrated that LSTM networks are very suited to handling data with long-term dependencies (Leon and Gavrilescu, 2021).

A common LSTM unit is comprised of a memory cell, an input gate, a forget gate, and an output gate. The memory cell stores the cell state vector, which summarizes the information from the past sequence data. Meanwhile, the selective remember-forget gating mechanism offered by the LSTM unit through the input gate, output gate, and forget gate controls the information flow between the input, output, and cell state, allowing the network to update the cell state given new data, as illustrated in Figure 5-1. The formulations of LSTM are provided in Equations (5-15) through (5-20). According to the equations and Figure 5-1, at time step t, the LSTM unit takes the cell state from the previous step (i.e., c_{t-1}), the hidden state vector from the previous step (i.e., h_{t-1}), and the input vector from the current step (i.e., x_t) to generate the new cell state (i.e., c_t) and hidden state vector (i.e., h_t), which are to be used

iteratively in the next time step. Interested readers can refer to (Hochreiter and Schmidhuber, 1997, Gers et al., 2000, Gers et al., 2002, Sherstinsky, 2020) for more technical details on LSTM.

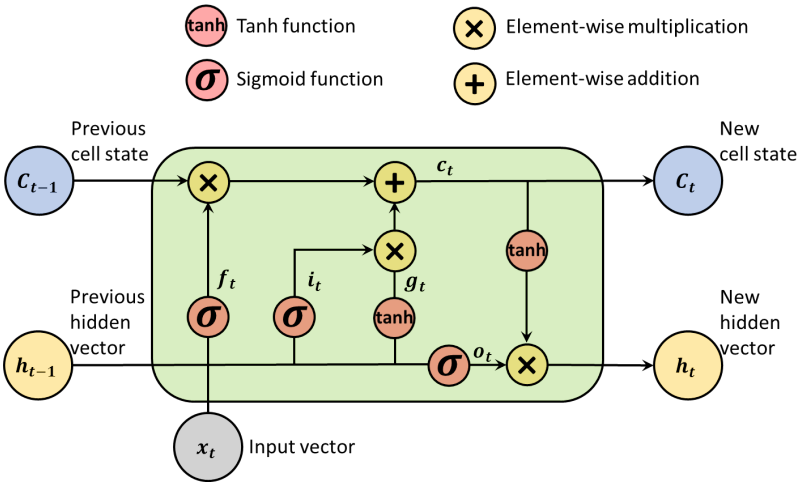


Figure 5-1. Schematic of an LSTM unit.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{5-15}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{5-16}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{5-17}$$

$$g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5-18}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5-19}$$

$$h_t = o_t \odot \tanh(c_t) \tag{5-20}$$

where x_t, c_t, and h_t denote the input vector, cell state vector, and hidden state vector at time step t, respectively; W and b denote the learnable weights and biases, respectively; $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid function, and tanh function, respectively; $\odot$ denotes element-wise multiplication.

5.4. Proposed LSTM network candidates for vehicle trajectory prediction

To explore the impact of different architecture layouts on the prediction performance, four different types of LSTM network architectures with different layout types are proposed as candidates, namely the linear architecture, densely connected architecture, multi-branch architecture, and feature pyramid architecture, as illustrated in Figure 5-2, Figure 5-3, Figure 5-4, and Figure 5-5, respectively. By creating variations in the layout types, the proposed LSTM networks explore the efficacy of different architecture layouts in sequence data processing and feature extraction. The proposed LSTM network structures are novel designs in the sense that their layer depth and number of hidden units in each LSTM layer are updated during optimization, and the optimal "layout type" of the proposed LSTM network is determined through Bayesian optimization, too.

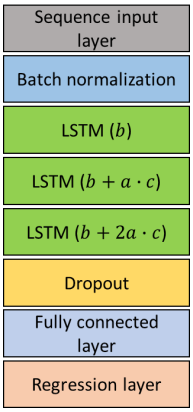


Figure 5-2. Proposed LSTM network candidate: linear architecture.

The first network candidate, a linear architecture (see Figure 5-2), represents a simple yet straightforward connectivity pattern where multiple LSTM layers are sequentially connected in a feed-forward manner for feature extraction. Note that the factors a, b, and c in each LSTM layer, as shown in Figure 5-2, Figure 5-3, Figure 5-4, and Figure 5-5, are designed to collaboratively determine the number of hidden units in each LSTM layer, which also control the dimension of the hidden state output (i.e., h_t) in Equation (5-20). In addition to the LSTM layer and fully connected layer (Goodfellow et al., 2016) for feature extraction, some auxiliary layers, including batch normalization (Ioffe and Szegedy, 2015, Santurkar et al., 2018), and dropout (Srivastava et al., 2014), are employed to regularize the input data and avoid overfitting. In the proposed linear LSTM architecture, the layer depth is defined as the number of LSTM layers, which is one of the hyperparameters participating in the optimization process. In this study, a mathematical relationship between the layer depth and the number of hidden units in an LSTM layer can be characterized as follows:

$$n_i = b + (i - 1) \cdot a \cdot c \tag{5-21}$$

where n_i denotes the number of hidden units in the LSTM layer at layer i; i denotes the layer depth; a, b, and c are the factors defined in Table 5-1.

The second network candidate proposed in this study is a densely connected architecture, as illustrated in Figure 5-3. This type of architecture is the same as a linear architecture except that each LSTM layer in a densely connected architecture receives information from all preceding LSTM layers through an element-wise addition operation. Such a connectivity pattern is defined in (Huang et al., 2017) as a "dense connection". Compared to a linear connectively pattern, the "dense connection" encourages feature reuse from all preceding layers with potentially fewer layers or a smaller number of hidden units, which alleviates the issue of "vanishing gradients" by strengthening backpropagation (Huang et al., 2017). In the proposed densely connected LSTM architecture, the layer depth equals the number of LSTM layers.

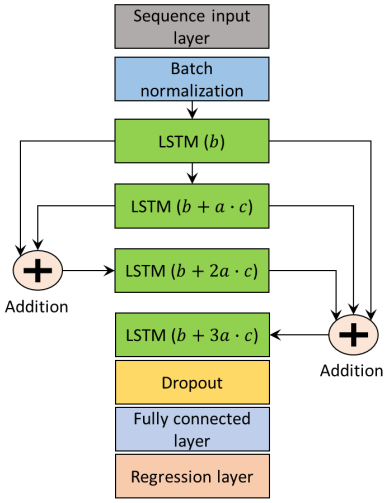


Figure 5-3. Proposed LSTM network candidate: densely connected architecture.

Illustrated in Figure 5-4, the third network candidate is called a multi-branch architecture, where multiple branches of feature extraction modules with different number of LSTM layers (i.e., different layer depths) are employed. Compared to a linear feed-forward architecture, the proposed multi-branch architecture utilizes multiple feature extraction modules (i.e., branches) arranged parallelly to adapt to the input data of varying complexities. That is, when the input data embody a simple feature pattern, the shallow branch in the proposed multi-branch architecture will be activated during the learning process, thus avoiding data overfitting; as the complexity in the training data increases, the deeper branch then comes into action, by applying multiple LSTM layers for more sophisticated feature extraction. For the proposed multi-branch architecture, layer depth is the number of LSTM layers in the longest/deepest branch. It is also worth noting that the number of branches in the proposed multi-branch architecture is equal to the hyperparameter "layer depth", where the number of LSTM layers in each branch gradually reduces to represent different complexities of the hierarchical feature extraction modules, as depicted in Figure 5-4.

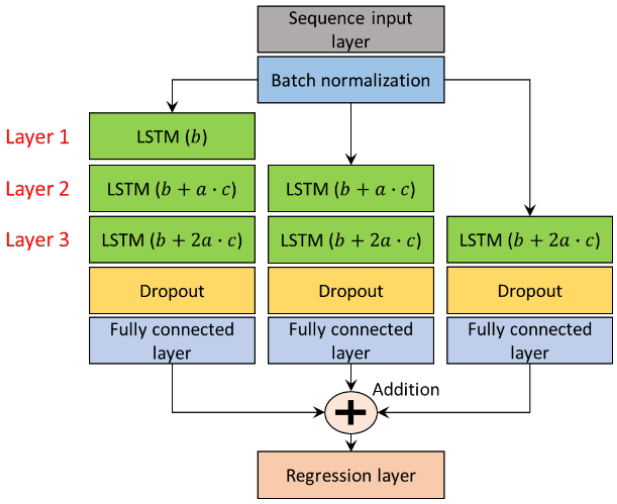


Figure 5-4. Proposed LSTM network candidate: multi-branch architecture.

The fourth and last network candidate is the feature pyramid architecture, as shown in Figure 5-5. The term “feature pyramid” refers to the fact that the proposed architecture adopts a sophisticated connectivity pattern of multi-scale feature extraction, where the hierarchical features are extracted at different stages of feature extraction and further inherited by the subsequent LSTM layers. Judging from the connectivity pattern for feature exploitation, the proposed feature pyramid architecture can be considered a hybrid of the proposed linear, densely connected, and multi-branch architectures. Note that the layer depth is defined as the number of LSTM layers in the longest branch, as illustrated in Figure 5-5.

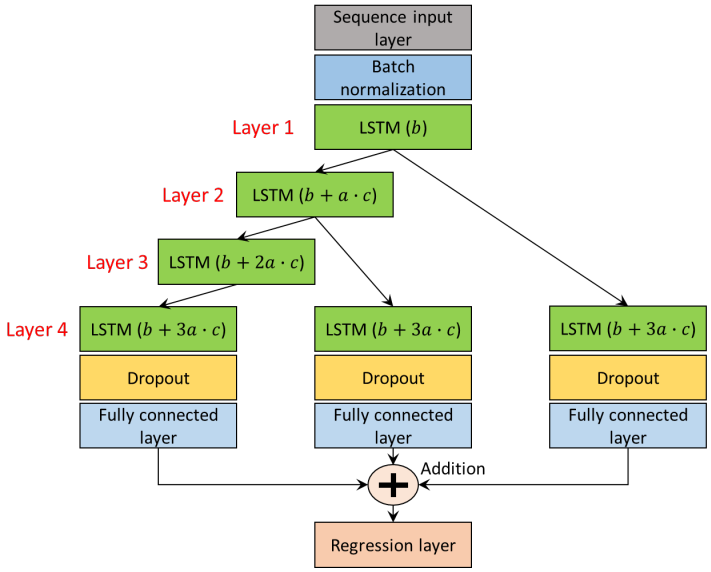


Figure 5-5. Proposed LSTM network candidate: feature pyramid architecture.

The proposed LSTM networks utilize the x and y coordinate and speed data of a single vehicle trajectory from the past 20 frames (i.e., equal to 2 seconds) as the input and then predict the x and y coordinates at the future timestep. In another word, the task of the proposed LSTM networks is to perform sequence-to-one regression. Note that some other research, such as (Chang et al., 2019, Buhet et al., 2021, Choi et al., 2021), also adopted 2 seconds as the observation time on the past trajectories for analysis. Besides, the proposed method assumes the initial condition (i.e., the first 20 frames of the vehicle trajectory) is given.

5.5. Hyperparameters

In constructing the proposed LSTM network candidates, 7 hyperparameters are considered the most critical factors that govern the network performance, as defined in Table 5-1. As explained previously, the first hyperparameter to be updated is the layout type of the proposed LSTM candidates, which consists of four options: “linear”, “densely connected”, “multi-branch”, and “feature pyramid”. The second hyperparameter is the layer depth. In both the linear and densely connected architectures, their layer depths are equal to the number of LSTM layers; in both the multi-branch and feature pyramid architectures, their layer depths are equal to the number of LSTM layers in the longest/deepest branch, as illustrated in Figure 5-4 (layer depth = 3) and Figure 5-5 (layer depth = 4). The third, fourth, and fifth

hyperparameters are designed to collaboratively determine the number of hidden units in the LSTM layers. Specifically, factor b used in each LSTM layer defines a base value for the number of hidden units, and factor c further assigns a range of variation in the base value. Factor a is the "variation pattern" whose data range is [-1, 0, 1]. When $a = 1$, the network embodies an "ascending" pattern in the number of hidden units used in the LSTM layers, meaning that when the layer depth increases, the number of hidden units in the LSTM layer increases accordingly. Similarly, $a = -1$ means a "descending" pattern in the number of hidden units in the LSTM layers; $a = 0$ corresponds to no variation in the number of hidden units. It is worth noting that the hyperparameters described above are all related to the network architecture.

Table 5-1. Hyperparameters to be updated in the optimization process.

Name	**Related to**	**Type**	**Range**
Layout type	Architecture	Categorical	Linear, densely connected, multi-branch, feature pyramid
Layer depth	Architecture	Integer	3, 4, 5
Factor a	Architecture	Integer	-1, 0, 1
Factor b	Architecture	Integer	[900, 1600]
Factor c	Architecture	Integer	[100, 200]
Learning rate	Training	Real	[0.001, 0.05]

The sixth and seventh hyperparameters under consideration are the learning rate and momentum (Goodfellow et al., 2016), respectively, related to the training configuration. The learning rate is one of the most critical factors in network training because it controls convergence (Goodfellow et al., 2016). A large learning rate may cause undesirable divergent behavior, reducing the generalization accuracy. Conversely, a small learning rate may cause the network to progress slowly in updating the weights, ending with a slower convergence. Thus, properly selecting the learning rate is crucial to promoting network performance during training. The momentum (Goodfellow et al., 2016) is a hyperparameter employed by the Stochastic Gradient Descent with momentum algorithm (Bengio, 2012, Bottou, 2012) for gradient-based supervised learning. The name momentum derives from a physical analogy, in which the negative gradient is a force moving a particle through parameter space, according to Newton's laws of motion (Goodfellow et al., 2016). During network training, the momentum controls the convergence rate by introducing inertia (i.e., information from the previous step) in a direction of the search space, thus allowing a faster convergence.

The other hyperparameters, such as the dropout factor, are considered less important and not updated through Bayesian optimization but rather determined based on prior knowledge and kept as constant. Such a strategy helps avoid a very high-dimensional input space, especially considering that Bayesian optimization is best suited for optimization of fewer than 20 dimensions (Frazier, 2018).

5.6. Objective Function

The goal of the proposed methodology is to minimize the objective function as expressed in Equation (5-22), which is the sum of the Root-Mean-Square-Error (RMSE) between the ground truth and

predicted *x* and *y* coordinates of the vehicle trajectories, evaluated on the validation dataset. Note that the hyperparameter vector v (defined in Table 5-1) serves as the direct input to the objective function; the intermediate input, $x_{i,v}^{(p)}$ and $y_{i,v}^{(p)}$, are the coordinates of the vehicle trajectories predicted by the LSTM network. The objective function is non-convex and does not have a closed-form expression.

$$f(v) = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i^{(g)} - x_{i,v}^{(p)}\right)^2}{N}} + \sqrt{\frac{\sum_{i=1}^{N}(y_i^{(g)} - y_{i,v}^{(p)})^2}{N}} \tag{5-22}$$

where x and y denote the coordinates along the horizontal and vertical directions, respectively; the superscripts (p) and (g) refer to the predicted and ground truth data, respectively; the subscript v in $x_{i,v}^{(p)}$ and $y_{i,v}^{(p)}$ means these two variables are dependent on the hyperparameter vector v (defined in Table 5-1); N denotes the number of observations in the validation dataset. An observation refers to a trajectory of a single vehicle object. Because this study is focused on sequence-to-one regression, each observation contains 20 frames of predictor data (i.e., network input) and 1 frame of response data (i.e., network output).

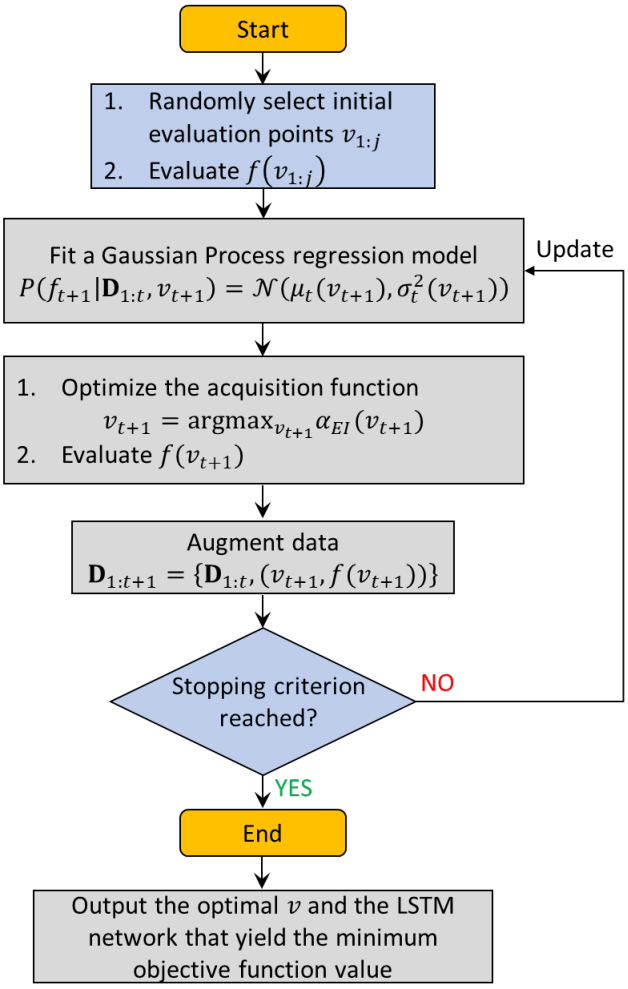


Figure 5-6. Flow chart of the proposed Bayesian optimization framework.

5.7. Flow Chart of the Proposed Methodology

To summarize, a flow chart is provided in Figure 5-6 to delineate the procedure of the proposed Bayesian optimization framework for hyperparameter tuning in LSTM-based vehicle trajectory prediction. As illustrated by Figure 5-6, the algorithm selects several initial evaluation points by randomly drawing samples from a uniform distribution for each hyperparameter. Subsequently, by incorporating the previous evaluations, Bayesian optimization updates the predictive posterior distribution by fitting a Gaussian Process regression model, which is further employed to construct the acquisition function. A proxy optimization determines the next query point by maximizing the acquisition function. Then, the augmented observation data are utilized to update the Gaussian Process regression model. Once the stopping criterion is reached, the optimal hyperparameter set and the corresponding network architecture can be obtained by locating the observed minimum objective function value throughout the optimization history.

5.8. Experimental Study and Results

This section presents an experimental study to validate the proposed methodology on real-world data. Firstly, in sections 5.8.1 and 5.8.2, data acquisition, processing, and dataset generation details are provided. Then, in section 5.8.3, the experimental setup process is introduced. Section 5.8.4 describes the performance metrics used for quantitative performance evaluation. Finally, section 5.8.5 presents the experimental results and discussions.

5.8.1. Data Acquisition and Processing

Raw LiDAR data were collected from a road intersection in Reno, Nevada, U.S., under complex urban traffic scenarios. As shown in Figure 5-7, a LiDAR sensor (i.e., Velodyne VLP-32C) was installed on a traffic light pole at a corner of the road intersection, 10 feet above the ground. Data acquisition lasted one hour at a rotational frequency of 10 Hz, obtaining 3D point cloud data of the surroundings.

A series of data processing techniques, including background filtering, object clustering, object classification, and data association, etc., are applied to the acquired raw LiDAR data, following the procedures described in Zhao et al. (2019b). As mentioned in section 5.3.1, “Research scope and assumptions”, the high-level features, including the coordinate and speed data of each vehicle trajectory, are considered to be given within this work. Thus, based upon Zhao et al. (2019b), vehicle trajectories are extracted from real-world LiDAR data for network training and testing purposes. The obtained vehicle trajectories are further examined by experts and trained personnel for quality control purposes.

Figure 5-7. Sensor instrumentation.

5.8.2. Dataset Generation

The vehicle trajectory data are cropped into multiple observations through a sliding window. Here, each "observation" refers to a trajectory of a single vehicle object containing 21 frames (or 2.1 seconds). Because this study is focused on sequence-to-one regression, the proposed LSTM networks are designed to take the coordinate and speed information from the first 20 frames of each observation (i.e., predictor) as the input, and predict the future coordinate of the vehicle object at the 21st frame (i.e., response). It is worth noting that other research (Chang et al., 2019, Buhet et al., 2021, Choi et al., 2021) also adopted 2 seconds as the observation time on the past trajectories for analysis. In total, 136,212 observations are obtained as the training dataset, and 15,135 observations are used for validation and testing, respectively.

5.8.3. Experimental Setup

(1) Computing hardware and software

The experiments were performed using MATLAB 2022a (MATHWORKS, 2022) with its deep learning toolbox under the following hardware specifications: CPU: AMD Ryzen 9 5900HX; GPU: NVIDIA GeForce RTX 3080 with 16 GB memory.

(2) Hyperparameter configuration

Regarding the hyperparameters related to the LSTM network architecture, their values are provided in the parentheses as follows: dropout factor (0.5) in each dropout layer, scale factor (1), and shift factor (0) in each batch normalization layer. The other hyperparameters, including the layout type, layer depth, and the number of hidden units, are optimized through Bayesian optimization.

The mini-batch Stochastic Gradient Descent with momentum algorithm (Bengio, 2012, Bottou, 2012) is adopted for training. The values for several hyperparameters related to this algorithm are determined as follows, based on empirical knowledge: mini-batch size (500), number of epochs (10), weight decay factor (0.0003). As explained previously, the optimal learning rate and momentum are determined through Bayesian optimization.

(3) Bayesian optimization configuration

To initialize the Bayesian statistical model, 4 sample points (i.e., $v_{1:4}$) are randomly drawn from the input space by assuming a uniform distribution for each hyperparameter (see Table 5-1), and the corresponding objective function values are calculated to serve as the prior observations. The stopping criterion for Bayesian optimization is set as 50 iterations.

5.8.4. Performance Metrics

The experimental study adopts the Root-Mean-Square-Error (RMSE) and the Average Displacement Error (ADE) for quantitative performance evaluation. The RMSE quantitatively measures the discrepancy between the predicted and ground truth future x and y coordinate data over the entire testing dataset. For concision, the formulation of RMSE is not provided herein.

As a popular metric adopted by many researchers (Chandra et al., 2020, Wang et al., 2021, Chen et al., 2021) for vehicle trajectory prediction, ADE is defined as the average distance error between the predicted and ground truth future position of the vehicle objects assessed over the entire testing dataset. The formulation of ADE is provided in Equation (5-23). The ADE metric provides a comprehensive evaluation by calculating the distance error. At the same time, the RMSE is used to assess the prediction accuracy along a single direction (i.e., x or y).

$$\mathrm{ADE} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{L}\sum_{t=t+1}^{t=t+L}\left(x_i^{(p)} - x_i^{(g)}\right)^2 + \left(y_i^{(p)} - y_i^{(g)}\right)^2} \tag{5-23}$$

where N denotes the number of observations; L denotes the sequence length of the future trajectory, equal to 1 (i.e., sequence-to-one regression); the superscripts (p) and (g) refer to the predicted and ground truth data, respectively; x and y denote the coordinates along the horizontal and vertical directions, respectively.

5.8.5. Experimental Results

In this section, firstly, results and discussions on Bayesian optimization are presented; then, a comparative study is performed to compare the performance of the optimal LSTM network obtained through the proposed methodology vs. a handcrafted LSTM network; subsequently, to further investigate the robustness of the proposed methodology, a sensitivity analysis is performed by creating different levels of perturbation around the optimal hyperparameter values.

- *Bayesian optimization results*

The optimization history is illustrated in Figure 5-8. In this figure, the horizontal axis refers to the iteration index up to 50; the vertical axis refers to the minimum objective function value evaluated on the validation dataset. The solid red line in Figure 5-8 represents the minimum objective function value observed from function evaluations. The blue dashed line represents the minimum objective function value estimated from the Gaussian Process regression model.

From Figure 5-8, it can be seen that the estimated minimum objective function value matches closely with the observed minimum objective function value, indicating a good fit of the Gaussian Process regression model in each iteration. At the 49th iteration, the minimum objective function value is

observed [i.e., $f(v_o) = 0.945$], resulting in the optimal hyperparameter configuration. The corresponding optimal values for the hyperparameters under investigation are tabulated in Table 5-2. It is worth noting that the proposed Bayesian optimization framework manages to find the optimal hyperparameters within 50 trials, indicating its high efficiency in terms of the number of trials required. On the other hand, traditional methods for hyperparameter optimization, such as grid search and random search, often need a much higher number of trials to find the optimal. For example, if grid search is employed, even just 3 possible values for each of the 7 hyperparameters would yield a total of 3^7 trials, making the optimization process extremely time-consuming and impractical.

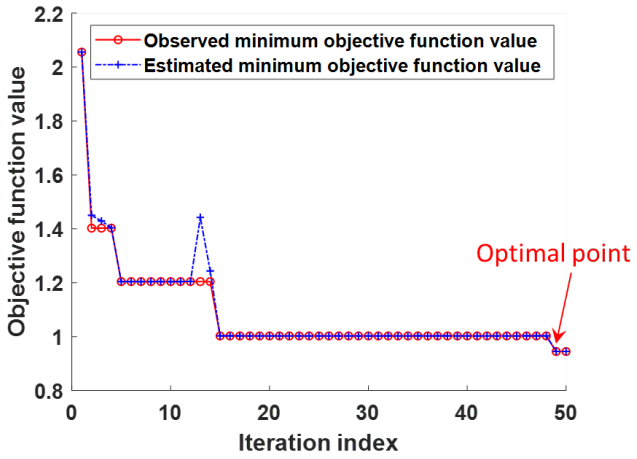


Figure 5-8. Plot of the iteration index vs. minimum objective function value.

Because the input space is a hyperspace containing 7 hyperparameters, it is not possible to visualize the variation pattern of the objective function value with respect to all the hyperparameters through a single 2D or 3D graph. Here, we choose to group the observed objective function values based on different hyperparameter combinations for visualization purposes and plot their marginal distribution to those hyperparameters, as displayed in Figure 5-9, Figure 5-10, and Figure 5-11.

Table 5-2. Comparison of the hyperparameters of the optimal LSTM network through Bayesian optimization vs. the handcrafted LSTM network.

Network	**Optimized through Bayesian optimization**	**Handcrafted**
Layout type	Multi-branch	Multi-branch
Layer depth	3	3
Factor a	-1	1
Factor b	929	1000
Factor c	144	100
Learning rate	0.0235	0.02

To gain insights into the impact of the hyperparameters related to the LSTM network architecture, the number of function evaluations (i.e., z axis) is separated into different groups and counted based on the layout type (i.e., x axis) and layer depth (i.e., y axis), as plotted in Figure 5-9. Note that in Figure 5-9, the height of each rectangular bar is equal to the number of function evaluations. Meanwhile, the color of the bar represents the average value of the objective function evaluations belonging to that bar. It is

clear that Bayesian optimization tends to allocate more computational resources to exploring the search space under the following hyperparameter combination: layer depth = 3 and layout type = “multi-branch”, as guided by the acquisition function. Meanwhile, the corresponding color suggests that such a hyperparameter combination will yield relatively low objective function values, indicating better prediction performance. This observation is in accordance with the acquired optimal values for the layout type and layer depth as tabulated in Table 5-2.

Furthermore, the objective function value is plotted against the total number of learnable parameters in each network, as shown by the scatter plot in Figure 5-10. In this figure, the horizontal axis is the total number of learnable parameters, which is jointly determined by the network layout and the number of hidden units in the LSTM layers [see Equation (5-21)]; the vertical axis refers to the observed objective function value. The results are further grouped using different markers based on their corresponding layout types. It can be observed that the majority of the sample points fall into the interval of [1, $4]\times 10^7$ for the total number of parameters. This range reflects the LSTM networks' desired complexity, which is determined by their adaptation to the training data. Based on the Bayesian optimization result, the number of parameters for the acquired optimal LSTM network is equal to 2.44×10^7.

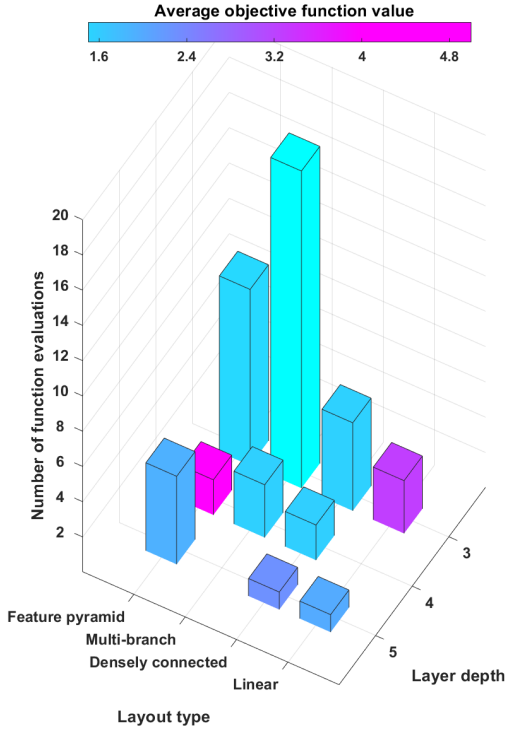


Figure 5-9. Histogram of the number of function evaluations with respect to the layout type and layer depth.

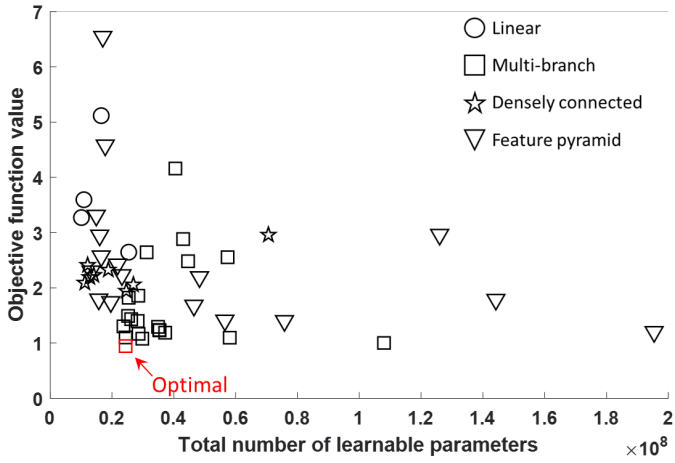


Figure 5-10. Scatter plot of the objective function value with respect to the total number of learnable parameters.

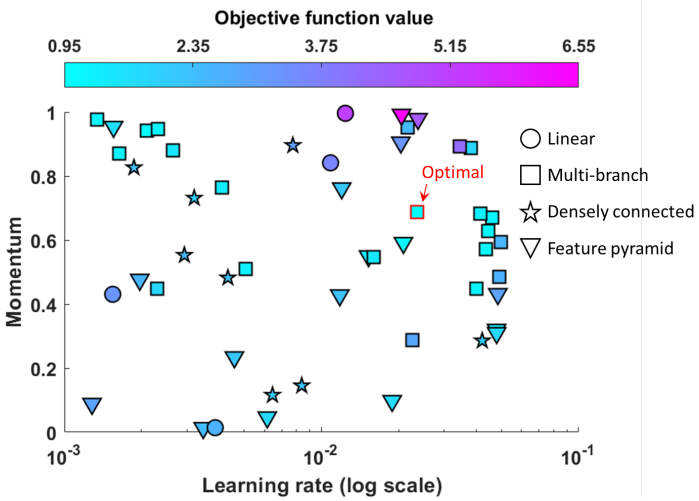


Figure 5-11. Scatter plot of the objective function value with respect to the learning rate and momentum.

In Figure 5-11, the observed objective function values are plotted against the hyperparameters related to the training process, namely the learning rate, and momentum. In this figure, the horizontal axis refers to the learning rate plotted in a log scale, while the vertical axis is the momentum. The objective function values are grouped by their layout types through different markers, and the color of each marker represents the corresponding objective function value. According to Figure 5-11, Bayesian optimization yields relatively low objective function values when the learning rate falls into the range of [0.01, 0.04], and meanwhile, the momentum value is within the range of [0.6, 0.8]. The optimal learning rate and momentum values are 0.0235 and 0.686, respectively.

- *Performance comparison between the optimal LSTM network and the handcrafted LSTM network*

Once the optimal LSTM network is obtained through Bayesian optimization, a handcrafted LSTM network based on prior experience is constructed for cross-comparison purposes. The hyperparameter configurations for the handcrafted LSTM network are also tabulated in Table 5-2. The hyperparameters

other than those mentioned in Table 5-2 are kept the same as the optimal LSTM network to provide a fair basis for cross-comparison. Then, the handcrafted LSTM network is trained and tested on the same datasets.

In Figure 5-12, the RMSE between the ground truth and predicted x and y coordinates of the future trajectories in the testing dataset are illustrated. The blue bars represent the prediction result from the optimal network, where the RMSE values along the x and y directions are 0.529 and 0.416 meters, respectively. For the handcrafted network, the corresponding values are 1.138 and 0.519 meters, respectively. Thus, by adopting the proposed Bayesian optimization framework for hyperparameter tuning, the prediction errors on the vehicle's future x and y coordinates are significantly reduced.

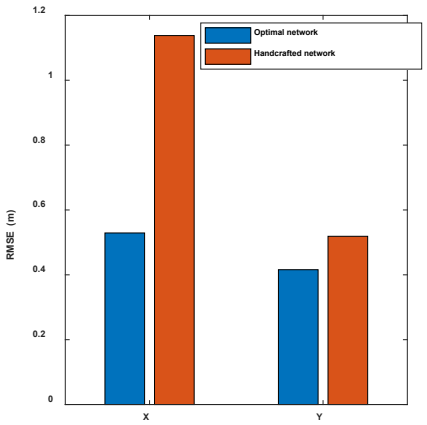


Figure 5-12. RMSE between the ground truth and predicted coordinates of the future trajectory.

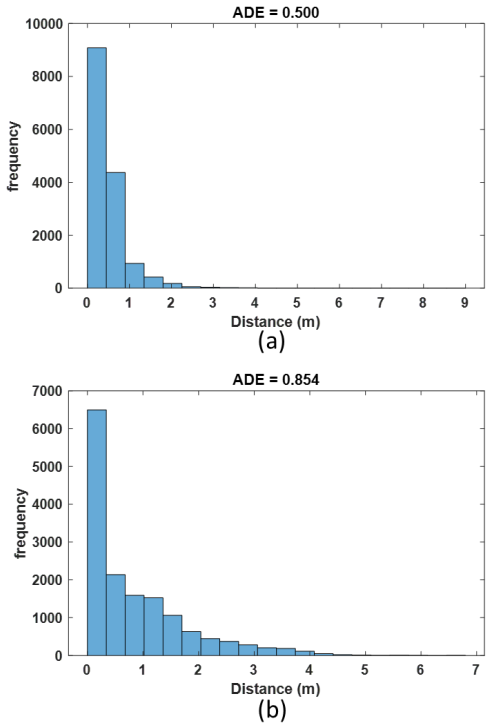


Figure 5-13. Histogram of the displacement error: (a) by the optimal network; and (b) by the handcrafted network.

To further demonstrate the advantage of the optimal LSTM network over the handcrafted network in reducing the prediction errors, the distances between the ground truth and predicted future vehicle position are evaluated on the testing dataset and displayed in histograms in Figure 5-13. Additionally, the ADE metric [see Equation (5-23)] is calculated and displayed on top of each plot in Figure 5-13. As can be observed, the data bins in the histogram generated by the optimal network through Bayesian optimization [Figure 5-13 (a)] are less spread out along the horizontal axis and more concentrated to the left-hand side of the plot than the results shown in Figure 5-13 (b). Furthermore, the ADE value by the optimal LSTM network is 0.5 meters, whereas the corresponding value for the handcrafted network is 0.854 meters, indicating the handcrafted network has a much lower accuracy in trajectory prediction. Judging from the graphical results of the histogram plots and the ADE metric, it is clear that the optimal LSTM network obtained through the proposed Bayesian optimization framework outperforms the handcrafted network with improved accuracy and consistency.

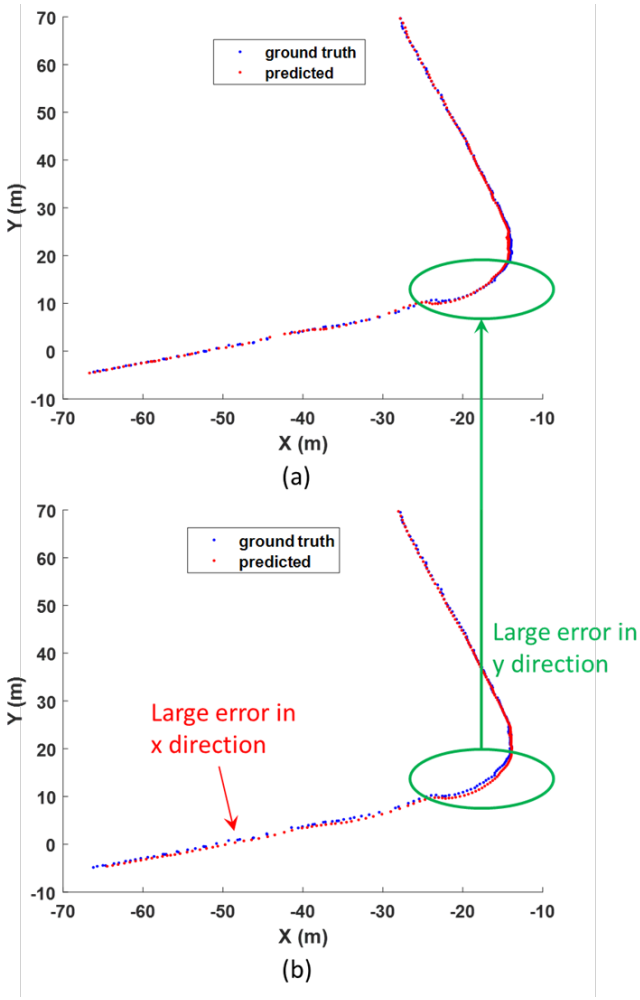


Figure 5-14. Example: prediction result on a long trajectory: (a) by the optimal network; and (b) by the handcrafted network.

Figure 5-14 presents an example of applying the obtained optimal network and the handcrafted network to predict the future trajectories of a long data sequence (i.e., sequence length = 288 frames) for performance demonstration. The prediction process is accomplished iteratively through a sliding

window. In Figure 5-14, the predicted and ground truth trajectories are plotted in red and blue colors. By comparing the prediction results by the optimal network [Figure 5-14 (a)] vs. the handcrafted network [Figure 5-14 (b)], it can be observed that the handcrafted network yields a larger error in both the x direction (indicated by the red arrow) and y direction (indicated by the green circle and arrow) than the optimal network. Figure 5-15 depicts the prediction result at a single frame, extracted from the trajectories in Figure 5-14. In Figure 5-15, the ground truth and predicted vehicle position are marked by green circle and red cross, respectively. The prediction result is further overlaid with the corresponding raw point cloud data. As illustrated by Figure 5-15 (a) and (b), the predicted position by the handcrafted network shows a large lateral deviation from the ground truth position. In contrast, the predicted position by the optimal network matches well with the ground truth position, indicating more accurate performance.

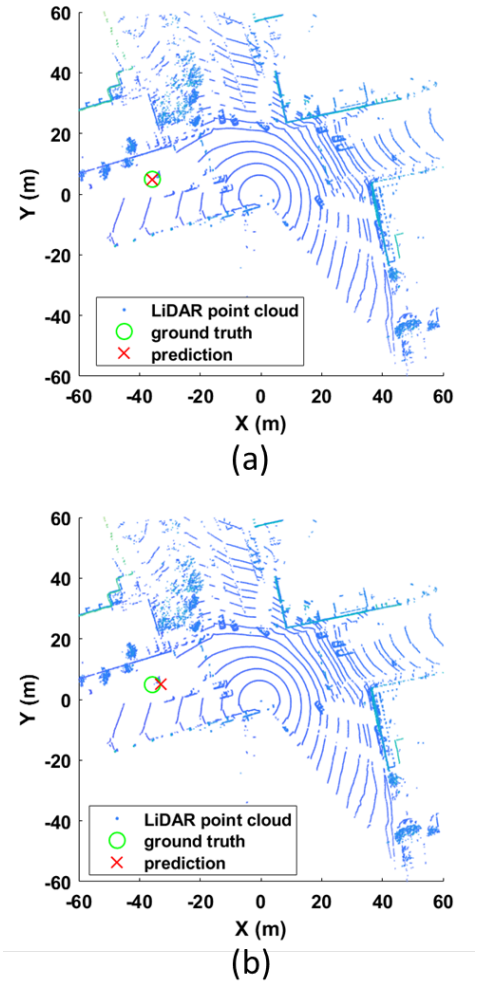


Figure 5-15. Prediction result at a single frame (overlaid with the LiDAR point cloud): (a) by the optimal network; and (b) by the handcrafted network.

- *Sensitivity analysis*

After Bayesian optimization, a sensitivity analysis is performed to measure the robustness of the proposed methodology to small changes in the optimal hyperparameter values. The following three

hyperparameters, including the learning rate, momentum, and the number of hidden units [see Equation (5-21)], are considered in the sensitivity analysis. This study adopts the one-factor-at-a-time (OAT) approach for sensitivity analysis, meaning only one hyperparameter is perturbated around its optimal value while the others remain unchanged. The formulation of the adopted OAT method is provided by Equation (5-24), which calculates the percentage change in the objective function value with respect to the percentage change in the hyperparameter value. Meanwhile, eight different levels of perturbation are designed, as expressed in Equation (5-25). Thus, in total, 24 subcases with different levels of perturbation are trained and tested.

$$\text{sensitivity} = \frac{f(v_p) - f(v_o)}{f(v_o) \cdot |\Delta v|} \tag{5-24}$$

$$\Delta v \in [-20, -15, -10, -5, 5, 10, 15, 20]\ (\%) \tag{5-25}$$

where $f(\cdot)$ is the objective function defined in Equation (5-22); v_o refers to the optimal hyperparameter set obtained through Bayesian optimization, as tabulated in Table 5-2; v_p refers to the hyperparameter set by creating different levels of perturbation around the optimal point; Δv denotes the perturbation level for each hyperparameter, measured by percentage.

Figure 5-16 illustrates the ADE metric calculated on the testing dataset under different perturbation levels. In this figure, the horizontal axis refers to the different perturbation levels, while the vertical axis refers to the corresponding ADE value. Bars with blue, orange, and yellow colors represent the results by creating perturbation in the learning rate, momentum, and the number of hidden units, respectively. The black dashed line indicates the ADE value of 0.5 meters from the optimal hyperparameter setting. From Figure 5-16, the change in the learning rate yields the highest ADE value among the three hyperparameters under investigation, in almost every perturbation level. Meanwhile, it can be observed that a higher perturbation level in the learning rate value leads to a larger negative impact in the prediction performance, as measured by the ADE metric. Similar observations can be made from the scenarios by imposing small changes in the momentum value and the number of hidden units, but the increase in their ADE values is not as significant as those induced by the variation in the learning rate. Furthermore, by creating perturbation in the optimal hyperparameter value, the resulted ADE value increases in all 24 subcases, which in turn shows that the hyperparameter set optimized through the proposed Bayesian optimization framework reflects the optimal configuration.

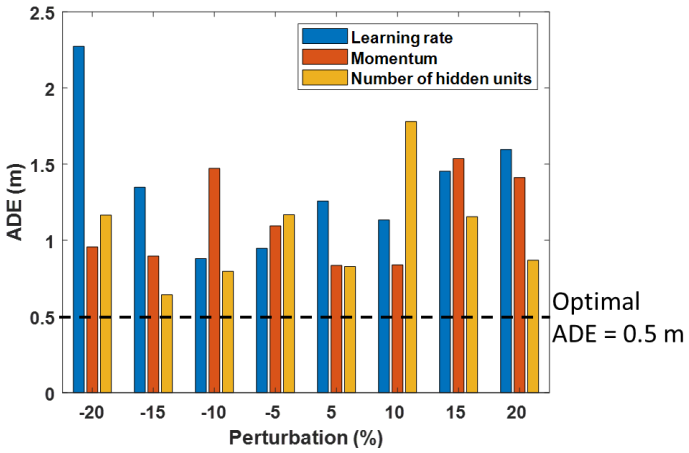


Figure 5-16. ADE at different perturbation levels.

The sensitivity at different perturbation levels is plotted in Figure 5-17, where the horizontal and vertical axes refer to the perturbation level and the corresponding sensitivity value, respectively. Similarly, based on the sensitivity plot in Figure 5-17, the proposed method is most sensitive to the changes in the learning rate value and least sensitive to the number of hidden units at almost every perturbation level. Meanwhile, it can be observed that as the perturbation level increases, the corresponding sensitivity value decreases for all the three hyperparameters, indicating the network performance does not degrade proportionally to the linear change in the optimal hyperparameter values. Specifically, in Figure 5-17, the proposed method shows robust performance at large perturbation levels (e.g., $\pm$20%), where the sensitivity to the change in the hyperparameter values is relatively low.

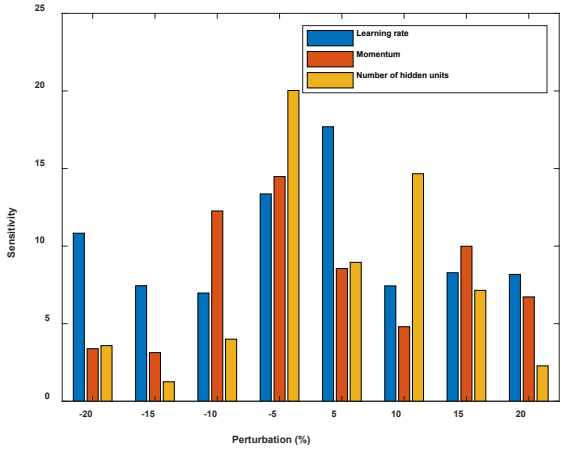


Figure 5-17. Sensitivity of the hyperparameters at different perturbation levels.

5.9. Summary

In this chapter, a LiDAR-based deep learning framework was proposed for vehicle trajectory prediction, which consists of two components: vehicle trajectory prediction through LSTM networks; and hyperparameter tuning through Bayesian optimization. The proposed LSTM networks utilize the high-level features including the coordinate and speed data of vehicle trajectories extracted from roadside LiDAR data to make predictions on their future trajectories. Meanwhile, to guide the search for the joint optimal hyperparameter configuration, Bayesian optimization is incorporated into the proposed framework. The following five hyperparameters related to the LSTM network architecture are tuned through Bayesian optimization: the layer type (i.e., "linear", "densely connected", "multi-branch", and "feature pyramid"), layer depth, and the number of hidden units (jointly determined by the factors a, b, and c). Additionally, two hyperparameters related to the training process, namely the learning rate and momentum, also participate in the optimization process.

In the experimental study, LiDAR data were collected from a road intersection at Reno, Nevada, U.S., under complex urban traffic scenarios. The obtained LiDAR data were further preprocessed to generate datasets for network training and testing. In the experimental study, the optimal LSTM network and its associated training configuration was acquired through Bayesian optimization. By cross comparing the prediction performance of the optimal LSTM network vs. a handcrafted LSTM network which is constructed based on prior knowledge, the results demonstrate that the proposed methodology is capable of determining the joint optimal hyperparameter configuration through optimization and lead to more accurate and consistent performance on vehicle trajectory prediction.

Furthermore, a sensitivity analysis is performed to measure the uncertainty in the network output subject to the uncertainty in the input hyperparameters, by creating different levels of perturbation in the optimal hyperparameter values. The results show that the proposed method is most sensitive to variations in the learning rate and least sensitive to the change in the number of hidden units, and that it shows robust performance when the perturbation level gradually increases.

In the process of LiDAR data acquisition and processing, the present study does not consider the impact of different weather conditions or environmental factors, because the captured LiDAR data are not subject to drastic environmental changes. Another limitation is that the present study does not consider the issue of data occlusion, which could potentially deteriorate the prediction performance of the LSTM networks. Besides, the proposed method currently adopts roadside LiDAR data for analysis. Expanding its applicability to more traffic scenarios and data types (e.g., autonomous driving LiDAR data) would further validate the efficacy of the proposed method for LiDAR-based vehicle trajectory prediction tasks.

Future research efforts need to be devoted to:

- Exploring the impact of varying environmental conditions on the proposed method.
- Considering the issues of data occlusion during object detection, data association, and prediction, and developing LSTM networks or other deep neural network architectures that are robust to data discontinuities caused by data occlusion.
- Expanding the applicability of the proposed method on different traffic scenarios and public datasets.
- Incorporating multi-modal data to further improve the accuracy and robustness of LSTM-based vehicle trajectory prediction.

CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

6.1. Conclusions

Recent technological advancements in computer vision algorithms and data acquisition devices have greatly facilitated research activities towards enhancing traffic sensing for traffic safety performance improvements. Significant research efforts have been devoted to developing and deploying more effective technologies to detect, sense, and monitor traffic dynamics and rapidly identify crashes in Rural, Isolated, Tribal, or Indigenous (RITI) communities.

As a new modality for 3D scene perception, Light Detection and Ranging (LiDAR) data have gained increasing popularity for traffic perception, due to its advantages over conventional RGB data, such as being insensitive to varying lighting conditions. In the past decade, researchers and professionals have extensively explored LiDAR data to promote traffic perception for transportation research and applications, especially in autonomous driving industry. Nevertheless, a series of challenges and research gaps are yet to be fully addressed in LiDAR-based transportation research, such as the disturbance of adverse weather conditions, lack of roadside LiDAR data for deep learning analysis, and roadside LiDAR-based vehicle trajectory prediction.

This project aimed to enhance vehicle sensing for traffic safety and mobility performance improvements using roadside LiDAR sensor data. To fulfill this overarching goal, the following research topics have been investigated through this technical report:

- Develop an effective analytical method to eliminate data outliers and noises from roadside LiDAR data captured under adverse weather conditions.
- Explore the feasibility of leveraging the domain knowledge of deep learning models trained on autonomous driving data for vehicle detection from roadside LiDAR data.
- Propose a novel deep learning model for vehicle trajectory prediction using positional information extracted from roadside LiDAR data and signal timing information.
- Explore the optimal hyperparameter configuration of deep learning models to boost the performance on vehicle trajectory prediction through Bayesian optimization.

Chapter 2 proposed a novel methodology called Dynamic Channel-wise Outlier Removal (DCOR) to de-noise LiDAR data corrupted by snow. The proposed methodology iterates over the LiDAR data based on different laser channels and marks a point of interest in a channel as an outlier if the number of neighboring points in the same channel (within a dynamic search radius) is fewer than a threshold. Unlike the existing LiDAR de-noising methodologies, the proposed methodology processes LiDAR data in a channel-wise manner, to reduce the data dimensionality and decouple the snow effects along the vertical axis; furthermore, upon searching for neighboring points, the proposed methodology adopts a dynamically adjusted search radius which is proportional to the point-to-sensor distance, to account for the varying point density that decreases as the distance to sensor increases. In the experimental study, the proposed DCOR filter is compared against some other methodologies, including DCOR-variant1, ROR, DROR, SOR, and DSOR, regarding their performance on snow removal. Through the cross comparison, it has been demonstrated that the proposed DCOR filter outperforms the state-of-the-art methodologies, including DROR and DSOR, in both accuracy and efficiency, where the average F1 score is 98.3%, 96.4%, and 95.4%, and the execution time per frame is 0.242 secs, 2.221 secs, and 3.652 secs, respectively.

Chapter 3 proposed a methodology based on CNNs, to explore the feasibility and challenges of reusing CNNs trained on autonomous driving datasets for vehicle detection from roadside LiDAR data. The proposed CNN architecture is modified from the established PointPillars object detection network, by adding dense connections between convolutional layers to promote feature fusion and extraction. In Case I of the experimental study, PandaSet, a publicly available large-scale autonomous driving dataset, was adopted for network training and testing. In addition to the proposed CNN, two extensively adopted object detection networks, PointPillars and YOLOv4, are also implemented in the experimental study for cross-comparison. The training and testing results show that the proposed CNN, PointPillars, and YOLOv4 are capable of achieving good detection performance on the autonomous driving dataset; moreover, the proposed CNN outperforms the other two CNNs on the autonomous driving dataset with an improvement of 7.0% and 2.7%, respectively, measured by the F1 score. In Case II, roadside LiDAR data were collected at an intersection in Reno, Nevada, the U.S, for performance evaluation. The trained CNNs from Case I were further utilized to make predictions on the captured roadside LiDAR data. It is shown that the proposed CNN yields higher F1 scores on the roadside LiDAR data than PointPillars and YOLOv4, indicating the best detection performance. Another observation from Case II is that the detection performance of all the three CNNs is slightly deteriorated compared to that in Case I, caused by the impact of different data features and characteristics between the autonomous driving data and roadside LiDAR data.

In Chapter 4, we proposed a deep learning-based methodology utilizing LSTM for pedestrian trajectory prediction and risk assessment. The proposed methodology leverages the trajectory data extracted from roadside LiDAR data and the corresponding signal phasing information to learn temporal dependencies between sequence data. In addition, risk assessment of pedestrian crossing behavior is incorporated into the proposed methodology, by formulating the risk factor as the input and output of the LSTM network architecture. In the experimental study, roadside LiDAR data and SPaT data were collected at MLK and Georgia Ave in Chattanooga, TN, under complex traffic scenarios. Pedestrian trajectory data, SPaT data, and risk factors were generated and pre-processed through a series of procedures, including data normalization, data transformation, and dataset generation. The RMSE values between the predicted and ground truth x and y coordinates calculated from the testing dataset are 0.225 meters and 0.377 meters, respectively, indicating high accuracy in trajectory prediction. Meanwhile, the F1 score value for the risk factor is 99.6%, which shows that the proposed methodology can achieve very accurate risk assessment as well.

Chapter 5 proposed a LiDAR-based deep learning framework for vehicle trajectory prediction, which leverages LSTM networks to predict vehicle trajectories and Bayesian optimization to determine the optimal hyperparameter configuration. The optimization scheme is designed such that both the deep learning model architecture and its associated training scheme are updated through Bayesian optimization. In the experimental study, a vehicle trajectory dataset extracted from roadside LiDAR data was utilized for network training and testing. The optimal LSTM network obtained through Bayesian optimization was compared against a benchmark LSTM network with handpicked hyperparameters. In the experimental study, LiDAR data were collected from a road intersection at Reno, Nevada, U.S., under complex urban traffic scenarios. The obtained LiDAR data were further preprocessed to generate datasets for network training and testing. In the experimental study, the optimal LSTM network and its associated training configuration was acquired through Bayesian optimization. By cross comparing the prediction performance of the optimal LSTM network vs. a handcrafted LSTM network which is

constructed based on prior knowledge, the results demonstrate that the proposed methodology is capable of determining the joint optimal hyperparameter configuration through optimization and lead to more accurate and consistent performance on vehicle trajectory prediction. Furthermore, a sensitivity analysis is performed to measure the uncertainty in the network output subject to the uncertainty in the input hyperparameters, by creating different levels of perturbation in the optimal hyperparameter values. The results show that the proposed method is most sensitive to variations in the learning rate and least sensitive to the change in the number of hidden units, and that it shows robust performance when the perturbation level gradually increases.

6.2. Recommendations

Based on the experimental studies, several practical recommendations are summarized as follows, to guide future similar research:

- In Chapter 2, the optimal values for the two parameters are obtained through grid search, which is a straightforward but time-consuming approach. Thus, further research efforts need to be devoted to developing more effective parameter tuning strategies to improve the efficiency and robustness of the proposed noise removal methodology.
- In Chapter 3, The proposed methodology may have difficulties detecting vehicles under data occlusion. For example, when making predictions on consecutive frames, some vehicle objects occluded by other objects may not be continuously detected by the proposed CNN due to the issue of data occlusion. Some future research directions are 1) improving the detection performance by refining the proposed CNN architecture to achieve more effective and efficient feature fusion and extraction; and 2) developing effective strategies through data fusion to reduce false-negative detections due to data occlusion.
- Based on the observations in Chapter 4, future research is needed in the following areas, including 1) improving the network prediction performance through more robust and efficient data pre-processing procedures and training schemes; and 2) developing LSTM encoder-decoder networks to perform long-term (output sequence > 2.5 secs) pedestrian trajectory prediction and risk assessment on roadside LiDAR data.
- For the methodology proposed in Chapter 5, the following improvements are suggested: 1) exploring the impact of varying environmental conditions on the proposed method; 2) considering the issues of data occlusion during object detection, data association, and prediction, and developing LSTM networks or other deep neural network architectures that are robust to data discontinuities caused by data occlusion; 3) expanding the applicability of the proposed method on different traffic scenarios and public datasets; and 4) incorporating multi-modal data to further improve the accuracy and robustness of LSTM-based vehicle trajectory prediction.

REFERENCES

2021. Hesai and Scale, "PandaSet".

Ahmed, S., Huda, M. N., Rajbhandari, S., Saha, C., Elshaw, M. & Kanarachos, S. 2019. Pedestrian and cyclist detection and intent estimation for autonomous vehicles: A survey. *Applied Sciences,* 9**,** 2335.

Alaba, S. Y. & Ball, J. E. 2022. A survey on deep-learning-based lidar 3d object detection for autonomous driving. *Sensors,* 22**,** 9577.

ASCE 2021. Infrastructure Report Card.

Asvadi, A., Girao, P., Peixoto, P. & Nunes, U. 3D object tracking using RGB and LIDAR data. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016. IEEE, 1255-1260.

Azim, A. & Aycard, O. Detection, classification and tracking of moving objects in a 3D environment. 2012 IEEE Intelligent Vehicles Symposium, 2012. IEEE, 802-807.

Baek, M., Jeong, D., Choi, D. & Lee, S. 2020. Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications. *Sensors,* 20**,** 288.

Barad, J. 2021. Roadside Lidar Helping to Build Smart and Safe Transportation Infrastructure. SAE Technical Paper.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C. & Gall, J. Semantickitti: A dataset for semantic scene understanding of lidar sequences. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019. 9297-9307.

Bengio, Y. 2012. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade.* Springer.

Bijelic, M., Gruber, T. & Ritter, W. A benchmark for lidar sensors in fog: Is detection breaking down? 2018 IEEE Intelligent Vehicles Symposium (IV), 2018. IEEE, 760-767.

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Bottou, L. 2012. Stochastic gradient descent tricks. *Neural networks: Tricks of the trade.* Springer.

Brochu, E., Cora, V. M. & De Freitas, N. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.

Buch, N., Velastin, S. A. & Orwell, J. 2011. A review of computer vision techniques for the analysis of urban traffic. *IEEE Transactions on intelligent transportation systems,* 12**,** 920-939.

Buhet, T., Wirbel, E., Bursuc, A. & Perrotton, X. PLOP: probabilistic polynomial objects trajectory prediction for autonomous driving. Conference on Robot Learning, 2021. PMLR, 329-338.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. & Beijbom, O. nuscenes: A multimodal dataset for autonomous driving. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020. 11621-11631.

Castleman, K. R. 1996. *Digital image processing*, Prentice Hall Press.

Chandra, R., Guan, T., Panuganti, S., Mittal, T., Bhattacharya, U., Bera, A. & Manocha, D. 2020. Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms. 5**,** 4882-4890.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S. & Ramanan, D. Argoverse: 3d tracking and forecasting with rich maps. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 8748-8757.

Charron, N., Phillips, S. & Waslander, S. L. De-noising of lidar point clouds corrupted by snowfall. 2018 15th Conference on Computer and Robot Vision (CRV), 2018. IEEE, 254-261.

Chen, J., Wang, Y., Wu, R. & Campbell, M. Spatial-Temporal Graph Neural Network For Interaction-Aware Vehicle Trajectory Prediction. 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 2021. IEEE, 2119-2125.

Chintalacheruvu, N. & Muthukumar, V. 2012. Video based vehicle detection and its application in intelligent transportation systems. *Journal of transportation technologies,* 2**,** 305.

Choi, D., Yim, J., Baek, M. & Lee, S. 2021. Machine learning-based vehicle trajectory prediction using v2v communications and on-board sensors. 10**,** 420.

Dai, S., Li, L. & Li, Z. 2019. Modeling vehicle interactions via modified LSTM models for trajectory prediction. 7**,** 38287-38296.

Deo, N., Rangesh, A. & Trivedi, M. M. 2018. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. 3**,** 129-140.

Deo, N. & Trivedi, M. M. Convolutional social pooling for vehicle trajectory prediction. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018. 1468-1476.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd.*

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters,* 27**,** 861-874.

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W. & Dietmayer, K. 2020. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems,* 22**,** 1341-1360.

Ferguson, S., Luders, B., Grande, R. C. & How, J. P. 2014. Real-Time Predictive Modeling and Robust Avoidance of Pedestrians with Uncertain, Changing Intentions. *ArXiv,* abs/1405.5581.

FHWA 2012. Moving Ahead for Progress in the 21st Century Act (MAP-21): A Summary of Highway Provisions. *In:* U.S. DEPARTMENT OF TRANSPORTATION (ed.). Office of Policy and Governmental Affairs, Federal Highway Administration.

Frazier, P. I. 2018. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Fünfgeld, S., Holzäpfel, M., Frey, M. & Gauterin, F. 2017. Stochastic forecasting of vehicle dynamics using sequential Monte Carlo simulation. 2**,** 111-122.

Gao, B., Pan, Y., Li, C., Geng, S. & Zhao, H. 2021. Are we hungry for 3D LiDAR data for semantic segmentation? a survey of datasets and methods. *IEEE Transactions on Intelligent Transportation Systems*.

Gargoum, S. & El-Basyouny, K. Automated extraction of road features using LiDAR data: A review of LiDAR applications in transportation. 2017 4th International Conference on Transportation Information and Safety (ICTIS), 2017. IEEE, 563-574.

Geiger, A., Lenz, P. & Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012. IEEE, 3354-3361.

Gers, F. A., Schmidhuber, J. & Cummins, F. 2000. Learning to forget: Continual prediction with LSTM. 12**,** 2451-2471.

Gers, F. A., Schraudolph, N. N. & Schmidhuber, J. 2002. Learning precise timing with LSTM recurrent networks. 3**,** 115-143.

Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A. S., Hauswald, L., Pham, V. H., Mühlegg, M. & Dorn, S. 2020. A2D2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*.

Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. 13th International Conference on Artificial Intelligence and Statistics, 2010 Sardinia, Italy. 249-256.

Gonzales, R. C. & Woods, R. E. 2002. *Digital image processing*, Prentice hall New Jersey.

Goodfellow, I., Bengio, Y. & Courville, A. 2016. *Deep Learning,* Cambridge, Massachusetts, MIT Press.

Graettinger, A. J., Kilim, R. R., Govindu, M. R., Johnson, P. W. & Durrans, S. R. 2005. Federal Highway Administration vehicle classification from video data and a disaggregation model. *Journal of transportation engineering,* 131**,** 689-698.

Habibi, G., Jaipuria, N. & How, J. P. 2018. Context-aware pedestrian motion prediction in urban intersections. *arXiv preprint arXiv:1806.09453*.

He, K., Zhang, X., Ren, S. & Sun, J. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence,* 37**,** 1904-1916.

Heinzler, R., Piewak, F., Schindler, P. & Stork, W. 2020. Cnn-based lidar point cloud de-noising in adverse weather. 5**,** 2514-2521.

Hochreiter, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. 6**,** 107-116.

Hochreiter, S. & Schmidhuber, J. 1997. Long short-term memory. *Neural computation,* 9**,** 1735-1780.

Hoermann, S., Stumper, D. & Dietmayer, K. Probabilistic long-term prediction for autonomous vehicles. 2017 IEEE Intelligent Vehicles Symposium (IV), 2017. IEEE, 237-243.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 4700-4708.

Huang, Y., Du, J., Yang, Z., Zhou, Z., Zhang, L. & Chen, H. 2022. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles,* 7**,** 652-674.

Ioffe, S. & Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jiang, Y., Zhu, B., Yang, S., Zhao, J. & Deng, W. 2022. Vehicle trajectory prediction considering driver uncertainty and vehicle dynamics based on dynamic bayesian network. 53**,** 689-703.

Jokela, M., Kutila, M. & Pyykönen, P. 2019. Testing and validation of automotive point-cloud sensors in adverse weather conditions. *Applied Sciences,* 9**,** 2341.

Kawakami, K. 2008. *Supervised sequence labelling with recurrent neural networks.* Technical University of Munich.

Kim, B., Kang, C. M., Kim, J., Lee, S. H., Chung, C. C. & Choi, J. W. 2017. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).* IEEE.

Kingma, D. P. & Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kurup, A. & Bos, J. 2021. DSOR: A Scalable Statistical Filter for Removing Falling Snow from LiDAR Point Clouds in Severe Winter Weather. *arXiv preprint arXiv:2109.07078*.

Kutila, M., Pyykönen, P., Holzhüter, H., Colomb, M. & Duthon, P. Automotive LiDAR performance verification in fog and rain. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018. IEEE, 1695-1701.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J. & Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 12697-12705.

Lefèvre, S., Vasquez, D. & Laugier, C. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal,* 1**,** 1.

Lefkopoulos, V., Menner, M., Domahidi, A. & Zeilinger, M. N. 2020. Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme. 6**,** 80-87.

Lei, B., Kirk, T. Q., Bhattacharya, A., Pati, D., Qian, X., Arroyave, R. & Mallick, B. K. 2021. Bayesian optimization with adaptive surrogate models for automated experimental design. 7**,** 194.

Leon, F. & Gavrilescu, M. 2021. A review of tracking and trajectory prediction methods for autonomous driving. 9**,** 660.

Li, P., Guo, H., Bao, S. & Kusari, A. 2022. A Probabilistic Framework for Estimating the Risk of Pedestrian-Vehicle Conflicts at Intersections.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D. & Li, J. 2020a. Deep learning for LiDAR point clouds in autonomous driving: a review. *IEEE Transactions on Neural Networks and Learning Systems*.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D. & Li, J. 2020b. Deep learning for lidar point clouds in autonomous driving: A review. 32**,** 3412-3432.

Li, Y., Xin, L., Yu, D., Dai, P., Wang, J. & Li, S. E. Pedestrian trajectory prediction with learning-based approaches: A comparative study. 2019 IEEE Intelligent Vehicles Symposium (IV), 2019. IEEE, 919-926.

Lin, L., Li, W., Bi, H. & Qin, L. 2021. Vehicle Trajectory Prediction Using LSTMs With Spatial–Temporal Attention Mechanisms. 14**,** 197-208.

Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018. 8759-8768.

Liu, S., Zheng, K., Zhao, L. & Fan, P. 2020. A driving intention prediction method based on hidden Markov model for autonomous driving. 157**,** 143-149.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A. C. Ssd: Single shot multibox detector. European conference on computer vision, 2016. Springer, 21-37.

Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W. & Manocha, D. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. Proceedings of the AAAI Conference on Artificial Intelligence, 2019. 6120-6127.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. 30th International Conference on Machine Learning, 2013 Atlanta, Georgia. 3.

MATHWORKS. 2019. *MATLAB deep learning toolbox* [Online]. Available: https://www.mathworks.com/products/deep-learning.html [Accessed].

MATHWORKS. 2022. *MATLAB deep learning toolbox* [Online]. Available: https://www.mathworks.com/products/deep-learning.html [Accessed].

Mekala, M., Park, W., Dhiman, G., Srivastava, G., Park, J. H. & Jung, H.-Y. 2021. Deep learning inspired object consolidation approaches using lidar data for autonomous driving: a review. 1-21.

Michaud, S., Lalonde, J.-F. & Giguere, P. Towards characterizing the behavior of LiDARs in snowy conditions. Proceedings of the 7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015.

Miglani, A. & Kumar, N. 2019. Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications,* 20**,** 100184.

Mo, X., Xing, Y. & Lv, C. Interaction-aware trajectory prediction of connected vehicles using cnn-lstm networks. IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, 2020. IEEE, 5057-5062.

Mockus, J. 1994. Application of Bayesian approach to numerical methods of global and stochastic optimization. 4**,** 347-365.

Moosmann, F. & Fraichard, T. Motion estimation from range images in dynamic outdoor scenes. 2010 IEEE International Conference on Robotics and Automation, 2010. IEEE, 142-147.

Park, J.-I., Park, J. & Kim, K.-S. 2020. Fast and Accurate Desnowing Algorithm for LiDAR Point Clouds. *IEEE Access,* 8**,** 160202-160212.

Pham, Q.-H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V. & Lin, J. A* 3D dataset: Towards autonomous driving in challenging environments. 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020. IEEE, 2267-2273.

Piewak, F., Pinggera, P., Schafer, M., Peter, D., Schwarz, B., Schneider, N., Enzweiler, M., Pfeiffer, D. & Zollner, M. Boosting lidar-based semantic labeling by cross-modal training data generation. Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018. 0-0.

Qi, C. R., Su, H., Mo, K. & Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017. 652-660.

Rasmussen, C. E. & Williams, C. K. 2006. *Gaussian processes for machine learning*, MIT press Cambridge, MA.

Redmon, J. & Farhadi, A. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. & Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 658-666.

Ridel, D., Rehder, E., Lauer, M., Stiller, C. & Wolf, D. A literature review on the prediction of pedestrian behavior in urban scenarios. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018. IEEE, 3105-3112.

Roriz, R., Campos, A., Pinto, S. & Gomes, T. 2021. Dior: A hardware-assisted weather denoising solution for lidar point clouds. 22**,** 1621-1628.

Royo, S. & Ballesta-Garcia, M. 2019. An overview of lidar imaging systems for autonomous vehicles. *Applied Sciences,* 9**,** 4093.

Rusu, R. B. & Cousins, S. 3d is here: Point cloud library (pcl). 2011 IEEE international conference on robotics and automation, 2011. IEEE, 1-4.

Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. How does batch normalization help optimization? Advances in Neural Information Processing Systems, 2018. 2483-2493.

Schulz, J., Hubmann, C., Löchner, J. & Burschka, D. Interaction-aware probabilistic behavior prediction in urban environments. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018a. IEEE, 3999-4006.

Schulz, J., Hubmann, C., Löchner, J. & Burschka, D. Multiple model unscented kalman filtering in dynamic bayesian networks for intention estimation and trajectory prediction. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018b. IEEE, 1467-1474.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & De Freitas, N. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE,* 104**,** 148-175.

Sherstinsky, A. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena,* 404**,** 132306.

Shorten, C. & Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data,* 6**,** 1-48.

Sighencea, B. I., Stanciu, R. I. & Căleanu, C. D. 2021a. A review of deep learning-based methods for pedestrian trajectory prediction. *Sensors,* 21**,** 7543.

Sighencea, B. I., Stanciu, R. I. & Căleanu, C. D. 2021b. A review of deep learning-based methods for pedestrian trajectory prediction. 21**,** 7543.

Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 2012. 2951-2959.

Song, X., Pi, R., Lv, C., Wu, J., Zhang, H., Zheng, H., Jiang, J. & He, H. 2021. Augmented Multiple Vehicles' Trajectories Extraction under Occlusions with Roadside LiDAR Data. *IEEE Sensors Journal*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research,* 15**,** 1929-1958.

Sun, L., Yan, Z., Mellado, S. M., Hanheide, M. & Duckett, T. 3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018a. IEEE, 5942-5948.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y. & Caine, B. Scalability in perception for autonomous driving: Waymo open dataset. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 2446-2454.

Sun, P., Sun, C., Wang, R. & Zhao, X. 2022. Object Detection Based on Roadside LiDAR for Cooperative Driving Automation: A Review. *Sensors,* 22**,** 9316.

Sun, Y., Xu, H., Wu, J., Zheng, J. & Dietrich, K. M. 2018b. 3-D data processing to extract vehicle trajectories from roadside LiDAR data. *Transportation Research Record,* 2672**,** 14-22.

Tan, D., Younis, M., Lalouani, W., Fan, S. & Song, G. 2023. A novel pedestrian road crossing simulator for dynamic traffic light scheduling systems. 1-15.

Theodose, R., Denis, D., Chateau, T., Frémont, V. & Checchin, P. 2021. A Deep Learning Approach for LiDAR Resolution-Agnostic Object Detection. *IEEE Transactions on Intelligent Transportation Systems*.

TRIP 2021. America's Interstate Highway System at 65: Meeting America's Transportation Needs with a Reliable, Safe & Well-Maintained National Highway Network.

Utriainen, R. 2020. The potential impacts of automated vehicles on pedestrian safety in a four-season country. 25**,** 188-196.

Vaquero, V., del Pino, I., Moreno-Noguer, F., Solà, J., Sanfeliu, A. & Andrade-Cetto, J. 2020. Dual-Branch CNNs for Vehicle Detection and Tracking on LiDAR Data. *IEEE Transactions on Intelligent Transportation Systems*.

Vizzari, G., Manenti, L., Ohtsuka, K. & Shimura, K. 2015. An agent-based pedestrian and group dynamics model applied to experimental and real-world scenarios. 19**,** 32-45.

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W. & Yeh, I.-H. CSPNet: A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020a. 390-391.

Wang, G., Xiao, D. & Gu, J. Review on vehicle detection based on video for traffic surveillance. 2008 IEEE international conference on automation and logistics, 2008. IEEE, 2961-2966.

Wang, H., Wang, B., Liu, B., Meng, X. & Yang, G. 2017. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. *Robotics and Autonomous Systems,* 88**,** 71-78.

Wang, J., Wang, P., Zhang, C., Su, K. & Li, J. F-net: Fusion neural network for vehicle trajectory prediction in autonomous driving. ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021. IEEE, 4095-4099.

Wang, S., Pi, R., Li, J., Guo, X., Lu, Y., Li, T. & Tian, Y. 2022a. Object Tracking Based on the Fusion of Roadside LiDAR and Camera Data. 71**,** 1-14.

Wang, S., Zhao, P., Yu, B., Huang, W. & Liang, H. 2020b. Vehicle trajectory prediction by knowledge-driven lstm network in urban environments. 2020.

Wang, W., You, X., Chen, L., Tian, J., Tang, F. & Zhang, L. 2022b. A scalable and accurate de-snowing algorithm for LiDAR point clouds in winter. 14**,** 1468.

Wang, Y., Liu, Z., Zuo, Z., Li, Z., Wang, L. & Luo, X. 2019. Trajectory planning and safety assessment of autonomous vehicles based on motion prediction and model predictive control. 68**,** 8546-8556.

Wu, B., Wan, A., Yue, X. & Keutzer, K. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018a. IEEE, 1887-1893.

Wu, J., Xu, H., Tian, Y., Pi, R. & Yue, R. 2020a. Vehicle detection under adverse weather from roadside LiDAR data. *Sensors,* 20**,** 3433.

Wu, J., Xu, H., Zhang, Y. & Sun, R. 2020b. An improved vehicle-pedestrian near-crash identification method with a roadside LiDAR sensor. *Journal of Safety Research,* 73**,** 211-224.

Wu, J., Xu, H., Zheng, J. & Zhao, J. 2020c. Automatic vehicle detection with roadside LiDAR data under rainy and snowy conditions. *IEEE Intelligent Transportation Systems Magazine,* 13**,** 197-209.

Wu, J., Xu, H., Zheng, Y. & Tian, Z. 2018b. A novel method of vehicle-pedestrian near-crash identification with roadside LiDAR data. *Accident Analysis & Prevention,* 121**,** 238-249.

Wu, Y., Wang, Y., Zhang, S. & Ogai, H. 2020d. Deep 3D object detection networks using LiDAR data: A review. *IEEE Sensors Journal,* 21**,** 1152-1171.

Xiao, H., Wang, C., Li, Z., Wang, R., Bo, C., Sotelo, M. A. & Xu, Y. 2020. UB-LSTM: a trajectory prediction method combined with vehicle behavior recognition. 2020.

Xie, G., Gao, H., Qian, L., Huang, B., Li, K. & Wang, J. 2017. Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models. 65**,** 5999-6008.

Xu, Q., Zhou, Y., Wang, W., Qi, C. R. & Anguelov, D. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 15446-15456.

Xue, P., Liu, J., Chen, S., Zhou, Z., Huo, Y. & Zheng, N. Crossing-road pedestrian trajectory prediction via encoder-decoder lstm. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019. IEEE, 2027-2033.

Yamauchi, B. Fusing ultra-wideband radar and lidar for small UGV navigation in all-weather conditions. Defense + Commercial Sensing, 2010.

Yan, Y., Mao, Y. & Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors,* 18**,** 3337.

Zhang, C., Berger, C. & Dozza, M. 2021a. Social-IWSTCNN: A Social Interaction-Weighted Spatio-Temporal Convolutional Neural Network for Pedestrian Trajectory Prediction in Urban Traffic Scenarios. *arXiv preprint arXiv:2105.12436*.

Zhang, G., Avery, R. P. & Wang, Y. 2007. Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras. *Transportation research record,* 1993**,** 138-147.

Zhang, J., Pi, R., Ma, X., Wu, J., Li, H. & Yang, Z. 2021b. Object Classification with Roadside LiDAR Data Using a Probabilistic Neural Network. *Electronics,* 10**,** 803.

Zhang, J., Xiao, W., Coifman, B. & Mills, J. P. 2020a. Vehicle Tracking and Speed Estimation From Roadside Lidar. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* 13**,** 5597-5608.

Zhang, J., Xiao, W. & Mills, J. P. 2022. Optimizing Moving Object Trajectories from Roadside Lidar Data by Joint Detection and Tracking. 14**,** 2124.

Zhang, L., Zheng, J., Sun, R. & Tao, Y. 2020b. GC-net: Gridding and Clustering for Traffic Object Detection with Roadside LiDAR. *IEEE Intelligent Systems*.

Zhang, Y., Xu, H. & Wu, J. 2020c. An automatic background filtering method for detection of road users in heavy traffics using roadside 3-D LiDAR sensors with noises. *IEEE Sensors Journal,* 20**,** 6596-6604.

Zhang, Z., Zheng, J., Xu, H. & Wang, X. 2019a. Vehicle detection and tracking in complex traffic circumstances with roadside LiDAR. *Transportation research record,* 2673**,** 62-71.

Zhang, Z., Zheng, J., Xu, H., Wang, X., Fan, X. & Chen, R. 2019b. Automatic background construction and object detection based on roadside LiDAR. *IEEE Transactions on Intelligent Transportation Systems,* 21**,** 4086-4097.

Zhao, J. 2019. *Exploring the fundamentals of using infrastructure-based LiDAR sensors to develop connected intersections.* PhD thesis, Texas Tech University.

Zhao, J., Li, Y., Xu, H. & Liu, H. 2019a. Probabilistic prediction of pedestrian crossing intention using roadside LiDAR data. *IEEE Access,* 7**,** 93781-93790.

Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y. & Wu, D. 2019b. Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. *Transportation Research Part C: Emerging Technologies,* 100**,** 68-87.

Zhao, J., Xu, H., Wu, J., Zheng, Y. & Liu, H. 2019c. Trajectory tracking and prediction of pedestrian's crossing intention using roadside LiDAR. *IET Intelligent Transport Systems,* 13**,** 789-795.

Zhou, S., Xu, H., Zhang, G., Ma, T. & Yang, Y. 2022. Leveraging Deep Convolutional Neural Networks Pre-Trained on Autonomous Driving Data for Vehicle Detection From Roadside LiDAR Data. *IEEE Transactions on Intelligent Transportation Systems,* 23**,** 22367-22377.

Zhou, S., Xu, H., Zhang, G., Ma, T. & Yang, Y. 2024. DCOR: Dynamic Channel-Wise Outlier Removal to De-Noise LiDAR Data Corrupted by Snow.

Zhou, Z., Kitamura, S., Watanabe, Y., Yamada, S. & Takada, H. Extraction of Pedestrian Position and Attribute Information Based on the Integration of LiDAR and Smartphone Sensors. 2021 IEEE International Conference on Mechatronics and Automation (ICMA), 2021. IEEE, 784-789.