Customizing Trustworthy Machine Learning and Advanced Computing Methods for Cooperative and Equitable Traffic Systems

Hao Yang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2023

Reading Committee:

Yinhai Wang, Chair
Xuegang Ban
Edward McCormack
Simon Shaolei Du

Program Authorized to Offer Degree:
Civil and Environmental Engineering

University of Washington

Abstract

Customizing Trustworthy Machine Learning and Advanced Computing Methods for Cooperative and Equitable Traffic Systems

Hao Yang

Chair of the Supervisory Committee:
Yinhai Wang
Department of Civil and Environmental Engineering

Urbanization offers opportunities for better lives and amenities globally, yet it brings considerable challenges. The combination of the network congestions, the inequity complaints, the unbalanced supply and demand of traffic resources, and gaps in integrating traffic needs with emerging technologies are converging to expand our understanding of smart traffic systems from a holistic point of view. To address these challenges, it is necessary to develop smart, cooperative, and equitable next-generation transportation systems. These systems need to integrate the machine intelligence to accomplish three key tasks: capturing real-time, accurate traffic information through sensors, integrating advanced data processing and analytical tools, and providing trustworthy services to roadway users without discrimination.

This dissertation explores the customization of vital technologies and requisite methods for constructing trustworthy next-generation traffic systems. Specifically, the thesis advocates for a tripartite framework that aligns trustworthiness across three essential components: the data, methodological, and system levels, which denotes the intelligent transportation systems that are more cooperative, accurate, reliable, and less discriminatory. These systems and methods exploit the combination of tailored machine learning methods, along with distributed computing systems,

to propose promising solutions for three key tasks: 1) Cooperative Traffic Perception – develop a privacy-aware cooperative sensing system through learning cross-sensor visual representations. A pioneering edge-empowered cooperative multi-camera perception system has been proposed and tested based on Internet of Things architecture. 2) Multimodality Data Fusion – pertain to advancing the customization of machine learning and feature fusion methodologies for multimodal data representations, with the aim of providing accurate and timely future traffic information for both roadway users and managers. Currently, the traffic forecasting approaches face obstacles such as limitations supporting multiple sensor modalities, insufficient traffic domain knowledge integrations, and poor generalization across heterogeneous traffic network structures. The works develop customized representation learning methodologies to predict traffic patterns of different scales and input formats. These proposed methods have outperformed sixteen cutting-edge machine learning approaches in predicting route speeds for hundreds of users, parking lot occupancy for nineteen lots, and network-scale congestion in cities worldwide. 3) Demonstrating Pilot Cooperative and Equitable Traffic Infrastructure Systems for mobility, safety, and equity enhancement. The systems utilize customized representation learning and edge computing approaches to address critical transportation needs. These include developing a human-machine cooperative automatic traffic signal assistance system for Vulnerable Road Users (VRU) and creating a vehicle recognition framework that takes unbalanced data samples into consideration. The outstanding performance of these representation learning methods significantly enhances both inter and intra-cooperation within traffic systems, from the perspectives of sensors, data, system, and their methodologies. Furthermore, advancements in the precision, generalization and privacy preservation underscore the substantial benefits of integrating transportation domain needs into machine learning and advanced computing research, which is pivotal for the future development of trustworthy smart traffic systems.

Contents

Listing of Figures

List of Tables

Acknowledgments

With profound gratitude, I reflect upon the journey that has culminated in the completion of my Ph.D. program in Transportation Engineering at the University of Washington. This journey has been extraordinary, a confluence of challenges, discoveries, and growth that I could not have navigated without the assistance and support of a constellation of individuals.

First and foremost, I wish to express my deepest gratitude to my advisor, Prof. Yinhai Wang. His continuous guidance, timely encouragement, and unflagging support have been foundational throughout my Ph.D. program. The breadth and depth of his knowledge, coupled with his insightful feedback and passion for transportation engineering, have shaped my research work at every turn. Prof. Wang's mentorship has been extraordinary, and I feel privileged to have learned under his tutelage. His generosity in sharing his expertise and his contributions to my academic and personal growth are deeply cherished. In addition, I am grateful to my committee members, Prof. Edward McCormack, Prof. Xuegang (Jeff) Ban, Prof. Simon Shaolei Du, and Prof. Sheng Wang. Their diverse expertise, unique perspectives, and incisive criticisms have significantly enhanced the quality of my research work. They have provided guidance through

the intricacies and complexities of the Ph.D. program, strengthening my resolve and honing my academic focus.

Extending beyond the realm of my committee, I am thankful for the camaraderie, support, and intellectual stimulation provided by my fellow Ph.D. students and colleagues. The enriching discussions and collaborations have rendered my time at the University of Washington an invaluable experience. I extend special gratitude to Mr. Chenxi Liu, Dr. Ruimin Ke, Dr. Ziyuan Pu, Dr. Zhiyong Cui, Dr. Yifan Zhuang, Dr. Meixin Zhu, Dr. Wei Sun, Dr. Wenbo Zhu, Dr. John Ash, and all the STAR Lab current and previous members. Their insightful feedback, generous assistance, and active collaboration on research projects and coursework were pivotal for the successful completion of my journey.

My heartfelt thanks extend to my family and friends, whose unwavering love, support, and encouragement have been my bedrock throughout this program. Their enduring belief in me, their sacrifices, and their understanding during this demanding period have kept me motivated and inspired. Among them, I wish to extend my deepest gratitude to my beloved wife, whose endless reservoir of love, encouragement, and support has been my beacon, inspiring and strengthening me during the toughest of times.

As I reflect on the broader context of my journey, I owe a significant debt of gratitude to several funding agencies and organizations that played a critical role in my Ph.D. program. Among them, the National Science Foundation, the Federal Highway Administration, the Washington State Department of Transportation, and the Pacific Northwest Transportation Consortium stand out for their substantial support. Their financial backing made it feasible for me to delve into my research work, participate in a range of conferences, and accumulate a wealth of invaluable experience within the diverse and dynamic research community. Simultaneously, I wish

to express my immense appreciation to the University of Washington. The institution has not only provided an exceptional academic environment conducive to rigorous study and innovative thinking but has also granted me the opportunity to pursue my passion for transportation engineering fervently. The support I received throughout this journey was unwavering and instrumental in shaping my academic and personal growth. The roles played by my advisor, committee members, staff, colleagues, funding agencies, family, and friends have been immeasurable, each contributing uniquely to my journey. Their support echoed in every academic milestone I achieved and resonated in each personal development I experienced.

Currently, as I stand on the threshold of this new phase in my life, the anticipation is both overwhelming and exhilarating. My roles as a researcher, transportation engineer, and lifelong learner are set to evolve and expand, and I eagerly anticipate engaging with new challenges. I look forward to continuing my journey—a journey replete with uncertainties and promises—and am keen on making meaningful contributions to the field. The future beckons, and I am ready to embrace it, fortified by the knowledge and experiences I gained during my time at the University of Washington.

1

Chapter 1. Background and Introduction

1.1 Background

1.1.1 Advancement of Smart Transportation Systems

Urbanization, while affording numerous opportunities globally, simultaneously presents considerable challenges due to escalating demands and limited resource in urban regions. Fortunately, the advent of the fourth industrial revolution, propelled by transformative technolo-

gies such as Artificial Intelligence (AI), Information and Communication Technologies (ICT), and the Internet of Things (IoT), has enhanced the capability of urban Cyber-Physical Systems (CPS) for advanced perception, interaction, and control among users, vehicles, and infrastructure systems. As urbanization and technological evolution coalesce amid shifting societal and political landscapes, current limitations - traffic network congestions [3], discrimination complaints [4], data biases and privacy issues [5], imbalanced network resource distribution in rural and urban regions [6], and the integration gaps between domain knowledge and technologies applications [7] - collectively push our holistic understanding of smart cities and future traffic systems.

The integration of AI and IoT technologies has empowered cities with more intelligent sensing and adjustment capabilities. This has enabled cities to collect data through various sensors and utilize automated analysis technologies to make better decisions and improve multiple perspectives such as safety, efficiency and equity. A smart city can be primarily divided into three layers: the technology/sensor base, the middle layer responsible for data fusion and mining, and the broad application layer that most residents interact with [8]. Intelligent Transportation Systems (ITS), a significant component of smart cities, aim to provide innovative services related to various modes of transport and traffic management. They enable users to be better informed, making their use of transport networks safer, more coordinated and more efficient [9]. Daily commuters can access real-time traffic information and adjust their travel plans, while traffic management agencies can automatically acquire and analyze dynamic traffic data using edge sensors and AI algorithms. Big traffic data helps build robust prediction and controlling models to optimize traffic flow and help more diverse users. By 2025, ITS is expected to reduce commute time by over 15% on average, depending on the city's scale, transit system construction,

and commuting patterns [10].

1.1.2 Traffic Perception Sensors and Technologies

Data collection and analysis constitute essential elements of smart and cooperative traffic systems, as they furnish vital information for numerous traffic services, including traffic flow prediction and traffic signal timing optimization [11]. Typically, these operations encompass two phases carried out on distinct platforms. The initial phase involves data gathering through an assortment of sensors, each boasting unique advantages and adaptability to different circumstances. Vision-based sensing systems, like RGB cameras, are predominantly employed in ITS due to their extensive coverage, scalable deployment, and mature analytical algorithms [12]. Cameras hold a crucial role in contemporary data collection frameworks as they provide visual data analogous to human observations, which is convenient for manual examination. Furthermore, the maturity of vision-based analysis algorithms and tools facilitates a swift automated analysis process. With the advent of IoT technologies, cameras find utility in numerous ITS applications, including traffic enforcement, parking space monitoring, traffic flow estimation, vehicle tracking, vehicle reidentification, and collision avoidance [13].

Additionally, automotive radars have found wide application in parking assistant systems and automated speed-camera enforcement, where only coarse object features are necessary [14]. Meanwhile, Light Detection and Ranging (LiDAR) presents more precise distance measurements and denser data points by deploying laser pulses, rendering it suitable for 3D reconstruction, albeit at a higher cost [15]. In terms of price and scalability, magnetic sensors outshine the others, yet they remain susceptible to fluctuations induced by passing heavy vehicles. Recent advancements in sensing technologies have diversified data collection alternatives, leading to system

deployment decisions based on factors such as cost, accuracy, and application scenarios. Consequently, sensor fusion, which involves the amalgamation of multiple sensors, is a preferred strategy to tackle complex situations. Autonomous Vehicles (AVs), for instance, integrate radar, LiDAR and cameras in their perception systems for superior 3D scenario reconstruction. Here, radar can be employed to perceive depth information. And LiDAR provides 3D information, cameras contribute detailed target data, such as surfaces and categories. However, the variety of sensor types, massive data volume, and diverse analysis tasks require precise and efficient data processing and modeling capabilities, presenting many challenges to researchers. Given the escalating demand for autonomous vehicles and intelligent data-driven services, the choice, deployment, and integration of suitable sensors, attuned to the requirements of specific tasks, are progressively becoming pivotal to the success of ITS.

1.1.3 Machine Learning and Edge Computing for Traffic Data Analytics

The expansion of urban areas has led to the construction and upgrading of transportation facilities such as roadways and transit centers. Monitoring these regions is crucial to ensure traffic safety and efficiency, and transportation agencies and commuters also require real-time traffic information to make informed decisions. To sense traffic, various sensors such as cameras, radars and LiDARs are installed, and different regions may require distinct sensing tasks, such as vehicle counting and pedestrian tracking. So, the multimodality data refers to data that comes from different sources or modalities, such as images, videos, sensor data, text, and audio [16]. In transportation systems, multimodality data can come from various sources, including: traffic cameras, GPS and trajectory data, social media data, users' attributes data, weather data transit data. [17]

Presently, traffic managers and agencies depend on cloud servers for data processing. Standard cloud computing-based traffic sensing systems have two distinct stages - data collection and transmission, followed by data analysis on cloud servers [18]. Yet, with broader monitoring coverage and diverse sensing tasks comes heightened data volume and workload on the computing platform. This demand necessitates increased computer resources and transmission bandwidth, particularly with the growing number of sensors. AVs, reliant on onboard computing with weaker processing power compared to cloud platforms, exacerbate this issue [19]. Although enhancing computing power by adding more devices or upgrading existing ones is a solution, it is challenging to align the server-side computing power with growing traffic data within the current framework. Additionally, data processing delays can be significant and intolerable for time-sensitive applications such as traffic safety monitoring and anomaly detection.

Current research in this domain has primarily focused on refining the accuracy of algorithmic frameworks. However, this trajectory raises a concern: the unique characteristics inherent to transportation applications might be overshadowed or undervalued. As the data volumes in ITS applications burgeon, there's a discernible shift toward the urgent requirements of efficiency, privacy, and reliability [20]. This evolution underscores a mounting demand for proficient algorithms and frameworks to adeptly handle the expanding data volumes. Emphasizing the increasing need for algorithmic efficiency, especially in systems lacking high-performance computing capabilities, highlights the urgency to decrease their reliance on such infrastructures [21]. Achieving this mandates a focus on algorithmic efficiency and the repercussions on privacy when integrating data processing algorithms in ITS. This perspective becomes even more important when implementing methodologies that leverage deep learning; these typically require greater computational resources than conventional machine learning algorithms using hand-

crafted feature descriptors. Additionally, employing high-resolution imagery and feature pyramids to amplify object detection precision results in augmented computational operations and a concomitant dip in inference speed. Prioritizing the creation of efficient algorithms can reduce dependency on high-end computing hardware, consequently amplifying the accessibility and scalability of intelligent systems within ITS applications.

The adoption and integration of edge computing technology are becoming increasingly promising solutions [21]. Edge computing, a distributed computing technology, enables data processing and analysis at the network edge, closer to the data generation source. Unlike traditional cloud computing, which relies on centralized data processing and storage in remote data centers, edge computing leverages the computing power and storage capacity of edge devices such as sensors, mobile devices, gateways and even moving platforms. By processing and analyzing data locally, edge computing reduces latency, improves data privacy and security, and enables real-time decision-making in time-sensitive applications. Therefore, reducing dependency on high-performance computing machines by adopting cost-effective edge computing approaches and improving algorithm efficiency and data privacy are crucial.

1.1.4 Learning Multimodality Data Representations for Traffic Tasks

Machine learning (ML), especially with the advent of deep learning techniques, is revolutionizing transportation. A noteworthy branch of deep learning that has significantly impacted this area is representation learning. Representation learning is crucial in traffic research community, as it enables machines to understand and create a compact representation of complex, high-dimensional data. The learned representations or features can enhance the performance of downstream tasks. The necessity for representation learning stems from the inherent complex-

ity, high-dimensionality, and lack of structure in raw data common in real-world applications. Traditional ML algorithms often necessitate handcrafted features or representations. This approach can be labor-intensive and may fail to capture the most relevant information in the data.

In contrast, representation learning empowers machines to autonomously discern the most pertinent and discriminative features from the raw data, bypassing the need for human intervention. This is made possible through neural networks capable of hierarchically representing the data. This strategy is particularly effective when dealing with complex, multi-modal data prevalent in transportation systems. The multimodal data collected by diverse sensors can be harnessed to learn effective representations encapsulating key traffic pattern features. By utilizing varied data sources like loop detectors, images, videos, weather conditions, and attribute data, traffic managers can attain a more comprehensive understanding of traffic patterns, culminating in more accurate predictions and enhanced decision-making. In recent years, Representation learning has been used in transportation to perform tasks such as traffic prediction, routing optimization, and vehicle detection [22, 23]. By learning the underlying structure of the data, representation learning algorithms can extract and fuse more diverse and accurate insights from the heterogeneous data. These improvements can lead to more precise results, reduced emissions, and better overall transportation efficiency and reliability.

1.2 Challenges of Customizing Trustworthy Intelligent Transportation Systems

With the encouragement of urbanization, transportation agencies have been using various sensors to monitor newly constructed transportation facilities, including high-resolution cameras, LiDAR and radar sensors that require larger transmission bandwidth and more computing re-

sources to collect information. Despite the significant benefits that advancements of sensor and machine learning technologies have brought to traffic tasks in ITS, emerging challenges have hindered its next-step development and large-scale implementation, as previously highlighted in studies such as [24, 25, 26, 27]. Today, there is a growing demand for more detailed and diverse information for both the public and transportation agencies and roadway users.

Obviously, with the rapid expansion of the sensor network and increased sensing tasks, the centralized computing framework may not be able to keep up. Besides, the insufficient cross-sensor cooperation hugely limited the network-scale data collection and information estimation. Further, the machine learning community is also challenged by the problem of multimodality data format, unbalanced training data, the algorithms' reliability and robustness which can lead to poor generalization ability in transferring algorithms to real-world application scenarios. Additionally, ITS applications' data quality is not always stable due to adverse environmental conditions that affect traffic sensing. To address these issues, this dissertation aims to design customized representation learning algorithms that can be adapted to edge devices for ITS applications. Specifically, four challenges related to traffic perception, modeling, and applications for roadway infrastructures is involved.

1.2.1 Insufficient Multi-Sensor Cooperation

With the rapid development of video processing and communication technology, different types of sensors are widely deployed in the current ITS. However, current perception systems, e.g., the broad-coverage traffic surveillance systems, have not been fully exploited: the installed cameras are separated and only can extract information from their own Field of Views (FoVs). Although video processing methodologies have been well-investigated by previous researchers, most of

them focus on the traffic parameters estimation and video analysis from a single-camera input [28, 29, 30, 31]. In most cases, the cross-camera traffic information, i.e., link travel time, link speed, origination and destination distribution, still can not be obtained automatically. Manually extracting and summarizing them are time-consuming and labor-intensive for most agencies.

1.2.2 Limited MultiModality Data Feature Extraction and Fusion Approaches

Transportation data comes from various sources, including RGB cameras, radars, GPS location sensors, and even weather sensors. To effectively analyze this data, machine learning algorithms must be able to extract relevant features from multiple modalities and fuse them to make accurate predictions. A crucial requirement is to build a unified framework to handle various traffic data modalities. This is motivated by practical demands, as the types of traffic sensors used by the Department of Transportations contain a broad spectrum, including magnetic loops, surveillance cameras, on-board sensors in vehicles, and cell phone applications, among others, and they vary region by region. The collected data could contain traffic attributes, static road attributes, and beyond.

In general, the collected traffic data can be naturally divided into two types w.r.t. the data acquisition/storage modality: the GPS grid modality and the detector graph modality [32, 33]. The GPS grid modality data [34, 35, 36] mainly resort to the GPS signal of a large fleet of probe vehicles. It usually divides the city map into tractable regular grids, then summarizes the road and traffic attributes into each grid cell. On the other hand, in the detector graph modality data, the traffic flow parameters are mainly measured by the loop detectors, which are then summarized into a topological graph based on the road network connection [37, 38, 39]. Current solutions are only designed based on one specific type of modality, for example, the Spatial–

temporal Grid Model (SGM) are specialized in the GPS grid modality and the Spatial–temporal Graph Sequential Model (SGSM) are specialized in the detector graph modality. The attempt to simultaneously fuse and use two types of modalites has not been forthcoming. With the increasing complexity of measurable traffic parameters within one city, developing an all-inclusive model to support both data types will naturally reduce the system development labor, scales up the prediction in urban networks, and pave the way for future data fusion.

1.2.3 Discriminative Modeling Ability for Traffic Minorities

With the increasing application of deep learning technologies for taffic tasks, domain experts have raised several concerns related to model biases and effectiveness [40, 41]. Those concerns are especially crucial when applying traffic perception technologies [42, 43, 44]. Object recognition, one of the most fundamental traffic perception tasks, imposes the key challenges into following two perspectives:

Real world data are naturally imbalanced with sample bias. The complexity and variability of traffic systems present unique challenges, particularly when viewed through the lens of deep learning [45]. Within these systems, certain events occur with high frequency, such as vehicles stopping at red lights. These frequent events, happening predictably and consistently, contribute to the standard rhythms of urban traffic. Conversely, other events, such as traffic accidents, road closures, and instances of jaywalking, occur with far less predictability and frequency. Despite their irregular occurrence, these events can significantly impact the overall flow and efficiency of the traffic system when they do take place. This distribution of event frequency becomes particularly impactful when considering visual object recognition in traffic surveillance systems, which employ deep learning techniques. Neural networks in these systems have shown

impressive proficiency in accurately classifying common objects like pedestrians and vehicles, which frequently appear in the traffic environment. Nevertheless, these systems often struggle when it comes to less frequently seen objects or individuals, such as wheelchair users or other Vulnerable Road Users (VRU). This poses significant concerns regarding traffic equity, as the systems might not equally serve all road users. Further challenges arise from sample imbalances in both real-world scenarios and the training data used to 'teach' these surveillance systems. These imbalances can lead to the misclassification of unexpected objects, such as animals straying into the road. For example, a surveillance system might misidentify a bear or an elephant as a pedestrian or a vehicle due to their uncommon appearance in the traffic environment. Such misclassifications can lead to impractical results and even pose potential safety hazards [46]. Thus, addressing these issues remains a crucial challenge for researchers.

Deep-learning models are likely to amplify the biases present in the training data [47, 48, 49, 50]. Another challenge immediately arises – how does the neural network capture features from the imbalanced distributions? Will the initial sample bias of the dataset be amplified by the learning procedure? Unfortunately, several state-of-the-art (SOTA) methodologies pessimistically indicate that the performances of the neural network for the majorities in a training set always surpasses that of the minorities [51, 52]. Similar research conclusions can be found in many of the existing AI applications in traffic perceptions, including pedestrian and disability user recognition [44], vehicle and truck classification [53] and AVs' road user detection and classification [54]. The fascinating performances reported in the previous research heavily relies on the model performance for the majority class instead of the results for the minority class. However, in real-life situations, the ignorance to the tail classes can cause minority objects to be misclassified, which can cause detrimental equity and safety concerns that could result in lethal

consequences.

1.2.4 Limited Reliability & Robustness in Open-world Scenarios

The transportation tasks, e.g. video-based traffic perception, are always designed with sufficient training data. However, transportation is a diverse and plural society. Currently, the visual perception tasks using in real world, for both AVs and surveillance systems, are dedicated to some certain targets, i.e., traffic sign recognition, vehicle recognition and non-motorized users detection [13]. Mostly, the AVs can easily distinguish the traffic signs in the region where the real classes and distributions of signs are close to the training data. Meanwhile, some of the traffic signs in various regions and countries are different, i.e., the school zone alert sign can always be seen in downtown regions but they are infrequent in most of the rural areas, on the other hand, the alert signs for the existences of deer and bears can be easily found in rural regions instead of cities. In order to deploy Autonomous Vehicles (AVs) in real-world scenarios, it is crucial for the object recognition algorithms to be generalized enough to distinguish all entities on the roads, including signs, road users, and traffic facilities. If the neural network is trained to learn a specific distribution, such as signs in downtown areas, it might result in seemingly optimistic performance metrics (since the training and testing datasets are closely related), but the effectiveness and scalability of the algorithm could be compromised. Such an algorithm could face overfitting issues and may not perform adequately in other regions. Therefore, it is beneficial to combine several specific tasks with similar backbone architectures to address real-world challenges. The gaps identified above provide the motivation for this study to investigate more advanced technologies, including sensor fusion solutions to improve system robustness under challenging lighting conditions, along with novel methods to handle system reliability, especially

in uncommon situations.

1.3 Research Objectives

1.3.1 Definition of Trustworthiness in ITS

Trustworthy ML refers to the development and application of machine learning models that are reliable, transparent, interpretable, fair, and accountable [55, 56]. The idea is to create models that humans can trust to make decisions or take actions, particularly in high-stakes domains like healthcare, finance, or autonomous systems. Specifically, there are four key elements of trustworthy ML:

- **Explainability:** This refers to the ability to understand and interpret the internal workings of a machine learning model and how it makes its decisions or predictions.
- **Fairness:** This means that the ML model should not be biased or discriminatory, particularly with respect to sensitive attributes like race, gender, or age.
- **Privacy:** In the context of machine learning, this refers to the protection of sensitive data used to train the model or make predictions.
- **Robustness:** This refers to a model's ability to maintain performance when faced with adversarial inputs, changes in the input distribution, or other challenging conditions.

While the conventional evaluation criteria for Trustworthy ML – explainability, fairness, privacy, and robustness – are undeniably critical, they may not comprehensively encapsulate the unique needs and challenges inherent in ITS. ITS are characterized by their operation within

complex, dynamic real-world environments abundant with uncertainties. Consequently, simply ensuring the robustness of a model may be insufficient. Additional factors such as real-time adaptability, cooperative behaviors, and resilience to a plethora of traffic uncertainties warrant consideration.

To transform the notion of trustworthiness from AI methodology into a comprehensive, system-wide standards in traffic systems, this dissertation proposes a reimagining of the conventional understanding of trustworthiness. It advocates for a tripartite framework that aligns trustworthiness across three essential levels: the data, methodological, and system levels. When evaluating the trustworthiness of systems such in ITS, these three levels become crucial. Each level is associated with specific criteria, cumulatively forming a set of nine fundamental criteria 1.1. These criteria collectively provide a comprehensive framework for assessing trustworthiness across multiple aspects of the system, thereby facilitating a more nuanced and robust evaluation of its overall trustworthiness.

Data Level: This pertains to the quality and handling of data used in the ML model.

- **Unbiased data acquisition:** This ensures that the data collected accurately represents real-world conditions, without favoring any particular outcome. This is critical in transportation systems to ensure accurate predictions and decisions.
- **Privacy:** Refers to the safeguarding of personal information. In transportation systems, this might involve protecting the identities of individuals in video data, or anonymizing GPS data collected from vehicles.
- **Security:** This ensures that the data used and produced by the system is protected from unauthorized access, manipulation, or malicious attacks.

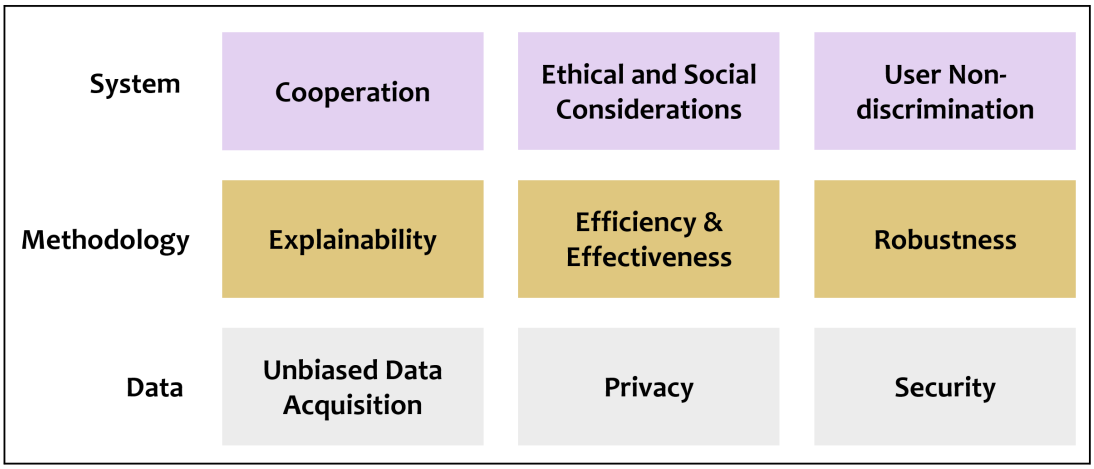


Figure 1.1: Proposed nine evaluation criteria of trustworthy ML and advanced computing methods for ITS.

Methodology Level: This involves the actual algorithms and models used for machine learning.

- **Explainability:** The decision-making process of the model should be understandable to humans. This is important in ITS to justify decisions like routing strategies, traffic light timings, etc.
- **Efficiency & Effectiveness:** The model's performance should be accurate and timely. For instance, a high-precision model could accurately predict traffic conditions or detect obstacles for autonomous vehicles in real-time manner.
- **Robustness:** The model should be able to handle variations or changes in the input data or environment. In ITS, a robust model might successfully handle adverse weather conditions, traffic variations, etc.

System Level: This considers how the model interacts with the broader transportation system and society.

- **Cooperation:** The system should work well with other systems and with human users. For instance, an autonomous vehicle should cooperate smoothly with human-driven vehicles and pedestrians.
- **Ethical and Social considerations:** The system should consider the potential impacts on society and ensure its usage is responsible. This might involve considering the system's impact on traffic congestion, pollution, accessibility, etc.
- **User non-discrimination:** The system should serve all users equitably, regardless of who they are. In transportation, this might mean ensuring that the system is accessible and useful for users of all demographics and abilities.

This nine-criteria framework provides a comprehensive way to evaluate and ensure the trustworthiness of machine learning and advanced computing applications in traffic systems, and they are also the key research targets link each chapters for this dissertation.

With the trustworthy ML and advanced computing methods, my general research target is to build future cooperative and equitable traffic infrastructure systems. As Figure 1.2, In this dissertation, the author merges the proposed principles of trustworthiness in ML with traffic tasks, thereby indicating that through the adaptation of novel methods, traffic systems can become more cooperative, equitable, accurate, and privacy-preserving. Specifically, the objects include improving the sensor cooperation for large-scale traffic perception, customizing representation learning for traffic pattern fusion and

prediction, enhancing the equity and robustness of connected and autonomous infrastructure systems, and promoting traffic infrastructure systems to a more scalable and cost-effective edge computing architecture. These goals coalesce towards the ultimate aim of constructing more intelligent, efficient, and equitable traffic infrastructure systems. Such advancements stand to substantially improve traffic mobility, safety, equity, and accessibility, thereby benefiting all road users.

1.3.2 Improving the System Cooperation for Large-Scale Traffic Perception

This research emphasizes building a cooperative perception system for large-sacle traffic data collection. The central goal is to devise algorithms that recognize and re-identify objects using multiple sensors across diverse locations with privacy-preserving manner. Faced with challenges such as fluctuating lighting conditions, vehicle occlusions, and differing driving orientations, the goal is to propose novel representation learning algorithms that consistently re-identify the same traffic objects across the sensor network by extracting all their distinctive features. By leveraging sensor and feature fusion methodologies, the overarching goal is to enhance the sensing scale, accuracy, and efficiency within ITS.

1.3.3 Mitigating the Perception System Bias and Enhancing the Accuracy and Reliability

As we steer towards the revolution of connected and autonomous infrastructure systems, there's an increasing emphasis on their robustness and bias. **The primary objective here is rectifying biases in traffic perception models empowered by ML.** A shift towards designing systems specifically for Vulnerable Road Users (VRU) is aimed at enhancing safety and equity. This

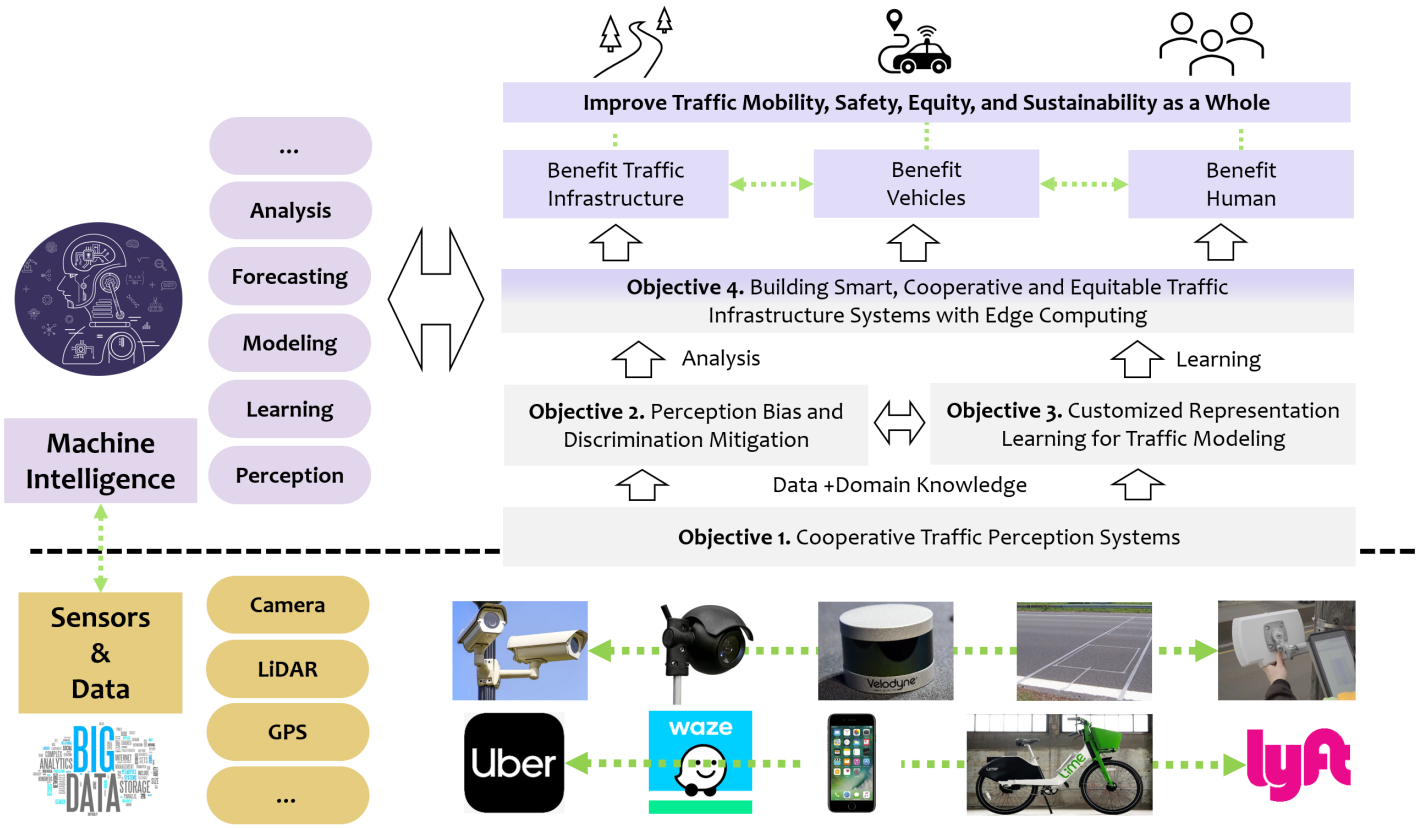


Figure 1.2: The illustration of research objectives and core steps in this dissertation.

involves:

- Harnessing new perception technologies for improved VRU detection, tracking and pose estimation.
- Analyzing system accessibility and affordability.
- Evaluating model fairness and potential biases.
- Collaborative initiatives with community stakeholders to ensure an non-discriminatory design.

1.3.4 Customizing Representation Learning for Traffic System Modeling

This research objective with a focus on spatial-temporal traffic input, are to develop and optimize algorithms that can effectively extract and fuse features from various data sources to accurately predict traffic patterns. The technical objectives include:

- Developing new representation learning methods that can effectively capture the complex spatial and temporal dependencies in traffic data. This will involve exploring the use of novel deep learning architectures, such as graph neural networks and attention-based models, that can capture the relationships between spatial and temporal features.
- Investigating the impact of incorporating external factors, such as weather data and social events, on traffic pattern prediction. This will involve developing new fusion methods that can integrate data from multiple sources, including traffic cameras, GPS data, and external data sources, to improve the accuracy of traffic pattern prediction.

- Customizing the representation learning process to the specific characteristics of traffic data, such as traffic patterns during rush hours and seasonal variations. This will involve developing new techniques for adapting the representation learning process to the specific characteristics of traffic data and investigating the impact of these techniques on traffic pattern prediction accuracy.

- Developing new methods for uncertainty quantification in traffic pattern prediction. This will involve exploring techniques for estimating uncertainty in model predictions and using this information to inform decision-making processes.

1.3.5 Promoting the Connected and Autonomous Traffic Infrastructure Systems to Edge Computing Architecture

Considering the pivotal role of efficient traffic infrastructure systems, this research emphasizes leveraging the Internet of Things (IoT) capabilities. The focus lies in traffic sensing, data processing, and communication. The main objectives encompass:

- Investigating efficient methodologies for traffic sensing and processing via edge devices.

- Weighing the merits of edge computing in traffic systems against established centralized models.

- Crafting strategies for optimal resource allocation in edge-centric structures, ensuring reliable and efficient traffic systems.

- Seamlessly merging new edge-based systems with the existing infrastructure.

1.4 Dissertation Organization

This dissertation investigates three key tasks for building novel infrastructures, including perception, operation, and building real demo systems. The contributions can be divided into three finished components: **1) proposing novel multimodal data representation learning system for cooperative traffic perception. 2) Learning multimodal data representations to improving traffic operations for roadway users and managers. 3) Demonstrating pilot cooperative and equitable infrastructure systems for mobility, safety and equity enhancement.** The overall framework is depicted in Figure 1.3, which consists of nine chapters, including the introduction and literature review. In Figure 1.3, the purple blocks indicate the key work theme of the dissertation, and the gold blocks represent the completed works. The figure also includes the chapter number and topic. A summary of each chapter, except for the first, is provided below.

Chapter 2 Literature Review Representation learning is a machine learning technique that involves automatically learning useful representations or features from raw data, which can then be used for various downstream tasks such as classification, clustering, and prediction. This technique has gained prominence in transportation, aiding in traffic prediction, object detection, and other tasks. This chapter delves into representation learning in transportation, touching on spatial-temporal data modeling, visual & LiDAR-based traffic object recognition, cooperative perception visuals, multi-modality data for unbalanced distributions, and edge AI in ITS.

Part I. Learning Representations for Cooperative Traffic Perception

Chapter 3 Cooperative Multi-camera Vehicle Tracking and Traffic Surveillance with Distributed Edge Artificial Intelligence. To track vehicles across multiple cameras and help pub-

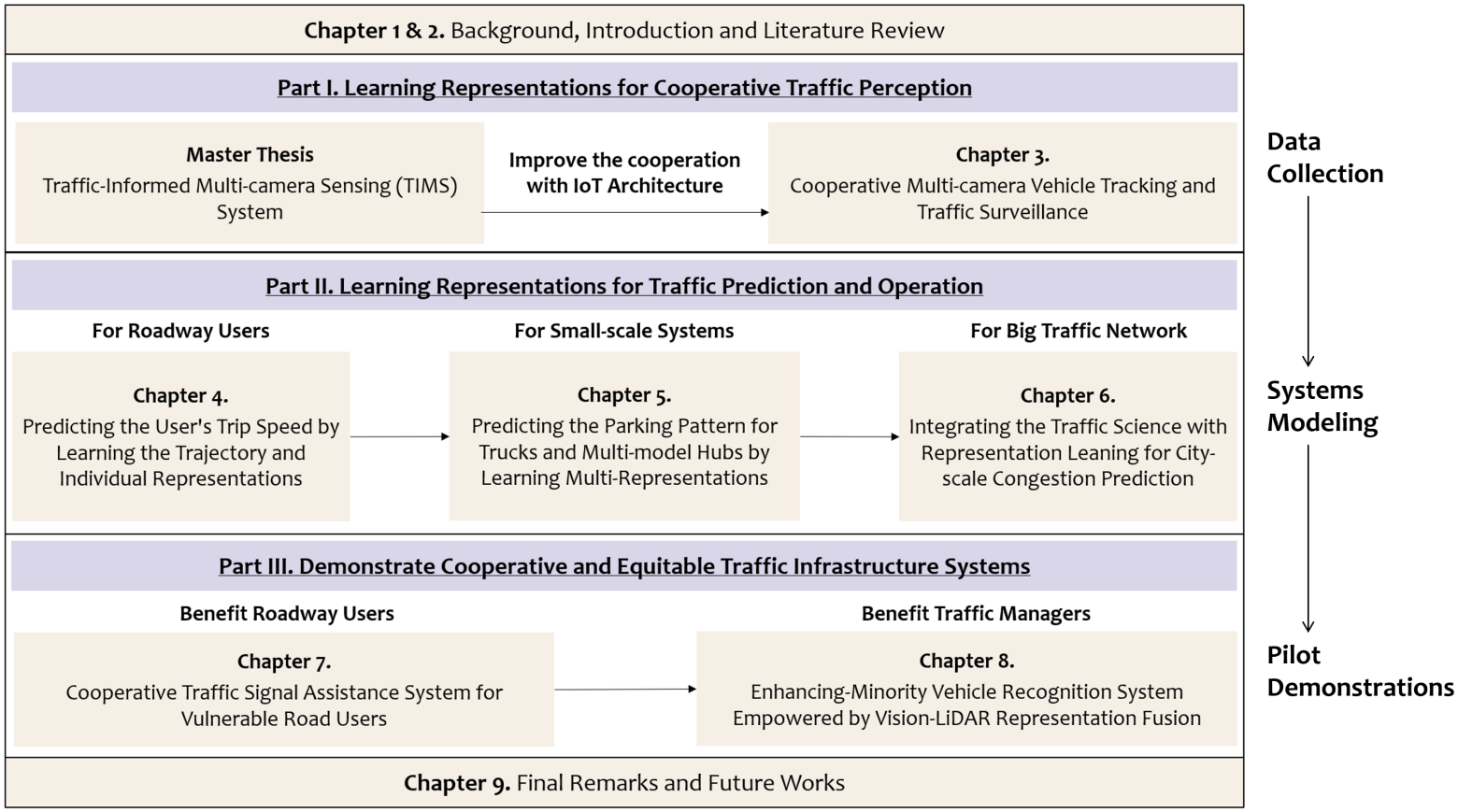


Figure 1.3: The illustration of the research topics and logic organization in the dissertation.

lic agencies collect link travel time and speed information, an Edge-empowered Cooperative Multi-camera Sensing (ECoMS) System is proposed. ECoMS system presents a novel algorithmic and edge-server cooperative system construct to push edge computing and multi-camera re-identification workflow serving for traffic sensing based on Internet of Things (IoT) architecture. On the algorithm side, ECoMS system proposes a light-weight edge-based computer vision framework for vehicle detection, tracking, and features selection process in a real-time manner. Then, by only sending the objects' representations to the server, the high-bandwidth data transmission and the heavy post-processing system can be abandoned. Furthermore, a hierarchical clip-based deep vehicle re-identification framework is proposed and integrated into the ECoMS system, and significantly outperforms other state-of-the-art methods by 4%-8% on Rank-1 accuracy. Finally, to balance the accuracy level of different camera pairs, a collaborative cross-camera traffic information estimation framework based on kernel density estimation with kernel smoother is implemented, which can get the precise and reliable link and region traffic information and distributions (less than 1.01 KL distance). By maximizing the cooperation of the computational resources, orchestrating the data transmission and integrating road network graph features in the system, the ECoMS can precisely model the network-scale traffic information in a flexible, cost-effective, and scalable workflow. To the author's best knowledge, ECoMS is the first multi-camera vehicle tracking and traffic monitoring system based on IoT architecture.

Part II. Learning Multimodality Data Representations for Traffic Prediction and Operation

Chapter 4 Predicting the User's Customized Trip Speed by Learning the Trajectory and Individual Attributes Representations. Customized path-based speed prediction is an eventful tool for congestion avoidance, route optimization and travel time prediction for navigation apps, cab-hailing companies and autonomous vehicles. However, the path-based speed prediction is very challenging since the speed is always changing in different path locations and is jointly affected by lots of complicated factors. This chapter presents a novel deep learning framework for customized path-based speed prediction. A Path-based Speed Prediction Neural Network (PSPNN) is designed to achieve speed predictions for a given path and attributes information. A hierarchical Convolutional Neural Network (CNN) and deep Bidirectional Long Short-Term Memory (Bi-LSTM) structure for different kinds of feature extraction are applied for multiple levels: the path cell, sub-path and the whole path. The method narrows down the prediction unit from road segments to customized path cells (mean length: 59.52m) and achieves a mean absolute error (MAE) of 1.94 m/s and Mean Absolute Percentage Error (MAPE) of 18.14%, showing the potential of serving rigorous data-driven applications. So far, PSPNN is the first made-to-order path-based speed prediction algorithm and can help both travelers and managers to obtain large-scale bespoke paths speed information in advance.

Chapter 5 Predicting the Parking Pattern for Trucks and Multi-model Hubs by Learning Attributes and Spatial-temporal Representations. Managing and estimating the availability information of parking lots is of great importance to travelers and managers. However, the task is very challenging since the occupancy rate is affected by various factors, including spatial-temporal features, parking lot attributes features, and environmental changes. Previous

studies mostly focus on the short-term prediction by capturing the historical sequential dependencies among inputs and outputs, which leads to low estimation accuracy for long-term prediction and limited scalability for real deployment in parking lots. To address the challenges, a comprehensive framework for real-time Smart Parking Information Management and Prediction (SPIMP) System is proposed in this chapter. Three types of data sources, including historical sequential data, real-time sequential data, and attributes category data, are sufficiently integrated into a customized Parking Availability Prediction (PAP) neural network by representation learning and heterogeneous feature embedding. Specifically, instead of using parking lot property information and environmental data directly, the authors design an Attributes Tensor Embedding Component (ATEC) to integrate the intra-class affection and inter-class representations and correlations by two steps of customized embedding process. To balance the impact of the indiscriminate features for various prediction targets, this study proposes a multi-factor attention mechanism to learn the weights and help the PAP achieve a stable performance for time-various tasks. From extensive experiments on two real-world large-scale datasets collected in China and USA (including 19 urban parking and 2 truck parking lots), PAP achieves superior prediction results on both urban parking (6.69% average MAPE) and truck parking prediction (7.63% average MAPE) for five prediction time slots (5min, 10min, 30min, 1h and 2h). Furthermore, even with limited training data, PAP still shows better transferability and adaptability for various types of lots. The proposed SPIMP is selected and used as a pilot test bed by the Washington State Department of Transportation (WSDOT) for truck parking management.

Chapter 6 Integrating the Traffic Science with Representation Leaning for City-scale Network Congestion Prediction. Recent studies on traffic congestion forecasting have paved a promising path toward the reduction of potential economic loss. However, at the city-wide

scale, current approaches face substantial hurdles, such as being unable to support the multiple sensors modalities, insufficient congestion fluctuation and propagation modeling, and weak generalization to heterogeneous traffic network structures. To address these pitfalls, this chapter investigates how to integrate the missing urban planning domain priors into a general sequence prediction model, and proposes the Traffic-informed Transformer (TinT). To prevent receptive field bias, a novel mixture of long and short range information routing mechanism is proposed with the traffic-informed tokenization. To capture the unbalanced traffic flow propagation, an original anisotropic graph aggregation is developed to differentiate the traffic fluctuation based on orientations. Extensive results demonstrated TinT's outstanding performance over other state-of-the-art models and its broad applicability to multiple data modalities in six well-known cities.

Part III. Demonstrating Pilot Cooperative and Equitable Traffic Infrastructure Systems

Chapter 7 Cooperative Traffic Signal Assistance System for VRU. To serve the users in an unbiased and automated way, a novel cooperated signal phase and timing (SPaT) services infrastructure – Vision Enhanced Non-motorized Users Services (VENUS) smart node is proposed. With customized up-to-date computer vision algorithms and an artificial intelligence pipelines on the edge, VENUS smart node can collect necessary active-user information (including location, class, pose direction and mobility status), and generate directional crossing request for every pedestrian and cyclist in real time. Meanwhile, the improved communication system makes the VENUS node a reliable information hub to share the SPaT messages and carry interactions to/from the signal controller, connected vehicles and user personal information devices (i.e., cell

phones, wearable devices) through various protocols. Based on extensive experimentation, 1076 testing users from six intersections, the VENUS sensing achieves 90.24% accuracy on directional-aware crossing trigger generation and 89.87% accuracy on mobility status estimation for normal users and four types of disabled persons. Furthermore, the VENUS smart node is fully compatible with the connected vehicles environment, and improves the signal system at low cost, mainly due to its flexibility and adaptability with existing infrastructure. The VENUS smart node is the first connected infrastructure architecture that integrates traffic sensing, data processing and information dissemination together for the self-operating indistinguishable signal services based on edge computing.

Chapter 8 Enhancing-Minority Vehicle Recognition System Empowered by Vision-LiDAR Representation Fusion and Edge AI. Leveraging micro high-speed pulse LiDAR mounted overhead of travel lanes, this chapter proposes **C**ompact **L**iDAR **E**mpowered **V**ehicle **E**nhancing-minority **R**ecognition (**CLEVER**) system, a real-time cost-effective vehicle detection and classification framework that is empowered by edge Artificial Intelligence (AI). Based on the customized minority-enhancing vehicle classification deep neural network, the CLEVER system outperforms cutting-edge LiDAR-based vehicle classification methods up to 15.98% true-positive rate in classifying ten types of vehicles. Furthermore, by highly integrating the hardware, the pre-processing algorithm and the classification neural network into an edge computing node, the CLEVER system only consumes 10% of the cost, 10% to 20% of energy compared to other scanning LiDAR systems and works perfectly in a plug-and-play mode with a negligible sub-second inference time (212ms to 459ms). The proposed CLEVER system offers an affordable end-to-end solution that can benefit traffic operators by collecting more accurate and reliable vehicle classification data streams and that can lead to a more efficient and flexible ITS.

Chapter 9 Final Remarks and Envisioning the Future.

2

Chapter 2. Literature Review

2.1 Representation Learning Overview

Representation learning techniques can be broadly categorized into two types: unsupervised and supervised learning [23, 57]. In unsupervised learning, the algorithm learns to extract features from raw data without the need for labeled data. Autoencoders are a common type of unsupervised learning algorithm that learn to encode input data into a lower-dimensional representation and then decode it back to the original input [58]. In supervised learning, the algorithm

learns to map input data to output labels based on a labeled training dataset. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are commonly used supervised learning algorithms that are particularly effective for image and sequence data, respectively [59, 60].

Representation learning boasts several advantages, making it a cornerstone for tasks rooted in learning-based paradigms. Firstly, it acts as a catalyst in amplifying model accuracy and in mitigating overfitting issues [61]. This enhancement arises from the innate proficiency of representation learning algorithms to autonomously navigate through vast data expanses, singling out and extracting the most salient features. These are not just any features, but ones that have been distilled from the data. Once identified, these features become instrumental in amplifying the precision of tasks they're associated with. Secondly, representation learning doesn't just improve models – it makes them more transparent. It somehow boosts the interpretability of machine learning models [62]. By translating raw data into meaningful and structured feature representations, it sheds light on the often convoluted decision-making process of models, clarifying the logic that steers a model's predictions [63]. This attribute of 'transparency' is not just a luxury but a necessity, especially in sectors like safe-related decision making where a wrong prediction can be life-altering. Lastly, the benefits of representation learning extend to the robustness and versatility of models. It carves out pathways for models that are not just effective but are resilient and adaptive [64]. By training models to recognize and use features that have broad applicability across diverse tasks and datasets, representation learning ensures these models are not pigeonholed into narrow functionalities. Instead, they can be generalized and adapt to new, previously unseen data with efficacy. This adaptability becomes the bedrock of success in domains where data might be at a premium or in situations demanding rapid model recali-

bration to cater to evolving scenarios [58].

Despite the many advantages of representation learning, there are also several disadvantages associated with this technique. One major challenge is the need for large amounts of labeled data, particularly in supervised learning. Without sufficient labeled data, representation learning algorithms may not be able to learn meaningful features. Another challenge is the difficulty of interpreting learned representations [65]. While representation learning can improve model interpretability, the learned features themselves may not be easily interpretable by humans. This can make it difficult to understand why a model is making certain predictions or to diagnose model errors [66]. Finally, representation learning can be computationally expensive, particularly when dealing with large datasets or complex models. This can make it difficult to scale representation learning algorithms to real-world applications.

In this chapter, the discussion will be focused on the overview and mechanism of representation learning and their applications in transportation, including the representation learning for spatial-temporal data modeling and forecasting, learning visual & LiDAR representations for traffic object recognition, learning visual representations for cooperative perception and learning multi-Modality data representations for unbalanced data distributions, together with learning representations on distributed edge devices in ITS.

2.2 Learning Spatio-temporal Representations for Traffic Forecasting

Spatial-temporal traffic pattern learning and forecasting are critical tasks in transportation management and planning. Accurately forecasting traffic conditions can help reduce congestion, optimize routes, and improve safety [67, 68]. With the advent of smart city technologies, there is an abundance of data available that can be leveraged to develop accurate traffic prediction models.

Spatial-temporal data, which describes the location and movement of objects over time, is particularly useful for this task [69]. However, modeling this type of data is challenging due to its high dimensionality and complex dependencies. Representation learning, a technique for automatically learning useful features from data, has shown promise in improving spatial-temporal data modeling. In this section, the use of representation learning for spatial-temporal data modeling, focusing on its application to traffic prediction will be further discussed.

2.2.1 Traffic Parameters Prediction by Representation Leaning

Traffic parameters prediction is a challenging problem due to the complex impacts of traffic flow [69]. Traffic patterns are affected by a wide range of factors such as road network topology, weather conditions, time of day, and events such as accidents or road closures [70]. To accurately predict traffic, it is necessary to model the complex relationships between these factors. Traditional approaches to traffic prediction have relied on statistical models such as autoregressive integrated moving average (ARIMA) or regression-based models such as support vector regression (SVR) [71]. While these methods can be effective in some cases, they have limitations in capturing the complex spatial-temporal patterns of traffic flow. Representation learning has emerged as a promising technique for traffic prediction as it can capture the complex spatiotemporal patterns in traffic data. In recent years, a variety of representation learning methods have been developed specifically for spatiotemporal data modeling [70, 72].

CNNs have been widely used in computer vision tasks such as image classification, object detection, and segmentation [73, 74]. CNNs are also well-suited for spatiotemporal data modeling as they can capture spatial patterns across multiple time steps. In the context of traffic prediction, CNNs can be used to extract spatial features from traffic flow data, which can then

be used to predict future traffic patterns. One popular approach to using CNNs for traffic prediction is to use a 3D CNN, which operates on 3D spatiotemporal data (x, y, time) [75, 76, 77]. 3D CNNs have been shown to be effective in capturing the complex spatiotemporal patterns in traffic data. For example, in a recent study, a 3D CNN was used to predict traffic flow in the city of Hangzhou, China. The results showed that the 3D CNN outperformed traditional methods such as ARIMA and SVR.

RNNs are another popular ML method for extracting the spatiotemporal data representations [78, 79]. RNNs are well-suited for modeling temporal sequences as they can capture the dynamics of time-series data. In the context of traffic prediction, RNNs can be used to model the temporal dependencies between traffic flow data across multiple time steps.One popular approach to using RNNs for traffic prediction is to use a Long Short-Term Memory (LSTM) network, which is a type of RNN that is designed to capture long-term dependencies in time-series data [27, 80]. LSTMs have been shown to be effective in capturing the complex temporal patterns in traffic data. For example, in a recent study, an LSTM was used to predict traffic flow in the city of Los Angeles, USA. The results showed that the LSTM outperformed traditional methods such as ARIMA and SVR.

Graph Neural Networks (GNNs) have shown great potential for learning spatial-temporal patterns in traffic data modeling. GNNs can represent traffic data as graphs, where nodes are locations and edges are connections between locations. GNNs can learn spatial relationships between locations and incorporate temporal information to make predictions. GNNs can also handle missing data, noise, and irregular data structures. One example of using GNNs for traffic prediction is STGCN (Spatial-Temporal Graph Convolutional Network) proposed by Yan et al. (2018) [81]. STGCN uses a GNN to learn spatial relationships between traffic sensor

locations and temporal patterns in traffic data. The GNN consists of several layers of spatial-temporal convolutional filters that operate on a graph representation of traffic data. STGCN can make accurate predictions for multiple traffic parameters, such as traffic volume, speed, and occupancy. Another example is AGCN (Adaptive Graph Convolutional Network) proposed by Li et al. (2018) [82]. GCRNN is designed to model the spatial-temporal dependencies in traffic data using a GNN with RNNs. GCRNN can handle irregularly spaced and missing data and achieve state-of-the-art performance in traffic prediction tasks.

Autoencoders are unsupervised neural networks that can learn compressed representations of data. Autoencoders consist of an encoder that maps input data to a low-dimensional representation, and a decoder that maps the low-dimensional representation back to the original data [83]. Autoencoders can be used for feature extraction, data denoising, and dimensionality reduction. One application of autoencoders in traffic prediction is the stacked autoencoder Levenberg-Marquardt model proposed by Yang et al. (2016) [83]. The stacked autoencoder Levenberg-Marquardt model uses a convolutional autoencoder to learn spatial-temporal patterns in traffic data. Besides, the proposed can encode the input traffic data into a low-dimensional representation and decode the representation to make traffic predictions. In 2020, the variational autoencoder (VAE) is proposed by Boquet et al., which can handle missing data, noise, and irregular data structures and achieve state-of-the-art performance in traffic prediction tasks [84].

In addition to the above studies, there have been many other studies exploring the use of representation learning for traffic prediction. These studies include deep learning-based methods such as autoencoders [85], variational autoencoders [84], and generative adversarial networks [86], as well as graph-based methods such as graph convolutional networks [87] and graph attention networks [88, 17]. These studies demonstrate the great potential of representation learning

in improving the performance of traffic prediction in ITS. Liao et al. proposed a deep spatiotemporal residual network (DSRN) for traffic prediction, which combines CNNs and RNNs to capture both spatial and temporal features of traffic data [89]. The DSRN model was evaluated on real-world traffic datasets and achieved better prediction accuracy than traditional time series models.

In summary, representation learning is a powerful approach for spatial-temporal data modeling, especially for traffic prediction under complicated impacts. Various deep learning techniques have been developed to learn meaningful representations of traffic data, including CNN, RNN, GNN, and autoencoders. These techniques can handle the complexities of traffic data, such as spatial relationships, temporal dependencies, missing data, noise, and irregular data structures. By learning informative representations of traffic data, deep learning models can make accurate predictions for various traffic parameters, such as traffic volume, speed, and occupancy. However, there are still challenges in applying representation learning to traffic pattern learning and forecasting, such as data scarcity, high-dimensional data, and model interpretability. Future research can focus on developing more efficient and interpretable models for traffic prediction.

2.2.2 Parking Pattern Prediction by Leaning Representations

Based on the literature review, the parking prediction methodologies based on machine learning have been made great progress. Combining the characteristics and skills of the neural network with the features and challenges of the parking perdition problem, several customized neural network is designed to complete the forecasting task. Generally, current researches mainly use existing machine learning models, i.e, CNN [90, 91], RNN [92, 93, 94, 95], and made some improvements to adopt to the real parking challenges. These models always include several features

extraction parts: spatial features extraction, temporal features extraction, and attribute information. Some of them also consider the traffic parameters as part of the input, but not well-used [94]. The output of the model is usually sequence-based occupancy level. There is no doubt that these approaches have achieved impressive prediction results.

Among all the state-of-art methods, several of them are worth discussion in detial. Zhang et.al. (2020) [90] proposed a Semi-supervised Hierarchical Re-current Graph Neural Network (SHARE) in the top artificial intelligence conference (AAAI-20). They designed three features extraction modules, including spatial auto-correlation, dynamic temporal features and the scarcity of information about real-time parking availability and integrated these three features set into the SHARE. Then, SHARE achieved impressive result for city-wide parking availability prediction on two real dataset. Another research group proposed a large-scale urban parking prediction methods based on Graph-Convolutional Neural Networks (GCNN) and stacked LSTM [94], which had the ability to incorporate the traffic speed and weather condition, achieved testing MAPE of 10.6% when predicting block-level parking occupancy for 30 minutes advanced period. Also in 2019, a research group in Google proposed a parking difficulty estimation methodology for parking prediction [96]. The original idea is transform the continuous occupancy value and other category and app information into probability distributions representing the parking difficulties. A customized reward matrix and a two layer feed-forward deep neural network model is implemented to finish the classification task. This work shows impressive adaptability in many cities and inspired the researchers a lot. Instead of using the sequential data to train the prediction network directly, an simple representation features transfer procedure is used here. Only a simple procedure can make the model achieve both good prediction result as well as stronger transferability.

As previous research mentioned [94, 92], parking availability information is not only closely related with historical pattern, the attributes information and the variance of prediction time slots (i.e., 5 minutes ahead and 2 hours ahead) also shows significant impact to the utility level. So, another key issues worth to review is current hybridizing attention and LSTM networks combination for sequential data related research. In general, the attention architecture combined with RNN can be divided into two tasks: sequence data classification and prediction. For the sequential classification, attention approaches, and other feature representation extraction models are integrated into the sequential neural network models for global pattern learning, including linear-attention mechanism [97, 98, 99], multi-head attention [100, 101], soft attention [102], gating mechanism [103, 104], Fully Convolutional Network (FCN) [105] and the hybrid approaches [106, 107]. Furthermore, the previous research successfully indelicate the mentioned attention combination with RNN structure can significantly improve the global features extraction. However, for the sequential prediction tasks, limited research have been published, and most of them are using basic attention model in a brute-force way, i.e., adding an attention layer after the input for filtering the low impact sequence record [99]. In general, The classification tasks always more rely on universal patterns and show less dependency on the local sequential correlations [108, 69], while prediction is rely on both features, and even more impact on historical dependency. So in this research, a new parking oriented, especially with the potential for involving attributes information impacts on multi-timescale prediction task is necessity.

However, even the some of the parking prediction methods have been investigated be researchers, challenges are still obvious. Generally, parking prediction are often affected by three types of information, spatial and temporal features, attributes features and environmental fea-

tures. Current cutting-edge parking prediction are mainly used in the urban area, which they pay more attention to integrated the spatial-temporal features [90, 95]. Also, at present, without the attributes pattern integration, parking prediction methods overemphasize short-term feature extraction (i,e, 5 minutes or 10 minutes ahead) and ignore long-term prediction, which is not suitable for general use of parking prediction and management. To proposed a method for general parking utility prediction, the attributes information, especially the long term pattern (workday or weekend, night time or day time, weather condition) significantly impact the prediction result [109, 110, 90]. How to effectively integrated the heterogeneous information sources, including spatial-temporal features, the attribute features and environmental information, which becomes a new topic for neural network design and algorithm development.

2.3 Learning Visual & LiDAR Representations for Traffic Object Recognition

Traffic perception is one of the most critical tasks in Intelligent Transportation Systems (ITS). It involves analyzing the traffic data collected by various sensors and cameras to obtain real-time traffic status information from stationary and mobile platforms. In recent years, there has been a surge in research on representation learning and computer vision techniques for traffic perception, including video-based object detection, tracking, and crowd monitoring. Besides the video sensors, the LiDAR is also being implemented by many agencies to perception the 3-D information. This section aims to provide a comprehensive review of the recent developments in learning visual and LiDAR representations for traffic perception, including the state-of-the-art models used, datasets, evaluation metrics, and success applications.

2.3.1 Learning Visual Representations for Traffic Object Recognition

Vision-based object recognition has always been a critical topic in the traffic research community. The recognition task can be easily defined as a classification task [13], and the visual inputs can be an on-board (moving) camera or surveillance (static) camera [13, 111]. The popular applications can be found almost everywhere in the ITSs and AVs, which include vehicle recognition [42], traffic facility and sign recognition [51, 112], user type recognition [113], road surface detection [114] and etc.

Currently, the vision-based object recognition task is always modeled by supervised learning process. The key target here is to learn a mapping function that analyzes the training data and produces an inferred mapping, which can then be used for matching new examples in real life. The traffic-related classification methods can be divided into traditional machine-learning approaches and deep-learning approaches. The machine-learning approaches include support vector machines (SVMs) [115], k-nearest neighbors algorithm (k-NN) [116], decision trees (DT) and random forest (RF) [117], and all of them elucidate improvements to the computational scalability [118, 119], reduced sizing for training data [119], easy handling of missing values [120], robustness to outliers in input spaces (except SVM) [121], and good interpretability (except SVM) [121]. Due to the improvements outlined above, the classical machine-learning methods have been widely tested, and obvious disadvantages are found including 1) low accuracy levels, 2) poor ability to extract linear combinations and non-linear correlations of features [122] and 3) limited capacity to deal with high-dimensional features [122].

With the development of deep learning methodologies and improvement in computational power, CNNs architecture present remarkable advantages for visual recognition tasks, not only in the traffic research community. Back in 2012, Krizhevsky, Sutskever, and Hinton proposed

the AlexNet [123], which can classify images into 1000 object categories. The original AlexNet contained eight layers, with five convolutional layers (the first, second and the fifth were combined with max-pooling layers) and three fully connected layers. Furthermore, the AlexNet was also the first neural network trained by GPUs, which later inspired more researchers to employ the deep neural networks and GPUs to accelerate deep learning. After AlexNet and CNN-based architecture proclaimed promising results for image classification tasks, the golden age of deep learning began. In the following years, several well-known CNN-based deep neural networks have been suggested, including Visual Geometry Group (VGG) [124], Residual neural network (ResNet) [125], MobileNet [126], EfficientNet [127] and etc. Cascading the downstream tasks based on the backbones' outputs becomes popular for various traffic application.

With the powerful deep neural network backbones, the features contained in the images are capable of being extracted. Therefore, the downstream traffic tasks based on image and video inputs have become a main focus for researchers. These tasks include vehicle classification [128, 43, 129], traffic light recognition and status classification [130], road user classification [44] and sign recognition [131]. It is evident that the backbones used to complete downstream tasks, especially for classification, tend to have high accuracy for categories with a plethora of training samples. Although many experiments have achieved approximately 100% accuracy on some datasets, those algorithms are not effective for many minority targets, i.e, the non-motorized users [42]. Since minority classes have fewer tests and images, many researchers held on to common categories in the training and testing set to achieve higher accuracy. Meanwhile, some studies identified the challenges and attempted to use basic techniques, i.e., balance data augmentation or two phase-training to improve the accuracy for the minority classes. However, in general, the previous researchers did not have a deep understanding of the data imbalance in the

real world, and without optimizations or improvements for the minority class, an increased bias in the backbones is inevitable. To achieve an equal and effective traffic perception for all road users, the debiased research for current deep learning schemes is necessary.

2.3.2 Learning LiDAR Representations for Traffic Object Recognition

LiDAR Sensors and Technologies

Based on the operating mechanisms, LiDAR can be generally divided into two types: continuous-wave and pulse-based [132, 132]. Continuous-wave LiDAR calculates the phase difference between the reflected light and the reflected light to calculate the target distance, achieving ultra-precise results, e.g., error less than 0.1mm [133]. To avoid multiple matches in one phase, measurement frequency needs to be low enough for phase distinguishing (always less than 50Hz) and thus this type of LiDAR is suitable to scan stationary or slow-moving objects [133, 134]. Pulse LiDAR sends out laser pulses at a high frequency and then detects corresponding reflective information [135, 132]. By contrast, pulse LiDAR controls pulse frequencies and the pulse energy is relatively concentrated in space. Therefore, pulse LiDAR is able to detect fast-moving objects even in challenging conditions (rainy or snowy conditions), reaching ultrahigh detection frequency (up to 10kHz), fast post-processing time and reliable accuracy (less than 1% error) [132].

2.3.3 Classification Systems and Methodologies Using LiDAR

According to the application scenario, the use of LiDAR in traffic sensing can be divided into two categories [132]: vehicle-mounted and infrastructure-mounted. For LiDAR installed on vehicles, related research investigates 3D object detection [144], object tracking [145], collision

Table 2.1: Roadside LiDAR-based vehicle classification frameworks

Framework Component	Classification Methodology	Venue	Year	Ref
The AUTOSENSE II laser range image system (manufactured by Schwartz Electro-Optics, with a scan rate of 720 lines per second).	Pre-defined rule-based classification. Using height, width, length, and other sensing features for vehicle classification.	TR_C	2001	[136]
Two side-fire LiDAR sensors (unknown brand, 37Hz scanning frequency) were each mounted at a height of about 6.7 ft above the ground and are 4.6 ft apart, which can scan a vertical plane across the roadway.	Decision tree with pre-defined rules for classification. Using height, width, length, and other sensing features for vehicle classification.	TRR	2012	[137]
A pair of Sick LMS 291-S05 scanning laser rangefinders, acting as vertical planar scanners, mounted approximately 2.2 m above ground, at an update rate of approximately 37Hz.	Pre-defined rule-based classification.	J. Intell. Transp. Syst	2015	[138]
Two VLP-16 LiDAR (16 beams, 30° view angle with 360° rotation) sensors manufactured by Velodyne was installed on top of the pedestrian signal head at a corner of an intersection.	Naïve Bayes, K-nearest neighbor, decision tree, support vector machine, and random forest (with the best performance) were included in this research.	TRR	2019	[139]
Two side-fire Lidar units are deployed on the road shoulder. Each Lidar unit emits a single laser beam perpendicular to traffic.	This research is for truck body classification. Methods included Naïve Bayes, decision tree, support vector machine, probabilistic neural network	TRR	2019	[140]
One VLP-16 LiDAR (16 beams, 30° view angle, adjustable 5-20Hz, 360° rotation) unit manufactured by Velodyne is used, installed at approximately 2.8 meters above the ground	density-based spatial clustering of applications with noise (DBSCAN) was used for object classification.	TRR	2019	[141]
Three VLP-16 LiDAR (16 beams, 30° view angle, adjustable 5-20Hz, 360° rotation) unit manufactured by Velodyne is used, installed 1.83 m above the ground without inclination.	Three fully connected layers back propagation neural network for two classes classification: pedestrian and vehicle.	TR_C	2019	[142]
One VLP-16 LIDAR (16 beams, 30° view angle, 360° rotation, set at 10Hz) unit manufactured by Velodyne is used. The sensor was installed at 6.71 m above ground and 6.10 m away from the lane.	This research is for truck body classification. Methods included K-nearest neighbors, adaptive boosting, multilayer perceptron, and support vector machine (with the best performance)	J. Intell. Transp. Syst	2020	[143]

warning [146], 3D view segmentation [147], etc. The purpose of the first category is for autonomous vehicles to obtain accurate and detailed environmental information for vehicle control. The second category focuses on identifying and detecting vehicles passing by fixed infrastructures, in order to generate traffic flow information for traffic operators. This paper summarizes a list of related studies in Table 2.1, including their LiDAR system information, classification methodology, venue, and publication year.

The earliest research that used LiDAR sensors to classify vehicles can be traced back to 2001. Harlow and Peng proposed a rule-based vehicle classification method based on the laser range image system at the time [136]. Due to technological and price constraints, this system was difficult to deploy, but it fully validated the possibility of classifying vehicles based on LiDAR sensing. For the first time, Lee and Coifman used LiDAR to classify vehicles in the form of side-fire [137]. This system used two LiDAR sensors, each scanning the road at 37Hz with 360° rotation. The vehicle information was extracted from the obtained 3D point cloud, and vehicles were classified by pre-defined rules. From the experiment, the rule-based classification method and system had good accuracies (75.6% on motorcycles and 99.8% on passenger vehicles), and researchers used it to validate the performance of vehicle classification stations. However, the limitations of this rule-based classification system were also very straightforward. The classification accuracies heavily relied on the rules customized for pre-defined scenarios, and thus the parameters can not be easily generalized. The expensive hardware, brute-force classification method, and limited classification categories significantly restrict its scalability.

With the development of LiDAR technologies and the advent of AI algorithms, vehicle recognition using LiDAR data is closely combined with machine learning classifications methodologies. LiDAR sensors with more beams, longer detection range, faster scanning speed, and ad-

justable frequency are used for vehicle detection and classification. Meanwhile, learning-based methods, including decision trees, Density-based Spatial Clustering of Applications with Noise (DBSCAN), k-Nearest Neighbor (k-NN), Support Vector Machine (SVM), and neural networks are implemented into the vehicle classification system based on LiDAR input [139, 140, 141, 142, 143].

In spite of their capacities, current SOTA systems are facing various challenges. Table 2.1 shows the VLP-16 LiDAR is the most popular sensor in the research community. Even researchers claim this LiDAR sensor is much cheaper than other models. Two thousand dollars per unit without post-processing system significantly limits its real-world deployment [139], especially in small municipalities with limited budgets. Previous research always used two or more LiDAR sensors in their experiments to obtain more reliable and precise data, and thus the system price is often doubled or even tripled. Furthermore, the LiDAR sensing output is in a 3D point cloud format. A heavy post-processing architecture is required to transform the raw input to practical classification pipeline formats. Additionally, the implemented classification algorithms are generally out of date, brute-force, and ignore the unbalance class distribution in the real application [140, 142]. Even though deep learning-based classification models are gaining promising and encouraging results on a wide range of tasks, a customized LiDAR-oriented vehicle recognition model is missing, not to mention a model that can answer imbalanced datasets and that supports real-time processing [142]. As summarized in the Introduction Section, several key issues, including affordable LiDAR integration, plug-and-play data processing algorithms, and customized online minority-care classification methods, are worth investigating in this research.

In the traffic area, many surveillance cameras have been installed. It would be advantageous to use these surveillance cameras for traffic information extraction and estimation comparing with other specialized hardware. The data from these cameras have been used extensively to handle vehicle detection problems. Right now, if people want to collect information through different cameras, a large amount of brute-force human labor work is necessary. However, vehicle Re-ID researches have escalated in the past few years and now they are booming.

Generally, the multi-camera cooperative traffic sensing system includes three cascading components: single-camera multi-object detection, single-camera multi-object tracking, and cross-camera object re-identification. Currently, the deep learning-based approaches are popular for vehicle detection and show promising results. Such models can be divided into two categories, two-stage detector (i.e., Fast R-CNN [148], Faster R-CNN [149] and Mask R-CNN [150]) and single-stage detector (i.e., You Only Look Once (YOLO) [151], Single Shot Detector (SSD) [152]). In general, a two-stage detector can achieve better accuracy with region proposal networks. At the same time, the single-stage algorithms show much faster processing speed and lower false positive error. Considering the balance of edge-capable processing capacity and the real-time detection accuracy, the YOLOv4, TinyYOLOv4 and MobileNet-SSD are propitious and encourage running on edge devices. For single-camera tracking algorithms, the algorithms can be divided into online and offline. Deep SORT [153], and MOANA [154] are well-known as online tracking frameworks with light structure and dependable performance. To achieve more accurate and reliable object tracking results with lower ID switches and better performance in high occlusion areas, Tracklet Net Tracker (TNT) [155] has been proven to be a dependable

and high-precision tracking algorithm by many state-of-art frameworks with offline design.

For cooperation perception, object ReID is the fundamental task, which refers to the efforts of associating a particular object across different observations. As for vehicle ReID, the process is to identify and match the target vehicle in different sensors. When a target vehicle appears, vehicle ReID will tell if the vehicle has been observed by other sensors, such as cameras, radars and other wireless sensors. Generally, vehicle ReID methods can be divided into two categories: sensor-based and vision-based methods [156]. The early-stage vehicle ReID research matches vehicle signatures detected by multiple traffic sensors. The sensor types include magnetic sensors, inductive loop detectors [157], wireless sensors (GPS, RFID, WiFi and Bluetooth MAC address) [158, 159, 160], and even sensor fusion and hybrid methods [161, 162]. So, the vehicle Re-ID technology breaks the ice that each camera installed at different locations works isolated. Besides sensors-based approaches, with the increase in computation power, vision-based methods emerged and have shown a lot of potential. With the vehicle Re-ID, the surveillance cameras can be used together to detect and track the same object at different locations. The emergence and boom of vehicle Re-ID technology are because (1) the increasing public safety and video information extraction needs and (2) the extensive use of surveillance camera networks in the road network, university campuses, parking garages and streets. With the vehicle Re-ID technology, spot a query vehicle or track the vehicle cross multiple cameras in the surveillance networks that can be done accurately and efficiently. In the remaining part of the literature review, I will focus on vision-based approaches, which includes the classic-feature-based methods and deep-feature-based methods.

Visual-based vehicle ReID algorithms based on classical features generally use traditional empirical rules. They extract and identify differentiated features in different images, which are then

used to match the same target objects. These traditional features include license plate number, color, texture, size, and the Histogram of Oriented Gradients (HOG). The main advantage of classical feature-based methods is that it is easy to interpret and explain the matching results [163, 164, 165, 166]. However, the accuracy of classical feature-based approaches is limited since traditional features may not be sufficient for vehicle ReID across different cameras. Since different feature extraction approaches may be used for different camera views, the matching between different features is not a simple linear relationship. When multiple features are used, sophisticated algorithms are needed to fuse them. Also, lots of manual work is needed to label a large number of outline features and key-point features. Currently, traditional feature-based methods have gradually faded away.

The rapid development of CNNs in recent years has dramatically promoted research topics on vehicle recognition. The task of vehicle ReID and retrieval with traffic cameras has always been a challenging subject. The focus of the former researchers tried to extract vehicle features based on the whole image. However, the sizes of vehicles in surveillance cameras are generally not large enough to support these methods, which leads to a bottleneck for vehicle ReID. Therefore, some researchers have started to pay attention to local scales. Commonly used ideas for extracting local features are vehicle key-point localization and region segmentation. Based on the key-point localization and alignment results, some methods extract the features of the key part of the object and make a detailed comparison to achieve good results [163, 167, 168, 169, 170, 171].

Since 2018, metric learning has become more and more popular in vehicle ReID research [172, 173, 174, 175]. Key target of using metric learning for ReID task is to maximize inter-class similarity and minimize intra-class differences. The challenges comes from subtle inter-class differences and significant intra-class differences in vehicles. For example, the same vehicle looks

different due to variations in lighting conditions, background, and orientation. Meanwhile, different vehicles with the same brand and color can look similar. Therefore, using appearance features alone may not be enough. To address this, [172, 176] introduced spatial-temporal features and information from roads, routes, trajectories, and vehicle attributes to vehicle ReID research. Specifically, deep networks are used to learn features with the purpose of maximizing the distance between different classes, while minimizing the distance within the same class. In particular, the triplet constraint is introduced for learning feature embedding, based on the principle that "samples belonging to the same vehicle ID are closer than samples belonging to different IDs." This triplet constraint has been widely used for pedestrian ReID and face recognition tasks. Based on triplet loss, [172] customized the temporal-attention model that fuses the inter-class features (different models, brands, years of manufacture, etc.) as the ranking module to improve the generalization ability of the vehicle representations. Besides, some related works focus on the hybrid features, the combination of deep features, empirical features and related traffic information, and achieve reliable results for vehicle ReID tasks on the public datasets [177].

2.5 Learning Multi-Modality Data Representations for Unbalanced Data Distributions

Learning representations of multi-modality data is a significant and challenging problem in the field of machine learning, particularly when dealing with unbalanced data distributions. The unbalanced data distribution poses a significant challenge to traditional learning methods, which are not capable of recognizing the small number of rare data points. Multi-modality data represents data with multiple and diverse characteristics, such as images with different orientations,

resolutions, and colors [178]. The mechanism of learning multi-modality data representations involves integrating different modalities into a unified feature space that captures the most relevant information for a given task. One approach to learning multi-modality data representations is to use deep learning models that can learn hierarchical features and capture complex relationships between different modalities minimize the distance between intra-class samples [179, 180]. These models can be trained using various loss functions, such as triplet loss, contrastive loss, or cross-entropy loss, which aim to maximize the distance between inter-class samples and minimize the distance between intra-class samples [181, 178].

Several state-of-the-art methods have been proposed to learn multi-modality data representations for unbalanced data distributions. One of the most popular approaches is to use transfer learning, which involves pre-training a deep neural network on a large dataset and then fine-tuning it on a smaller dataset [182, 178]. This approach has been shown to be effective in improving the accuracy of models trained on unbalanced data distributions. Another approach is to use ensemble learning, which involves combining the outputs of multiple models trained on different modalities [183]. This approach has been shown to be effective in improving the robustness of models to different types of data distributions.

Besides, the few-shot Learning is a branch of Meta Learning in the field of supervised learning to deal with the unbalanced datasets. Meta Learning, also known as learning to learn, decomposes the dataset into different meta-tasks in the meta training phase, and then targets to learn a more generalized model when the category changes [184, 185]. In the meta testing phase, when facing a limited-sample or new category, the classification can be completed without need to retrain or change the existing model. Formally, the few-shot training set contained many categories, with each including multiple samples. During the training phase, C categories with K

samples in each category (total $C * K$ data), will be randomly selected from the training set, and a meta-task will be constructed as the input of the model's support set; a batch of samples are extracted from the remaining data in the model as the prediction object (batch set). This task is known as the C-way K-shot problem, and it requires the model to learn how to distinguish the C categories from $C * K$ data [184].

During the training process, each training (episode) will sample different meta-tasks and contain different category combinations [184]. The mechanism enables the model to learn similar parts of different meta-tasks, such as how to extract important features when comparison samples are similar. The models learned through the learning mechanism can also classify well with new meta-tasks. In general, Few-shot Learning models can be divided into three categories [186]: Model Based, Metric Based and Optimization Based. The Model Based method aims to quickly update parameters on a small number of samples through the customized model structure, and it can directly establish the mapping function between the input X and the predicted value P. The Metric Based method uses the nearest neighbors to measure the distance between the samples in the batch set and the samples in the support set. Finally, the Optimization Based method believes that the ordinary gradient descent method is difficult to fit in few-shot scenarios, so it adjusts the optimization method for a small sample classification.

Learning multi-modality data representations has several advantages, particularly when dealing with unbalanced data distributions. By integrating different modalities, these methods can capture the most relevant information for a given task, improve the robustness of models to different types of data distributions, and enhance the accuracy of models by learning more informative representations. However, one of the main disadvantages of learning multi-modality data representations is the increased computational complexity and the need for large amounts

of training data. Deep learning models are computationally expensive, and training them on large datasets can be challenging. Additionally, these models require significant expertise in data preprocessing, feature engineering, and hyperparameter tuning, which can be time-consuming and difficult.

2.6 Learning Representations on Edge Devices in ITS

Currently, the state-of-the-art development of traffic sensing has always been closely related to IoT technologies. Well-processed summary from the edge detectors present more clear and organized information than raw materials. However, due to the constraints in computing power and communication technology, only limited studies adopted the IoT architecture for traffic sensing. Back in 2014, Jin et al. proposed a Network-Centric IoT architecture with a sensing paradigm used for traffic control and information estimation, which brings the inspiration to researchers using IoT sensors in ITS systems [187]. Li et al. proposed a policy-based secure sensing system, which can hugely improve the safety level and defend the fake alerts generated by attackers [188]. In 2017, Ling et al. proposed an automated object detection algorithm and fully experimented on the urban surveillance system based on edge computing. Their proposed method help cameras detect object vehicles accurately and can be used to reduce the data volume needed to be transmitted, processed, and managed in the surveillance systems. In 2019, a research group from the University of North Carolina at Chapel Hill proposed a hybrid architecture REVAMP^{2}T using multi-camera pedestrian tracking [189]. In REVAMP^{2}T, each camera is equipped with a computing unit, and hierarchical information extraction and sharing system are made, including single-camera detection, tracking, and multi-camera human Re-ID. This framework achieved network-scale pedestrian tracking with much lower cost and time la-

tency. However, considering real traffic networks, the pedestrians' travel speed, activity range and scale are much lower than vehicles. Ke et al., implemented a hybrid system with edge AI for monitoring parking status using a single camera and achieved promising results [190]. To the author's best knowledge, the authors are pioneers who design and implement video-based hybrid IoT systems for large-scale traffic sensing.

Part I

Learning Representations for Cooperative Traffic Perception

3

Chapter 3. Cooperative Multi-camera Vehicle Tracking and Traffic Surveillance with Edge AI and Representation Learning

This chapter is modified from the published works:

- H. Yang, J. Cai, C. Liu, R. Ke, and Y. Wang*. "Cooperative Multi-camera Vehicle Tracking and Traffic Surveillance with Edge Artificial Intelligence and Representation Learning." in Transportation Research Part C: Emerging Technologies, Volume 148 (2023), 103982, https://doi.org/10.1016/j.trc.2022.103982

- H. Yang, J. Cai, M. Zhu, C. Liu and Y. Wang*, 2022. "Traffic-Informed Multi-Camera Sensing (TIMS) System Based on Vehicle Re-Identification." in IEEE Transactions on Intelligent Transportation Systems, https://doi.org/10.1109/TITS.2022.3154368

3.1 Challenges and Motivations

In contemporary Intelligent Transportation Systems (ITS), surveillance cameras are ubiquitously installed, yet remain isolated, extracting data only from their individual fields of view. Existing research mostly emphasizes traffic parameter estimation and video analysis from single-camera inputs, leaving cross-camera traffic information, such as link travel time, speed, and origination and destination distribution, untapped. Addressing this, one approach is to track roadway users across different cameras, leading to the extraction of cross-camera traffic information. However, this proves challenging due to current limitations in video-based Re-Identification (Re-ID) methodologies, computational architecture, and methods for collaborative traffic parameter estimation.

The prevalent Re-ID methodologies are not accurate enough for traffic sensing due to disparities among individual cameras. The resource-intensive nature of these methodologies places heavy demands on data processing and storage hardware. Furthermore, automatic cross-camera surveillance requires precise cross-camera vehicle Re-ID methods which consider the physical particularities of the traffic system. The second hurdle is the computational cost associated with object re-identification. The requirement for powerful GPUs and complex deep feature extractors poses a significant barrier. A more efficient hardware architecture and algorithm workflow would greatly benefit the traffic agencies and research communities. The final challenge lies in the absence of a methodology for collaborative traffic parameter estimation. Current studies largely focus on single-camera data, but a comprehensive methodology that accounts for variances between different camera pairs and locations is needed.

Emerging edge computing technology provides potential solutions to these challenges. By

shifting some storage and processing resources closer to data sources, edge computing could enable video pre-processing at network peripheries, reducing the load on central servers and diminishing the cost of server GPUs. To realize this vision, both edge nodes and servers require innovative cross-camera vehicle Re-ID algorithms. While edge devices can handle real-time multi-object detection, real-time and reliable multi-object tracking needs further investigation. Simultaneously, vehicles' representations need to be selected and transmitted in a cooperative edge-server manner. On the server side, customized Re-ID feature extractors must capture individual peculiarities from concise inputs. A more precise re-identifier and efficient traffic parameter calculations are also essential.

In light of these requirements, I propose the Edge-empowered Cooperative Multi-Camera Surveillance (ECoMS) system. This novel system applies IoT architecture to network-scale traffic perception and vehicle tracking. The ECoMS system offers an edge-server cooperative workflow for multi-camera vehicle tracking and traffic perception. The system is optimized for edge devices, and it features a multi-task feature extraction workflow. This system also introduces a novel clip-based deep vehicle Re-ID model with a hierarchical feature extraction and fusion mechanism. Additionally, a precision-aware kernel density estimator is integrated into ECoMS for information extraction from various camera pairs. Finally, the ECoMS system is cost-effective, reducing the need for server hardware, data storage, and communication bandwidth. Overall, the ECoMS system facilitates cooperative cross-camera vehicle tracking and smart traffic surveillance in an affordable, scalable, and power-efficient manner.

3.2 Methodology – the Proposed ECoMS System

3.2.1 ECoMS System Overall Framework Architecture Illustration

In general, ECoMS includes three key components: the edge nodes with embedded cameras, a communication component based on TCP/IP socket and the edge server for multi-node information processing. The users need to select the target camera pair or road segment to extract the traffic information. The Figure 3.1 shows the overall architecture of ECoMS System, and the intermediate variables and parameters are summarized in Table 3.1. In ECoMS system, the edge server is the whole system's control unit. The server can admit the input from traffic managers, including attributes information (weather condition, road type), choosing camera groups, and re-identifying vehicles from multiple cameras. The server is also the central node for listening and summarizing the belonging edge nodes' information and finishing the cross-camera traffic information estimation tasks. The tasks include clips features extraction, candidates filter and selection, cross-camera vehicle Re-ID, traffic information estimation, and communication of the necessary parameters to all edge nodes. Then, n edge nodes are separated in different locations of the road network. Each of them is equipped with an NVIDIA Jetson AGX Xavier to support real-time object detection, single-camera tracking, object representations, and features selection. Via the TCP/IP socket based on WiFi, tracker clips and the features are summarized and sent to the edge server.

Table 3.1: Definitions of key intermediate variables and parameters in ECoMS system.

Variable	Definition
$0, 1, \ldots, m, n$	The index of the edge nodes. Each edge node is equipped with one camera. Since the cross-camera Re-ID process always uses two or more cameras for comparison, the m represents the query camera, and n is the gallery camera.
RV_m	The raw video input of edge m with continuous video inputs.
O_{mi}	The i^{th} detected object of the edge node m through the MOD.
T_{mi}	The i^{th} aggregated track of edge node m through SCT.
C_{mi}	The i^{th} vehicle representation clip generated by camera m. Each clip consists of $\mathcal{N}$ cropped vehicle image.
$f^k_{C_{mi}}$	The k^{th} frame level of feature belong to k^{th} frame of C_{mi} $(0 < k < \mathcal{N})$.
$f_{C_{mi}}$	The clip-level of feature belongs to the C_{mi}.
P_{mn}	The camera pair from camera #m to camera #n.
$ReID_{mn}$	The final vehicle Re-ID results by clips, #m is the query camera and #n is the gallery camera.
Top^{mi}_{ni}	The top-one candidate, using C_{mi} as query and C_{ni} is best match candidate from the gallery set.
TT_{mn}	The estimated travel time information form camera #m to camera #n.
LS_{mn}	The estimated link travel speed information form camera #m to camera #n.
OD_{mn}	The estimated cross-camera origination to destination information form camera #m to camera #n.

3.2.2 Multi-task Edge-based Algorithms

Real-time Multi-object Detection and Tracking

For each edge node, the MOD is the first step to localize the objects. The inputs of the MOD algorithm are raw video captured by the camera RV_m and the outputs are the object sets O_{mi}. To achieve fast and reliable processing speed on the edge computing units, I choose to use the single-stage detector You Only Look Once (YOLO) version four (YOLOv4) [191]. YOLOv4

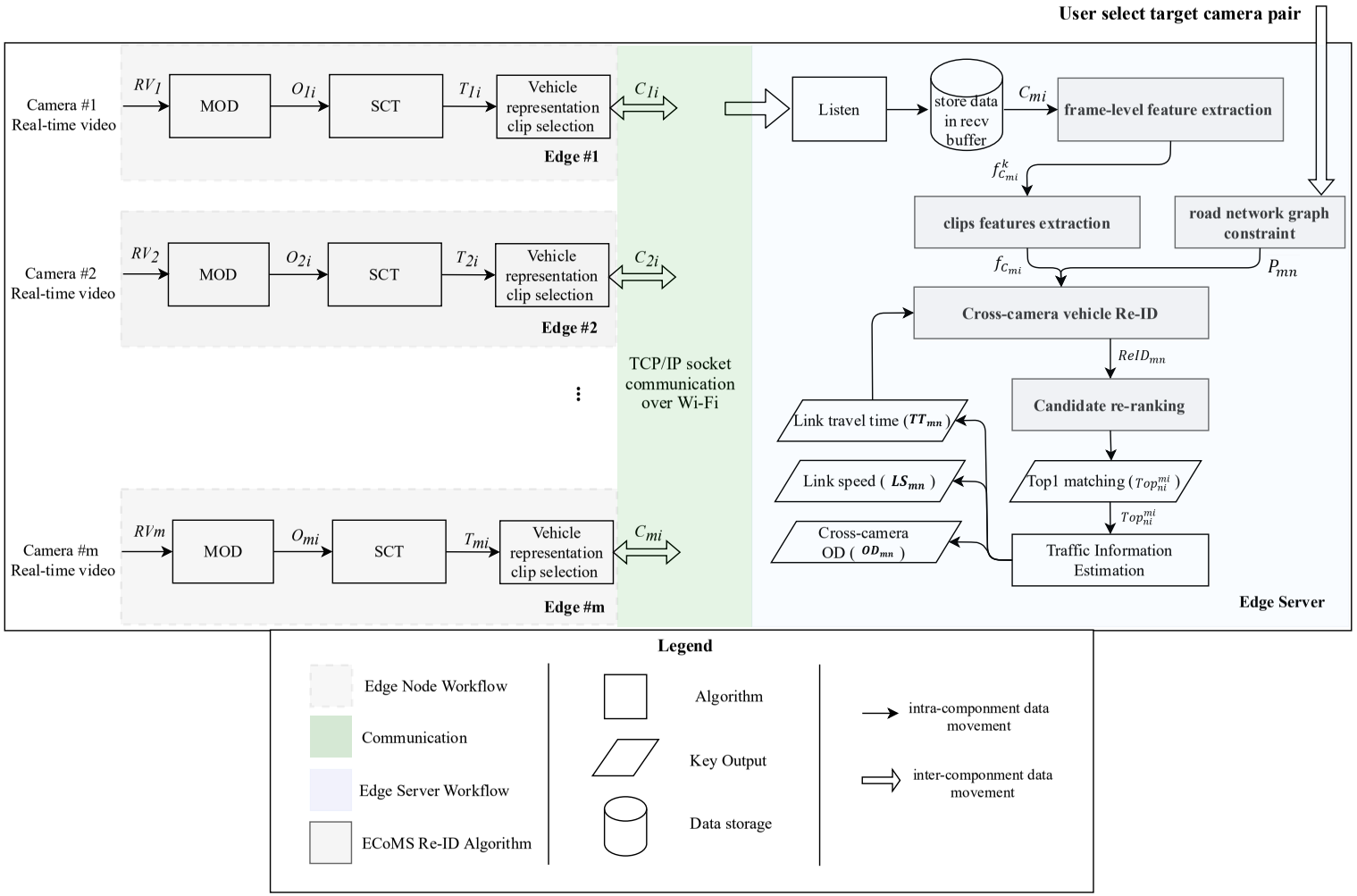


Figure 3.1: The workflow of the ECoMS system, which includes three components: m edge nodes in the road network, a communication component, and an edge server. The nodes run object detection, single-camera tracking, clips selection, and feature extraction simultaneously on the edge devices. Clips are summarized and sent to the edge server via the TCP/IP socket based on WiFi. Then, the edge server conducts heavy computation tasks, which consist of the cross-camera objects Re-ID, traffic information estimation, and sending the necessary parameters back to all edges and the TMCs.

[153] is a famous detector with a CSPDarknet53 backbone, SPP neck [192] and inherits the head design of YOLOv3 [193], which achieves the balance of processing speed and accuracy on multiple open datasets [191]. In this research, I use a well trained YOLOv4 by the coco dataset [194], and then retrained the detector by two open-source datasets: MIO-TCD [42] and LHTV [111].

Deep learning feature extractors are usually the bottleneck for online tracking algorithms, making them unscalable for real-time applications, not to mention running on the edge devices. Inspired by the current state-of-art joint detection and embedding [195] tracker, Tracktor [196] and Fast MOT [197], I implemented a boosting mechanism for the tracking feature extractor based on the structure of deep Simple Real time Tracker (SORT). As shown in Figure 3.2., in practice, the feature only extracted every K frames and used as the frame-level features. Since the surveillance cameras are permanently mounted at a certain point with a stable background, optical flow [198] is then used to fill in the gaps. Towards a lightweight and reliable feature extractor for clips, I chose to implement the Omni-Scale network (OSNet) [199]. In OSNet, to efficiently capture the spatial dependencies and avoid overfitting, the building block is operated through point-wise and depth-wise convolutions. A unified Aggregation Gate (AG) is integrated into the tracking framework for fusing multi-scale features with various input-dependent channel-wise weights dynamically. OSNet achieves impressive feature extraction results with a feather-light architecture. The input of the online tracking algorithms are the detected objects sets O_{mi} and the outputs are the aggregated object tracks T_{mi}. In the next step, the T_{mi} is further selected by the clip selection and generation module.

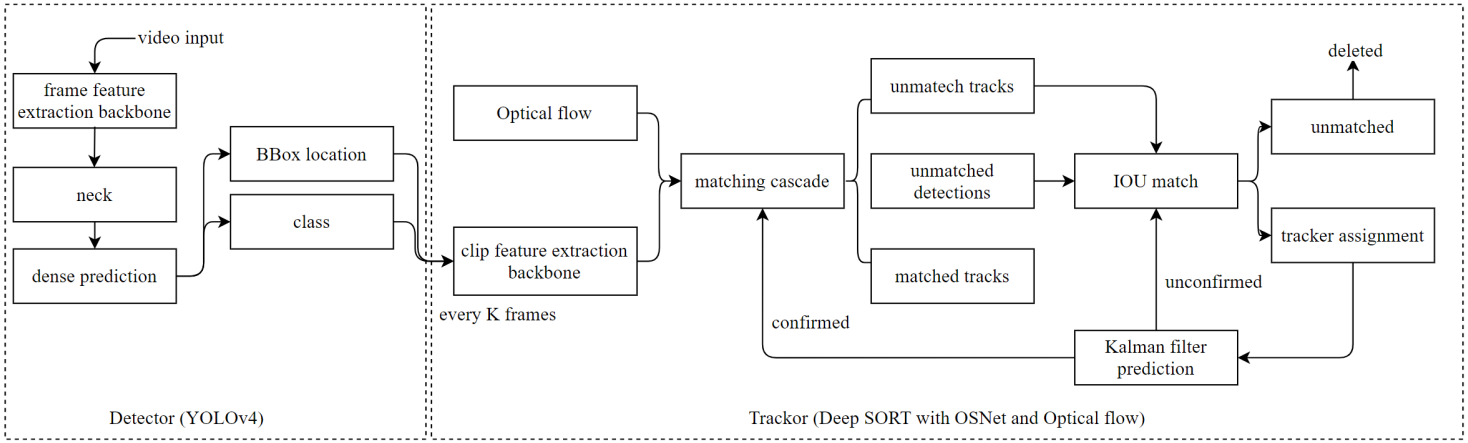


Figure 3.2: The optimized MOD and SCT framework deployed on the edge device. The YOLOv4 detector and Deep SORT tracking with OSNet and Optical flow are customized for real-time processing.

Vehicle's Representation Clip Selection

After the real-time detection and online tracking procedures running on the edge side, the raw video input of camera m (RV_m) are cut into different tracks T_{mi}. The next step is to select $\mathcal{N}$ frames from a track with the high-quality objects' representativeness and consist of a clip C_{mi} for object i. Here, the selection criteria depend on several aspects, including the object detection confidence (θ), object size ($\mathcal{S}$), and the attributes input (road speed limit and camera frame rate). The bounding boxes are normalized by the largest size of a track. Then, a confidence θ_i is used as the threshold to filter the low-confidence frames. Denoting the object frame k score as ρ^k, the score for each track is calculated as:

$$\rho^k = \{\theta^k * \mathcal{S}^k | \theta^k > \theta_i\} \tag{3.1}$$

With the value of ρ^k, the next step is to select top $\mathcal{N}$ frames with the largest result of ρ^k as representative frames to consist of a C_{mi}. The last task for the edge is to send the C_{mi} for each track to the server for further processing by TCP socket protocol. It is worth noting that for different attribute inputs, the θ_i and $\mathcal{N}$ can be changed by the actual situation. The detailed parameter settings are in the experiment section. The final output for each edge node m is the C_{mi} and belonged features, including the captured time, location, and the associated camera information.

3.2.3 Clip-based Multi-camera Vehicle Re-ID Algorithm Customization

After receiving the clips C_{mi}, C_{ni} and the corresponding features from edge nodes m and n, the server runs four threads simultaneously to re-identify the vehicles among the clips. The whole

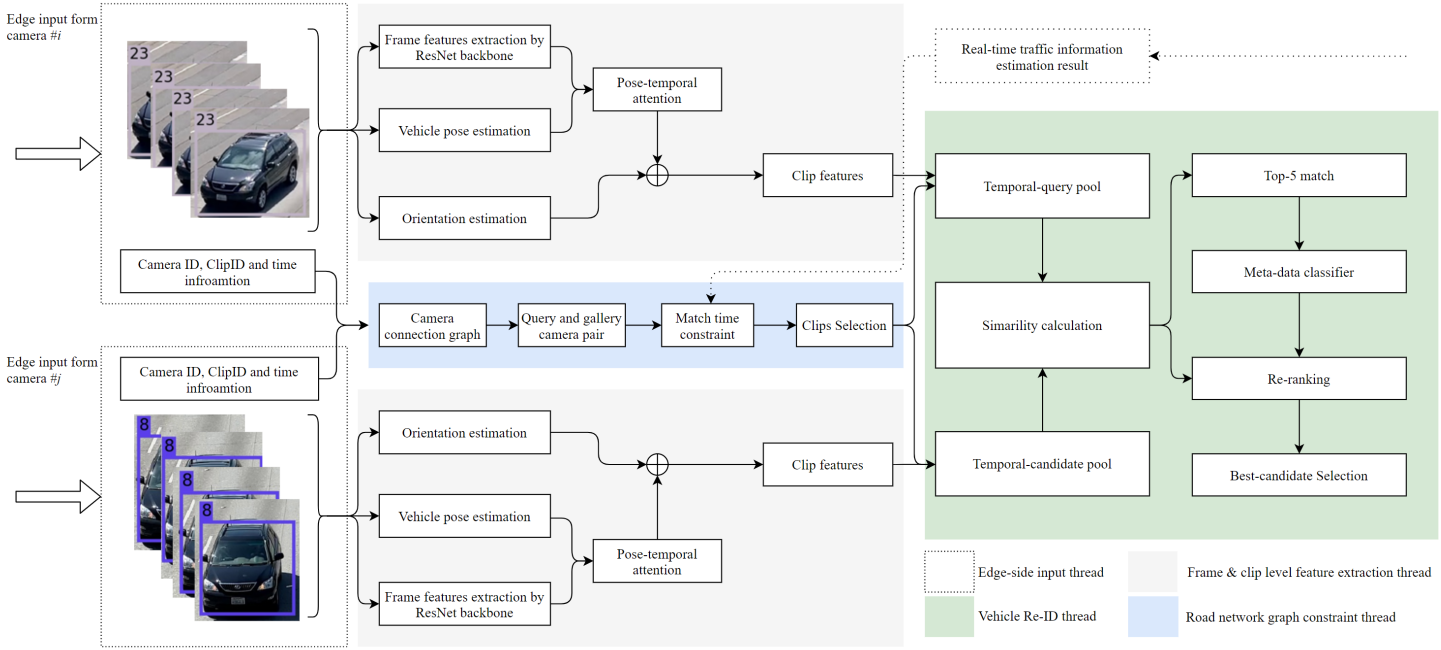


Figure 3.3: The ECoMS_Re-ID workflow on the server.

process is shown in Figure 3.3. The proposed clip-based ECoMS vehicle Re-ID method includes frame-level feature extraction, clip-level feature extraction, road network graph constraint filtering, and attributes-aware re-ranking.

Frame-level Feature Extraction

After receiving the clips (C_{mi}) from edge nodes, the frame-level features extraction is designed for capturing spatial information. Here, the features of each single frame are extracted using a ResNet50 [125] backbone pre-trained on the ImageNet [200] and CityFlow [201] datasets, where the 2048-dimension vector from the fully-connected layer represents the appearance features ($f^k_{C_{mi}}$) for frame k in camera C_{mi}. Prior research suggests that pose and orientation significantly impact the human and vehicle Re-ID results. Therefore, I utilize the car key points estimation to represent the pose and structural features. Specifically, with the car key points estimation network proposed by [202], it predicts 18 surfaces for each vehicle, which can be converted into a 72-dimension vector by concatenating the nodes of surfaces in order to infer 3D information. The 72-dimension vector is then projected to 2048-dimension as the vehicle's pose features ($fp^k_{C_{mi}}$) with a learnable weight matrix $\mathbf{W} \in \mathscr{R}^{72*2048}$.

Clip-level Feature Extraction

After obtaining the frame-level features, I implemented the attention mechanism to fuse $\mathscr{N}$ frames features into a clip-level feature ($f_{C_{mi}}$). The detailed process is shown in Figure 3.4. To fuse the appearance features, a 2D convolution is used to capture the spatial correlations of neighboring frames. Then, I apply pose-attention to the two vector sets with the same size of $\mathscr{R}^{2048*T}$ and fuse them into a vector. Then, to leave the raw pose information for further com-

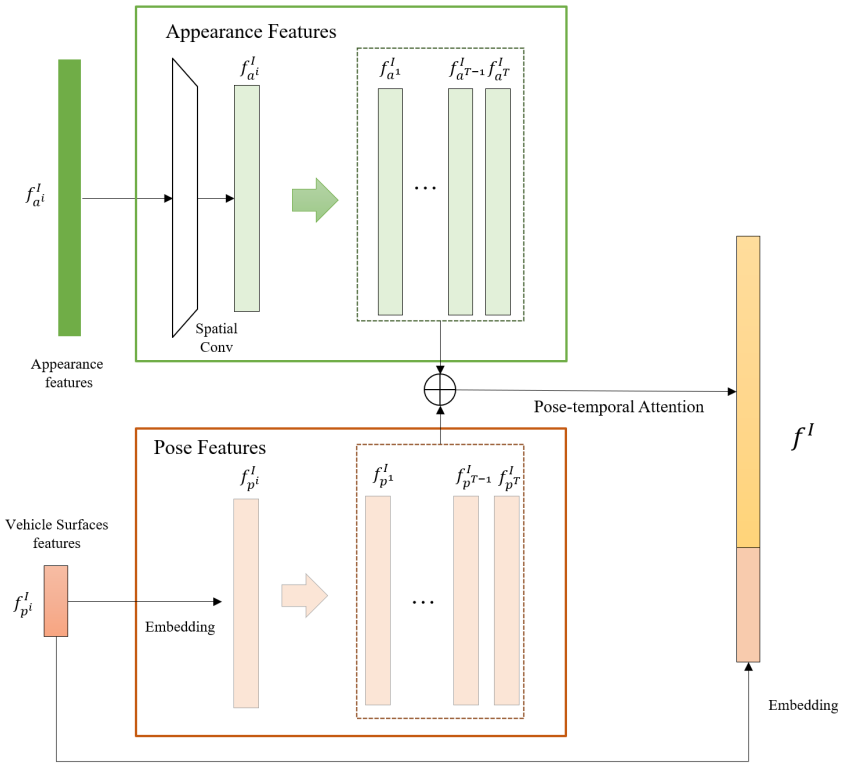


Figure 3.4: The pose-aware clip-level feature extraction component.

parison, another smaller embedding is used to map the pose features $fp_{C_{mi}}$ into a 256-dimension vector and then concentrate with the output of the pose-attention. The final clip feature ($f_{C_{mi}}$) of clip C_{mi} consists of both the appearance and post vectors, belonging to object i.

Road Network Graph Constraint Integration

In order to save computing resources, I used the road graph restrictions and travel time restrictions to filter the set of potential matching targets before calculating the similarity of objects between camera m and camera n. The first step is to build the road network adjacent matrix for the installed cameras. Based on the pair connection information, the second step is to set up the

query and gallery camera pair P_{mn}, which means the camera m is used as the query set and n as the candidate set. The third step is to set time constraints for matching. I use a time window by considering the real-time travel time reliability to filter impossible matching pairs. Denoting the average travel time at time t from edge m to n as t_{mn} and a buffer index (b_{mn}, varies with time of a day), the matching time window length is

$$TT_{P_{ij}} = (t_{mn} \cdot (1 - b_{mn}),\ t_{mn} \cdot (1 + b_{mn})). \tag{3.2}$$

The value of the b_{mn} varies for different cities at different times of day. In this chapter, the b_{mn} is summarized and used to represent the city's congestion level and the travel time reliability summarized in [203].

Vehicle Re-ID by Clips Features

After obtaining the features of each clips and generating the query and gallary object Re-ID sets, the next step is to calculate the pairwise similarity between targets. Here, the pairwise object similarity is measured based on the Euclidean distance. Only top five closest targets will be used as the alike candidates sets ($\mathscr{A}$) and then considered for further re-ranking.

For the ECoMS Re-ID, I train the clip-based Re-ID model end to end. The loss functions are consists of two parts. Considering the task somehow contains inter-object classifications, the cross-entropy (*Xent*) loss is included. In the following equation, i and j represent the query and gallery object.

$$\mathscr{L}_{Xent} = -\sum_{i=1}^{q} \log\left(q(i)\right)g(i)), \ g(i) = \delta_{i,j}, \tag{3.3}$$

At the same time, inspired by for human Re-ID research, the triplet loss is widely adopted to

distinguish the intra-class and inter-class differences. Specifically, the Batch Hard (BH) triplet loss [204] shows promising results. However, a significant fundamental difference between vehicle and human face Re-ID tasks is that people do not typically have the same appearance, while there are cars with exactly the same appearance features. If training the vehicle Re-ID model using BH loss, the process would be too tough to obtain reliable results if the same vehicle exists. So, in this research, I use the batch sample (BS) instead [205, 172, 111] of BH triplet loss to train the model. In BS triplet loss, a mini-batch $\mathscr{B}$ is defined as

$$\mathscr{L}_{BS}(\theta;\mathscr{B}) = \sum_{\text{all batches}} \sum_{a\in B} l_{tri}(a), \tag{3.4}$$

Where

$$l_{tri}(a) = [m + \sum_{p\in P(a)} w_p D_{ap} - \sum_{n\in N(a)} w_n D_{an}]_+, \tag{3.5}$$

In equation 3.5, w_p are the weights of positive samples, w_n are the weights of negative samples, D_{ap} are the distances of anchor sample to the positive samples, D_{an} are anchor samples to the negative samples, and m represents the predefined margin value. The final loss function of the proposed ECoMS_Re-ID methodology is a linear combination of both BS triplet loss and cross-entropy loss, as shown below,

$$\mathscr{L}_{ReID} = \lambda \mathscr{L}_{BS} + (1-\lambda)\mathscr{L}_{Xent}. \tag{3.6}$$

So the output of the clip features Re-ID procedures are the several candidates with the top closest Euclidean distance, represented by $ReID_{mn}$. Here, I select only the top five in the following re-ranking procedure and then find the best candidates.

Attribute-aware Re-Ranking

As mentioned, the toughest task for ECoMS is to obtain cross-camera traffic information automatically from reliable matching results. In this research, I integrate a light attributes classifier of the final alike candidates' sets to choose the best one. Since the appearance features and pose features have already been well used in previous steps, object attributes features, including vehicle types, brand, color, and module are included in the comparison. To balance the computing time and classification accuracy, Light-CNN, proposed by Wu et al. [206], is adopted into this process. The expanded 256-dimensions *MFM_fc* layer is used for final classification features with pre-defined class. Using x represents total class numbers, q_i and g_i as the inputs of the query and gallery objects, $c(q_i)$ represents confidence of the object q_i, the metadata probabilities for q_i and g_i are p_{q_i} and p_{g_i}, then the re-ranking distance for a Re-ID pair among all class can be shown in the following equation:

$$\mathscr{D}_R(q_i,g_i) = \sum_x d_x(q_i,g_i), \tag{3.7}$$

where q_i represents a query object clip input and g_i represent the i_{th} candidate in gallery set. In detail, the distance for a certain class x is:

$$d_x(q_i,g_i) = c(q_i) \cdot c(g_i) \cdot (-logP(c_{p_i} = c_{g_i}|p_i,g_i)). \tag{3.8}$$

And the $c(p_i)$, $c(g_i)$ are the KL distance between p_i,g_i and the uniform classification distribution. The n is the number of classes defined in the re-ranking classification dataset. Finally, the candidates are re-ranked by the distance in the equation 3.9

$$\mathscr{D} = \mathscr{D}_e(q_i,g_i) + \beta \cdot \mathscr{D}_R, \tag{3.9}$$

where the $\mathscr{D}_e$ is the euclidean distance obtained from the clips Re-ID. The β a hyper-parameter that can be tuned in the experiment. After the ECoMS cross-camera vehicle Re-ID, only the best match candidate Top_{ni}^{mi} is used for the cross camera information extraction.

3.2.4 Cooperative Cross-camera Traffic Parameters Estimation based on Vehicle Re-ID

Link-level Traffic Information

Cross-camera vehicle Re-ID enables ECoMS to estimate link-level and network-level cross-region traffic information (i.e., link travel time and speed) at a high penetration rate. For the link information estimation, the inputs are the best matched candidate (Top_{ni}^{mi}) pairs' attributes, including the captured time and the camera id. Assuming that the time for object vehicle i to pass the camera m is t_i^m and then to pass the camera n is t_i^n, then the travel time TT_{mn}^i of vehicle i from m to n is $t_i^m - t_i^n$. Suppose I have $\mathscr{I}$ vehicles in the pool, then each matching vehicles' link travel time and link speed value can be obtained. Then, the link average travel time and average speed is calculated by the sum of the value over the $\mathscr{I}$.

Also, based on the obtained object-level traffic parameters, ECoMS can estimate the cross-camera traffic information distribution, i.e., travel time and speed distributions. Parametric probability approaches are commonly used to estimate the cross-camera traffic information traditionally. The basic assumption of the proposed approach is that the traffic information follows specified distributions, which can be estimated based on the observed data in theory. However, inevitable errors caused by the limited capability of modeling complex scenarios make these methods inapplicable for cross-camera scenes.

Here, I establish the distribution estimation approach based on Kernel Density Estimation

(KDE) and Kernel Smoother (KS) [207], as a non-parametric approach for estimating probability distributions. The proposed approach enables the construction of travel time distribution and speed distribution with the advantages of fewer assumptions and a more flexible fitting structure with fast processing speed. In detail, the density of the evaluated variable ($\mathscr{X}_{mn}$) by the KDE and KS is shown mathematically as:

$$f(\mathscr{X}_{mn}) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_o}{h}). \tag{3.10}$$

In equation 3.10, $\mathscr{X}_{mn}$ are variables that need to be estimated, including the link speed (LS_{mn}) and travel time (TT_{mn}). n is the number of observations and h is the kernel bandwidth that controls the smoothing level of Probability Distribution Function (PDF). K represents the kernel function and the I use the Gaussian kernel here. With the distribution of LS_{mn} and TT_{mn}, traffic information can be obtained.

Network-level Traffic Information

After obtaining the vehicle Re-ID information between adjacent cameras and combining with the graph of the road network, the vehicles' trajectories along with the camera link can be extracted. Aggregating the trajectories by each period (network information extraction thread time length, i.e. per five minutes), area Origination and Destination (OD) information can be extracted with the channelized characteristics of roads. Subsequently, the utility level, the traffic flow distribution, and the OD distribution can be extracted in real-time. To the authors' best knowledge, this is the first reliable traffic flow and OD distribution extraction framework in real-time with a high penetration rate.

3.3 Experiment and Discussion

3.3.1 Dataset Description

Cityflow

The Cityflow Vehicle Re-ID dataset is proposed by NVIDIA Research in 2019 [201], which was captured from 40 traffic cameras, including the scenario of intersections, local roads, arterial roads and highways. In this study, the Cityflow dataset is used to train and evaluate the detection and vehicle Re-ID algorithm performance deployed on the edge server. In detail, a total of 666 vehicles annotated with distinct vehicle IDs are used for training and testing the cross camera vehicle Re-ID based on clips. License plates are masked in black for privacy consideration. As the original testing set includes single images and the training set includes short video clips, to establish clip-based evaluation, I use 20% of the training set as the testing set and report the ReID performance.

Freeway Sensing Video (FSV) Dataset

Since the time length of each surveillance video clip in Cityflow dataset is too short (average length is only 4.88 min) and does not include the ground truth timestamp, a whole new dataset Freeway Sensing Video (FSV) dataset is proposed by authors to demonstrate the traffic sensing and information estimation evaluation.

FSV dataset includes 4.31 hours (258.35 minutes) of video and includes four different cameras on the Interstate 5 freeway, with the video format of 1080p/30 fps. The illustration of the FSV dataset is shown in the Figure3.5, with the FSV cameras (from camera 1 to camera 4) location, orientation and view. All the image frames include ground truth time and the cam-

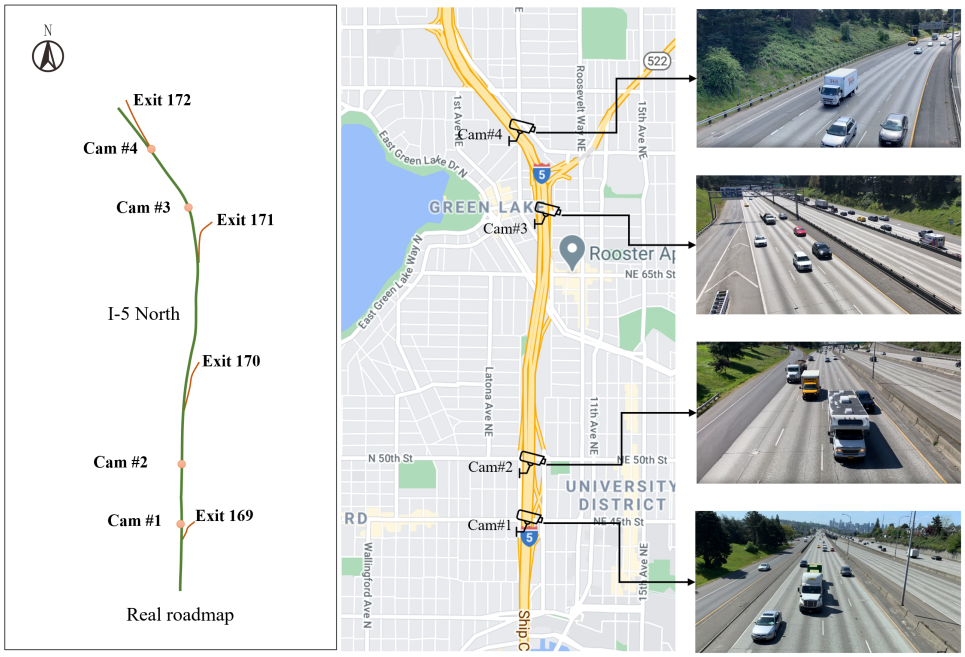


Figure 3.5: The detail information illustration of the FSV dataset, including four cameras locations, view point and the network spatial constraints.

era location GPS information. Among the camera links, the longest distance is 3216 m, and the shortest is 405 m. In this research, 40.12 minutes of video is used to fine-tune the detection, tracking and Re-ID algorithms, and 138.23 minutes of video are used to evaluate the traffic sensing framework performance. In evaluation, using the long video as input of the edge devices for real-time detection and tracking, then sending the clips to the server with spatial-temporal information for cross camera vehicle Re-ID and traffic information estimation. For the cross-camera vehicle Re-ID, a total of 1547 vehicles captured from the different four cameras are annotated with distinct vehicle IDs, and then used for training and testing the cross-camera vehicle Re-ID algorithms (70% of for training, 10% for validation and the rest 20% for testing.).

3.3.2 Edge-side Experiment

Edge-side Workflow and Algorithm Adaptation

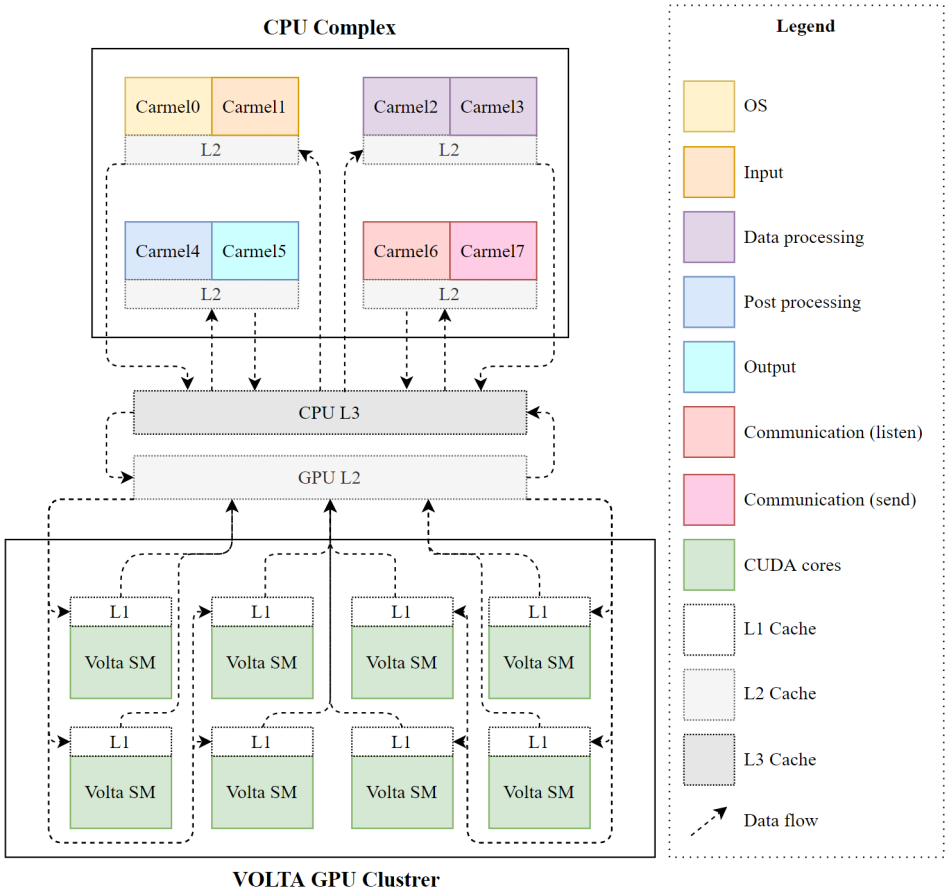


Figure 3.6: The illustration of integrating and deploying the ECoMS algorithms workflow with the computational resources on the Nvidia Jetson AGX Xavier edge node.

The Nvidia Jetson AGX Xavier is chosen as an embedded edge platform for real-time vehicle detection, tracking and clip selection, integrated with a 1080P camera. The Xavier has eight Carmel CPUs and eight Volta SM GPU to consist of the CPU and GPU cluster. To fully stimulate the computing potential on edge nodes, the entire workflow running on the edge side of

ECoMS is combined with the reasonable Xavier resources without overlap in one cycle. As Figure 3.6. shows, the CPUs cluster are divided into six groups for 1) handling OS system, 2) input preparation, 3) data processing for algorithms, 4) output preparation and control, 5) communication receiving orders from TMC and 6) transmit data to edge servers. Then, the GPU cluster is in charge of handling the detection and tracking script. The CPU and GPU clusters share the information through the cache for each cycle. To speed up the whole edge framework, the detector and feature extractor fully implemented using the TensorRT backend (TRT 7.1.3, with Jetpack 4.4) and perform asynchronous communication inference. To minimize the computation delay and energy consumption of the edge node, I implemented a hybrid deployment offloading strategy [208] to optimize the performance of service caching placement and system resource allocation, including the CPU processing frequency and communication transmission power allocation. In addition, all code finished by Python language, including Kalman filter, optical flow, and data association, are optimized using Numba [209].

Edge-side Parameters

On the edge nodes and server, final parameters are set as follows:

- In the multi-object tracking, the algorithm extract features every K frames, and add the associations by optical flow. Here the K is equal to four.
- $\mathscr{N}$ frames of a track with the high quality representativeness is five.
- the object detection confidence (θ) base line is equal to 0.75.
- object size ($\mathscr{S}$) here is no less than 500 pixel.

Edge-side Detection and Tracking Performance

For the real-time multi-object detection, I pre-define three classes, including truck, bus, and car. The average precision of detection are 81.40%, 78.33% and 96.92% for the four camera views together, respectively. Set the Intersection over Union (IoU) threshold as 50%, then the Area-Under-Curve (AUC) for each unique recall mean average precision (mAP@0.50) is 85.55%. I also evaluate the real-time multi object tracking result separately under different traffic conditions light traffic (less than 10 vehicle/lane/min), normal (10 vehicle/lane/min to 20 vehicle/lane/min) and busy (20 vehicle/lane/min to 30 vehicle/lane/min) heavy (above 30 vehicle/lane/min), with the accurate result and the computational performance. The results are summarized in Table 3.2.

Table 3.2: The optimized multi-object detection and tracking performance summary on the ECoMS system edge nodes.

Traffic condition	Camera 1			Camera 2			Camera 3			Camera 4		
	IDF1	MOTA	FPS	IDF1	MOTA	FPS	IDF1	MOTA	FPS	IDF1	MOTA	FPS
Light	81.76	64.31	34.4	82.12	65.43	33.6	78.24	61.52	34.2	85.24	67.45	33.9
Normal	76.23	61.61	26.1	77.07	63.05	25.4	73.42	56.62	25.2	79.78	63.92	26.9
Busy	74.47	60.01	19.3	75.04	61.78	18.2	68.97	54.81	18.7	77.14	62.13	20.1
Heavy	71.23	59.22	16.3	74.05	60.63	17.1	66.17	51.48	16.8	74.23	60.61	17.4

As shown in Table 3.2, The scripts are deployed on edges can perform real-time MOD, MOT and candidates selections process with reliable MOT IDF1 score. The camera 4 performs best, which gets 85.24, 79.78, 77.14 and 74.23 on the light, normal, busy and heavy traffic conditions with 33.9, 26.9, 20.1 and 17.4 FPS, respectively. Even the camera 3 performs lowest among four nodes, the average IDF1 still scores 71.70. Considering the general surveillance video input always use 10-15 FPS as standards, such a performance can satisfy all kinds of real time traffic

multi-object detection and tracking with post processing.

3.3.3 ECoMS Cilp-based Vehicle Re-ID Experiment

Vehicle Re-ID Hardware and Environmental Setting

After the vehicle representations are selected and sent to the edge server by the internet, the clip-based Vehicle Re-ID framework is triggered. In ECoMS, the edge server is equipped with A CPU of Intel Core-i9 9900K and two GPUs manufactured by NVIDIA. In the experiment, four edge nodes are involved in the system testing for cross-camera Re-ID and traffic information estimation. The cost of the server is around $3000 at the year of 2022. The server can support up to eight ECoMS edge nodes input. The operating system is Linux system and the ECoMS_Re-ID is implemented by PyTorch.

ECoMS Vehicle Re-ID Parameters

The parameters using on the edge nodes are as follows:

- The clip level features ($f_{C_{mi}}$) is fused by $\mathcal{N}$ frames, and here the $\mathcal{N}$ is equal to four.
- Each vehicle can be obtained two clip level features ($f_{C_{mi}}$) by fuse frame zero to three and one to four, then used for re-ranking together.
- The λ in equation 3.6 is 0.5.
- The β in equation 3.9 is 0.4.
- b_{ij} in this work is used as the travel time reliability index in the data collection region, which is open source and provided on Digital Roadway Interactive Visualization and

Evaluation Network (DRIVE Net) platform developed by the Washington State of Department of Transportation and University of Washington [210]. In this research, I use the b_{mn} for camera #1 to camera #2 is 0.1, camera #2 to camera #3 is 0.35 and camera #3 to camera #4 is 0.1.

Vehicle Re-ID Results Summary and Comparison

To show the proposed RSITS_Reid's potential in vehicle Re-ID tasks, the research introduced four SOTA methods.

BoT [211]. Bag of Tricks (BoT) is a well-known baseline for deep human Re-ID. Here, I re-trained the model by a 256*256 ResNeXt50 as a backbone, with Global Average Pooling (GAP). Cross-entropy and triplet loss are integrated into the BoT.

AGW [212, 213]. Attention Generalized mean pooling with Weighted triplet loss (AGW) is proposed in 2020. Here, I re-trained the model by a 256*256 ResNeXt50 as backbone, with Attention Generalized Mean Pooling (AGMP). Cross-entropy and weighted triplet loss are used in the model.

BoT_ibn [211, 214]. The Instance-Batch Normalization (IBN) network is integrated into the upgrade version of BoT and called BoT_ibn in this research as a baseline.

SBS [212]. SBS is proposed in the Fast Re-ID framework by integrating the SOTA tricks used in the Re-ID tasks. Here, I re-trained the model by a 256*256 ResNeXt50 as the backbone, with on-local block (NL). The Generalized Mean Pooling (GMP) is used to downsample the vector. Circle Softmax is used instead of traditional linear classification layers. Both Cross-entropy and weighted triplet loss are integrated into the BoT.

The results are summarized into the Table 3.3 and visualized on the Figure 3.7, including

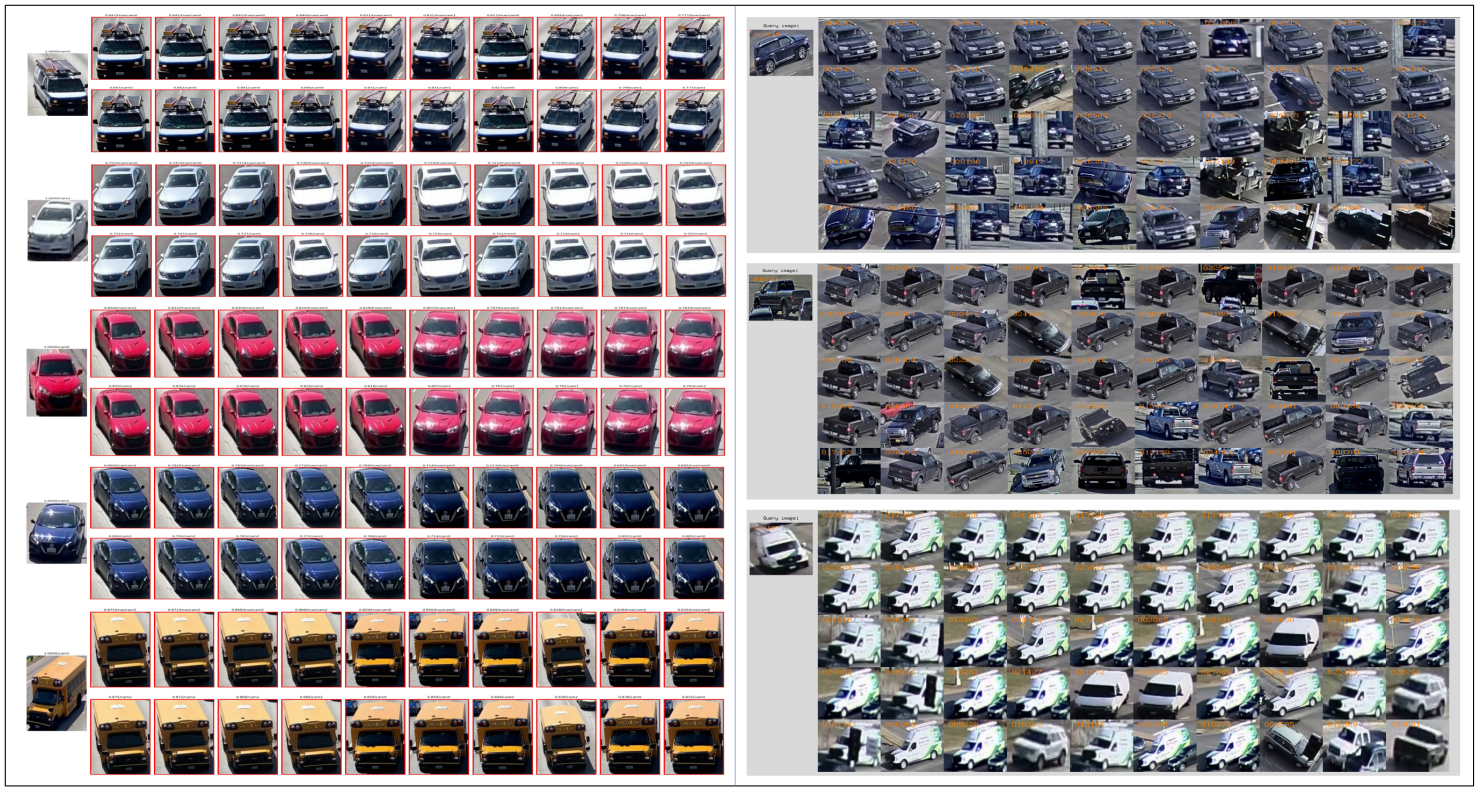


Figure 3.7: The result visualization for the proposed ECoMS_Re-ID algorithm on the FSV and Cityflow dataset.

Type	Method	Accuracy			
		Rank-1 ↑	Rank-5 ↑	mAP ↑	Infer time (s/item) ↓
FSV dataset	BoT	80.03	82.92	69.58	0.00314
	AGW	80.55	84.05	68.22	0.00353
	BoT_ibn	82.57	85.23	71.68	0.00315
	SBS	82.08	85.95	72.62	0.00247
	ECoMS_Re-ID	**90.14**	**93.72**	**77.01**	**0.00241**
Cityflow2021	BoT	85.45	87.86	70.11	0.00221
	AGW	86.74	88.98	72.31	0.00257
	BoT_ibn	88.79	90.58	75.57	**0.00167**
	SBS	88.13	90.72	75.04	0.00196
	ECoMS_Re-ID	**92.43**	**94.87**	**80.09**	0.00189

Table 3.3: The result comparison for the proposed ECoMS_Re-ID algorithm with other state-of-the-art baselines.

the Rank-1, Rank-5, mAP and inference time (second per batch). ECoMS_Re-ID framework significantly outperforms other SOTA methods on both datasets, especially on the Rank-1 accuracy. On the FSV dataset, the ECoMS achieves 90.14% Rank-1 and 93.72% Rank-5, with the best mAP accuracy. As mentioned before, to extract link traffic information, Rank-1 accuracy is the most important measurement need to be improved. The customized attributes-aware multi-query and re-ranking mechanism play an essential role in enhancing the Rank-1 accuracy. Such a result indicates that the ECoMS Re-ID results can sufficiently support traffic information estimation.

Ablation Study for ECoMS Vehicle Re-ID

For the proposed clip-based ECoMS Re-ID algorithm, a whole new clip-level feature extraction and an attributes-aware re-ranking components are integrated into the proposed Re-ID model. Besides, some popular techniques are also used in the model customizing and training process [211], including freeze backbone, learning rate warm-up, and learning rate cosine decay. The

Dataset	Freeze Backbone	LR Warm up with Cosine Decay	Attributes-aware Re-ranking	Clip-level Feature Fusion	Rank-1 ↑	mAP ↑
FSV Dataset	✔	✗	✗	✗	80.07	69.91
	✔	✔	✗	✗	83.93	71.24
	✔	✔	✔	✗	85.46	72.49
	✔	✔	✗	✔	87.34	75.92
	✔	✔	✔	✔	**90.14**	**77.01**
Cityflow 2021	✔	✗	✗	✗	83.35	72.01
	✔	✔	✗	✗	85.74	74.49
	✔	✔	✔	✗	88.24	77.08
	✔	✔	✗	✔	91.76	79.17
	✔	✔	✔	✔	**92.43**	**80.09**

Table 3.4: The ablation study for the proposed ECoMS cross-camera Re-ID algorithm.

ablation study for each components' contributions to the final result are summarized in Table 3.4. From the before and after comparison, the clip-level feature fusion and attributes-aware re-ranking play significant roles in improving the Rank-1 and the mAP in ECoMS cross-camera Re-ID. The leaning rate warms up with cosine decay, and freeze backbone settings also contribute to the cross-camera Re-ID accuracy.

3.3.4 Traffic Information Estimation Evaluation

The final target of ECoMS is to sense the information at multi-level of traffic networks, including node, link and local area. Here, I mainly evaluate the link and network level of traffic information estimation, including link travel time and speed, as well as network OD flow distribution.

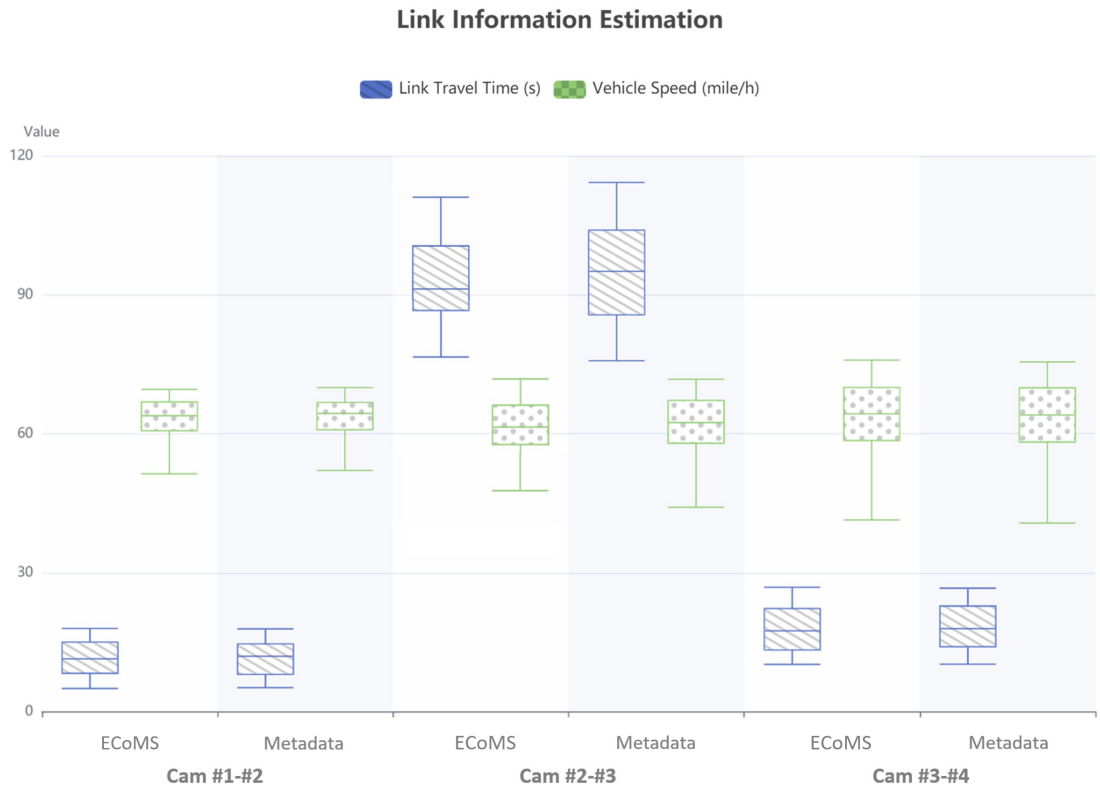


Figure 3.8: The result visualization and comparison with ground truth data for the ECoMS link travel time distribution and average speed distribution estimation.

Link Information

The link speed and travel time are two key parameters to be estimated. Different from the traditional method, for ECoMS, I estimate the distribution instead of the average value. The Figure 3.8. shows the box plot of metadata and estimated distribution. From the result, a similar distribution (with less than 1.01 KL distance) is observed for both travel time and speed for three camera link. For the link travel time, the distribution of Cam#1-#2 and Cam#3-#4 are obtained 0.09, 0.11 KL distance respectively, which can be treated as the same. For the Cam#2-#3, due to two exits are located in the road section, as well as the road network graph constraint, the es-

timated distribution are more concentrated compared with the metadata. However, the general distribution is quite close (KL distance 1.01 for link travel time distribution and 0.94 for link average speed distribution). Such an accuracy level can satisfy all kinds of challenging traffic-related services inducing travel time reliability analysis, road network resilience analysis and etc.;

Area OD Flow Distribution

Area OD estimation is a long-lasting challenge for traffic perception. The previous methods mainly rely on vehicle GPS data captured by the on-board device. However, the GPS tracking equipment always with the complex installation procedure. The GPS shifting in the urban area is also a serious challenge, which decreases the accuracy of network OD estimation [215]. However, with ECoMS, it can be used to kill two birds with one stone. The ECoMS can not only answer how many vehicles are passed on each node but also where they will go in with less than 2% distribution error. As shown in Figure 3.9, the ECoMS can obtain the route trajectory data by continuous Re-ID vehicles and then summarize the traffic flow distribution. With the road graph constraint combination, the traffic engineers can obtain the area OD with high precision.

3.3.5 System Performance Comparison

Due to the ECoMS is the first proposed IoT system for network-level traffic sensing framework by multi-camera Re-ID, I mainly compared with two other current frameworks, video-based framework and motion-trigger-based framework. For the previous framework, the basic structure is the cameras collect and transmit the original real-time video to the TMCs. Then the TMCs deal with the original video and then extract the traffic information by implementing MOD, MOT and multi-camera Re-ID. Such a framework is the current most popular structure

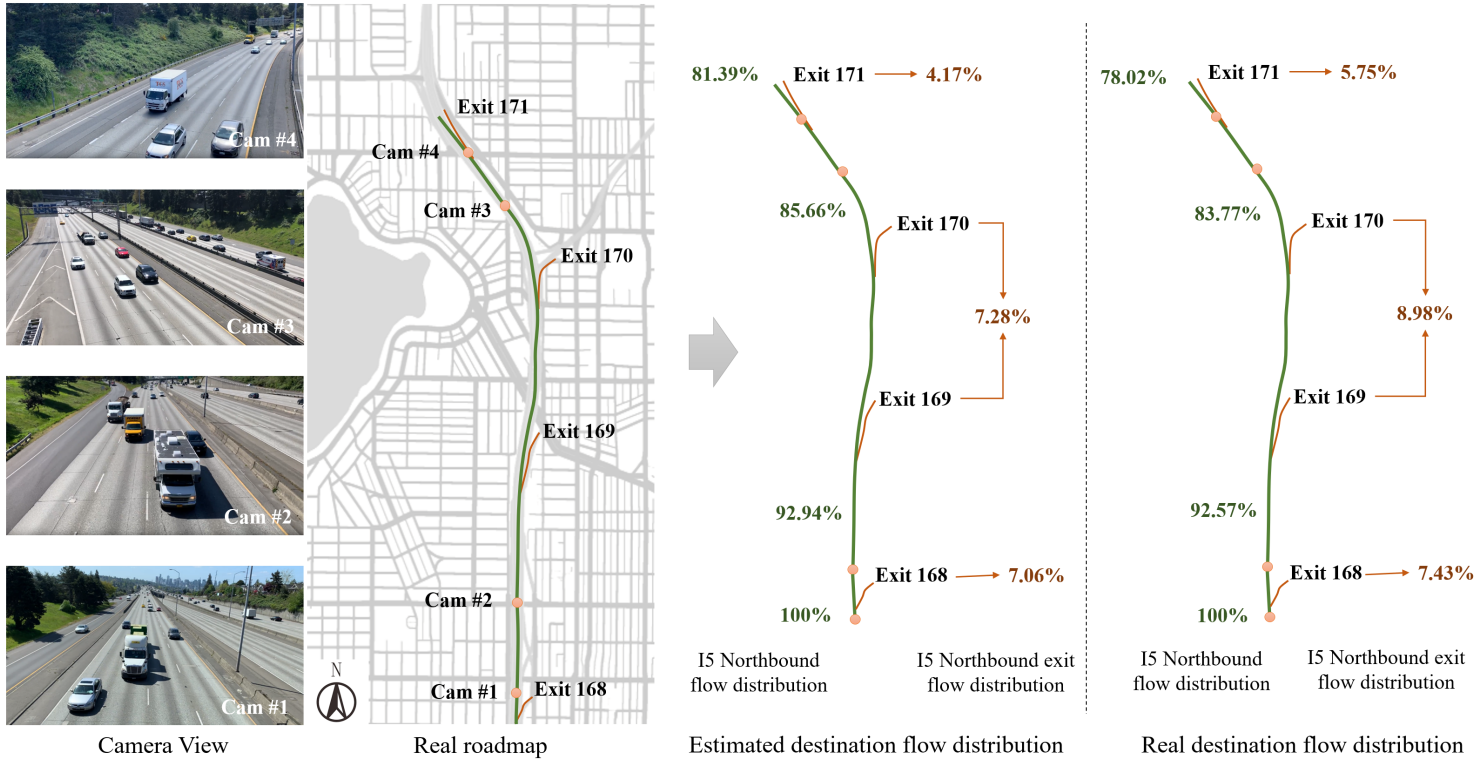


Figure 3.9: The ECoMS system area OD estimation visualization and comparison with the ground-truth data.

and implemented by pioneer researchers [176]. The motion-trigger-based methods are proposed in recent years [216, 217]. By running basic motion-driven detection or tracking methods on the edge node, the TMCs can receive video clips with target objects (i,e, vehicles, pedestrians and etc.). Then post-processes the video clips by the same workflow. Such a framework can reduce the communication stream, especially in rural areas. However, only triggering by motion detection can not be sufficient for many traffic sensing tasks, including incident detection and monitoring, traffic counting, speed and occupancy estimation. So if a comprehensive traffic perception is necessary, streaming live videos and post-processing in the back-end servers are still the only choice for traffic agencies.

In this research, I conducted a comparison with video-based and motion-trigger frameworks by conducting same workflow in MOD and MOT. For the vehicle Re-ID component, due to the ECoMS_Reid is customized and can not suit for the video based Re-ID framework, I using the SOTA video-based SBS Re-ID framework in the workflow. The detail system information can be found in Table 3.3.5.

Table 3.5: System information for three multi-camera traffic monitoring architecture.

Name	Video-SBS	Motion-SBS	ECoMS
Hardware	two Titan Xp	two Titan Xp	four Jeston Xavier and one Titan Xp
Archt.	online MOD and MOT, Re-ID	online MOD, offline MOT, Re-ID	online MOD and MOT, Re-ID
Sys Power (watt)	846W	825W	4*34.2W+402W
Cameras	4	4	4
End-to-end Latency	198.98s	42.74s	10.21s

3.3.6 Communication Efficiency

The communication band-with requirements are always the traffic engineers worried about when anywhere is needed to install surveillance cameras. Even with motion-trigger transmission, the volume of data is still quit high when the traffic condition is quit busy and the stream need high performance transmission system. Such video-based framework significantly limited the vehicles Re-ID utility in transportation information collection. However, with the ECoMS based on IoT structure, useful objects representations can be well selected by edge nodes, only small amount of data need to be send to TMCs for cross camera Re-ID. From the experiment, the average data stream is only 8.1% of video method and 24.4% of motion-triggered IoT system. The ECoMS makes traffic data in the rural and low communication band-width area also can enjoy better transportation services by SOTA IoT technologies. The Figure 3.10. shows the detail communication data stream volume comparison of the three workflow.

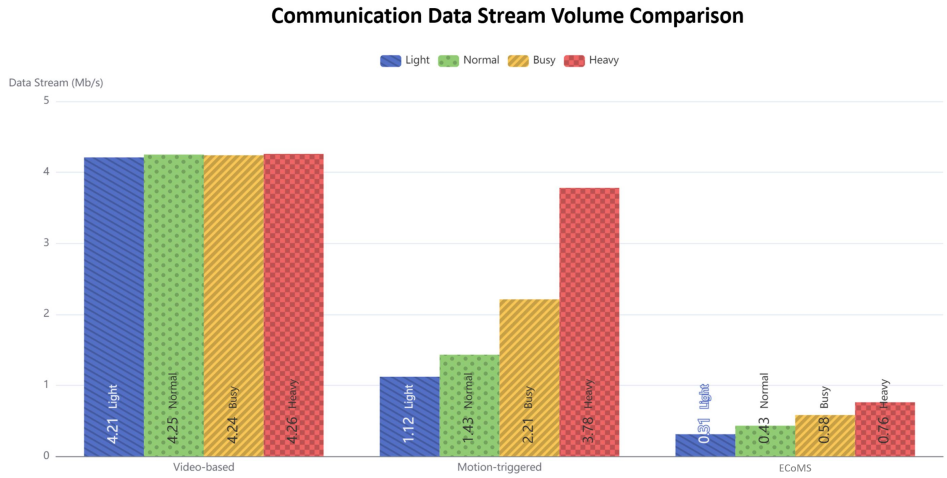


Figure 3.10: The transmitted data stream volume comparison for the three multi-camera traffic monitoring architecture.

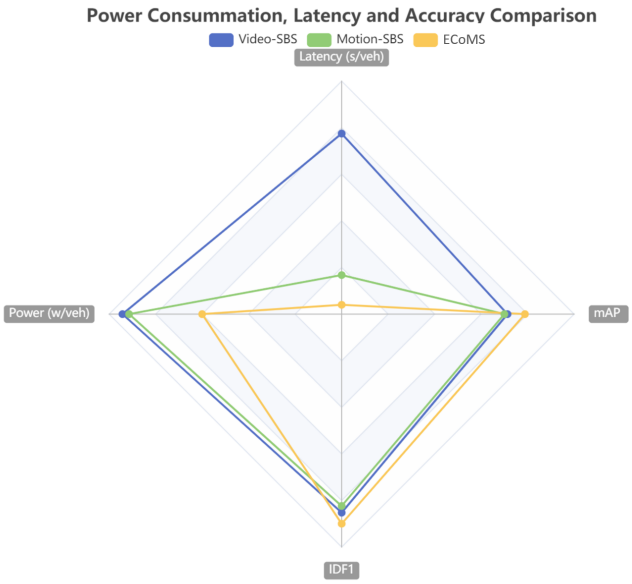


Figure 3.11: Power consumption, computational latency and cross-camera Re-ID performance comparison for three multi-camera traffic monitoring architecture.

With the help of edge and center IoT structure, the power consummation, computational latency can be much more saved based on the experiment. Here, I evaluated the computational latency from the video clip input to obtain the traffic information estimation results, and then average the latency on each vehicle. I found that the traditional video-based method are with the highest latency due to the server is in charge of multiple video workflow simultaneously. Even two Titan Xp is installed, such a framework is the highest latency with the maximum power consummation. Due to selecting the video clips with vehicles, the motion-SBS can significantly reduce the computational time, but the power consummation is still close to the video-based method. The ECoMS performs only 5% of the latency, 63% of the power consummation, with the highest Re-ID accuracy. The Figure 3.11. shows the detail comparison of the three workflow.

3.3.7 System Scalability and Real Implementation

When the traffic managers consider actual deployment, scalability becomes one of the most significant issues for consideration. For the video sensing system, the department not only need to pay for the hardware system, but also the fee of database, communication services, power supply, and periodical maintenance. Here, I compared the ECoMS with current video-based multi-camera sensing system [172, 176]. From the comparison with current video-based multi-camera perception system, it is easy to find that ECoMS has multiple advantages. For the original video processing framework, using Titan Xp as the standard server GPU, one GPU can only deal with upto two video streams. A huge amount of cost needs to be spent on GPUs and data storage. The motion-trigger method can reduce around half of the communication and data volume, but the cost is higher when nodes are limited. Compared with the previous two frameworks, ECoMS has four advantages. 1) Due to the hybrid Iot architecture, ECoMS can save much more money on the server hardware, including both GPUs (only about 30%) and data storage (the vehicle clips only). 2) Easy to scale up. If the manager installs more nodes, much more money will be saved. 3) The communication requirement is easy to reach, and flexible. Only daily bandwidth can fit into the system. 4) The power consummation is much more user-friendly. The total cost will be only about 35% or less if twenty nodes are included in the ECoMS. Considering the number of cameras installed in the current surveillance system for traffic monitoring, the ECoMS is a solution with high reliability, low price, and easy management.

With the mentioned advantages, the proposed ECoMS has been installed and being tested by multiple DoTs and will be the key information source for the link travel time estimation. Specifically, the installations of the proposed ECoMS system in Oslo, the capital of Norway, at late 2021, and City of Bellevue in Washington State, USA, at April of 2022 have been finished.

At the late April of 2021, three ECoMS edge nodes was installed at the E-6 freeway, for the travel time estimation from Kløfta (small town located in Ullensaker, Akershus, Norway) to Oslo. At April of 2022, four ECoMS edge nodes was installed at four intersections in the city of Bellevue and another two are planed to install at the city of Lynwood, in the Washington State, at Nov of 2022. Currently, all of the seven installed ECoMS smart nodes are being tested for the cross-camera vehicle tracking and link travel time estimation and work as I expected.

3.4 Chapter Summary and Future Works

In this research, I proposed an IoT workflow for cooperative multi-camera vehicle tracking and traffic surveillance. By optimizing the computational resources on the edge device and adapting and accelerating the multi-object detection and tracking algorithms, the ECoMS edge node can achieve the real-time object detection and tracking even under heavy traffic condition. In addition, in order to realize re-identify objects across different cameras, the representations of the vehicle can be automatically extracted on the edge and sent to the server. Furthermore, a novel clip-based deep vehicle Re-ID model, with the hierarchical feature extraction and fusion mechanism is proposed and integrated into the ECoMS system workflow. The experiments on two vehicle Re-ID datasets, the Freeway Sensing Video (FSV) and Cityflow 2021 achieves 90.14% and 92.43% rank-1 accuracy respectively, which significantly outperforms other SOTA methods 4%-8%. Besides, the cross-camera traffic parameter and distribution estimation algorithms based on KED and KS are also fully tested and evaluated, and the distribution difference can achieve less than 1.01 KL distance. Further, the hardware cost of ECoMS system is significantly reduced including the GPUs (only with 25% of the original GPUs) and data storage (with less than 10% of original data volume). With the help of the ECoMS system, the traffic surveillance

system can share the detected object features and cooperate with each other to track vehicles and extract traffic information together, which can provide valuable regional data sources for both travelers and traffic agencies.

The ECoMS system is not perfect. Firstly, the current ECoMS still has some labor work that needs to be done manually in the real deployment. These tasks include setting the region of interest in each camera view, selection of some parameter thresholds (e.g., tracklet length in tracking), and manually pairing road network information with cameras, etc.. Although these parameters are not difficult to set, they also the inconvenience to the public agencies. Then, some privacy considerations is raised up by the DOTs when track the vehicles across multiple cameras. Although the current cross-camera matching mechanism using in ECoMS is based on the representative images (without the licence plate information) of the target vehicle extracted on the edge device, transmitting the vehicle images can still bring some privacy disputes (for example, in the Nordic region). In addition, the data collection of the camera is sometimes affected by the environmental lighting and weather conditions. Although the environmental diversity of deployments has been taken into account when training the deep learning-based cross-camera Re-ID model, the ECoMS system still faces some unhandled situations in practice, for example, the serious camera glare during the sunrise. Besides, during the snow days, obvious appearance vehicle features are sometimes covered by the white snow and cannot be accurately extracted and matched, etc. The mentioned hurdles and practical challenges, but not limited to them are part my future work.

The future research beyond the ECoMS can be divided into two types of tasks, the optimization of edge-server cooperative system and the downstream research using the ECoMS sensing outputs. The first task is to further strengthen the edge computational load and optimize the

edge-sever cooperative workflow for reducing the privacy concerns. For example, deploying the Re-ID based features extractor on the edge nodes, then only sent the uninterpretability deep features to the server for cross-camera matching. Currently, since the feature extraction backbones used in cross-camera Re-ID are often very deep and heavy (i.e., ResNet-50, ResNet-101, etc.) [156], how to accelerate and optimize the neural network with the edge device computational constraints are very worth the time and effort. Besides, a variety of cameras and edge nodes, such as Unmanned Aerial Vehicle (UAV) cameras and on-board cameras, as well as thermal cameras can be further integrated into the ECoMS to build up a more inclusive surveillance system. Furthermore, after obtaining accurate cross-camera information, it will be very interesting topics to develop downstream control and evaluation algorithms for traffic road networks, such as travel time reliability assessment, and road network accessibility prediction. In addition, there are some longer-term studies worth considering. For example, the regional OD information captured by the deployed ECoMS can be used to verify the validity and reliability of the residents' household survey data; or to mine the regional correlation of the traffic road network to alleviate traffic congestion during peak hours.

Part II

Learning Representations for Traffic Prediction and Operation

4

Chapter 4. Predicting the User's Customized Trip Speed by Learning the Trajectory and Individual Attributes Representations

This chapter is modified from the published work:

- H. Yang, C. Liu, M. Zhu, X. Ban and Y. Wang*, "How Fast You Will Drive? Predicting Speed of Customized Paths By Deep Neural Network." in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 3, pp. 2045-2055, March 2022, https://doi.org/10.1109/TITS.2020.3031026

Amid the worldwide rapid urbanization process, human beings endure daily traffic congestion. Recently, thanks to significant advancements in data collection and processing technologies [218], predicting traffic trends and future conditions has become a common strategy for transportation managers and travelers. Traffic parameter prediction methods are undergoing rigorous research and are progressively applied in real scenarios. Undeniably, speed prediction is a paramount aspect.

In the realm of traffic speed prediction, previous research has been categorized into three tiers: road segment [219, 220], corridor [221], and network [222, 223, 224, 225]. All these domains have received considerable attention over the past two decades. In every prediction algorithm, the fundamental unit of prediction is a road segment, the value of which typically represents average velocity. Such methods aid traffic managers and engineers in illustrating and analyzing the overall traffic status. However, individuals tend to be more interested in the traffic status of their particular routes. Traditional macroscopic speed prediction algorithms fail to efficiently inform personal path planning. It's notable that many commercial navigation systems suggest the optimal route based on real-time segment estimation data [226, 227]. For instance, leveraging real-time travel time and speed estimation, most commercial GPS navigation systems like Google Map can provide functions including optimal travel path selection, real-time road condition visualization, and optimal departure time recommendations. The so-called "real-time best route" offered by these apps may not be the truly best route due to the time delay from the users checking the app to their arrival at the target road segment. In reality, many people are guided by navigation software to "currently smooth road segments" only to subsequently experience sig-

nificant traffic congestion [227]. As a solution, I proposed the development of a traveler-centric individual level traffic information prediction algorithm.

In line with the progress in data science and deep learning in transportation, researchers are increasingly focusing on custom-scaled prediction algorithms, aimed at equipping both managers and travelers with more useful predictive information [224]. Customized path speed prediction is exceptionally challenging as it is influenced by numerous complex factors, such as spatial and temporal elements (e.g., locations, peak/non-peak hours, departure time), human elements (e.g., drivers' behaviors, driving habits), internal conditions (e.g., vehicle performance), and external conditions (e.g., weather, road conditions). Traditional speed prediction methods struggle to perform efficiently under these conditions. Hence, a novel algorithm is needed that can not only integrate all the aforementioned factors but also accommodate a new prediction scale. Traditional speed prediction methods typically employ the road segment as the basic prediction unit. However, in this chapter, I introduced a new deep neural architecture for customized path-based speed prediction to better assist both traffic managers and individual travelers.

In the novel method, I proposed a new fundamental unit – path cell for speed analysis and prediction, which is flexible and customized based on a given path across both spatial and temporal domains. Every path is composed of a chain of custom path cells linked end-to-end. The speed within a path cell is influenced by various factors such as location, departure time, weather, and road conditions. I proposed a deep neural network based on hierarchical Convolution Neural Network (CNN) and deep Long Short-Term Memory (LSTM) – termed the Path-based Speed Prediction Neural Network (PSPNN). I implemented the PSPNN to a real open dataset for testing in this chapter, and the results validate the impressive capabilities of the PSPNN

4.2 Model Preliminary

4.2.1 Definition

Definition 1 Historical Trajectory (H_i): a historical trajectory H_i is a sequence of continuous historical GPS points generated by vehicle i with a fixed length of time gap, i.e., $H_i = \{h_1, h_2 ... h_{m-1}, h_m\}$. Each point record of h_m includes latitude, longitude and timestamp. In this research, the trajectory is the basic information for the proposed model.

Definition 2 Path (P_i): a passable pathway with given origination, destination, and route on the map. In this research, P_i is generated by users, and it is the fundamental entity for researchers to predict the speed sequence information. Each P_i includes multiple path cells and sub-paths.

Definition 3 Path cell ($C^n_{P_i}$): the customized isometric sections of each path (P_i). Path cell is the basic path speed prediction unit. N represents the cell number and belongs to the P_i ($0 < n < j$) and $Dis_{C_{P_i}}$ is the distance of each path cell.

Definition 4 Sub-path ($S^N_{P_i}$): sub-path is the combination of k adjacent path cells. Sub-path is used to reflect and summarize and the neighboring path cells connections and relationships. The sub-path length equals to $k \cdot Dis_{C_{P_i}}$.

4.2.2 Research Objective

Randomly given a customized path (path) (P_i) on a map with origination, destination, path starting time, route and attributes information, then predict the speed ($V_{C^n_{P_i}}$) belonging to each path cell ($C^n_{P_i}$).

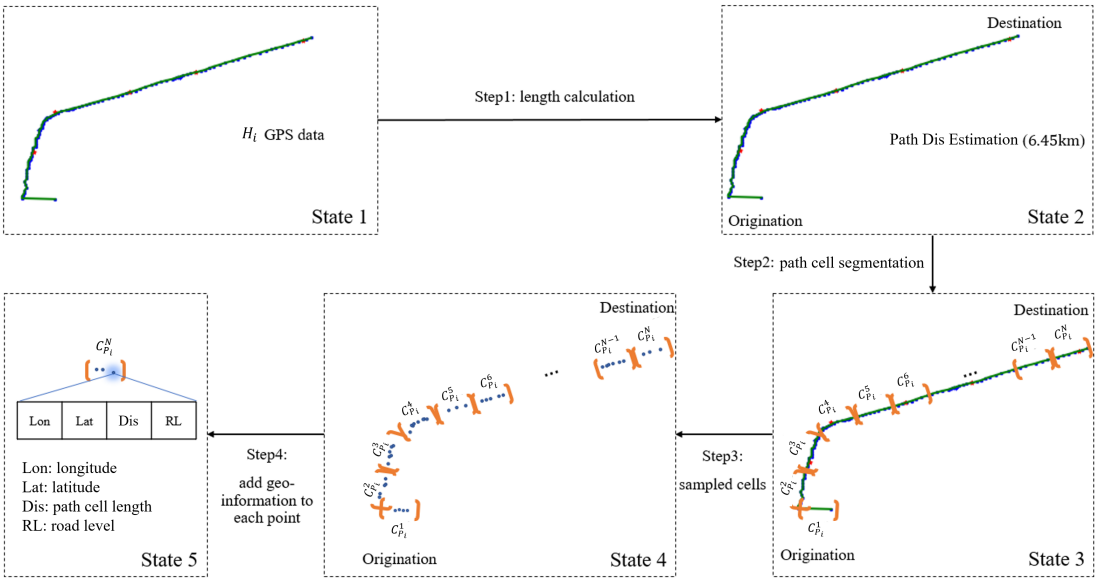


Figure 4.1: Data pre-processing, using one real path as an example

4.2.3 Data Pre-processing

In this research, path information and the path cell information are obtained from the trajectory data. The historical trajectory (H_i) GPS points are the original records holding the spatial-temporal information. Based on the Definition part, after one historical trajectory obtained, the first step is to calculate the length (Dis_{P_i}) of H_i. Based on the Dis_{P_i}, the second step is to divide H_i into n GPS points clusters. Each cluster covers the same distance, which is treated as path cell length. The third step is to sample each cell sequence cluster and make the total records equal to j (if the sequence is not long enough, add 0 to each cell record). Generally, GPS data is recorded by a relatively fixed time gap. Here, I re-sampled the data set since:

- To avoid network learning error (e.g., just predicting the speed based on counting the GPS record numbers);

- To save computing resources and increase training speed (some of the path records including more than 3k GPS records);
- To improve the model applicability to both long and short customized path speed prediction.

Using the sampled GPS points in each isometric cluster represent each path cell ($C^n_{P_i}$). The fourth step is to add the cell information (cell length $Dis_{C_{P_i}}$) and the geographic information (road level) to each point. An example of the overall data pre-processing process is shown in Figure 4.1, for Vehicle ID: 00f8c1861b883d4a9bde0babcbbe2f96.

After the data pre-processing, I obtained the path P_i, consists of j records representing for n path cells, and each data record in the path cells $p^j_{C^n_{P_i}}$ includes longitude, latitude, the customized cell length $Dis_{C_{P_i}}$, and the road level (RL), showing in equation 4.1.

$$p^j_{C^n_{P_i}} = (Long, Lat, Dis_{C_{P_i}}, RL) \tag{4.1}$$

4.2.4 Attributes

In this research, attributes information is an important input of the model, including: weather ($weatherID$), week date ($weekID$) and driver's info ($driverID$). These attributes are presented as attributes set, represented as A, defined in equation 4.2. Here, weatherID represents weather conditions, such as rain, snow, or clear, and weekID indicates the week day from Monday to Sunday. DriverID is used to distinguish the drivers.

$$A = (weatherID, weekID, driverID) \tag{4.2}$$

Besides, some path attributes are summarized including:

- starting time: in the unit of second, over a range of (0, 86400), as a day is divided into 86400 intervals in the chapter.
- current average speed of different road levels (update every one minutes).

4.3 Representations Extraction and Fusion Architecture

The architecture of PSPNN is shown in Figure 4.2. It essentially consists of two modules: the Feature Extraction Module (FEM) and the Memory Module (MM). The FEM is responsible for extracting and converting features of a particular path. There are four parts in this module: the Cell-Conv layer, the Sub-path Conv layer, the Path-Conv layer and the attributes embedding. The MM is used to memorize and summarize connectivity of cells. The basic structure of this module is deep Bi-LSTM. Then the outputs of the two modules will input in four fully-connected layers to map the vector matrix into prediction speed sequence.

4.3.1 Feature Extraction Module (FEM)

Cell-Conv Layer

The Cell Convolution layer is used to capture features including in a sequence of path cell records. It is responsible for converting the raw sequence to a series of feature vectors. Inspired by the Geo-conv Layer of the DeepTTE model proposed in 2018 [228], the Cell-Conv Layer is used to capture the microscopic spatial correlation among consecutive trajectory cell records. The detailed architecture of Cell-conv is shown in Figure 4.3. Here, a non-linear mapping is applied to integrate the three parts into the Cell-Conv layer, a 1-D convolution layer and con-

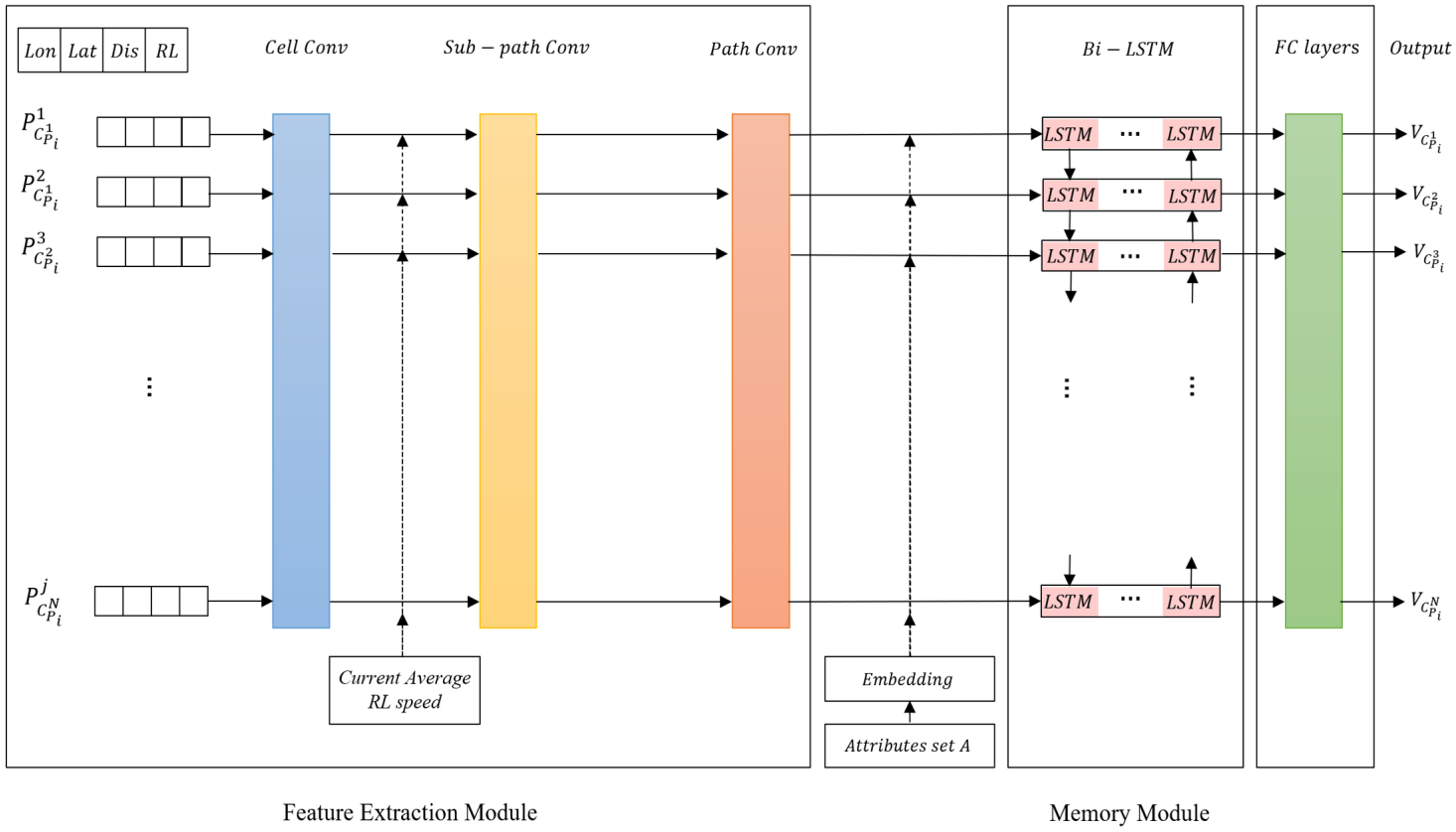


Figure 4.2: Path-based Speed Prediction Neural-Network (PSPNN) Architecture

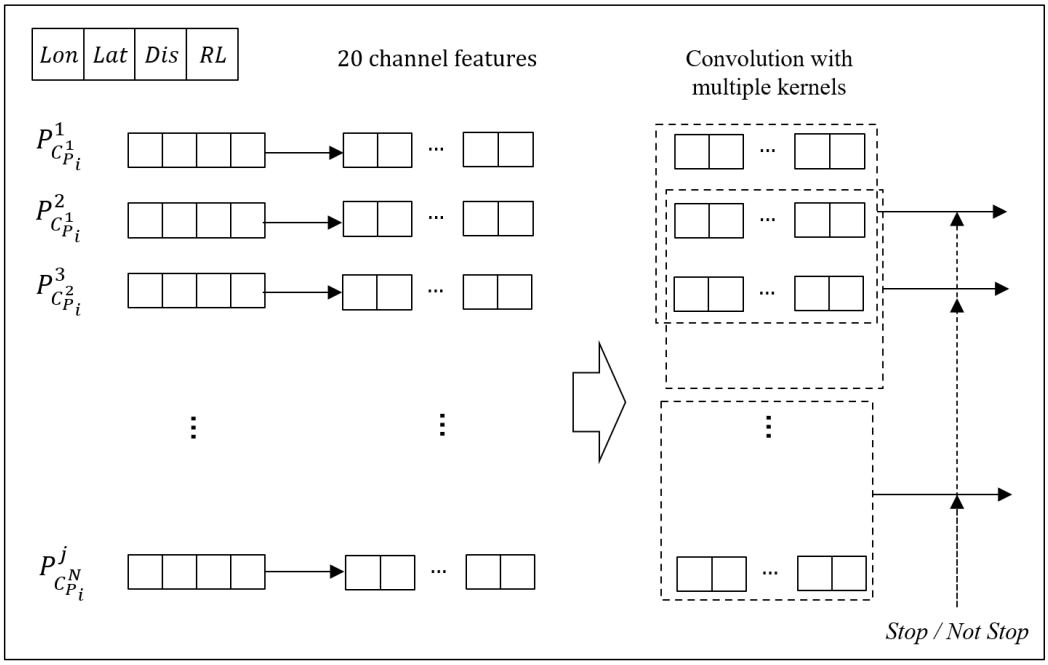


Figure 4.3: The structure of the Cell-Conv Layer

catenate. Before the convolution calculation, I mapped the input vector $p^j_{C^n_{P_i}}$ from R^4 to R^{20}. This step is used to enlarge the records dimension for better records feature extraction. Thus, the input sequence of the convolution layer can be treated as 20 channels. The mathematical approach of the Cell-Conv layer is in the following:

$$\mathfrak{f}_{p^j_{C^n_{P_i}}} = tanh(W_{geo}[long \circ lat] \circ W_{dis}[Dis] \circ W_{rl}[RL]) \tag{4.3}$$

$$\mathfrak{f}^{cell}_{p^j_{C^n_{P_i}}} = \sigma_{cnn} \cdot (W_{conv} * \mathfrak{f}_{p^j_{C^n_{P_i}} : p^{j+k_c-1}_{C^n_{P_i}}} + \varepsilon) \tag{4.4}$$

In the equation 4.3, the $\mathfrak{f}_{p^j_{C^n_{P_i}}}$ means the output of the non-linear mapping of output. The W_{geo} W_{dis} W_{rl} are learnable weight matrix. The non-linear mapping method used *tanh* function. In the equation 4.4, $*$ represents the convolution calculation with kernel size k. ε is the bias item. $p^j_{C^n_{P_i}} : p^{j+k_c-1}_{C^n_{P_i}}$ is the input sequence of a path from records j to $j + k_c - 1$. The k_c is the kernel size of the convolution layer. σ_{cnn} represents the corresponding activation function of CNN. The output of the convolution layer $\mathfrak{f}^{cell}_{p^j_{C^n_{P_i}}}$ shows as equation (4).

Sub path Conv Layer

Generally, a vehicle's speed on a certain road section is always affected by the adjacent road sections. Therefore, it is necessary to use the adjacent road segments' traffic information to assist the algorithm in predicting in the target cell. Here, a Sub-path convolution layer is designed to capture the features among adjacent cells. Here, before transmitting the cell convolution layer's output into the Sub path convolution layer, the road network average speed of different road levels AS^j_{rl} is concatenated to each path cell at the trip time. This is aimed to help the network

capture the current traffic status better. And the output of sub-path convolutional layer is $\mathfrak{f}^{sub}_{p^j_{C^n_{P_i}}}$ in following equation 4.5 and 4.6:

$$\mathfrak{f}^{insub}_{p^j_{C^n_{P_i}}} = (\mathfrak{f}^{cell}_{p^j_{C^n_{P_i}}} \circ AS^j_{rl}) \tag{4.5}$$

$$\mathfrak{f}^{sub}_{p^j_{C^n_{P_i}}} = \sigma_{cnn} \cdot (W_{conv} * \mathfrak{f}^{insub}_{p^{j:j+k_s-1}_{C^n_{P_i}}} + \varepsilon) \tag{4.6}$$

Path Convolution Layer

The path convolutional layer is used to capture a high level of path features across several cells. Here, I used a 1-D convolution layer to finish the feature extraction. The math equation of path convolutional layer is in equation 4.7:

$$\mathfrak{f}^{path}_{p^j_{C^n_{P_i}}} = \sigma_{cnn} \cdot (W_{conv} * \mathfrak{f}^{sub}_{p^{j:j+k_p-1}_{C^n_{P_i}}} + \varepsilon) \tag{4.7}$$

Attributes set and embedding

The traffic system can be treated as a subsystem of the modern social system. The parameters will be affected by many general factors. Travel patterns on holidays and weekends also vary greatly from working days. Furthermore, the travel speed can be different in different weather conditions. To address this, the model includes and integrates the attributes set into the path speed prediction. Here, the attributes set includes weatherID (rainy, snowy, sunny, etc.), weekID (from Monday to Sunday), departure time ID and the driverID.

However, the values of the attributes are always categorical values and not satisfied with the

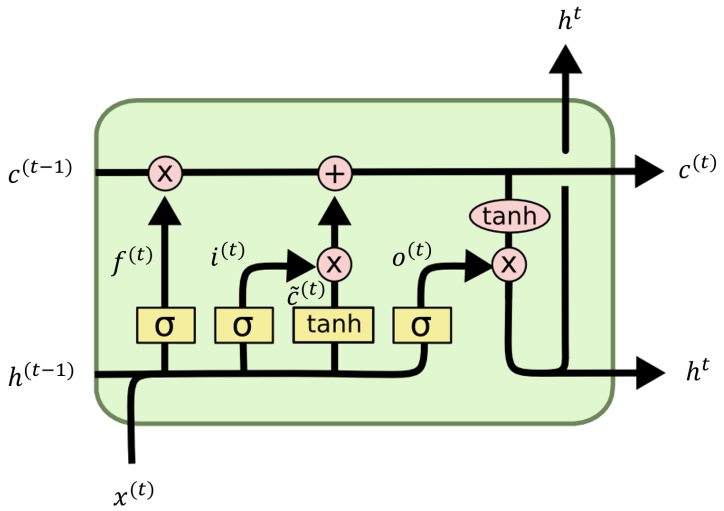


Figure 4.4: The structure of the LSTM neural [1]

neural network input format. A learnable procedure is necessary to address the problem since the impact of the attributes to the output is always complicated and associated. Inspired by feature learning techniques in natural language processing (NLP), mapping the words or phrases from the vocabulary to vectors of real numbers, embedding becomes a bridge to connect these discrete values to a vector dimension. In the framework, I adopted the low dimension embedding method proposed by [229] to transform categorical factors into a neural network input sequence. Using $E(A)$ represents the four attributes vectors. The overall output of the memory module is:

$$\mathfrak{f}^{FE}_{p^j_{C^n_{P_i}}} = \mathfrak{f}^{path}_{p^j_{C^n_{P_i}}} \circ E(A) \tag{4.8}$$

4.3.2 Memory Module (MM)

The change in velocity between different traffic cells can be interpreted as a change in the spatial and temporal domain. After the feature extraction module extracts and summarizes the spatial features, a module with time memory properties is needed. LSTM is a well-known network to capture the temporal dependencies among these cells. The LSTM is a special form of RNN, which is being capable of learning long-distance dependencies and handling long-term memory information. Generally, each LSTM neural is contains three gates, which are input gate $i^{(t)}$, output gate $o^{(t)}$ and forget gate $f^{(t)}$ inside a neural. Each gate is controlled by their own weight $w^{(t)}$ and the previous neural output $h^{(t-1)}$. Also, the memory cascading process can be divided into two parts: new memory generation $\widetilde{c}^{(t)}$ and final memory generation $c^{(t)}$. After the last memory is generated, the new hidden state $h^{(t)}$ is raised by the control of output gate $o^{(t)}$. $o^{(t)}$ makes the assessment regarding what parts of the memory need to be shown in the $h^{(t)}$. Figure 4.4 shows the detailed structure of LSTM neural and the mathematical formulations of 4.9 to 4.14 show the LSTM units working procedures in the following [1]:

$$i^{(t)} = \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) \tag{4.9}$$

$$f^{(t)} = \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) \tag{4.10}$$

$$o^{(t)} = \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) \tag{4.11}$$

$$\widetilde{c}^{(t)} = tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) \tag{4.12}$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \widetilde{c}^{(t)} \tag{4.13}$$

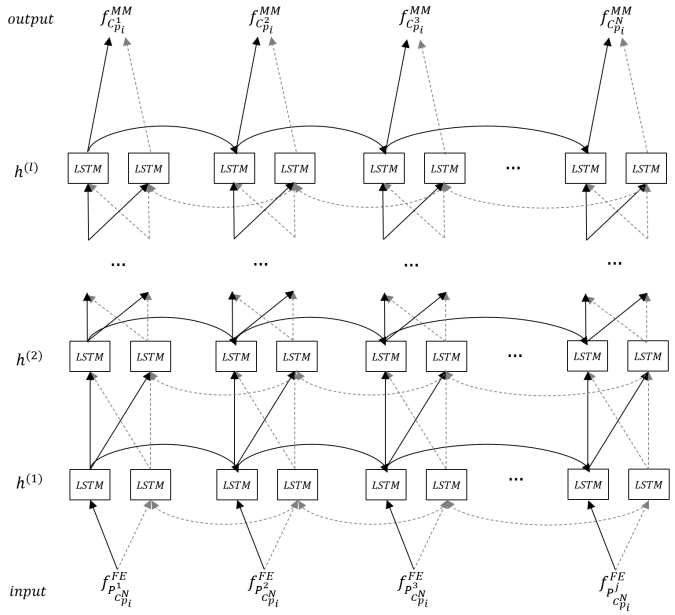


Figure 4.5: Deep bidirectional LSTM neural network architecture

$$h^{(t)} = o^{(t)} \circ tanh(c^{(t)}) \tag{4.14}$$

To capture the time dependency better, PSPNN uses deep bidirectional LSTM as the memory module. Each training sequence is passed the LSTM neural from two directions, one forward and another backward. The mathematical formulations of deep bidirectional LSTM are showing below:

$$\overrightarrow{h_t^{(i)}} = f(\overrightarrow{W^{(i)}}\, h_t^{(i-1)} + \overrightarrow{V^{(i)}}\, h_{t-1}^{(i)} + \overrightarrow{\varepsilon^{(i)}}) \tag{4.15}$$

$$\overleftarrow{h_t^{(i)}} = f(\overleftarrow{W^{(i)}}\, h_t^{(i-1)} + \overleftarrow{V^{(i)}}\, h_{t+1}^{(i)} + \overleftarrow{\varepsilon^{(i)}}) \tag{4.16}$$

$$\hat{y^{(t)}} = g(\mathcal{U}[\overrightarrow{h_t^{(i)}}; \overleftarrow{h_t^{(i)}}] + \hat{\varepsilon}) \tag{4.17}$$

The Figure 4.5 and the equations 4.15 to 4.17 show the bidirectional hidden layer architecture parameters updating process. Generally, the bidirectional LSTM consists of two parts, the forward sequence and backward sequence. During the calculation process, the network will compute the forward hidden sequence $\overrightarrow{h_t^{(i)}}$ and the backward hidden sequence $\overleftarrow{h_t^{(i)}}$. Finally, the output integrates both forward and backward hidden features and summarizes into one sequence as output.

To narrow down the dimension of memory module output and obtain the cell speed sequence, multiple Fully Connected (FC) layers are used here. These layers' task is to map the high dimension feature sequence to a specific scale of value. Based on the mapping results, the predicted value of each path cell $V_{C_{P_i}^n}$ can be obtained.

4.4 Experiment

4.4.1 Environment Description

The PSPNN was implemented with PyTorch 0.3.1. The work station for training was equipped with a GPU (NVIDIA TITAN Xp) and the CPU is Intel Core i7 8700. The operating system is Linux Ubuntu 16.04.

4.4.2 Data Description

Overall description of the research dataset: Didi GIYA data from Chengdu, China. The original data size was about 132 GB and contained more than two billion trajectory records from November 1st, 2016 to November 30th, 2016. There are 9,050,774 trips in total and an average of 301,699 orders per day. After the data cleaning, the effective path travel time ranges between

3.25 and 59.97 minutes. The trip distance is ranging from 0.12 km to 35.74 km. The GPS points recording time gap is 2-4 seconds.

Also, I match the trajectory with the Chengdu city road network to extract the road level information. Here, I divided the road into five different RL as input, including: freeway, arterial streets, sub-arterial streets, collector's streets and local streets.

4.4.3 Parameter Setting

The parameters in the PSPNN experiment are as follows:

- The customized cell amount n in PSPNN for paths is fixed as 128. The customized point records amount j in PSPNN for paths is fixed as 256. The average length of each cell is 59.52m.
- In the Cell-Conv layer, the kernel size k_c is fixed as 3. The number of filters is set as 128. The activation function in Eq (4) σ_{cnn} is Exponential Linear Unit (ELU) [230] function, which can converge cost to zero faster and produce more accurate results. Also, ELU has an extra alpha constant which should be positive number.
- In the sub-path convolution layer, I fixed the kernel size as 2. Therefore, the sub-path is composed of three adjacent path cell clusters and represent the transmission and connection among the two neighboring cells clusters. The number of filters of the sub-path convolution is set as 128. The activation function σ_{cnn} is ELU function.
- In the path convolution layer, I fixed the kernel size as 3. The number of filters of the sub-path convolution is set as 96. The activation function σ_{cnn} is ELU function.

- The size of the embedding vector for each attribute is settled in the following: weatherID mapping into R^3, weekID mapping into R^3, departure time ID mapping into R^{16} and the driverID mapping into R^{10}. The total dimension size of $E(A)$ is R^{32}.

- In the memory module, the number of hidden neural in the bidirectional LSTM is fixed as 256. Here, three hidden layers are used in the memory module. The activation function in Eq (9-11) σ_{rnn} is *tanh* function. The mathematical expression of *tanh* is $tanh(x) = e^x - e^{-x}/e^x + e^{-x}$

- In prediction module, the number of fully connected layers is fixed as 4. The four layers down sample to the 128 dimension vector to represent as the predicted cell speed ($V_{C^n_{P_i}}$).

4.4.4 Loss Function

In PSPNN, I used the mean absolute percentage error (MAPE) to train PSPNN model. And then using the mean absolute error (MAE), root mean squared error (RMSE) and MAPE together to evaluated the model. The loss function and evaluation methods for the cell speed prediction are defined in the following equations:

$$\text{MAE} = \frac{1}{N} * \sum_{n=1}^{N} |V_{C^n_{P_i}} - \hat{V}_{C^n_{P_i}}| \tag{4.18}$$

$$\text{MAPE} = \frac{1}{n} * \sum_{n=1}^{N} |\frac{V_{C^n_{P_i}} - \hat{V}_{C^n_{P_i}}}{V_{C^n_{P_i}} - \varepsilon}| * 100\% \tag{4.19}$$

$$\text{RMSE} = \sqrt[2]{\frac{1}{N} * \sum_{C=1}^{N} (V_{C^n_{P_i}} - \hat{V}_{C^n_{P_i}})^2} \tag{4.20}$$

In the equations 4.18 to 4.20, the $V_{C^n_{P_i}}$ represents the predicted speed value of each path cell.

The $\hat{V}_{C^n_{P_i}}$ represents the ground truth speed of each path cell. n is the cell number. In equation 4.19, ε in MAPE is used to prevent the 0-denominator appearance and ε is small enough compared with the average road network speed.

4.5 Performance Evaluation and Comparison

The training process is reasonable. The MAPE training curve dropped slowly and converged to a limit at 18.14%. During the evaluation, the MAE is down to 1.94 m/s (6.98 km/h). Additionally, to better show the performance of PSPNN, the research also comparing with a traditional method (AVG) and other cutting-edge deep neural network architecture, including deep RNN, deep LSTM, deep Bi-LSTM, deep Conv-LSTM, and PSPNN without embedding. The results are summarized in Table 4.1. Also, the parameters of each deep neural network using in the comparison are shown in the following items.

- AVG [231]: Calculating average speed for each path is the most traditional method for path-based speed prediction. In this method, I calculated the average speed of each path and used the average speed as the predicated cell speed. Then the paper estimates the errors based on real path cell speed and the average speed.

- Deep RNN [222]: RNN can use internal memory units to process arbitrary sequences of inputs, and thus grants the RNN the capability of learning temporal sequence. In the comparing process, I used a primary recurrent neural network (RNN) to predict cell speed and compared it with the real path cell speed. And the neural number of RNN is set as 256 with three hidden layers.

- Deep LSTM [221]: For comparing, I used deep LSTM neural network to predict the cell

speed directly. Here, I set the number of hidden units as 256 and in three hidden layers. Then based on the LSTM output, the paper estimates the errors based on real path cell speed and the predicted cell speed.

- Deep Bi-LSTM [223]: For comparing, I also used deep Bi-LSTM neural network to predict the cell speed directly. Here, I set the number of Bi-LSTM hidden unites as 256 and in three hidden layers. Used Bi-LSTM neural network to predict the cell speed directly and then estimate the errors.

- Deep Conv-LSTM [232]: Combined both CNN and LSTM as a new neural network into Convolutional LSTM (Conv-LSTM) is tested here. Comparing with traditional LSTM, convolutional layers are used for feature extraction, which can extract not only the temporal correlation but also the spatial correlation. Here, I used a 1D-CNN and a deep LSTM network to capture both temporal and spatial features. For the 1D convolution layer, I set the kernel size equals to 3 and the number of filters is 128. Then, passed a three hidden layers deep LSTM with 256 hidden units.

- PSPNN without embedding: in this research, I implemented the embedding methods into the path based on speed prediction and try to integrate features such as driver behavior, day of week, and the weather impacts. To see the actual influence, I trained and evaluated a PSPNN without the attributes of information embedding.

From the comparison of table 4.1, I can see that the PSPNN prediction accuracy is significantly better than other methods. Comparing with the traditional deep RNN and LSTM, the feature extraction module do extract more useful information. Also, I found that the convolution layer do help the spatial features extraction since the Conv-LSTM performs around 9.94% comparing

Table 4.1: PSPNN Performance Comparison with Other Deep Neural Network

Method	MAE (m/s)	RMSE (m/s)	MAPE
AVG	4.37 (15.44km/h)	6.32 (25.76km/h)	87.98%
Deep RNN	3.32 (11.95km/h)	5.14 (18.50km/h)	52.78%
Deep LSTM	3.13 (11.27km/h)	4.77 (17.17km/h)	47.89%
Deep Bi-LSTM	3.07 (11.05km/h)	4.58 (17.49km/h)	46.02%
Conv-LSTM	2.89 (10.40km/h)	4.22 (15.19km/h)	37.95%
PSPNN (no embed)	2.13 (7.67km/h)	3.36 (11.38km/h)	19.96%
PSPNN	**1.94 (6.98km/h)**	**2.98 (10.73km/h)**	**18.14%**

with only using the same hidden layers of deep LSTM network. More detail analysis about the PSPNN components can be found in the next section.

4.6 Result Discussion and Analysis

4.6.1 Integration of Attributes Representations

In real life, people's travel is always affected by the many external factors such as weather, departure time, day of the week and etc. How did these factors impact the prediction result? Here, I did an evaluation with and without the attributes set, and I found that the full attributions information with embedding methods using in the PSPNN can improve the overall prediction MAPE of 1.82%. Among the four attributes, the weekID shows the most significant contribution to the result with a 1.02% improvement. Then, the weatherID and the timeID provide with 0.91% and 0.73% error decrease for the MAPE. The influence of the driverID is neglectable, which shows about 0.05%-0.08% contribution to the MAPE accuracy. The result shows that the driving habit still not be fully found out and integrated into the neural network. A more detailed attributes to represent driver's behavior is needed in future research.

4.6.2 Cell Convolution Layer

For the cell convolution layer, I tested multiple parameter combinations. For the kernel size k_c, I tried different values $k_c = 2, k_c = 4, k_c = 5$ and the prediction MAPE were worse than $k_c = 3$. Because the cell convolution layer is mainly used to extract the spatial dependency in the small cell clusters. It is obvious that the speed in a path cell is highly dependent on the former and latter cells. Also, the vehicle 's speed change will show an influence on the neighborhood cells. The same result of the kernel size attempt was mentioned in the [228] Geo-Conv layer.

Sub-path and Path Convolution Layer

For the sub-path convolution layer, I finally set the kernel size $k_s = 2$. Here, I also tested several different kernel sizes that $k_s = 3, k_s = 4$, and I obtained different MAPE results as 19.31% and 20.23%. The possible reason leading the result is that the sub-path convolution layer is mainly responsible for capturing the higher dimension features, especially for the connections to each cell clusters. The $k_s = 2$ potentially shows that the speed change in a path cell is closely to the neighborhood cells status. Also, for the path convolution layer, I set the kernel size as 3. I also tried the different kernel size $k_p = 2, k_p = 4$ and $k_p = 5$. I found that the MAPE are 21.07%, 19.85%, 19.98%.

I also tried different number of filters in the three different convolution layers. After considering the prediction performance and network efficiency, I finally decided the number of filters as showing in the section VI.

4.6.3 Memory Module

In the PSPNN, the memory module is a significant part of capturing the temporal dependency for the speed changes among different cells. The research tries different models and different hidden layers, and finally decides to use Bi-LSTM since its performance is the best. I can infer this result from the table 4.1 from the comparison of different types of RNN. For the hidden layers, I tried the $l = 2$ and $l = 4$. I found that when $l = 2$, the MAPE result (19.04%) is pretty close to the best one. When the $l = 4$, the MAPE is 18.09%, whose improvement can be neglected. However, increasing one Bi-LSTM layer with 256 hidden neural will lead into huge parameter growth. So, I finally decided to use three hidden layers Bi-LSTM as the memory module.

4.6.4 Path-based Speed Prediction Case Analysis

It can be seen from the plots that the PSPNN has captured the changes in the temporal and spatial aspects of the entire path speed in Figure 4.6. Here, I randomly selected six vehicles and six testing paths, of which three in non-peak hours and three in the peak hours. The x-axis indicates the cell index (from 0-127) and the y-axis is the speed value (m/s). The blue line shows the real speed captured by the historical trajectory records and the orange line shows the predicted cell speed of the path. Form Figure 4.6, I can see that the overall forecasting trend is very close to the actual speed of the path, especially when the driving speed is close to the mean road network speed . The IDs of the testing vehicles' paths in Figure 4.6 are:

- (a) 612aba8f428371175142de88293cd43f,
- (b) 664721206a2abc0cf15e567565ebd826,
- (c) 0f5ce372be702eae7b85bfb3afcc534e,

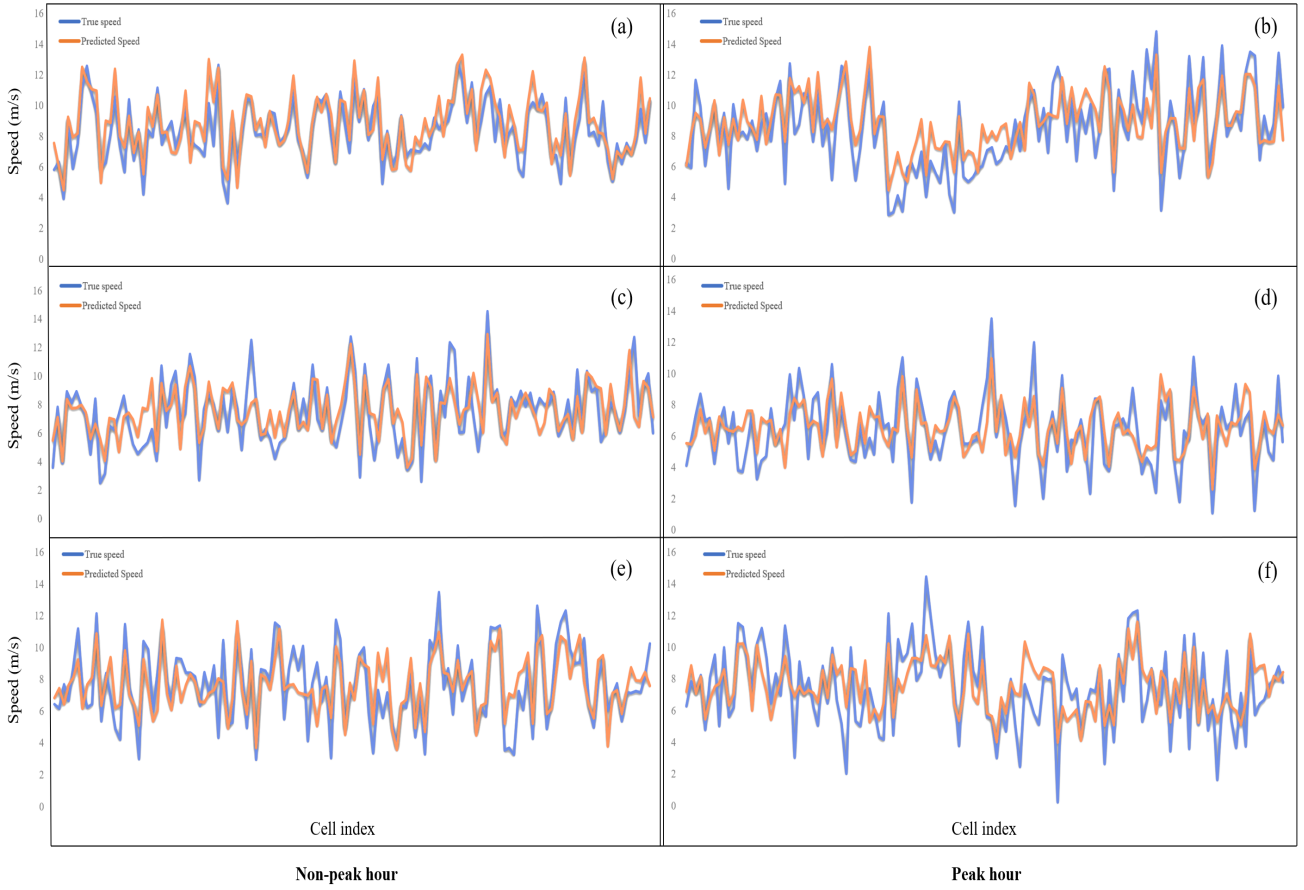


Figure 4.6: Comparison of predicted path cell speed and ground truth of six random paths.

- (d) 90bf9e76e133c0199689 4e9041a74381,
- (e) a9e84ad69afadd53e6a3545259b29d7c,
- (f) 70b24f91b809d7b84bedb16c20258334.

The success of the PSPNN shows that the combination of PSPNN has an outstanding performance in modeling complicated spatial-temporal features.

However, several findings still worth for an open-minded discussion. The first and foremost is that the cells, usually with a higher speed is easy to predict comparing with the low speed cells (under 4m/s). The details can be found from the Figure 4.6. This results could be explained by the following points. Firstly, in the road network, the stable high travel speed always indicates the high spatial dependency. Generally, in the situation, the road surface and the transportation infrastructure are both in good conditions. Also, the speed limit and the level of services of the road segments are higher. These obvious spatial information can be treated as useful features and learned by the neural networks. Also, these features can be linked with the velocity change well. Secondly, in the congested situations, due lots of sudden speed drops accumulated in different road segment, it is difficult to capture the acceleration and deceleration feature effectively through the network. From the result analysis, the overall network prediction of cells with higher speed is better than that at lower speeds. In summary, the PSPNN performs not good enough when there are many acceleration and deceleration happened like peak hour periods.

4.6.5 Departure Time Analysis

In the road network, paths with different starting times always lead to different real speed distribution. In order to evaluate and demonstrate the speed prediction accuracy of PSPNN in

Table 4.2: Analysis of PSPNN for Different Departure Time and Error

Departure Time	MAE (m/s)	RMSE (m/s)	MAPE (%)
5:00-7:00	1.85±0.05	2.89±0.05	17.19±0.2
7:00-9:00	2.31±0.3	3.32±0.2	21.13±0.4
9:00-11:00	1.93±0.1	2.92±0.1	18.19 ±0.25
11:00-13:00	1.91±0.05	2.93±0.05	19.06 ±0.2
13:00-15:00	1.89±0.05	2.95±0.05	17.76 ±0.2
15:00-17:00	2.01±0.15	3.02±0.1	20.01 ±0.25
17:00-19:00	2.55±0.4	3.45±0.5	25.31 ±0.5
19:00-21:00	1.90±0.1	3.01±0.04	17.28 ±0.2
21:00-23:00	1.87±0.05	2.89±0.05	17.45 ±0.2

different time periods, parts of the test data sets (3 days) are divided into different time period according to the starting time. Table 4.2 shows the specific results. It can be seen that PSPNN has the ability to predict travel speed around the clock. During the off-peak hours, PSPNN can achieve an accurate travel speed prediction, whose error fluctuation can reach 2-3%. The highest prediction accuracy time domain is during 6:00-7:00. The MAPE is 17.14% and the MAE is 1.83 m/s. However, during the morning peak (7:00-8:00) and the evening peak (18:00-19:00) hours, the prediction error of PSPNN is relatively large. The MAE during the morning peak period is 2.34 m/s, and the MAPE is 22.25%. The MAE is 2.64 m/s in the evening peak period, and the MAPE is 26.02%. The speed prediction errors are relatively large in the morning and evening peak hours. The possible reasons are: 1) serious traffic congestion, 2) due to changes in signal lights, accidents and other external factors, the road network predictability deteriorates, and 3) the acceleration and deceleration are more frequent than other time period whose features are hard to capture.

4.6.6 Path Length Analysis

Table 4.3: Analysis of PSPNN for Different Path Distance and Error

Dis(km)	MAE (m/s)	RMSE (m/s)	MAPE (%)
0-5	2.17±0.15	3.65±0.01	23.19±0.15
5-10	1.83±0.1	2.92±0.05	18.11±0.1
10-15	1.89±0.1	2.96±0.05	18.59 ±0.1
15-20	1.97±0.15	3.06±0.1	21.49 ±0.15
20-25	2.31±0.2	3.35±0.2	26.76 ±0.25
25-30	2.45±0.2	3.83±0.2	29.31 ±0.3
30+	NA	NA	NA

As in Table 4.3, I analyzed the trip to some extent, the accuracy level of PSPNN depends on the lengths of trajectories. For short-distance paths within 5 km, the prediction error of PSPNN is the relatively large (MAPE 23.19% ± 0.15%, MAE 2.17 m/s ± 0.15 m/s). The possible reasons can be summarized as three points: 1) There are few sampling points for paths less than 5 km, and the size of the re-sample is not sufficient for the trajectory characteristics. 2) The speed distribution of shorter journeys is often affected by signal lights, real-time road conditions, so the statistical patterns are not obvious. 3) Some Short path records are always with a higher level haphazardness and uncertainty. For paths length are in 5 km-10 km and 10 km-15 km, that people travel most frequently in the urban area, PSPNN performs excellent prediction results and the error is stable. For the 5-10 km paths, the prediction result (18.11%± 0.1%, 1.83 m/s ± 0.1 m/s) is the best. When the path length increasing, the prediction accuracy drops slowly since each cell length becomes longer. For the path over 30 km, the size of data in both training and testing sets is so limited that I exclude this part from the experiment.

4.7 Chapter Summary and Future Works

Individual path-based traffic parameter prediction is a necessary research scope in the transportation field. In this chapter, I present a path-based speed prediction based on deep neural network structure PSPNN for a given path by integrating spatial-temporal and attributes information. With the combination of individual and systematical features, path-based methods can predict flexible and useful information for both travelers and managers. This work narrows down the minimum prediction unit to path cells and achieves an encouraging and promising speed prediction results.

In practice, the path-based speed prediction method proposed in this research can be served for many applications, including e-hailing taxi dispatch optimization, autonomous vehicle route planning, congestion avoidance and etc. Meanwhile, since the PSPNN narrows down the research scope to the individual path level, human behavior and preference can not be neglected. In the real world, human beings are the host of the transportation system. Therefore, to improve the prediction accuracy with human features and interactions can be a future research topic. More useful information will be integrated and extracted for prediction based on the path scope in the future.

5

Chapter 5. Predicting the Parking Pattern for Trucks and Multi-model Hubs by Learning Attributes and Spatial-temporal Representations

This chapter is modified from the following published works:

- H. Yang, R. Ke, Z. Cui*, Y. Wang*. and K. Murthy, 2022. "Toward a real-time Smart Parking Data Management and Prediction (SPDMP) System by Attributes Representation Learning." International Journal of Intelligent Systems, 37(8), pp.4437-4470. https://doi.org/10.1002/int.22725

- H. Yang, et al. "Truck parking pattern aggregation and availability prediction by deep learning." IEEE Transactions on Intelligent Transportation Systems 23.8 (2021): 12778-12789. https://doi.org/10.1109/TITS.2021.3117290

5.1 Challenges and Motivations

Currently, long-last parking challenges attract more and more human-being's attention. In urban area, "cruising for parking" is becoming a common problem for many curbside spaces and parking lots, which always causes congestion and additional air pollution [233, 234]. Given the limited parking facilities supply and the increasingly growing car ownership, the parking issue is particularly severe in cities' hot-spot areas where it is extremely hard to build new parking spaces. The situation results in incredibly high cost in time and others regarding drivers searching for parking, which costs Americans $73 billion a year [235]. Meanwhile, parking shortages and challenges are not only very serious for cars in the city region, but also for trucks. Although not reflected by parking research, the unbalance of supply and demand for the truck parking has become a common concern [236]. Taking USA as an example, in 2015, the total tonnage of trucks was 10.49 billion tons, accounting for 70.1% of the total tonnage of the domestic freight industry [236, 237]. Among the transport, 30%-40% are long-distance shipments (more than 750 miles) with over 50% [238] cargo values. In order to complete such long-distance driving tasks safely and efficiently, truck drivers need adequate and high-quality rest. No doubt, timely truck parking information would be necessary and valuable for them.

To solve the ubiquitous parking challenges for cars and trucks, the first and foremost approach is to make full use of existing parking resources instead of building new ones. In this case, disseminating real-time and future parking availability information to drivers would significantly help them schedule their stops at parking facilities. Therefore, the Parking Information Management System (PIMS) has been gradually used to improve the parking efficiency and utility [239, 240, 241, 242, 243, 109]. PIMS involves typically three major components: parking detec-

tion, data processing, and information dissemination [241, 244, 109]. First, real-time parking data is collected by parking sensor technologies. Then, the sensor data is processed to calculate the current parking availability. Finally, the real-time parking utility information is disseminated to drivers via approaches such as dynamic message signs, websites, mobile apps, etc.

With the deployment of PIMS, high quality parking data can be collected. Then, data-driven methodologies for solving parking issues are being developed and investigated. Among them, parking occupancy prediction is a meaningful topic for both driver and manager. Generally, parking utility prediction can be summarized into two categories [110, 245, 190, 246], the traditional approaches and machine learning based approaches. Traditional models include many classical statistical models, such as the Markov process and Kalman filter prediction. These models' core idea is to fit the relationship among the parking occupancy distribution and other required variables by simulation data or actual collected parking data. Under normal circumstances, these models can show an intuitive analysis of the overall parking characteristics from a macroscopic perspective, and can provide managers with meaningful conclusions.

However, in real life, there are so many factors showing impacts on the parking utility. The complex features sets can be summarized into three categories, the spatial-temporal set [245], the attributes information set (i.e., weather impact, hour of a day and day of a week) [247], and the driver's behavior set (i.e., parking duration information, parking habit, etc.) [110]. The integration of these factors with heterogeneous formats into the traditional prediction methods becomes a mission impossible. Luckily, the rapid development of machine learning algorithms can spark inspiration by giving researchers a new way of looking at the parking availability prediction features selection and integration.

In state-of-art studies, some effective feature extraction backbones are widely used in traffic in-

formation prediction research, and encouraging and promising results have been achieved. Recurrent Neural Networks (RNN) [248, 249], especially the Long short-term Memory (LSTM) network, are very popular models for temporal pattern extraction and parameter prediction [250, 251, 252]. Compared with the normal neural network, RNN can use the elements of sequential data as input, and generate outputs that solely dependent on the previous computations. To achieve traffic prediction tasks, these networks need to be well-customized to successfully capture the spatial-temporal dependencies.

The cutting-edge learning-based models perform well in certain scenarios. However, their limitations are also obvious that several key challenges need to be overcome. The first and foremost one is the attributes information integration. Previous studies indicated that the attribute information has a significant impact on the parking pattern [94, 233, 234, 109]. However, different from the spatial-temporal record, the attributes information always has heterogeneous values with different formats, including category values, clusters, and even text information. If using these features directly as the network input, these characteristics' impact will be greatly limited and even trigger negative impacts [253, 254, 255]. Besides successfully integrated the attributes features into training process, balancing attributes and historical impacts and fusing features to a reliable prediction result need to be investigated. An attributes-aware attention mechanism is necessary to be integrated. The second challenge is the limited scalability issues. Usually, neural networks are designed with a particular motivation, especially in parking problems. It is difficult for different sensor systems to share the same predictive model, which is especially obvious in deep learning approaches. Therefore, a systematic prediction scheme that shares common input and output and can serve all sensor systems (centralized and decentralized) on the market is a necessity. The third hurdle is the limited transferability of the current spatial-temporal neu-

ral networks. As known, the neural network prediction result is trained based on the ground truth data. If a new parking lot with limited parking data needs to be integrated into the predicting framework, the whole model needs to be retrained or even rebuilt, which is very time consuming and inconvenient. These limitations seriously restrict the field deployment of the neural network-based parking prediction algorithms.

Therefore, in this research, I targeted to solve the three main challenges from both the theoretical and practical aspects. Representation learning is introduced and integrated into the model to better capture the attributes features and increase the flexibility of prediction framework. Instead of using original information as input, the goal of representation learning is to learn a representative features and measure the correlations among objects feature sets by transferring the raw data into a high dimension vector space [256, 257]. The success of the method is tested and implemented by many computer science topics such as Natural language processing (NLP) [258, 259], Computer Vision (CV) [260, 261, 172] and Recommendation System (RS) [262, 263]. Thus, with the heterogeneous feature embedding and representation learning, the author can better capture and combine the spatial-temporal features, attribute features, and the customized information provided by the users (i,e, the arriving time) into back-propagation training process. Such a modeling structure can make the results more universal and accurate for truck parking and urban parking. Instead of using raw sensor data as the neural network input, a general pre-processing workflow is designed using the summarized parking lot occupancy sequences as the prediction input. With the feature learning process, the transferability of the prediction framework is greatly improved. Also, inspired by the broad learning architecture [264, 265], the proposed prediction framework takes the future flexibility and applicability into account, which can easily integrate the attributes features with different formats and pre-

dict the target based on various external condition inputs instead of only using spatial-temporal information.

5.2 Smart Parking Information Management and Prediction (SPIMP) System Illustration

In this research, I proposed a general SPIMP system for current private and public parking lots with the demand on parking utility monitoring and forecasting. The SPIMP system includes four sub-components: sensing component, data processing component, attributes information collection component and prediction component. Detail procedures illustration can be found in Figure 5.1. All the components is discussed in the following sections.

5.2.1 Sensing Component

The sensing component plays a role as the eyes and ears in the intelligent parking system, which is used to detect the status of the target parking region and then summarize into occupied/non-occupied status. In general, the current parking sensor system can be divided into two types of approaches: decentralized (space-by-space) sensing and centralized (entrance/exit-based) sensing architecture. Examples can be found in Figure 5.2. The decentralized sensing always consists of multiple sensors installed on parking slots to monitor the slot status, each sensor is in charge of one or more slots. The slot-based sensor sensing, i.e. radar senor, magnetic detector, laser detector, and video sensor. Under the circumstance, the decentralized system is more accurate and reliable, but usually in a high cost with a complex installation process since the parking sensors need to be mounted at each parking slot. Another centralized sensing system is more straightforward. The sensors are always installed at the parking lot entry/exit way, and the numbers of

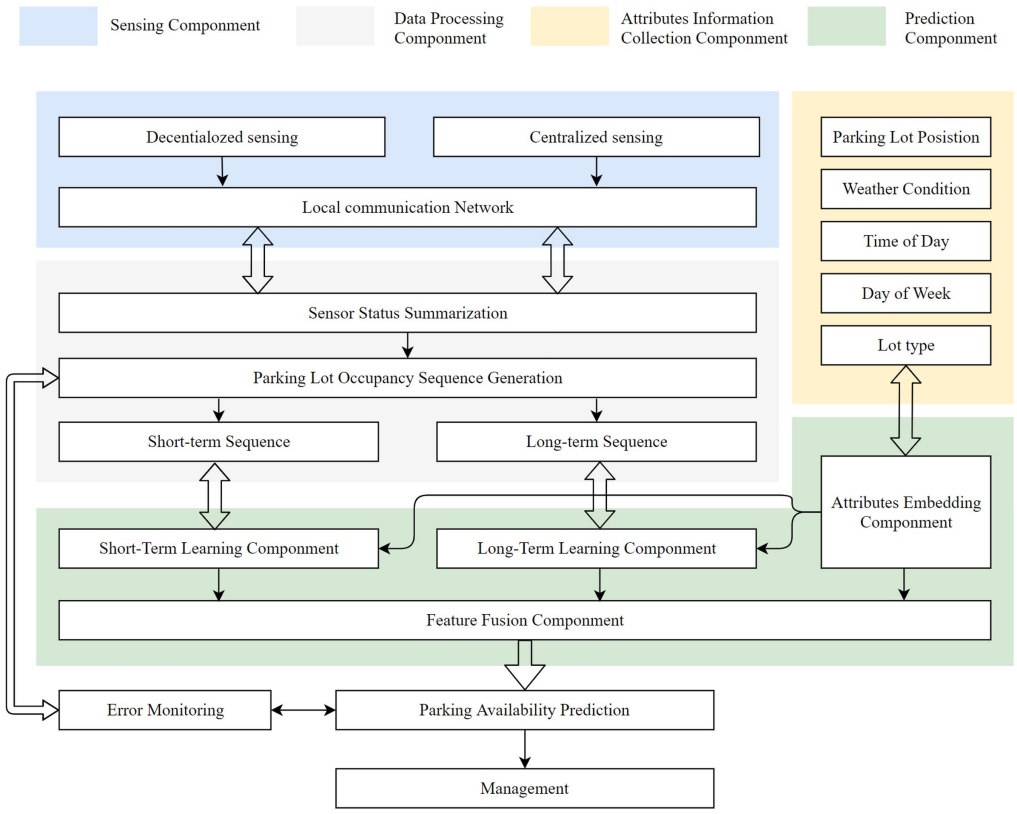


Figure 5.1: Overall framework of the parking availability prediction system design, four components are included: sensing, data processing, attributes information collection and prediction component.

driving in and out of the lot can be directly obtained. Compared with the slot-based approach, the entry/exit system is cheap and easy to deploy but may lead to traffic congestions. However, slot level occupancy information is not feasible. The one-by-one pass of the parking lots sometimes becomes a bottleneck. Also, if the sensing result is not very reliable, the error accumulation be a big problem when using the entrance/exit count approach.

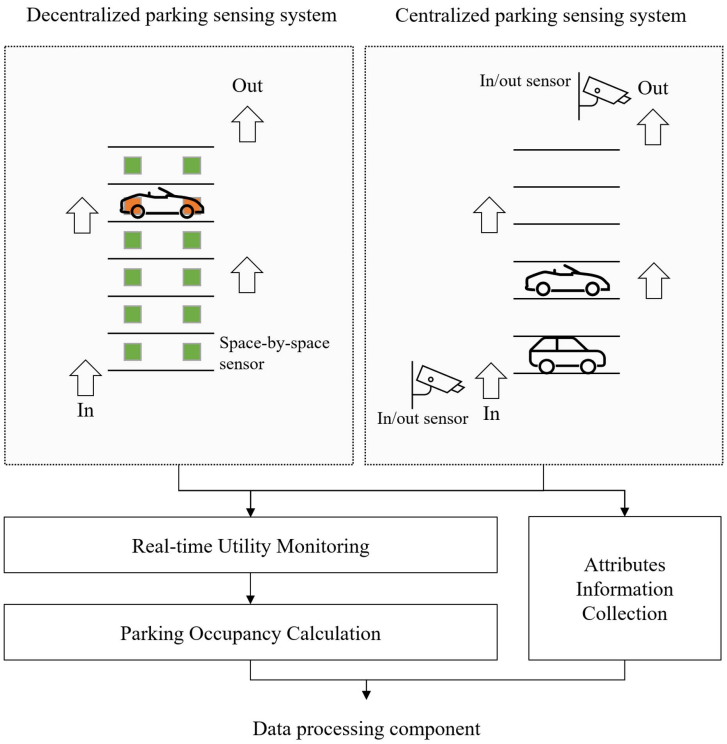


Figure 5.2: Illustration of centralized and decentralized parking sensing system architecture. Both system can support the proposed SPIMP system.

5.2.2 Data Processing Component

After the sensing component sends the occupancy status to the local server, the post-processing procedure is started to prepare the input sequences. In general, the decentralized sensing status, as well as the centralized sensing results, are summarized into the overall parking lots occupancy. Then, two sequences, one short-term real-time parking sequence (R_n^i) and long-term historical parking occupancy sequence (H_m^i) are calculated and ready to use.

5.2.3 Attributes Information Collection Component

Parking activity is highly impacted by the status of the attributes including, time of day, day of week, weather condition, the parking lot location and etc. Collected such information in a real-time manner and fully integrated the information into the prediction system is necessary. Here, the author sets up an attributes information collection component served for capturing the necessary external factor, including parking lot type (business, working, shopping, rest area, etc.), weather condition (fair, cloudy, light rain, rain, etc.), time of day, day of week and level of roads (freeway, arterial streets, local streets etc.), then processed as the input for the prediction component.

5.2.4 Prediction Component

For the prediction component, four parts are summarized into the system, including the short-term feature learning component, long-term learning component, the attributes embedding component, and the feature fusion component. The detailed architecture will be discussed in next section. Then, after the target output sequence is obtained, the error monitoring component is used to watch the predicting result and compare it with the ground truth, and report the comparison result for the parking lot manager. If the error is larger than expected, auto re-training will be activated and the model will be re-trained with the data in the recent eight weeks.

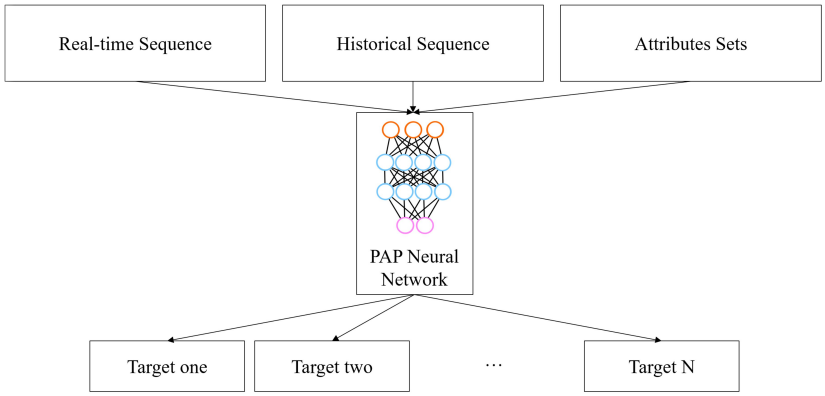


Figure 5.3: Overall framework of the parking availability prediction design

5.3 Prediction Representations Extraction and Fusion

5.3.1 Model Preliminary

In this section, the author first shows some preliminaries in the prediction task. The overall idea is to build a parking prediction neural network with three types of input. The network can predict the future occupancy status for various prediction time targets accurately (idea framework shows in the Figure 5.3).

Definition 1 Historical occupancy data sequence (H^i_m): a historical occupancy is a sequence of m continuous historical occupancy records generated by parking lot i. The H^i has a fixed length of time gap (gh, here gh=60 minutes, equal to one hour). Each point record of h^i_m includes occupancy level and timestamp.

Definition 2 Real-time occupancy data sequence (R^i_n): a real-time occupancy is a sequence of n continuous short-term occupancy record generated by parking lot i with a fixed length of time gap (gr, here gh=5 minutes,), i.e., $R^i_n = \left\{r^i_1, r^i_2, \ldots r^i_n\right\}$. Each point record of r^i_n includes occupancy level and timestamp.

Definition 3 Attributes set (A^i): the attributes information of the parking lot i is an important input source of the model. Each occupancy record of the H^i and R^i, as well as the prediction target all have the belonged attributes information set. Here, A^i includes: parking lot id i, parking lot class i, weather condition of the parking lot ($weatherID^i$), week date ($weekID$) and Time of a day ($TimeID$).

5.3.2 Parking Availability Prediction (PAP) Neural Network Architecture

In this section, I would like to show the four components of PAP in detail. As shows in the Figure 5.4, the PAP neural network is consists of four parts: Input Embed Component (IEC), Attributes Tensor Embedding Component (ATEC), Temporal Learning Component(TLC), and the Feature Fusion Component (FFC).

Input Embedding Component (IEC)

In order to better enable the network to fully extract the long-term and short-term temporal dependency of parking patterns, in the data input component, I use two sets of sequence data as input for the prediction task. The two sets of sequences are real-time sequence (R^i) and history sequence (H^i). The real-time sequence represents the most current parking situation, consists of n occupancy data records. Meanwhile, the history sequence means at the prediction target time flag, the past parking occupancy record at the same time of a day and day of a week (i.e., if the prediction target of time is next Wednesday 8:00 AM of parking lot i, the H^i will be consists of m historical occupancy records captured in the past several Wednesday 8:00 AM).

Then, to better extract the hidden pattern of the two sequence for each prediction target, the author designed a input embedding process for the input sequences. To obtain each prediction

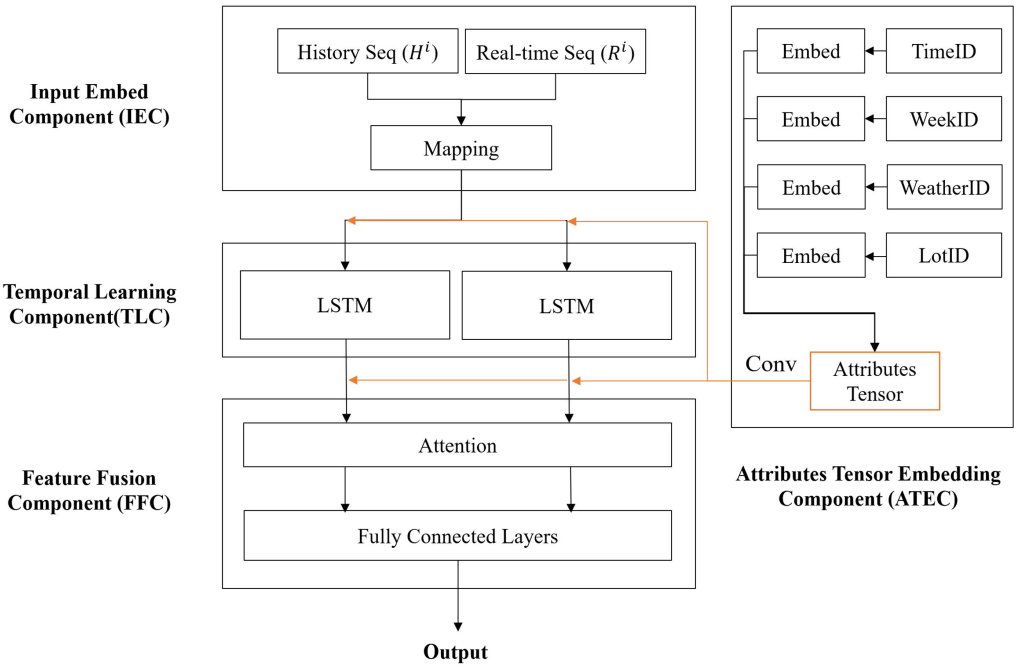


Figure 5.4: Architecture of the PAP neural network, including four components: Input Embed Component (IEC), Attributes Tensor Embedding Component (ATEC), Temporal Learning Component(TLC), and the Feature Fusion Component (FFC)

target input sequences (In_{T_i}), a matrix embedding:

$$In_{H^i} = tanh(W_{H^i} \cdot h^i_m) \tag{5.1}$$

$$In_{R^i} = tanh(W_{R^i} \cdot r^i_n) \tag{5.2}$$

is used to map the m_{th} records of historical and n_{th} real-time occupancy sequence into a high-dimensional vector space $In_{T_i} \in \mathscr{R}^{16}$. And in the Equation (1), the W_{H^i} and W_{R^i} are the learnable embedding weight matrix. Specifically, in this research, I chose the m equals to n as the same length. So, the output of the Equation (1) and (2) In_{T_i} and In_{R^i} are used as the input for

occupancy prediction, whose size are $\mathscr{R}^{16*|\mathbf{m}|}$.

Attributes Tensor Embedding Component (ATEC)

From the previous research, the attributes information, including the time of a day, day of a week, weather condition heavily impacts on the parking occupancy and pattern [94, 109]. For example, for some parking lots, the daily packing lot occupancy status of the workday is much higher than the weekend. Also, the frequency of short-term parking (less than 20 minutes) in the mid-afternoon are much higher than evening and night. Obvious to see, these attributes clusters information has connections and relations with each other. Only vector mapping can not sufficiently capture the hidden pattern inside of the sets. Furthermore, since the attributes information are always collected as discrete bracket values, directly using them as neural network input is not feasible [266, 267]. To fully utilize these information, in this research, I design a particular embedding approach, called Attributes Tensor Embedding Component (ATEC) to extract and integrate the features into the neural network at the utmost level (detail structure showing in Figure 5.5). The ATEC including two steps, value embedding, and tensor embedding.

First and foremost, the author needs to map the discrete category value into a continuous space. Transferring the format of information into high dimension space has been widely used in the NLP algorithms. Inspired by the word to vector algorithms [268] and the latent topological feature vectors extraction method [264], the author design a learnable process to map the category values into a continuous vector space for further mathematical operation.

$$Att_{a_j^i} = W^{a_j^i} \cdot a_j^i \tag{5.3}$$

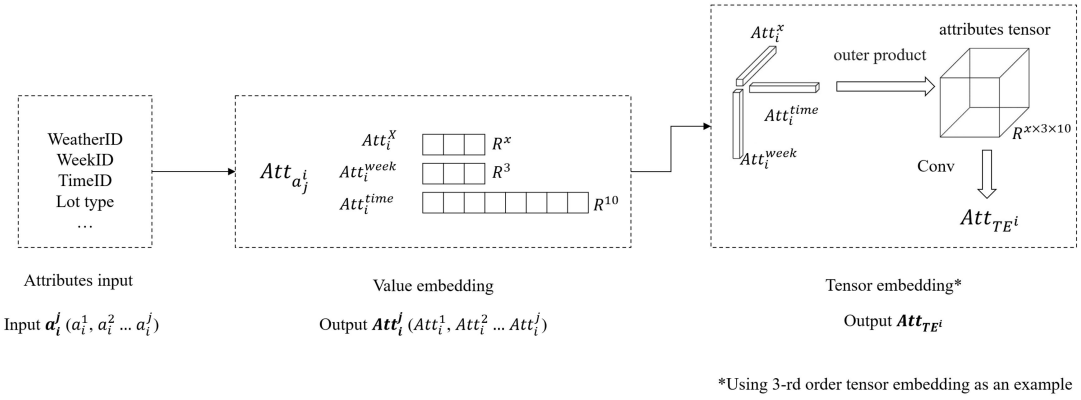


Figure 5.5: The detail structure of proposed ATEC.

In Equation 5.3, the a^i is the attributes information that belongs to each occupancy data record. W^{a^i} is the learnable matrix, whose dimension is $\mathscr{R}^{\mathbf{A}*\mathbf{E}}$, is used to transform the input attributes category values a^i ($a^i \in A$) from a original space A to the embedding space E. Each attribute information has its own corresponding mapping matrix to ensure the feature can be map into a suitable space.

The second step aims to capture the combination and correlations inside of the attributes vectors. Here, the author designs a tensor embedding procedure to extract the latent features. The first sub-step is to build an attribute tensor by using the vector outer product

$$\mathscr{T}_{a^i} = Att_{a_j^1} \otimes Att_{a_j^2} \cdots \otimes Att_{a_j^i} \tag{5.4}$$

The output of Equation 5.4 is a tensor $\mathscr{T}_{a^i} \in \mathscr{R}^{E_1 \times E_2 \ldots \times E_j}$. Then, using multidimensional convolution operation to extract the attributes tensor features and map into a one-dimension vector. This sub-step is mainly to capture and emphasize the inside correlations of attributes.

The math expression of the operation is in equation 5.5.

$$Att_{TE^i} = \sigma_{cnn}(W_{conv} * \mathcal{T}_{a^i} + b) \tag{5.5}$$

Then, I got the output of the attribute set embedding vector Att_{TE^i}, which includes j different factors. Specifically, j is the number of different attributes includes inside of the attributes set. I will use the Att_{TE^i} to represent the attributes features in the following components.

Temporal Learning Component (TLC)

After both attribute information and sequence records are transformed into a vector features set, then, the temporal learning component is used to capture the time dependency features. Here, I design two separate stacked LSTM modules for capturing both the historical sequence and the real-time occupancy sequences together. The input of the TLC for a prediction target is:

$$In_{TLC_{R^i}} = In_{R^i} \circ Att_{TE_{R^i}} \tag{5.6}$$

$$In_{TLC_{H^i}} = In_{T_i} \circ Att_{TE_{H^i}} \tag{5.7}$$

From the Equation 5.6 and 5.7, the input of TLC can be divided into two sequences, and each of them including the occupancy sequence features and the attributes features. Both features vector is linked by the concatenate operation ($\circ$) and then send into the temporal leaning network. Here, to further captures the dependencies of the input sequence, I introduced the recurrent neural network (RNN) into the model. RNN is a widely-used artificial neural network for learning the temporal features in sequences, especially the transportation data sequence. In the model, a well-known special RNN – LSTM is designed to capture the parking sequence de-

pendency features [1]. In each LSTM neuron, three gates are included: input gate $i_{(t)}$, output gate $o_{(t)}$ and forget gate $f_{(t)}$. Each gate is controlled by a combination of their own trainable weight $w_{(t)}$ and the previous neural output $h_{(t-1)}$. Furthermore, with the designed memory cascading process, both new memory $\widetilde{c}_{(t)}$ and final memory $c_{(t)}$ for a neuron can be updated during the training process. After the last final memory generation, the hidden state $h_{(t)}$ is proposed under the control of output gate $o_{(t)}$. In detail, Figure 5.6 shows the operating structure of LSTM neural. The mathematical formulations for each LSTM show in equation 5.8 to equation 5.13 as follows:

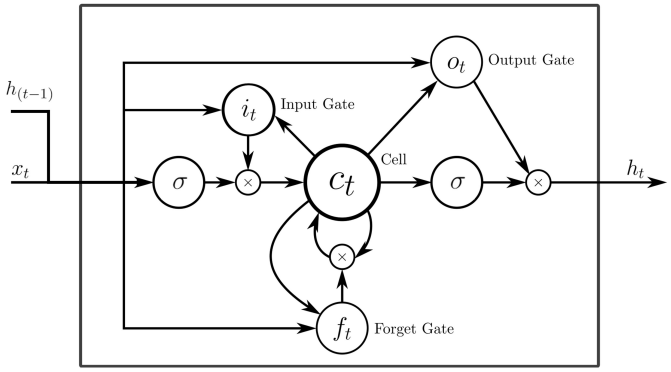


Figure 5.6: Detail structure of LSTM neuron [2].

$$i_{(t)} = \sigma(W_{(i)}x_{(t)} + U_{(i)}h_{(t-1)}) \tag{5.8}$$

$$f_{(t)} = \sigma(W_{(f)}x_{(t)} + U_{(f)}h_{(t-1)}) \tag{5.9}$$

$$o_{(t)} = \sigma(W_{(o)}x_{(t)} + U_{(o)}h_{(t-1)}) \tag{5.10}$$

$$\widetilde{c}_{(t)} = tanh(W_{(c)}x_{(t)} + U_{(c)}h_{(t-1)}) \tag{5.11}$$

$$c_{(t)} = f_{(t)} \circ c_{(t-1)} + i_{(t)} \circ \widetilde{c}_{(t)} \tag{5.12}$$

$$h_{(t)} = o_{(t)} \circ tanh(c_{(t)}) \tag{5.13}$$

To capture the temporal features, PAP uses two stacked LSTM as the memory module for each input sequence. Each training sequence is passed multiple LSTM neural together. The mathematical formulations of stacked LSTM are showing in equations 5.14 to 5.15:

$$h_i^{H^i} = \sigma_{lstm}(W_x \cdot In_{TLC_{H^i}} + W_h \cdot h_{i-1}^{H^i} + W_a \cdot Att_{TE^{H^i}}) \tag{5.14}$$

$$h_i^{R^i} = \sigma_{lstm}(W_x \cdot In_{TLC_{R^i}} + W_h \cdot h_{i-1}^{R^i} + W_a \cdot Att_{TE^{R^i}}) \tag{5.15}$$

Where the $In_{TLC_{Ri}}, In_{TLC_{h^i}}$ are the two input data sequences. $Att_{TE^{R^i}}$ and $Att_{TE^{H^i}}$ are the attributes features vector after the tensor embedding of each occupancy record. And the W_x, W_h and W_a are learnable parameter matrices used in the LSTM.

Feature Attention Component (FAC)

Since the author prefers to use the PAP neural network to integrated heterogeneous latent features, a feature fusion and attention component is necessary. From the previous research, the parking utility prediction features can be roughly divided into two parts: short-term features and collective features. Formally, the short-term features mainly serve for predict close-target time availability information. These information always can be investigated and summarized from the real-time continuous input sequence (i,e, the occupancy sequences or parking status sequences). Meanwhile, when the traveler want to predict long term availability parking information (much longer than the time gap of the input sequence), the long-term features are

play roles. Generally, the collective features include the long-term parking pattern features and the attributes conditions. Hence, the model not only needs to refer to the current occupancy sequence, but also the historical occupancy records with the similar attribute conditions. Meanwhile, sufficiently integrating the attributes information at the target prediction time is quite necessary since the long term parking pattern is closely related to the attributes conditions, i,e, day of the week, time of the day and etc.

In PAP, to better fuse and use the two types of features, the FAC is designed and implemented. The first step is the attention mechanism. Here, the author designs an attention mechanism considering the attributes factors instead of mean pooling. The attention includes three procedures. The first part is the attributes vector similarity calculation. In each prediction target, attributes information includes two part, the input occupancy sequence (O_i) attributes ($Att_{TE_{R^i}}$, $Att_{TE_{H^i}}$) as well as the target prediction attributes Att_{TE_T}. The cosine similarity is used here in equation 5.16:

$$Sim(T, O_i) = \frac{\mathbf{Att_{TE_T}}^\top \cdot \mathbf{Att_{TE^{O_i}}}}{|Att_{TE_T}||Att_{TE^{O_i}}|} \tag{5.16}$$

Then, the attention value α_i can be calculated by the following equations 5.17 to equations 5.18:

$$h_i = h_{HAtt} \circ h_{RAtt} \tag{5.17}$$

$$\begin{aligned} \beta_i &= < \sigma_{Att}(Att_{TE^{O_i}} \cdot Sim(T, O_i)), h_i > \\ \alpha_i &= \frac{e^{\beta_i}}{\sum_j e^{\beta_j}} \end{aligned} \tag{5.18}$$

In the Equation (17), the $\circ$ is the concatenate operation. In Equation (18), σ_{Att} is the nonlinear mapping that maps the attention into same dimension with h_i. With the attributes at-

tention, I summarized the two sequence features into an attention-based vectors: h_i. Then, the final features vector ($\mathscr{F}_{\mathscr{T}}$) for the prediction target T is:

$$\mathscr{F}_{\mathscr{T}} = \sum_{i=1}^{2m} \alpha_i \cdot h_i \tag{5.19}$$

Finally, the author used four fully connected layers to down sample the $\mathscr{F}_{\mathscr{T}}$ from 64 to only one single neuron, which represents the predict parking occupancy result (O_{T_i}).

5.3.3 Loss and Evaluation Function

In PAP, the author choosed to use the mean absolute percentage error (MAPE) to train the PAP neural network. In the evaluation process, the mean absolute error (MAE) and MAPE are used together to analysis the performance. The loss and evaluation functions for PAP are defined in the following equations:

$$\text{MAPE} = \frac{1}{N} * \sum_{i=1}^{N} \left| \frac{O_{T_i} - \hat{O}_{T_i}}{O_{T_i} - \varepsilon} \right| * 100\% \tag{5.20}$$

$$\text{MAE} = \frac{1}{N} * \sum_{i=1}^{N} |O_{T_i} - \hat{O}_{T_i}| \tag{5.21}$$

In the equations 5.20 to 5.21, the O_{T_i} represents the predicted occupancy value of the given target. The $\hat{O}_{T_i}$ represents the ground truth occupancy. In Equation (21), ε in MAPE is used to prevent the 0-denominator appearance and ε is small enough (ε=0.05) compared with the average occupancy level.

5.4 System Experiment

5.4.1 Environment Description

The PSPNN was implemented with PyTorch. The work station for training was equipped with a GPU (Nvidia RTX 2080 Ti) and the CPU is Intel Core i7 8700. The operating system is Linux Ubuntu 16.04.

5.4.2 Data Sets Description

Urban Parking Data: In the research, I collected 19 parking lots data including 1534 urban parking slots in Zhengzhou, a midsize city in the central area of China. The parking lots are all in the busy urban area of Zhengzhou (inside of the third ring road), and whose locations can be divided into four types: nine of them are shopping mall parking lots, five are in working and business area and two of them are located in the residential area. Then, three of the parking lots are in the comprehensive service areas, including shopping, business, working region, and even the areas for transit sharing. All of the parking lots are opened for public citizens. The formats of the original data records are vehicle parking information, including parking start time, end time, parking duration, parking price, and vehicle information. Then, I processed the parking records into parking lot occupancy based data, for 24/7. The records are form March 1^{th} to Aug 31^{st} of 2018. The time gap of the occupancy records is five minutes. Besides, I also collected the real time weather information at the same time period with the time interval of five minutes. Eight categories of weather conditions were summarized for future analysis: fair, cloudy, light rain, rain, heavy rain, light snow, snow, and fog.

Truck Parking Data [269]: In the research, 49 truck parking slots data in two rest areas of

near the Interstate 5 freeway from the January 01^{st} to March 31^{th} of 2020, and January 01^{st} to April 30^{th} 2021 is collected. The data are summarized into two formats, parking event data and occupancy data. The event data is used to record each parking slot status, which includes the status change, the starting time of parking and the time duration. For the occupancy data, I processed the event data and obtained the occupancy rate of each parking lot based on per minute interval. Besides, I also collected the real-time weather information from the closest weather station during the same period and recorded every minute. Weather conditions were summarized into eight categories: fair, cloudy, fog, light rain, light snow, rain, snow and heavy rain.

5.4.3 Parameters

The parameters using in the PAP experiments are as follows:

Parameter settings in the IEC

- The historical sequence (H_m^i) length m in PAP is fixed as 16. If the sequence length is not long enough, duplicating the earliest record till 16. The time gap for each record is 60 minutes (one hour).
- The real-time short-term sequence length (R_n^i) n in PAP is fixed as 16. The time gap for each record in the R_n^i is five minutes.
- The input vector embedding space is $\mathscr{R}^{16}$.

Parameter settings in the ATEC

- The attributes information for urban parking availability used in PAP are including four sets: weather condition (fair, cloudy, light rain, rain, heavy rain, light snow, snow, and

fog), day of a week (Monday to Sunday), Time of a day (288 time ID per day), and lot attributes (id and type (shop, business and working, residential and comprehensive utility)). The size of the embedding vector for each attribute set is settled in the following: weatherID mapping into R^3, weekID mapping into R^3, departure time ID mapping into R^8 and the lotID and location mapping into R^6. The total dimension size of $E(A)$ is R^{20}.

- The attributes information for truck parking availability used in PAP is generally the same setting as the urban parking. However, as the truck parking lots location and attributes characteristics are very similar (both are near the Interstate 5 freeway and all used for truck parking purposes), I used including three sets: weather condition (fair, cloudy, light rain, rain, heavy rain, light snow, snow, and fog), day of a week (Monday to Sunday), Time of a day (288 time ID per day). The size of the embedding vector for each attribute set is the same setting as the urban PAP experiment.

- The convolution operation setting for the attributes tensor embedding is in the following. For the urban parking availability, the author uses a 3-D convolution operation with the kernel size of $2 \times 2 \times 1$. For the truck parking prediction, a 2-D convolution operation is used with the kernel size of 2×1. The output dimension of the ATEC for both urban parking or truck parking attributes sets are $\mathscr{R}^{16}$.

Parameter settings in the TLC

In the TLC, the number of hidden neural in the stacked LSTM is fixed as 16. Here, two hidden layers are used in the memory module. The activation function in Eq (8-13) σ_{rnn} is $tanh$ function. The mathematical formulation of $tanh$ used in the research is $tanh(x) = e^x - e^{-x}/e^x + e^{-x}$.

In prediction module, the number of fully connected layers is fixed as 4. The four layers down sample to the 64 dimension vector to a neuron as the predicting parking occupancy result (O_{T_i}).

For each dataset, I use the occupancy data records for both urban parking and truck parking to train (70%), validate (15%) and test (15%) the PAP model. I train five different predict target time (5 minutes later, 10 minutes later, 30 minutes later, 1 hour later, and 2 hours later) to validate the PAP for both short-term and long-term prediction ability. I adopt Nesterov-accelerated Adaptive Moment Estimation (Nadam) optimization algorithm [270] to train the parameters. I train the model for 220 epochs and select the best models by five-fold cross-validation. For evaluating the PAP, I compared with other six methods, including,

- Kalman filter [271] I also use the Kalman filter as a baseline model for comparison. Generally, it is an efficient autoregressive filter, which can estimate the state of the dynamic system in the combined information with uncertain status. The method is widely used in traffic parameters prediction [272, 273]. In the experiment, I implemented a Kalman Filter on the parking occupancy dataset with weatherID and weekID features. Specially, a multivariant Kalman filter formulation proposed by [271] is implicated in this research.

- Random Forest [96] Random forest is a simple and efficient basic machine learning algorithm proposed by Breiman [96] in 2001. Generally, random forest is a classifier that contains multiple decision trees. As the name implies, a forest is built in a random manner. The forest is composed of many decision trees, and there is no correlation between these decision trees. For each tree, the training set they use is sampled from the total training set by means of replacement. When training the nodes of each tree, the features used are extracted from all the features randomly according to a certain proportion without

replacement. In the parking occupancy prediction, the number of decision trees used by the me is 400 and the number of seeds is 16.

- RNN (2-layer) [274]: For comparison, I used RNN as a baseline to predict the future parking occupancy rate and then compared with the ground truth records. In the research, I set the neural number of RNN as 16 with two hidden layers.

- LSTM (2-layer) [275, 250]: I also implemented the stacked LSTM neural network to as a baseline, with 16 hidden unites and cascading two hidden layers.

- Google Parking Difficulty Estimation (Google PDE) [96] Google PDE is proposed by Google Research in 2019. In the Google PDE framework, the parking availability estimation problem in transformed into the parking difficulty estimation. A reward matrix and a two layer feedforward deep neural network model that has two hidden layers (20 and 10 hidden units, respectively) is proposed. The activation function for each layer is applied the ReLU function. In the performance evaluation process, I used the prediction architecture proposed by Google PDE, but transformed the output from difficulty categories into the actual occupancy rate for further comparison.

- PewLSTM [93] Periodic Weather-Aware LSTM (PewLSTM) is proposed by Zhang et al. in IJCAI 2020, which is a sequential model incorporating the weather conditions for parking utility forecasting. Each of the PewLSTM neuron receives both weather information and parking record as inputs. By integrating the gating mechanism into traditional the LSTM structure, the PewLSTM successfully surpass the LSTM models in various weather conditions and environments. Here, I set the number of PewLSTM hidden neural as 16 and stacked two layers.

- PAP (no ATEC) in this research, I designed and implemented a attributes tensor embedding method for attributes information integration. To see the value of the component, I trained the PAP without the ATEC. So based on the condition, the attention component also simplified into a four fully-connected layers. Through the comparison, I would like to determine how significantly the ATEC and the internal attributes operations affect the actual prediction result.

The detail results and compassion with baselines are showing in Table 5.1 and Table 5.2. In order to show the results more intuitively and convenient for comparison, I random selected four parking lots and visualize the training loss and show prediction results of 10 min and 1h as well as the ground truth records in Figure 5.7 and Figure 5.8. From the results, I can see that the PAP performance are much better than other state-of-art baselines, especially for the long-term parking prediction (1 hour and 2 hours later). Meanwhile, in the experiment, several findings are worth to mention. Firstly, the sequential based prediction algorithms (RNN, LSTM, PewLSTM) work better for short-term features extraction and prediction (5 minutes, 10 minutes) rather than long term prediction (1 hour, 2 hours). Secondly, except PAP, the Google PDE has the best performance for long-term features capturing and prediction results. Sometimes the result is even better than the PAP with no ATEC. So, instead of sending the records directly into the sequential model, the Google PDE transform the original input as multiple difficulty level. This approach could balance the temporal features and general parking pattern features which make the network capture more long-term pattern for forecasting. However, such method is also cut some accuracy for the short-term prediction. Thirdly, the ATEC shows significant impact on the performance improvement on both datasets. With ATEC, attributes information is sufficiently integrated into the short-term and long-term parking prediction tasks. As the Fig-

ure 5.8 shows, even the weekday and weekend pattern are very different in some cases, the PAP can also capture the pattern for both 10 min and 1h. Then, the residential area always be the best predictable parking lot for every method since the relative regular activity pattern.

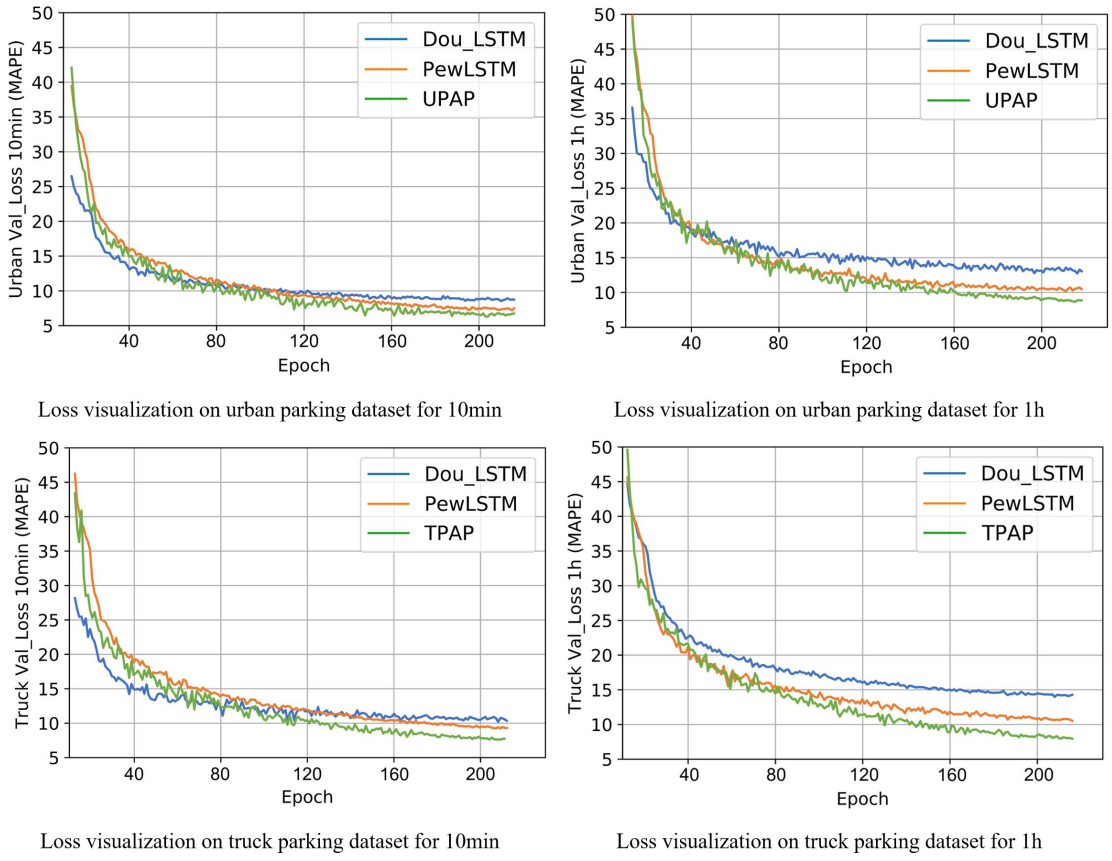


Figure 5.7: Training Loss visualization for 10 minutes and one hour prediction model, including double layer LSTM, PewLSTM, urban and truck PAP.

Table 5.1: Performance comparison on urban parking dataset

Time Ahead	5min		10min		30min		1h		2h	
Method	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**
Kalman filter	9.05	11.17%	10.35	12.78%	12.34	15.23%	14.68	18.13%	15.54	19.18%
Random Forest	8.20	10.23%	9.84	12.27%	11.38	14.19%	13.67	17.04%	14.10	17.59%
RNN (2-layer)	6.67	8.36%	7.18	9.01%	10.14	12.71%	12.60	15.79%	13.03	16.34%
LSTM (2-layer)	5.87	7.23%	6.24	7.69%	8.08	9.96%	11.19	13.78%	11.44	14.10%
Google PDE	6.27	8.01%	7.08	9.04%	8.20	10.47%	10.68	13.63%	9.85	12.57%
PewLSTM	5.83	7.31%	5.96	7.43%	6.59	8.21%	7.29	9.09%	7.99	9.96%
PAP (no ATEC)	5.03	6.24%	5.14	6.38%	6.10	7.56%	7.05	8.74%	7.65	9.48%
PAP	**4.33**	**5.35%**	**4.57**	**5.65%**	**5.51**	**6.80%**	**6.01**	**7.41%**	**5.94**	**7.26%**

Table 5.2: Performance comparison on truck parking dataset

Time Ahead	5min		10min		30min		1h		2h	
Method	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**
Kalman filter	8.36	17.06%	8.40	17.15%	9.71	19.82%	9.53	19.45%	9.73	19.86%
Random Forest	8.15	16.99%	8.26	17.21%	8.84	18.42%	8.65	18.03%	8.68	18.06%
RNN (2-layer)	5.86	11.73%	6.16	12.33%	6.64	13.27%	7.63	15.25%	7.96	15.92%
LSTM (2-layer)	4.53	10.07%	4.58	10.18%	4.74	10.54%	5.98	13.29%	6.16	13.70%
Google PDE	6.26	13.06%	6.29	13.10%	6.99	14.56%	6.84	14.26%	6.82	14.21%
PewLSTM	4.13	8.61%	4.26	8.89%	4.54	9.47%	5.07	10.57%	5.26	10.96%
PAP (no ATEC)	4.07	8.49%	4.11	8.58%	4.47	9.33%	5.01	10.43%	5.41	11.29%
PAP	**3.39**	**6.92%**	**3.46**	**7.07%**	**3.82**	**7.79%**	**3.94**	**8.04%**	**4.09**	**8.35%**

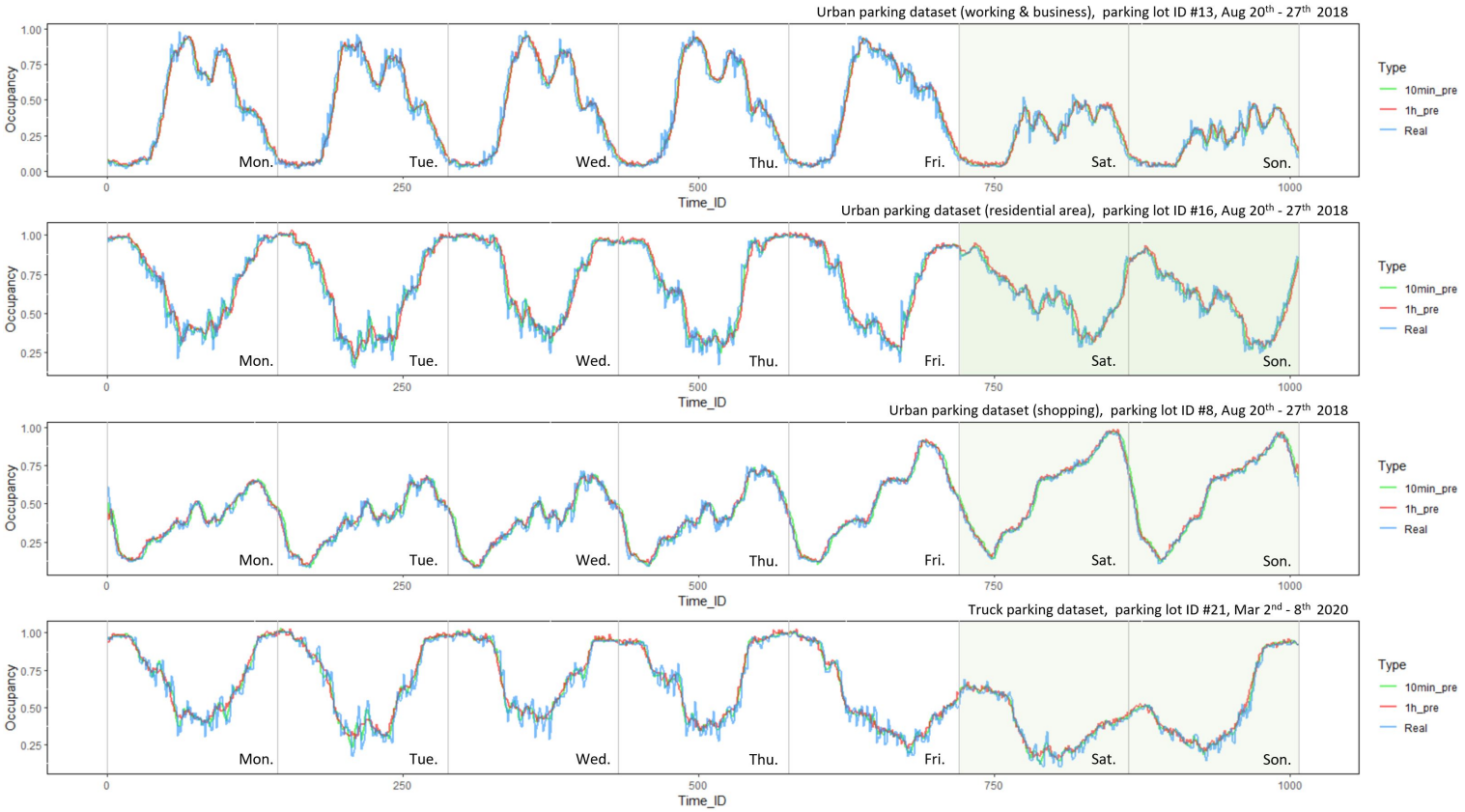


Figure 5.8: The PAP prediction result visualization for parking lot 13 (working business), lot 16 (residential area), lot 8 (shopping), lot 21 (truck parking).

5.5 Result Analysis and Discussion

5.5.1 Result Analysis

Types of Parking Lot

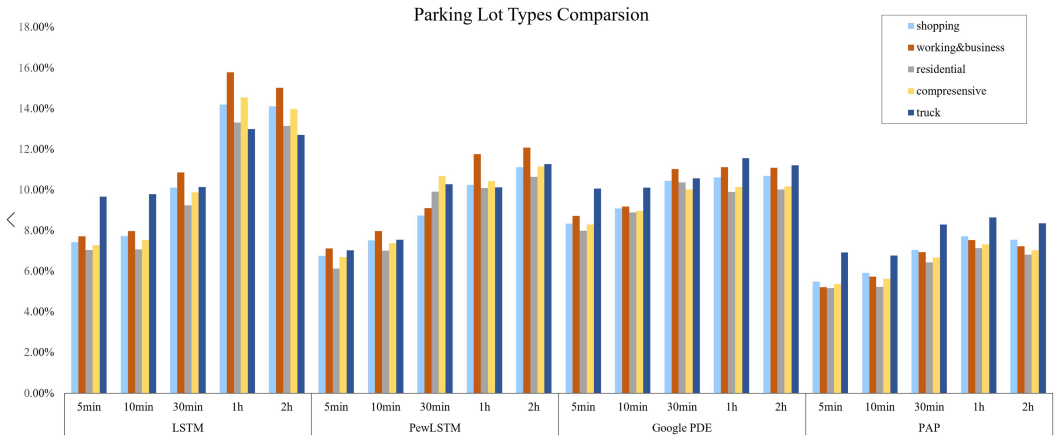


Figure 5.9: Types of parking lot prediction comparison for LSTM (2-layers), PewLSTM, Google PDE and PAP

To see the PAP neural network performance using for different types of parking lots prediction, the author finish the ablation study for five types including: shopping, working & business, residential, comprehensive service area and truck parking. From the analysis and comparison with the 2-layer LSTM, PewLSTM, Google PDE, PAP achievement the best performance among all the types of parking lot availability prediction (in the Figure 5.9). Several findings worth further discussion. First and foremost, comparing with urban parking, short-term truck parking prediction is relatively tough. The reasons could be the records limitation of truck parking dataset, as well as the low short-term correlation among the parking records. Specifically, in

the USA, the truck drivers need to follow the Hours-Of-Service rules and complete the logbook. So the truck drivers are very likely to start work in the morning (especially 7:00-9:00 AM) and find somewhere to park and rest in the evening (7:00-9:00 PM). During this time period, the short term pattern is significantly various time to time and sometimes the occupancy rate has changed greatly in five minutes, which obviously increase the difficulty of prediction. Secondly, for LSTM based algorithms (2-layer LSTM and PewLSTM), the working & business parking lot error is always the biggest, especially for the long term prediction (1h or longer). From the data analysis, the daily the parking occupancy rate fluctuation of the working business is the largest, indicating the huge long-term pattern change day by day. PAP shows excellent ability to capture these features and achieve the best performance for the working & business parking lot prediction for both long-term and short-term time period. For PAP, the largest prediction error shows on the shopping mall parking lots, especially on the holidays and weekends. This phenomenon can be explained by the increasing number of random trips and parking during holidays. The holiday parking pattern features are often difficult to be learned and captured by unspecified and small amount of training data. At the same time, historical data at the same time increases the difficulty of capturing the sudden changes by the neural network. This problem will be improved in future research.

Time of a day

To evaluate the applicability of the PAP neural network 24/7, I evaluated the framework based on different hour of a day by a week of urban and truck parking data. The detail result are showing in the Figure 5.10. In general, the PAP achieve very steady and reliable works for all time of a day (with less than 2% of the MAPE change between the best and worst hour). Specifically,

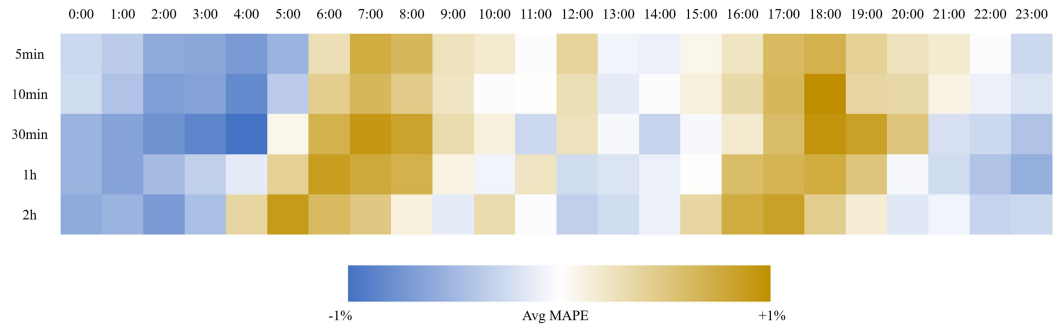


Figure 5.10: The PAP prediction accuracy analysis on different time period of a day

during the night time, especially 23:00 to 4:00 of next day, the prediction result is more accurate. The best prediction result is in the 4:00-5:00 time period for 30 minutes ahead, which is -0.978% better than the average MAPE. However, for the morning and evening peak, with the parking activity pattern is much more complex, the error is bigger than average MAPE. The largest error hour is 18:00-19:00, for 10 minutes prediction slot ahead. The MAPE increases to 6.67%, however, still significantly better than other state-of-art methods. Also, from the analysis, the author finds the 30 minutes prediction slots has the largest error vibration domain (1.926%) among all.

Weekday and Weekend

For PAP neural network, the performance for both truck and urban parking dataset on weekday is better than weekenD, as shown in Table 5.3. Specifically, the biggest error shows on truck parking 5 minutes prediction (1.33% higher than weekday) and urban parking 30 minutes prediction (0.45% higher than weekday). After investigation, the author summarizes the potential reason in the following two aspects. Firstly, for both urban and truck parking, during the weekends, with the random parking activity increased, the parking availability is relatively tough than

Table 5.3: The PAP neural network prediction performance comparison on weekday and weekend

	Prediction Time	Urban Parking	Truck Parking
Weekday	5min	5.31%	6.54%
	10min	5.57%	6.69%
	30min	6.54%	8.11%
	1h	7.23%	8.60%
	2h	6.91%	8.31%
Weekend	5min	5.52%	7.87%
	10min	5.75%	6.97%
	30min	7.00%	8.74%
	1h	7.55%	8.74%
	2h	7.33%	8.45%

weekday. However, the pattern change are not same. In the urban area, the long-term pattern, especially for 30 minutes, 1h and 2h prediction is greatly changed and sometimes various week by week. Meanwhile, for truck parking, especially during the weekend, the truck drivers are obvious in relax mode and the random parking for short-time relax (always less than 20 minutes) is increasing [109], and the short-term prediction challenge is increased. Even though, the overall PAP performance in both weekday and weekend are impressive.

Weather Condition

Weather condition also a factor shows impact in the PAP neural network prediction performance. Generally, the PAP performance can satisfy most of the weather conditions, with less than 2% of MAPE fluctuation. However, for the urban parking prediction, I found that the rain, heavy rain are heavily increased the parking availability level, as well as the prediction re-

sults. I evaluate three days with rain & heavy rain condition and found the average accuracy decrease from (1.35%, 1.77% and 3.32% (heavy rain at evening peak hour)). Considering with other information, I summarized the reasons into two aspects. Firstly, rain, snow or other challenge conditions for drivers shows trigger the heavy congestion in the road network, especially for the busy area of Zhengzhou. Sometimes the congestion will heavily impact the cars comings and goings of a parking lot, and shows stuck period on the parking occupancy records. The situation will heavily impact the parking accuracy, especially for the short term parking. Secondly, during the days with challenge weather conditions, the differences of the parking lots are magnified, which need more detail characteristics information to integrated. Only simple parking types and id can not help the PAP perform well in these heavily dynamic environmental factors. This would be a future research topic for the author to investigate. For the truck parking availability prediction, I did not find obvious change (less than 1%) from various weather input. Considering the data is obtained from the northwest region of USA with mild weather changes, the PAP performance is reasonable.

5.5.2 Heterogeneous Features Fusion Analysis

ATEC Attributes Representation Features Effect

In PAP neural network, an attributes representation extraction component is designed for extract the heterogeneous features. The author finished a before-and-after analysis for the effect. Among all of the attributes information, the lotID and types shows the biggest influence for the prediction result, with a average 2.32% MAPE improvement (up to 4.12%) for five prediction. Then, the timeID and weekID also shows significantly impact on the prediction results, with 2.04% (up to 3.37%) and 1.89% (up to 2.89%) average improvement. Specifically, the weekID

significantly help the truck parking prediction with 2.24% average improvement. For the urban parking, the weather condition also contribute average 1.01% MAPE (up to 1.89%) improvement, while, 0.103% contribution for the truck parking dataset. From the results, even weather are widely considered as one of important factors impact for parking [94], it is the lowest importance factor among all the attributes features selected by me. Needless to say, from the Table 5.1 and Table 5.2, the ATEC plays a role to transform the attributes heterogeneous features into representations sufficiently, and then well integrate by the framework.

Attention Effect

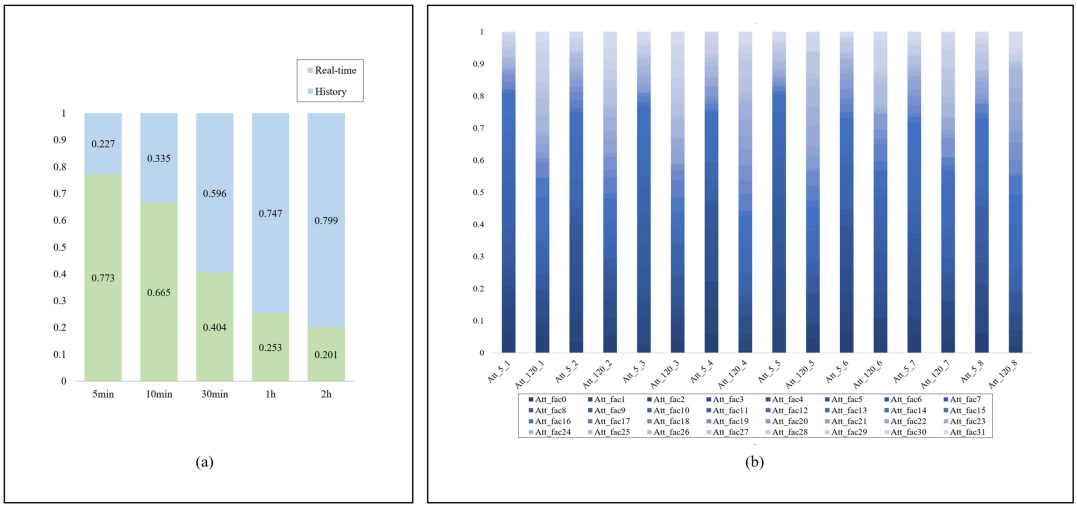


Figure 5.11: (a) The attention mechanism quantification and visualization for various prediction time target (b) The attention factor value visualization with same input records for 5-minute and 2-hour prediction

To fully utilize the attributes information and balance the long-term and short-term features captured by the LSTM blocks, the attention mechanism is integrated into PAP. To evaluate the contribution for both features vector, I used the well trained model for five different target

prediction time and randomly test 500 times. Then summarize the attention value (α_i in the Equation (19)) for real-time sequence representation features and historical sequence representation features. The sum of the all attention is equal to one. From the Figure 5.11 (a), I can see that for the short-term prediction (the prediction target time is similarly with the real-time input time gap), the attention on real-time sequence is much higher (0.773 and 0.665 respectively). Meanwhile, for the long-term prediction, using 1 hour or 2 hours later as target, the dominated attention will transform into the history sequence representation vector. Under this circumstance, the long-term pattern, especially the precious records with similar attributes information will contribute more. For the two hours parking prediction, the history representation features contribute almost 80% percent for the final result. In Figure 5.11 (b), I randomly chose eight attention records for five minutes (Att_5) and two hour prediction (Att_120) with same input sequences. Obvious value distribution difference can be found that in the five minutes prediction slot, the small number attention factor (Att_fac) with dark blue is significant. However, even with the same input sequences, the contributions of the 2 hours prediction are more in the light blue region. The result of the attention analysis also answer the question that why for predicting the long-term target, the LSTM based sequential model is not showing as good performance as short-term prediction.

Transferability

A big advantage for integrating the attributes representation features instead of using original data records is that the learning-based models trained by representation vectors shows much more strong transferability [276]. If the model need to be implemented into a new similar scenario, these models perform better adaptability. By integrated the ATEC to transform the

original attributes input into representation features, the PAP shows impressive prediction result even without using the target parking lot data for training. A better performance can be achieved by only using two weeks or four weeks of the target parking lot records, and can achieve significant improvement. Using the MAPE of integrating the target parking lot data into training process as $MAPE_{all}$, and no data integrate into training as the original MAPE performance $MAPE_{no}$, I evaluated the performance improvement tune by using 20% and 40% of the target parking lot dataset and summarized the result into the following Table 5.4. In the evaluation, the $MAPE_{all}$ minus $MAPE_{no}$ is considered as the largest accuracy improvement and equal to 100%. From the result, I can find out that the PAP neural network transferability is very strong, with only 20% of the target parking lot data (about 28.5 days) for fine tuning can achieve 63.37%, 64.82%, and 62.80% of the total improvement (average improvement value of five prediction targets) for shopping, business and working and comprehensive parking lots respectively. For comparison, the original 2-layer LSTM improvement performance is 20.16%, 22.35%, and 22.04%, only about one-third of the PAP performance. Specifically, with the attributes representation features, long-term features learning and capturing becomes much easier, and even limited data (20% of training data) can help the PAP achieve above 70% improvement for 1 hour and 2 hours prediction. Using the same amount of tuning data, the improvement for 2-layer LSTM is only 12.95% and 10.23%, respectively. From the evaluation, I successfully show the transferability improvement with heterogeneous representation features embedding. This research and the methodology using for attributes information integration shows a potential target to improve the adaptability of machine learning-based models in the transportation research community.

Table 5.4: The PAP neural network fine tune process with different levels of new parking lots data integration

LorID & Type	Prediction Time	No data	20% Training data		40% Training data		All Data
Lot8 (Shopping)	5min	6.50%	5.96%	51.43%	5.79%	67.62%	5.45%
	10min	6.82%	6.21%	54.95%	6.12%	63.06%	5.71%
	30min	9.22%	8.24%	73.81%	8.05%	82.86%	7.69%
	1h	9.79%	8.24%	73.81%	8.05%	82.86%	7.36%
	2h	9.09%	7.91%	71.52%	7.72%	83.03%	7.44%
Lot13 (Business & working)	5min	6.81%	5.92%	59.73%	5.77%	70.13%	5.32%
	10min	6.91%	6.23%	56.21%	6.12%	65.45%	5.71%
	30min	8.34%	7.37%	65.88%	7.20%	77.77%	6.87%
	1h	9.87%	8.15%	69.51%	7.89%	79.69%	7.39%
	2h	9.50%	7.87%	72.77%	7.58%	85.78%	7.26%
Lot19 (Multimodal Hub)	5min	6.73%	5.92%	55.46%	5.81%	63.32%	5.27%
	10min	7.23%	6.30%	57.32%	6.15%	66.74%	5.61%
	30min	8.34%	7.29%	61.14%	7.04%	75.73%	6.62%
	1h	9.21%	7.83%	69.87%	7.63%	79.86%	7.23%
	2h	9.16%	7.70%	70.21%	7.50%	79.98%	7.08%

5.6 Real Implementation

In the research, a pilot SPIMP system was implemented as a sub-system of the Truck Parking Information Management System (TPIMS) on two truck parking rest-areas in the Washington State of USA as test bed, showing in Figure 5.12. Each truck parking lot is monitored by real-time status sensors and the surveillance camera system. The radar-based TPIMS sensors were manufactured by Sensys Networks, Inc. and can provide the real-time occupancy status space by space. The raw inputs by the sensors are sent to the data processing and prediction component. Both the real-time and the forecasting occupancy information is disseminated to drivers and traffic managers for further reference.

In this research, the smart parking information management and prediction system for truck parking in Washington State was put into test in July of 2020. And until now, the entire system is operating stable, and the predicted error is consistent with the error reported in the previous chapter. The system is in testing and will officially open to drivers in late of 2021. Immediately, more parking lots will be connected to this real-time parking prediction system.

5.7 Chapter Summary and Future Works

In this work, I proposed a comprehensive SPIMP system including four components: sensing, data processing, attributes information integration, and prediction components. A deep neural network for integrating heterogeneous data sources for time-variant parking availability prediction is fully integrated into the system to provide managers and drivers with multi-timesacle parking utility prediction. Specifically, four components by learning data representations instead of directly using as input, attributes information and environmental changes are suffi-

ciently integrated into the learning process. To the authors' best knowledge, This article is a pioneer work for the integration of the heterogeneous features for transportation data science research.

Several potential improvements can still be further discussed and investigated. The first and foremost is how to integrate the driver information, especially the habit and behavior, into the prediction framework and used for providing personalized prediction results. How to well capture and integrate the human-activity features in a neural network model become a challenge. Secondly. integrated more dynamic traffic-related features (i,e, real-time flow, speed, and surface condition) into the prediction are also necessary. More generalized traffic factors embedding methods can be developed based on the attributes tensor embedding framework. Finally, in real parking lots, the price is always an important lever to balance the parking supply and demand. Considering the impact of dynamic price strategy and using the information to predict parking utility is necessary. This topic will be further investigated by the author.

Figure 5.12: The architecture of the smart parking information management and prediction system in the pilot test bed.
(a) the overall structure of the SPIMP used in the pilot project, including sensing, data processing and prediction, and information dissemination components;
(b) installation of the radar-based wireless detector made by the Sensys network;
(c) illustration of the in-ground sensor after installation;
(d) the wireless repeaters mounted on the top of streetlight pole;
(e) real-time slot level parking information visualization via customized website;
(f) parking user app visualization.

6

Chapter 6. Integrating the Traffic Science with Representation Leaning for City-scale Network Congestion Prediction

This chapter is modified from the published work:

- W. Zheng[1], H F. Yang*[1], J. Cai, P. Wang, X. Jiang, S. Du, Y. Wang and Z. Wang*. "Integrating the traffic science with representation learning for city-wide network congestion prediction." Information Fusion 99 (2023): 101837. https://doi.org/10.1016/j.inffus.2023.101837

[1]: The authors contributed equally.

6.1 Challenges and Motivations

With the intensification of urbanization, metropolises all over the world are suffering from severe traffic congestions. Only in 2018 Americans lost an average of 97 hours a year per person, costing them nearly $87 billion [112, 277]. In more developed regions, i.e. Boston, Washington D.C., New York City, the "cost of congestion" is significantly higher than average, and leading to more serious economic and environmental loss. To reduce the unnecessary cost, there is an increasing number of travelers who are used to check the current and future traffic conditions before they start the trips [278]. Therefore, assuring the forecasted traffic congestion to closely match the reality is highly integral to the economic efficiency demand.

In recent years, innovations and advances in the Deep Neural Network (DNN) have delivered mechanisms to effectively capture, model and predict the urban traffic patterns, and yielded new understandings about the non-linear transportation systems dynamics [27, 279]. Researchers often cast the traffic modeling problem into a spatial temporal prediction task [280, 27], then adopt various kinds of domain expert DNN models. The current DNN-based traffic prediction frameworks can be roughly summarized into three types: 1) Spatial-temporal Sequential Model (SSM) [250, 281, 282], 2) Spatial-temporal Grid Model (SGM) [32, 33, 283], and 3) the Spatial-temporal Graph Sequential Model (SGSM) [284, 37, 38, 39]. For SSMs [282, 68], the sequential spatial temporal fluctuations are fit by the recurrent neural network (RNN) structures such as the Long Short-Term Memory [250], Gated Recurrent Units [285]. However, RNNs only model the temporal dynamics, hence their abilities to capture the spatial dependencies among neighbor segments are limited. The SGMs are specialized for image-like grid type data and could not handle other data types properly, such as the irregular graph data. They are

usually variants of Convolutional Neural Networks (CNNs), and extract local spatial temporal correlations based on 1D or 2D convolutional kernels. The SGSMs simultaneously capture the spatial information with graph convolutional operation and the temporal dynamics with RNN or temporal convolutions [284, 37, 38]. These DNN models have witnessed successful traffic flow prediction on a handful of datasets crafted from real-world records. However, with the ever increasing scale of the road network and the demand in dealing with more complex traffic conditions, several fundamental challenges still remain to be solved in the current congestion prediction solutions, which we render in FIGURE. **6.1**.

The first crucial desideratum is to build a unified framework to handle various traffic data modalities. This is motivated by practical demands: the types of traffic sensors used by the Department of Transportations contain a broad spectrum – including the magnetic loops, surveillance cameras, vehicles on-board sensors, cell phone users applications etc, and they vary region by region. The collected data could contain the traffic attributes, the static road attributes and beyond. The collected data can be naturally divided into two types w.r.t. the data acquisition/storage modality: the GPS grid modality and the detector graph modality [32, 33]. The GPS grid modality data [34, 35, 36] mainly resort to the GPS signal of a large fleet of probe vehicles. It usually divides the city map into tractable regular grids, then summarizes the road and traffic attributes into each grid cell. On the other hand, in the detector graph modality data, the traffic flow parameters are mainly measured by the loop detectors, which are then summarized into a topological graph based on the road network connection [37, 38, 39]. Current solutions are only designed based on one specific type of modality, for example, the SGMs are specialized in the GPS grid modality and the SGSMs are specialized in the detector graph modality. The attempt to simultaneously predict two types of modalites has not been forthcoming. With

the increasing complexity of measurable traffic parameters within one city, developing an all-inclusive model to support both data types will naturally reduce the system development labor, scales up the prediction in urban networks, and pave the way for future data fusion.

The second challenge dwells in the limited spatial view and propagation modeling. In urban traffic, the origins and destinations of the traffic trajectories extend across the whole city, and cover a wide variety of time span. As a result, the congestion could propagate and exert city-wise (not only local) influences on the traffic flow distribution. However, most of the existing congestion prediction models adopt grid or graph convolution, which only captures local (short range) features [32, 33, 283, 250, 281, 282, 284, 37, 38, 39].

The third challenge is the weak generalization ability towards the heterogeneous road networks. The road networks display great heterogeneity across different cities, i.e., the average distance between intersections, the road alignments, the orientation and density of road segments, all diverse greatly. To assimilate and infer the traffic pattern, the graph models are widely adopted [284, 286, 37, 38, 287, 39]. However, previous methods mainly focus on improving the spatial temporal interactions with graph attention, without taking care of the aggregation mechanism. In other words, they leave the *isotropic* nature of the graph message passing mechanism unchanged. On the other hand, the traffic patterns and the traffic rules are *anisotropic* by nature: the number of vehicles per direction is not symmetric, and the traffic rules are directional. Previous studies revealed a strong relationship between the propagation direction and the physical locations of the road segment [112, 288, 289], and showed that the congestion has strong anisotropic distributions (e.g., traffic tidal flow) [289]. In this way, the current orientations-blind SGSMs [27, 279] are difficult to generalize to heterogeneous road networks.

To tackle the above mentioned challenges, a natural idea is to trace back why congestion always happens, then summarize appropriate priors that dominate the traffic congestion pattern, and inject these priors into the model design. In general, the pixel in the grid data or road section/intersection in the graph data are modeled as the traffic network unit. Systematic transportation science has revealed that the congestion is closely tied to three level of influences between units: the intra-unit interaction (within-self), inter-unit interaction (adjacent units influences), and the network-level propagation [290, 288, 289, 278]. Therefore, a good traffic informed model should be able to capture interactions with all three scales, without being biased to capturing only one or two of them.

In the deep learning community, the transformer family [100, 291, 292, 293] has demonstrated revolutionary potential over a broad spectrum of tasks in the last few years, including natural language processing [292, 100], computer vision [294, 295, 296, 297], graph learning [298, 299, 300], cyber physical systems [286, 301, 302], and more. Instead of solving these tasks with domain expert models individually, the transformer treats the language/vision/graph/sensor type data inputs as a unified and generalized data type: the *sequence*, then showcased that the most generalized sequence modeling architecture with fewest possible inductive biases could lead to even better performances.

In this research, we concretize a new take to address the traffic domain challenges with the generalizable transformer architecture. On one hand, the transformer offers flexible input tokenization, which naturally generalizes across the two data modalities and supports for unification. On the other hand, the *intra-unit interaction* are naturally captured by Transformer's feed forward layers, and the network-level propagation are captured by the attention mechanism, with the missing type of interaction being the *inter-unit interaction* [294, 303, 286].

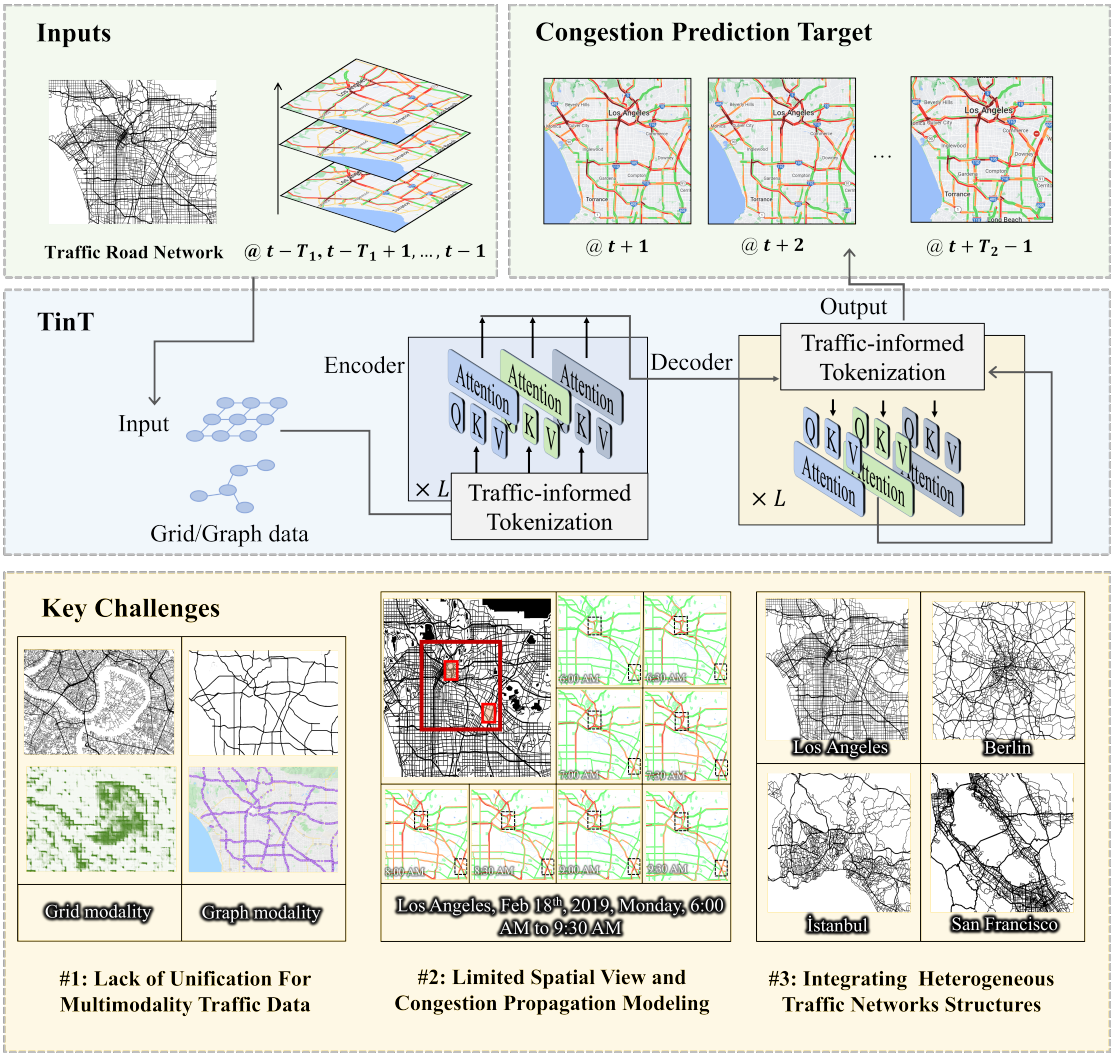


Figure 6.1: The formulation and key challenges of the congestion prediction problem. In this research, a general transformer-based deep learning model is proposed and mainly target to solve three key challenges for network-scale congestion prediction: (b) multimodality data inputs, (c) heterogeneous traffic network structures, (c) spatial-temporal long-term fluctuation and propagation (next page). Three figures are the key challenges for current city-wide congestion prediction.

To fit a sequence with complex distribution, recent researches have shown that the vanilla transformer could learned it well albeit with the cost of massive amount of training data [304, 293, 292]. Yet, we hypothesize that injecting domain specific structures into the transformer

architecture as priors could shed new light on efficient training and generalized inference. The blessings behind domain specialized structures are multiple-fold, as these structural constraints improves the training efficiency and regularize the model to generalize better in this domain [305, 286]. To this end, in order to count for the adjacent unit-unit transformation, we bring the merit of graph models to the transformer, and assign anisotropic graph kernels for neighbors from different physical orientations.

With these domain priors injected, we propose a novel transformer architecture, named the Traffic informed Transformer (TinT). We trained the TinT in six datasets collected belonged cities in North America, Europe and Asian, with different congestion levels, road network structures, cultures, and data modalities. The rendering of these data and the key contributions is presented in FIGURE. **6.2**, and the TinT architecture is depicted in FIGURE. **6.3**. Throughout rigorous evaluations, the TinT significantly surpasses both grid and graph based state-of-the-art models in both modalities, as we shall elaborate in section 6.4.

6.2 Preliminaries And Problem Setting

The overarching goal of traffic and congestion forecast is to predict the *traffic parameters* ahead of time. These traffic parameters contain vehicle attributes that effectively reflect the congestion condition, such as the speed, volume, and orientation, as discussed in the introduction section. These traffic parameters are first observed, then used to train the traffic prediction model as both the input and the prediction target ground truth. In most cases, the range of the prediction covers across the entire city or major rural areas, which contains hundreds of locations, and the prediction lead time lasts from minutes to hours in advance.

As mentioned in introduction 6.1, the traffic parameters are usually observed, organized, and interpreted in two modalities: the GPS grid modality and the detector graph modality. The GPS grid modality summarizes the *vehicle level* information. In the GPS grid modality, the city map is partitioned into regular mesh grids, and the information for each vehicle is measured and summarized into a unified representation within each grid cell ("pixel"). The detector graph modality summarizes the *sensor level* information. The sensors are usually the loop detectors densely populating the road and highways. In detector graph modality, the collection of sensors form a graph, with each sensor being a node at a specific physical location.

Previous approaches leverage the image sequence based model (SGM) to handle the GPS grid modality and use the graph sequence based model (SGSM) to predict for detector graph modality, and a model that unifies these two modalities has not been forthcoming. In this work, we unify these two cases by treating both spatial structure as a graph. In this way, the node of the graph is naturally defined as the grid cell in the GPS grid modality, or the sensor in the detector graph modality. For the edge definitions, in the GPS grid modality, the edge is defined on

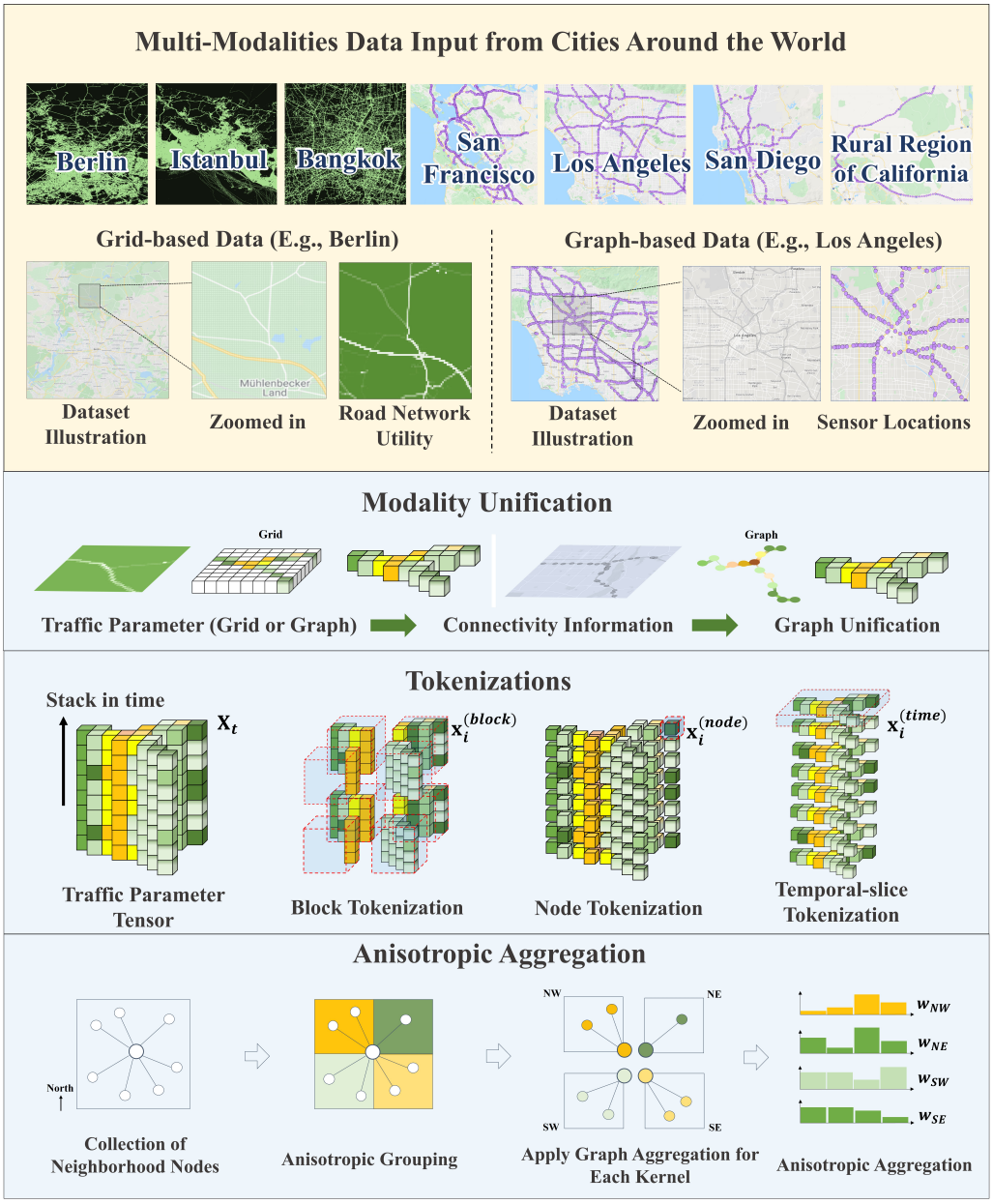


Figure 6.2: The rendering of the multi-modalities data inputs and the customized traffic-informed solutions. The proposed solution includes modality unification, multi-scale tokenizations and anisotropic aggregations. The traffic parameters, i.e., speed, volume, and road network features can be fused into a unified graph. Furthermore the multi-scale tokenization, including block, node and temporal-slice can enhance the regional congestion features. The anisotropic aggregation attention is used to modeling the directional impact of traffic congestion and propagation pattern, then to obtain a more accurate and orientation-aware forecasting reference.

road construction meta information (whether the two pixels have direct road connection), and in detector graph modality, the edge is defined in a similar way but more decided through measurements [284, 37, 38]: within a given time window, for two given nodes, if the ratio of the common passed vehicles versus the total passed vehicles is more than a given threshold, then an edge is added between these two nodes. Under this formulation, the traffic prediction can be naturally formulated as a spatial temporal sequence prediction problem, with the temporal sequence supported on this spatial graph.

There are two properties that make the unified traffic graph unique from the general sense graph: the graph is inserted on a 2D plannar surface, and each node in the traffic graph is tied to a physical location as its meta information, which exerts huge influence over the traffic distribution on these nodes, as we will leverage in the discussions in section the Architecture of the Proposed TinT.

In this research, we adopt the transformer as our base model architecture. Recent researches have demonstrated transformer's outstanding ability of sequence modeling, especially high dimensional sequence distributions at scale, including effective generalization in the domain of natural language processing [292, 100], computer vision [294, 295, 296, 297], graph learning [298, 299, 300], and cyber physical systems [286, 301, 302]. Given the diversity of successful applications of such transformer variants, we further examine yet another critical question: can the transformers be trained to handle not only temporal sequence prediction, but the mixture of spatial temporal sequence prediction problem, especially the economically important traffic sequence prediction task? In the following, we discuss details of the proposed spatial temporal learning transformer architecture, the TinT.

6.2.1 Notations

Before we formally start to introduce the TinT we make the following notations.

- $\mathbb{G}$ denotes the graph supporting the spatial temporal sequence of traffic flow;
- p denotes the physical positions of nodes in the graph $\mathbb{G}$ (which previous approaches are blind of);
- A denotes the adjacency matrix of the graph $\mathbb{G}$;
- t, t_0, t_1, t_2, t_n denotes the time slice indices;
- T_1, T_2 denotes the maximum length of historical temporal sequence and future temporal sequence;
- $\mathbb{X}^{(t-T_1:t+T_2)}$ denotes a sequence of traffic data from $t - T_1$ (inclusive) to $t + T_2$ (non-inclusive);
- $\mathbf{X}^t$ denotes the traffic data at a specific timestamp t (while $\mathbb{X}$ is used to denote a consecutive series of timestamps).

6.2.2 Problem Formulation

We denote the graph that unifies the two modalities as $\mathbb{G}$, and the number of nodes in this graph $\mathbb{G}$ as $|\mathbb{G}| = N_{nodes}$. In the congestion prediction task, the learning target is to predict future status of the *spatial temporal data* given its past status. The spatial temporal data is a discretely sampled sequence on the graph $\mathbb{G}$, where each node in $\mathbb{G}$ has different attributes (traffic flow statistics) at each sampled time slice. The spatial temporal data contains the following components: the

physical location of each node in $\mathbb{G}$, the spatial connectivity pattern (edges) of $\mathbb{G}$, and the spatial temporal flow records over a series of consecutive time slices.

The first element, the physical node locations, is denoted as $p \in \mathbb{R}^{N_{nodes} \times 2}$. In p, the j-th row corresponds to the physical location of the j-th node. The physical location information differs the TinT from previous models: it is highly correlated with the traffic pattern, but previous models fail to explicitly leveraged it. The proposed TinT however, explicitly use p to compute orientation-aware kernels, as specified in section 6.3.3. The second component, the spatial connectivity pattern $A \in \mathbb{R}^{N \times N}$ is a sparse matrix that encodes the adjacent node pairs. Finally, the spatial temporal flow records $\mathbb{X}^{(t-T_1:t+T_2)} \in \mathbb{R}^{(T_1+T_2) \times N_{nodes} \times d}$ stored the traffic parameters at a set of consecutive time slices:

$$\mathbb{X}^{(t_0-T_1:t_0+T_2)} = [\mathbf{X}^{t_0-T_1}, \mathbf{X}^{t_0-T_2}, \cdots, \mathbf{X}^{t_0-1}, \mathbf{X}^{t_0}, \mathbf{X}^{t_0+1}, \cdots, \mathbf{X}^{t_0+T_2-1}] \tag{6.1}$$

In the equation 6.1, $\mathbf{X}^t \in \mathbb{R}^{N_{nodes} \times d}$ represents the graph signal snapshot at time t, T_1 and T_2 are the number of time slices prior to and after the current time step, and d is the dimension of features at each node. To cast the congestion prediction task into a math form, it is to predict $\mathbb{X}^{(t_0:t_0+T_2)} = [\mathbf{X}^{t_0}, \mathbf{X}^{t_0+1}, \cdots, \mathbf{X}^{t_0+T_2-1}]$, given the previous graph sequence $\mathbb{X}^{(t_0-T_1:t_0)} = [\mathbf{X}^{t_0-T_1}, \mathbf{X}^{t_0-T_1+1}, \cdots, \mathbf{X}^{t_0-1}]$. Assume we use a function g to estimate the traffic distribution of future T_2 time steps based on previous T_1 observations:

$[\hat{\mathbf{X}}^{t_0}, \cdots, \hat{\mathbf{X}}^{t_0+T_2-1}] = g([\mathbf{X}^{t_0-T_1}, \cdots, \mathbf{X}^{t_0-1}])$,

and we have a dataset with $|\mathscr{T}|$ samples. The goal is to find a maximal likelihood estimator that best describes all samples of $t_0 \in \mathscr{T}$ in this dataset:

$$g = \arg\max_{g} \prod_{t_0 \in \mathcal{T}} \mathbb{P}([\hat{\mathbf{X}}^{t_0}, \cdots, \hat{\mathbf{X}}^{t_0+T_2-1}] = [\mathbf{X}^{t_0}, \cdots, \mathbf{X}^{t_0+T_2-1}] | \mathbf{X}^{t_0-T_1}, \cdots, \mathbf{X}^{t_0-1}) \quad (6.2)$$

This distribution g is not directly measurable, and we fit the proposed transformer model on available real-world traffic data to capture it.

6.3 The Architecture of The Proposed TinT

6.3.1 Traffic-informed Tokenization

To readily build the traffic domain priors into an actionable algorithm design, principled and automated tokenization is demanded to appropriately organize the inputs. In general, the traffic flow data is one type of sequence, to be specific, it is a temporal sequence supported on a graph. To organize the input for the sequence, the transformer family utilizes the tokenization [306, 307]. However, being a spatial temporal graph sequence, the traffic flow data has unique structures, which makes it intrinsically different from the sequences in other domains tasks that the current state-of-the-art transformers are good at, such as natural language processing (NLP), static graph learning or computer vision (CV). In contrast, in the most common case the sequences in graph or CV tasks extend in one direction, either spatial or temporal, while the traffic data naturally extends both in spatial and temporal directions. The co-existence of spatial and temporal directions enabled more ways to slice, view, and process the traffic data. Therefore, special designs are needed to adapt the tokenization to the traffic congestion prediction task.

In this research, we employ the collection of three tokenization approaches: the time slice

tokenization, the node tokenization, and the spatial temporal block tokenization. Each tokenization poses a unique way to slice/partition the traffic flow data, the $\mathbb{X}^{(t-T_1:t+T_2)}$, which is a tensor with shape $(T_1+T_2)\times N_{nodes}\times d$.

Each tokenization is linked to one type of processing unit: the time slice tokenization connects to the encoder/decoder (to be discussed in section 6.3.2), the node tokenization connects to the anisotropic aggregation (to be discussed in section 6.3.3), and the spatial temporal block tokenization connects to the global attention module (to be discussed in section 6.3.4).

The architecture of the TinT model is visualized in FIGURE. **6.3**. When the TinT processes the traffic data, it first uses the *node tokenization* and *spatial temporal block tokenization* to partition the data, then sent the resulting tokens to two different processing units to process in parallel, then, the outputs of these units are added back together. The three different tokenizations are visualized in FIGURE. **6.2**.

In the following, we mathematically formulate these tokenizations as partition functions: we write the expression of the i-th token of each type. Here i is an integer index ranging from o to *the number of tokens* -1 in each tokenization type, which is different for each tokenization. We use $\mathrm{vec}(\cdot)$ to denote the vectorization operation, which flattens any tensor into a vector. The **time slice tokenization** cuts the tensor $\mathbb{X}^{(t_0-T_1:t+T_2)}=[\mathbf{X}^{t_0-T_1},\cdots,\mathbf{X}^{t_0},\cdots,\mathbf{X}^{t_0+T_2-1}]$ into time slices, with each one indexed by the time step. The i-th token is hence:

$$\mathbf{x}_i^{(time)}=\mathrm{vec}(\mathbf{X}^t|_{t=i}) \tag{6.3}$$

The **node tokenization** further cuts the time slice tokens into nodes representations, and each output token of the node tokenization is indexed by the time index t and the spatial index j

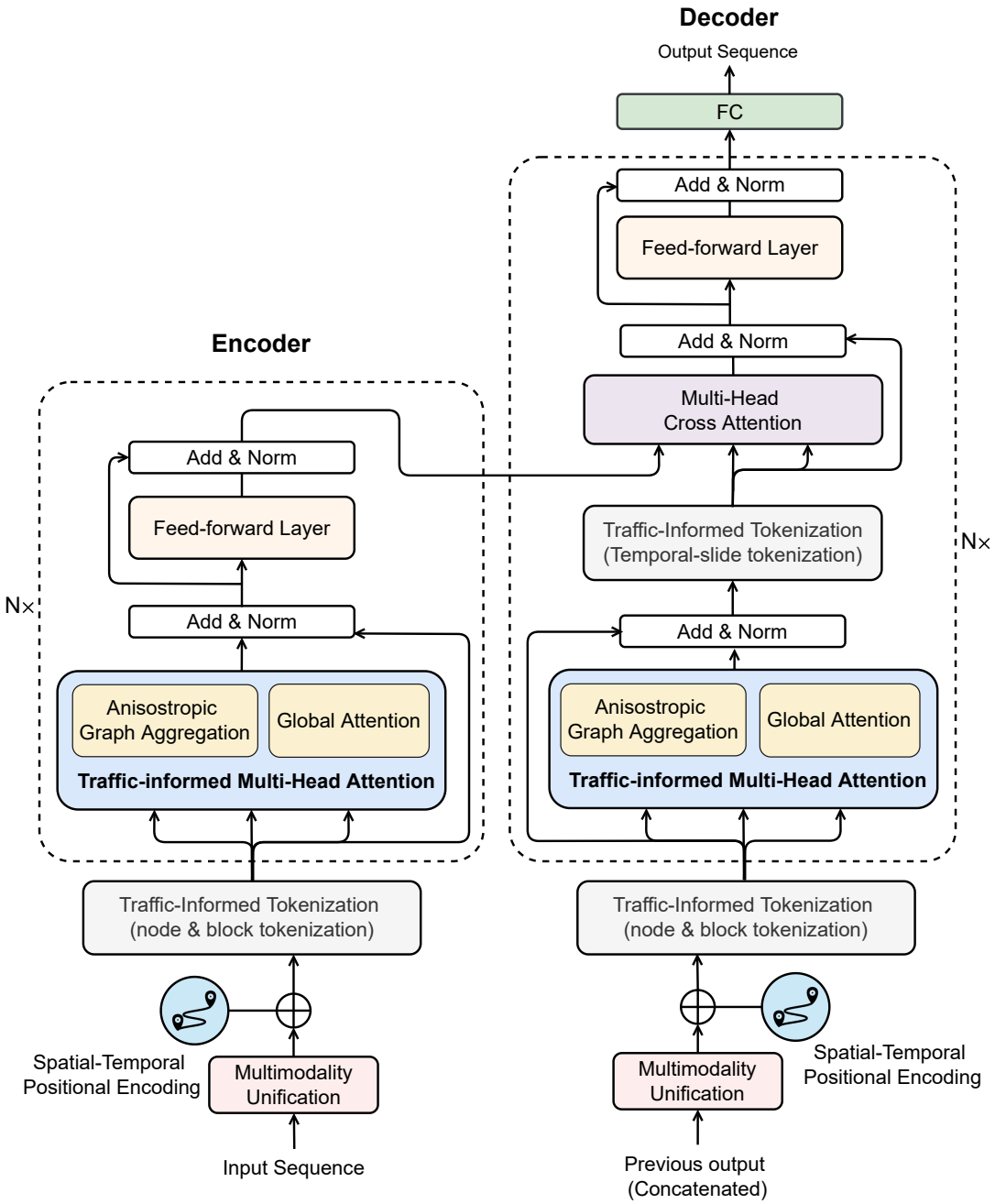


Figure 6.3: The architecture of the proposed TinT. Two key traffic-informed novel deep learning component is designed and integrated, including the multi-scale tokenization and traffic-informed multi-head attention.

(meaning the j-th node). The i-th token is:

$$\mathbf{x}_i^{(node)} = \text{vec}(\mathbf{X}^t[j]), i = t \times N_{nodes} + j \tag{6.4}$$

The **spatial temporal block tokenization** cuts the tensor $\mathbb{X}^{(t_0-T_1:t+T_2)}$ into a list of spatial temporal sub-regions, then each token is a serialized sub-region node representations. Each token contains a portion of nodes that are spatial temporal neighborhoods, and different tokens are also spatially and temporally adjacent but not overlapped. Consider a group of spatially adjacent nodes $[j, j+1, \cdots, j'-1]$, their respective time steps $[t, t+1, \cdots, t'-1]$ are partitioned into the same token. In this way, the i-th token is:

$$\mathbf{x}_i^{(block)} = \text{vec}(\mathbb{X}^{(t:t')}[j:j']), i = \frac{t}{t'-t} \times \frac{N_{nodes}}{j'-j} + \frac{j}{j'-j} \tag{6.5}$$

In practice, if N_{nodes} is not divisible by the node group size $j'-j$, then the remainder is padded by zero embeddings to make it divisible.

6.3.2 The Encoder Decoder Architecture

We take the encoder decoder architecture following the design of the transformer [100]. Here, the encoder transform the input token sequence $\mathbb{X}^{(t_0-T_1:t_0)} = [\mathbf{X}^{t_0-T_1}, \mathbf{X}^{t_0-T_2}, \cdots, \mathbf{X}^{t_0-1}]$ to a sequence of intermediate representations $\tilde{\mathbf{X}} = [\tilde{\mathbf{X}}^{t_0-T_1}, \tilde{\mathbf{X}}^{t_0-T_2}, \cdots, \tilde{\mathbf{X}}^{t_0-1}]$ for the decoder, while the decoder apply the causally masked attention to predict the final output. Given $\tilde{\mathbf{X}}$, the decoder generates the output sequence one element at one step. At each step, the model

is auto-regressive [308], consuming the previously generated symbols as additional input when generating the next one: in order to predict $\mathbf{X}^{t_2}$, it first leverages $\mathbb{X}^{(t_1-T_1:t_1)}$ to predict $\mathbf{X}^{t_1}$, then leverages $\mathbb{X}^{(t_1-T_1:t_1+1)}$ to predict $\mathbf{X}^{t_1+1}$, and so forth, until leveraging $\mathbb{X}^{(t_1-T_1:t_2)}$ to predict $\mathbf{X}^{t_2}$.

Similar to the original transformer model architecture, both the encoder and the decoder are composed of stacks of identical layers, where each layer contains the self-attention sublayer and the feed forward sublayer. For the decoder layers, it additionally inserts a third attention sublayer after the self-attention, which let the decoder sequence attend to the output of the encoder stack. The attention sublayers in the decoder stack are modified to prevent attending to subsequent data. This causal masking ensures that the predictions for time i can depend only on the known outputs prior to i. There is a residual connection around each of the two sub-layers, followed by a layer normalization. The TinT inherits the same encoder decoder structure as the original transformer [308]. The the model architecture is shown in FIGURE. **6.4**.

6.3.3 Modeling the Anisotropic Graph Aggregation

In general, the traffic prediction model is expected to leverage a certain design that allows interactions between neighboring regions. Among the neighborhood interaction mechanisms, the most successful one is the graph convolution in GCN [309], which is widely adopted in previous traffic prediction benchmarks [284, 37, 38, 39]. It aggregates the information from neighbors with the simple yet effective neighborhood aggregation function.

Despite the blessing of effectiveness and verified success of such graph aggregation in various tasks, it still faces an important gap: this type of aggregation is "isotropic", which means all neighborhoods are treated equally. This attributes works for a graph without physical location attributes, but under the location attributed traffic network, it fails to distinguish traffic flows

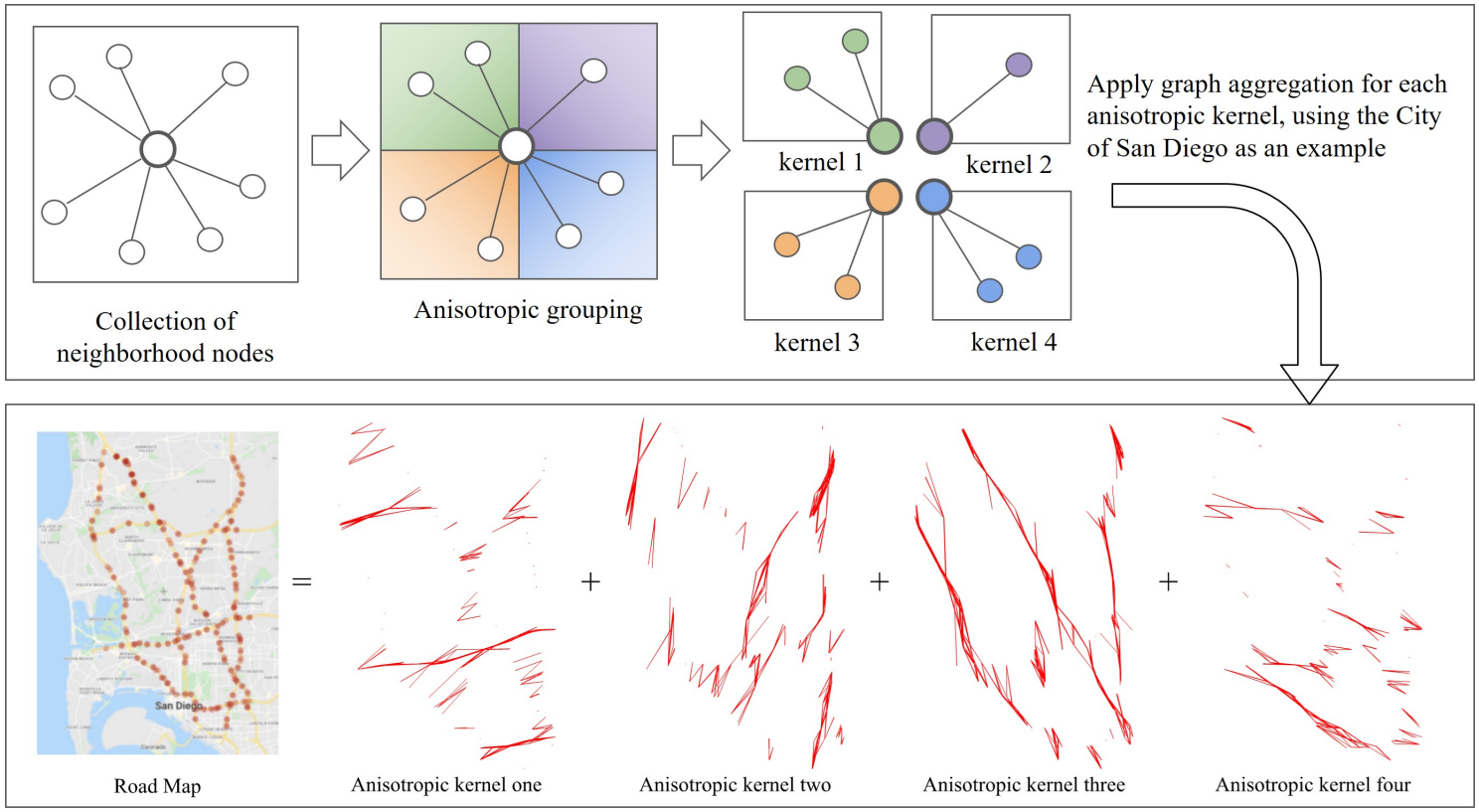


Figure 6.4: The visualizations of anisotropic graph aggregation and the traffic science integrated TinTand key components illustrations. The upper subfigure is a node centric visualization while the lower subfigure is a graph level visualization, displayed for a subset of San Diego nodes. In the figure, the colored layers represent the adjacent time stamps, the dots in each layer represent a node, and four adjacent time stamps are shown in this illustration. In the lower subfigure, the red lines means the anisotropic edge connection, which is used by the anisotropic graph kernels to perform aggregation. These two subfigures only shows the spatial division for one time stamp, while multiple time stamps are included in the implementation. More details are discussed in section 6.3.3

from different directions. This important gap urge us to design a novel physical location aware aggregation mechanism.

As discussed in section 6.1, in this work, we assume the anisotropic orientation distribution [290, 288, 289] and the non-ideal permutation invariance [284, 37, 38] of the traffic data. In other words, the traffic flow on north/south/west/east, or more fine-grained partitions of orientations, will affect the center node differently [289]. The vanilla transformer offers no mechanism to tie the token representation to the token's true physical locations. That is, the token representation and transformation will show no difference if the nodes are physically located elsewhere, hence the structural features of the road network are not fully leveraged.

In the following, we first mathematically illustrate the aggregation function of the traditional transformer and the GCN, explain their orientation unawareness, and then discuss the anisotropic graph aggregation.

The Orientation Unawareness of Transformer and GCN For a given node i of interest, denote $\mathbf{x}_i$ as its embedding and $\mathcal{N}(i)$ as the collection of its neighbors. Then, the aggregation/attention procedure of the GCN and Transformer for token $\mathbf{x}_i$ can be expressed as:

$$\text{GCN} : \mathbf{x}_i \leftarrow \sum_{j \in \mathcal{N}(i)} f^{(GCN)}(\mathbf{x}_j) \tag{6.6}$$

$$\text{Transformer} : \mathbf{x}_i \leftarrow \sum_{j \in \mathbb{G}} f^{(trans)}\left(\mathbf{x}_j, \mathbf{x}_i\right) \tag{6.7}$$

where $f^{(trans)}(\cdot,\cdot)$ is the binary attention based aggregation function for Transformers, which is:

$$f^{(trans)}(\mathbf{n},\mathbf{y}) = \bigcup_{h=1}^{H} \left(\underset{\mathbf{n}\in\mathbb{G}}{softmax} \left(\mathbf{y}^{\top} W_{Q,h}^{\top} W_{K,h} \mathbf{n} \right) W_{V,h} \mathbf{n} \right) \tag{6.8}$$

where $\mathbf{n}, \mathbf{y}$ are two arbitrary tokens after processing by $f(\cdot)$ at the same layer, h is the index of attention head, and $\cup$ is the concatenation operation, which concatenates all of H attention heads.

From the above equations 6.6 and 6.7, all nodes or neighborhoods, regardless of their relative orientation to the center node i, is processed by the *same* kernel (transformation function) $f(\cdot)$.

Orientation-aware Graph Kernels As shown above, GCN and Transformer both process the node interactions with a *unified* function. In this way, the graph kernels applied in previous traffic prediction researches [284, 37, 38, 39], though have been improved with domain specific modifications, all fall into this pitfall as the neighborhood aggregation function are the same.

In this research, we overcome the ignorance of orientation issue by assigning different kernels for different oriented node pairs. We achieve this by separating $f(\cdot)$ into different kernels, each one now with unique subscript. These kernels differ by the orientation between the center node and the neighborhood node, and the number of hops from the center node. The neighborhood nodes are pre-divided into different groups that form into concentric circles, and those with similar orientations from the center node and at the same number of hops stay within the same group.

Consider a center node i of interest. Define the tier τ as the collection of nodes that are ex-

actly τ+1 hops away from node i. In TinT, the kernels are assigned to neighborhoods with a maximum of Γ tiers (hops). At each tier $\tau, \tau = \{0, 1, \cdots, \Gamma - 1\}$, there are K kernels assigned to the neighboring nodes, which forms a family of multiple relational kernels $f_k(\cdot)(k \in \{\tau * K, \tau * K + 1, \cdots, \tau * K + K - 1\})$. In this way, a total of $\Gamma * K$ kernels are assigned, and each one is applied to one group of nodes. These groups (kernels) uniformly distribute at the subsequent tiers, evenly segmenting each tire into equally sized sectors.

Denote $\mathscr{ON}(i)$ as the *O*riented *N*eighborhoods of node i. Each neighbor node in $\mathscr{ON}(i)$ has an index j, $\alpha \in [0, 2\pi)$ as the orientation of node pair (i, j), and τ as the hop from i to j. We use kernel ID $n, n \in \{0, 1, \cdots, \Gamma * K\}$ to separate the isotropic aggregation $f^{(GCN/trans)}(\cdot)$ into anisotropic kernels $f_n^{(TinT)}(\cdot)$. The kernel ID contains the orientation part $\lfloor \frac{\alpha * K}{2\pi} \rfloor$, which is an integer from 0 to $K - 1$, and the hop part $\tau * K$. The proposed kernels are written as follows:

$$\text{TinT Anisotropic Kernel} : \mathbf{x}_i \leftarrow \sum_{(j;\alpha,\tau) \in \mathscr{ON}(i)} f^{(TinT)}_{\tau * K + \lfloor \frac{\alpha * K}{2\pi} \rfloor} \left(\mathbf{x}_j \right) \tag{6.9}$$

The kernel function architecture is an open design, and in this work we choose the simple yet effective non linear feed forward transformation:

$$f_n^{(TinT)}(\mathbf{x}_i) = \text{ReLU}\left(W_n \mathbf{x}_i + b_n \right) \tag{6.10}$$

The real example of spatial visualization for the anisotropic aggregation is displayed in FIGURE. **6.4**.

6.3.4 Capturing Long Range Dependency by Global Attention

Learning long range dependencies is a key challenge in many sequence transduction tasks [310, 311, 312]. In particular, the long range patterns ubiquitously exist in the traffic data, both spatially and temporally [286, 310, 313]. As a result, the ability to model long range traffic propagation patterns will have a critical impact on the model's performance. However, previous studies [284, 37, 38, 39] all rely on local view message passing, and there is no component that could directly pass information to far-away node pairs. These approaches adopt the graph convolution or graph attention, which only pass information to the immediate neighbors. In order to route information to H-hop away nodes, these instant neighbor message passing approaches need H layers. Therefore, the length of the paths where forward and backward signals have to traverse in the network is long. The *shorter* these paths between any combination of positions in the input and output sequences, the *easier* it is to learn long range dependencies [314], which put these approaches under suboptimal conditions.

In this research, we leverage the transformer's attention mechanism to directly route information to arbitrarily far away nodes. The long range attention is achieved by means of the proposed spatial temporal block tokenization. In this type of tokenization, nodes who are spatially and temporally near each other are concatenated and aggregated as a block using equation 6.5. Each block is treated as a new token for this module, and is passed through the transformer's attention layers. We evenly assign the spatial and temporally adjacent nodes into B spatial temporal blocks. When B is not an exact division of the number of nodes, we zero-pad the node features so as to make every spatial temporal block contain the same number of nodes.

Besides this long range attention mechanism, recall that the anisotropic graph kernels are responsible for short range aggregation. We assemble the long and short range information routing

mechanisms parallelly in each layer, and let them interleave the processing of global or local information across adjacent layers. In this way, the bias of either short or long range receptive fields are avoided. The interplay between the long and short range information routing mechanisms will naturally lead to a more versatile model capable to generalize better.

6.3.5 Complementary Components of TinT

In the transformers, there are repeated blocks with the same feed forward neural network and attention architecture. We aim to utilize the generality of the transformers and follow this framework, except that the transformer attention module is replaced by the parallel combination of the modules above. For the positional encoding in TinT, we add two sets of positional encodings, spatial and temporal positional encodings, into the node representation together. That is, for the i-th node at time step t, the representation after the positional encoding is $\bar{\mathbf{x}}_i^t + p_i + q_t$, where p_i is the spatial positional encoding for node i, and q_t is the temporal positional encoding for time t, and $\bar{\mathbf{x}}_i^t$ is the feature of node i at time t ($\mathbf{x}_i^t$) transformed by the input projection layer of the transformer.

6.3.6 Training approaches and configuration details

We train the model with a pre-training stage followed by a fine-tuning stage. In the first stage, we train the encoder and decoder separately, by minimizing MAE loss of the model output and the ground truth:

$$loss_{\text{ Encoder}} = ||\mathbb{X}^{(t-1:t+T_2-1)} - Encoder(\mathbb{X}^{(t-T_1):t})||_1 \tag{6.11}$$

$$loss_{\text{Decoder}} = ||\mathbb{X}^{(t:t+T_2)} - Decoder(\mathbb{X}^{(t-1:t+T_2-1)})||_1 \tag{6.12}$$

Then in the fine-tuning stage, the model is trained on the same dataset but in an end-to-end way: the encoder and the decoder are concatenated, and are trained to minimize the MAE loss of the final prediction against the ground truth:

$$loss = ||\mathbb{X}^{(t:t+T_2)} - TinT(\mathbb{X}^{(t-T_1):t})||_1 \tag{6.13}$$

In our model, we use three layers of encoder and decoder blocks, which means $N = 3$ as in Fig. 6.3. In the anisotropic graph aggregation, we choose the number of anisotropic kernels to be four, so as to mimic the four-way traffic flows in most intersections.

6.4 Experiment Results and Discussion

6.4.1 Dataset Summary

We follow a rigorous and systematic procedure to evaluate the performance of the proposed methodology in congestion forecasting. We compared the model with 13 state-of-the-art baseline models, across 8 datasets from 6 culturally diverse cities around the world, including both the detector graph modality datasets (Los Angeles, San Francisco, San Diego) and the GPS grid modality datasets (Berlin, Istanbul and Bangkok). Both the downtown and rural regions are included in these datasets, with the respective scales marked in Table 6.2. In the GPS grid modality

datasets, each pixel represents a 100m × 100m area. The detector graph modality datasets are collected from 01/02/2019 to 04/12/2019, and the GPS grid modality datasets are generated from 1/2/2019 - 3/2/2019. The measurement intervals for both modalities are 5 minutes, and the prediction target time spans are all one hour, i.e. 12 time steps, in the future. The input historical length is also limited as one hour [38, 37], which means we use 12 time steps in the past to predict 12 time steps in the future. All models are trained separately in each city. We randomly split 60% of the data for each city as training, 20% as validation, and 20% as testing set. The processing procedure of the GPS grid modality and detector graph modality data, as well as the rendering for all datasets are provided in Fig. 6.2. We lay out the quantitative results and analysis in the sequel.

6.4.2 Performance comparison with state-of-the-art models

On the GPS grid modality data, we compared the TinT with SOTA grid-based baseline models: UNet2D with LSTM [315], UNet3D [316], Vision Transformer [317], Video Transformer [318] and Swin Transformer [319]. On the detector graph modality data, we compared the TinT with the following current traffic prediction benchmarks: basic LSTM [250, 281], Local Spectral Graph Convolution LSTM (LSGC-LSTM) [281], Spectral Graph Convolution LSTM (SGC-LSTM) [320], Multi-View Spatial-Temporal Graph Convolutional Networks (MSTGCN) [37], Attention Based Spatial-Temporal Graph Convolutional Networks (ASTGCN) [37], Adaptive Graph Convolutional Recurrent Network (AGCRN) [321] and Attention based Spatial-Temporal Graph Neural Network (ASTGNN) [38].

We choose the following three principled evaluation metrics: the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), and the Mean Absolute Percentage Error (MAPE). The

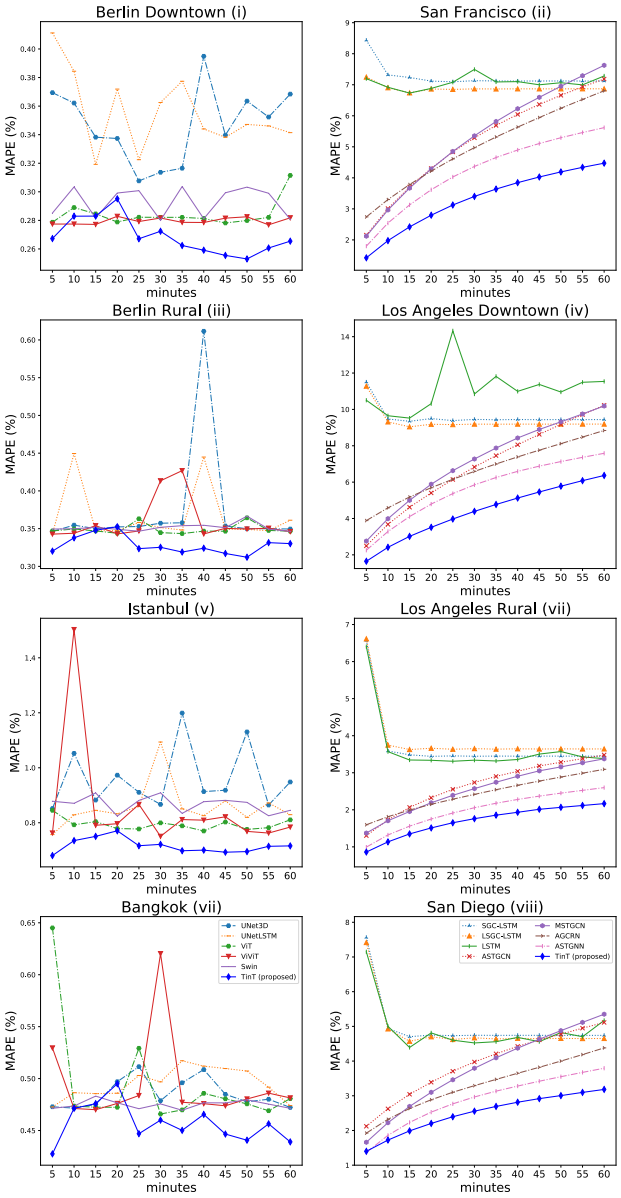


Figure 6.5: The prediction MAPE results expanded into different time spans in the future. Sub-figures i/iii/v/vii on the left are grid modality, and sub-figures ii/iv/vi/viii on the right are graph modalities. The legends for the grid data are shared and displayed in the subfigure of Bankok, and the legends for the graph data are shared and displayed in the subfigure of San Diego.

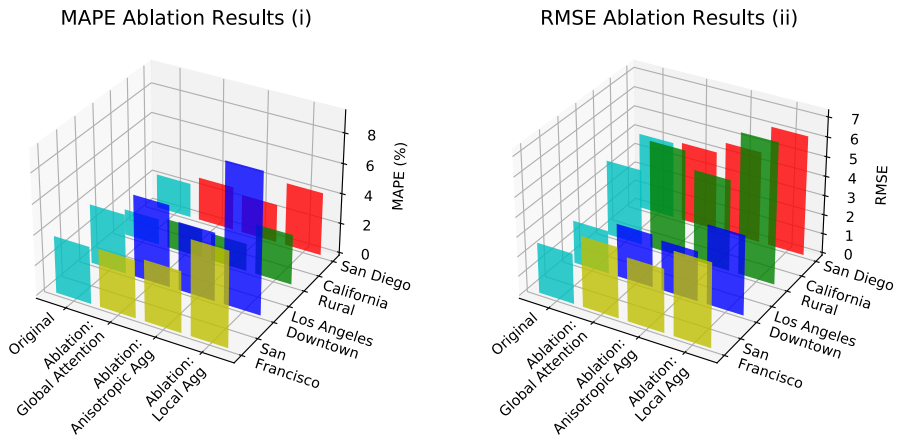


Figure 6.6: The ablation study results of the graph based data, reported in MAPE and RMSE. Compared to the original model, the three types of ablation studies are: transformer global attention, local graph aggregation, and the anisotropic kernals in the local aggregation, where we dropped the corresponding components and re-train the model. More details are discussed in section 6.4.3

overall comparison results are shown in TABLE. **6.1** and **6.2**. In addition, the prediction at every 5 minutes in the future are also plotted in FIGURE. **6.5**. As can be seen in TABLE.**6.1**, TABLE. **6.2** and FIGURE. **6.5**, the proposed TinT scores the best across the entire one hour forecasting range, with an average of 12% MAPE improvement than the second best state-of-the-art (SOTA) model.

Recall that the TinT is a general sequence prediction model by its nature, simply with traffic domain primitives injected. The fact that the TinT outperforms both the *graph expert models* in detector graph modality and the *image domain expert models* in the GPS grid modality again demonstrated the expressiveness power of a general sequence model. Besides better performance, another blessing from being a general sequence model is the a better potential towards future data fusion from both modalities. At current stage, what is lacking is the data availability, i.e., the two modalities are deployed in different regions due to local policies, managements and developments. With the increasing deployment of the internet of things, we envision a larger

family of data fusion algorithms based on TinT could be inspired when both data modalities are made accessible in the same city.

Table 6.1: The TinT Framework Result Summarization and Comparison with SOTA Methods on Grid-based Datasets. As can be seen in the table, the proposed method consistently achieved the best performance on the grid-based datasets.

Data Type	Cities	Methods	Evaluation Matrix		
			MAE ↓	MAPE (%) ↓	RMSE ↓
Grid-based Data	**Berlin Downtown** (1024 grid, 10.32 km^2)	UNet2D and LSTM [315]	0.241	16.2	0.356
		UNet3D [316]	0.235	15.8	0.347
		Vision Transformer [317]	0.192	13.0	0.284
		Video Transformer [318]	0.189	12.8	0.280
		Swin Transformer [319]	0.198	13.4	0.293
		TinT (proposed)	**0.182**	**12.3**	**0.269**
	Berlin Rural Area (TBD) (1024 grid, 10.32 km^2)	UNet2D and LSTM [315]	0.249	16.7	0.366
		UNet3D [316]	0.257	17.0	0.374
		Vision Transformer [317]	0.236	15.9	0.349
		Video Transformer [318]	0.243	16.4	0.359
		Swin Transformer [319]	0.238	16.0	0.352
		TinT (proposed)	**0.222**	**15.0**	**0.328**
	Istanbul Downtown (1024 grid, 10.32 km^2)	UNet2D and LSTM [315]	0.582	39.1	0.858
		UNet3D [316]	0.652	43.8	0.959
		Vision Transformer [317]	0.536	36.2	0.794
		Video Transformer [318]	0.591	38.9	0.852
		Swin Transformer [319]	0.586	39.6	0.867
		TinT (proposed)	**0.484**	**32.7**	**0.716**
	Bangkok Downtown (1024 grid, 10.32 km^2)	UNet2D and LSTM [315]	0.335	22.6	0.495
		UNet3D [316]	0.328	22.1	0.486
		Vision Transformer [317]	0.335	22.5	0.493
		Video Transformer [318]	0.335	22.5	0.494
		Swin Transformer [319]	0.321	21.7	0.475
		TinT (proposed)	**0.308**	**20.8**	**0.456**

The models' predictions are depicted in FIGURE. **6.7**, which visualizes the traffic flow on a

Table 6.2: The TinT Framework Result Summarization and Comparison with SOTA Methods on Graph-based Datasets. As can be seen in the table, the proposed method consistently achieved the best performance on the graph-based datasets.

Data Type	Cities	Methods	Evaluation Matrix		
			MAE ↓	MAPE (%) ↓	RMSE ↓
Graph-based Data	**San Francisco** (1358 nodes, 13 freeways and highways)	LSTM [250]	3.45	7.14	6.03
		LSGC-LSTM [281]	3.65	6.89	6.11
		SGC-LSTM [320]	3.89	7.25	6.29
		MSTGCN [37]	2.39	5.31	4.80
		ASTGCN [37]	2.30	5.18	5.09
		AGCRN [321]	2.23	5.01	4.89
		ASTGNN [38]	1.92	4.21	4.21
		TinT (proposed)	**1.69**	**3.60**	**3.70**
	Los Angles Downtown (1432 nodes, 18 freeways and highways)	LSTM [250]	4.43	11.11	7.63
		LSGC-LSTM [281]	4.60	9.46	7.63
		SGC-LSTM [320]	4.73	9.60	7.73
		MSTGCN [37]	2.84	7.17	5.47
		ASTGCN [37]	2.81	6.87	5.80
		AGCRN [321]	2.63	6.64	5.53
		ASTGNN [38]	2.25	5.62	5.02
		TinT (proposed)	**1.88**	**3.96**	**3.92**
	California Rural Freeway (1432 nodes, 18 freeways and highways)	LSTM [250]	2.19	3.66	4.10
		LSGC-LSTM [281]	2.36	3.90	4.20
		SGC-LSTM [320]	2.25	3.72	4.14
		MSTGCN [37]	1.31	2.56	2.92
		ASTGCN [37]	1.32	2.67	3.15
		AGCRN [321]	1.25	2.43	2.91
		ASTGNN [38]	1.01	2.00	2.71
		TinT (proposed)	**0.93**	**1.74**	**2.22**
	San Diego (862 nodes, 9 freeways and highways)	LSTM [250]	2.61	4.91	4.96
		LSGC-LSTM [281]	2.77	4.90	4.99
		SGC-LSTM [320]	2.81	4.99	5.07
		MSTGCN [37]	1.72	3.83	3.78
		ASTGCN [37]	1.68	4.03	3.91
		AGCRN [321]	1.54	3.31	3.69
		ASTGNN [38]	1.37	2.88	3.34
		TinT (proposed)	**1.15**	**2.32**	**2.59**

subset of nodes in San Diego, containing both ground truth and the predictions of four different models. Each visualization subfigure represents one specific time slice, and different colors represent the average vehicle speed values. As shown in FIGURE. **6.7**, the TinT is the most ac-

curate in picking the congestion points (i.e. the south-west direction of I-805 and I-94 freeway overpass near San Diego downtown, the west direction of I-8 and I-125 freeway overpass). In contrast, other models are either weak to capture the spatial temporal pattern (LSTM) or have large prediction error so that the long term prediction drift seriously (MSTGNN, ASTGNN).

6.4.3 Before-After Ablation Study and Analysis

In this section, extensive ablation studies are conducted to verify the contribution of each component: the local-view attention mechanism, the global-view attention mechanism, and the anisotropic graph aggregation. In the ablation for global attention and local attention, we remove the transformer attention mechanism and the graph kernels, respectively. In these two cases, the TinT reduces to the anisotropic graph model and transformer model, respectively. In the ablation for anisotropic kernels, we replace the anisotropic kernels into a single graph convolutional kernel. We report the MAE and the RMSE metrics in FIGURE. **6.6**.

The Effect of Mixed Global And Local Views

Almost all previous models are biased towards only capturing localized features. In the proposed TinT, we propose the mixture of global and local view to effectively avoid the model to fall into this trap. As can be seen in FIGURE. **6.6**, the TinT outperforms its counterparts where one of the global or local attention components is dropped. We observe that dropping the global attention is less harmful than dropping the local attention. This is not a surprising observation, but it explicitly verifies an assumption that is blindly adopted by previous researches [281, 320, 37, 321, 38]: the local pattern is more important than the global pattern in the traffic data. However, this does not mean that the global pattern is not important, on the contrary, we

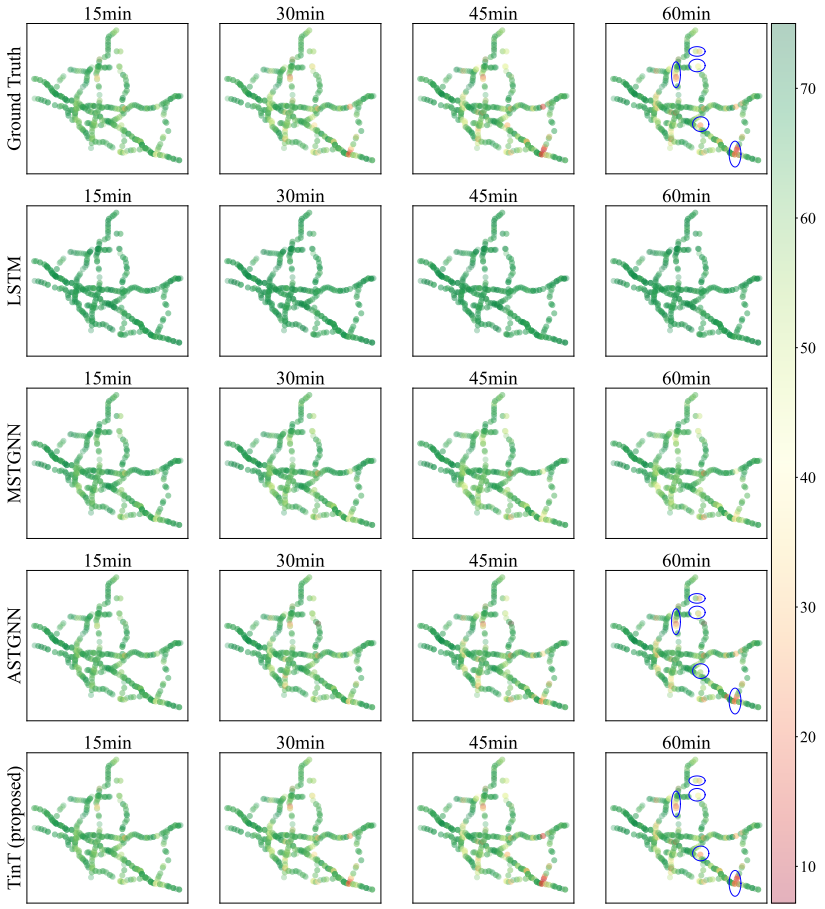


Figure 6.7: Visualization of different model predictions versus the ground truth. For better visualization, the nodes shown are sub-sampled from the SD traffic network. It can be seen from the results that the TinT captures the congestion conditions better than other models. Especially, in some sensitive regions marked with ellipse, other models are unable to make congestion predictions as accurate as the TinT.

see consistent improvements of the fully fledged TinT over the ablation model where the global attention mechanism is removed.

From another perspective, in FIGURE. **6.5**, the performance gap between TinT and other models increase when the time span grows larger. The global-view helps the TinT to drift less when the prediction time is longer ahead. This advantage could be attributed to the following reasons. First, the global view helps alleviate the bottleneck [322] and over-smooth [323, 324, 325] issues that are inherent to local-view graph convolutions. In graph convolution based models, in order to make long term prediction, one has to stack more layers. However, the amount of contexts that a node could receive grows dramatically at deeper layers, which tends to force the exponentially increasing information to be squeezed into a fixed small number of edges of this node, making it difficult to generalize [322, 326]. Second, the overall traffic is the composition of trips, and each trip has unique global Origination and Destination (OD). The local-view module cannot pass information globally and ignore these long range OD dependencies. Last but not least, the congestion is caused by the excessive traffic demand w.r.t. the road capacity, which does not surge immediately but builds up over time. Hence, for congestion prediction, the long temporal dependency is also important, yet missing in many graph-based models. So, with the combination of the short and long range views, both spatially and temporally, the TinT takes the advantage of both sides and achieves the best performance.

The Effect of Anisotropic Graph Aggregation

The TinT treats the node pairs from different spatial temporal directions differently, and use separate graph convolution kernels to learn over these directions. In this way, the TinT successfully links its forward processing procedure to the actual physical location of the node. In the results, we see that the anisotropic graph aggregation consistently improves the performance, as reflected in the MAPE and RMSE of FIGURE. **6.6**. The traffic flow distributions display

strong anisotropy properties, as both the trips and the traffic rules are spatially anisotropic. For the trips, the drivers' origins and destinations vary from place to place, and certain regions in the city receive much greater demand and in-flow than other regions. For the traffic control rules, the local traffic signals at the intersection give way to different traffic directions staggered time, leaving the local traffic patterns differ from directions. However, the current SOTA models [37, 321, 38] directly apply the vanilla isotropic graph convolution without careful treatment. In contrast, the proposed TinT incorporates the anisotropic graph aggregation, which separates the whole graph into multiple subgraphs based on physical orientations, and separately learns graph kernels. An illustration of the anisotropic subgraph partition is shown in FIGURE. **6.4**. With the customized anisotraphic graph aggregation, the model is informed with physical location interaction and orientation information, and is able to distinct traffic flow patterns from different physical directions under various road section connections.

6.5 Chapter Summary and Future Works

In this work, we address the city scale congestion prediction challenge. We orchestrate a transformer based model named TinT that effectively overcome the limitations of the current traffic prediction algorithms: inadequate propagation modeling, insufficient generalization, and inability to unify different modalities due to over specialized architecture design. We investigate the key attributes of the traffic flow that are not considered by previous studies, including the anisotropic traffic flow pattern and the long range and short range dependency. We selected the transformer to model the traffic data as sequence, and inject these attributes as traffic domain priors into the model architecture design, and present a new transformer-based model for spatial temporal data learning and prediction – the TinT.

With the help of novel anisotropic graph kernel, the TinT could better distinguish the traffic flow from different orientations. With a novel routing mechanism design that mixes global and local information, the TinT is not biased to either long range or short range traffic patterns. Using the tokenization designs carefully tailored from the traffic domain knowledge, the TinT for the first time learns to assimilate and infer for both two data modalities under the same model architecture. We extensively compared the proposed model with existing state-of-the-art approaches over diverse prediction range under two data modalities: the GPS grid modality data and detector graph modality data, which both come from challenging real-world scenarios. Our result shows that the TinT outperforms all other expert neural networks models for traffic flow prediction, and is the most accurate in picking the congestion point during the entire one hour forecast.

Our work promulgates new mechanisms that rope the traffic propagation and new factors that influence the model performance, opens up the opportunities to push forward the state-of-the-art congestion prediction, and brings about significant practical impact. On one hand, the TinT unified two data modalities, which offered one potential to fuse different data sources within the same city, when they are available with the increasing development of infrastructure. Even in the current data availability status, this unified framework still significantly reduce the labor of software development, as only one model is needed to develop, train and deploy, in order to address different cities/regions with incongruent data modalities. On the other hand, as the first transformer-based model for multi-modality traffic prediction, the TinT extends the success of transformer families to a brand new domain, with critical industry and economic values. We hope our framework could take the congestion prediction beyond better reliability and could eventually help the optimization of traffic route planning and reduce the economic cost.

Part III

Demonstrate Pilot Cooperative and Equitable Traffic Infrastructure Systems

7

Chapter 7. Cooperative Traffic Signal Assistance System for Vulnerable Road Users (VRU)

This chapter is modified from the following published work:

- H F. Yang, Y. Ling, C. Kopca, S. Ricord and Y. Wang, "Cooperative traffic signal assistance system for non-motorized users and disabilities empowered by computer vision and edge artificial intelligence." Transportation research part C: emerging technologies 145 (2022): 103896. https://doi.org/10.1016/j.trc.2022.103896

With the development of Information and Communications Technologies (ICT), smart infrastructure is generally considered one of the most promising approaches to improve traffic safety and efficiency. Much of the related work falls under the broader context of Connected Vehicle (CV) applications [327, 328]. In general, a CV environment allows vehicles, road users (e.g., pedestrians, bicyclists), and other hardware infrastructure (e.g., RoadSide Equipment (RSE) and RoadSide Units (RSU)), to communicate and interact in a local network. Such communication is often enabled via Dedicated Short-Range Communications (DSRC) protocols [329, 330].

Over 50% of serious roadway crashes occur at or near intersections [331]. The disproportionate risk in these locations substantiate a higher need for connected and smart infrastructure systems. Most state-of-the-art achievements related to intersection signal control, optimization and safety enhancement, are focused on the vehicular side (i.e., traffic signal phase and timing optimization [332], vehicle to vehicle collision warning [333, 334], lane-changing driving aids [335], and driver's decision assistance at the onset of yellow traffic lights [336], dynamic speed advisory [337, 338]), which leaves a big gap for non-motorized users in a CV environment [339, 340, 341]. The application of these new and emerging technologies to assist non-motorized users have been neglected by both the research community and industry. Non-motorized users do not have the same mechanisms for traffic information dissemination as vehicles through the current information sensing and sharing channels. Active users, without necessary CV hardware or software component, are less detected, less informed, and at greater risk at signalized intersections. This is particularly true for people with disabilities [342]. The

main hurdles can be summarized as follows:

- **Perception deficiency.** Currently, adaptive signal control systems are deployed in many major cities all over the world, where the signal phase and timing are adjusted based on the real-time traffic demand. However, most passive sensors (loop detectors, surveillance cameras) are installed to count the number of motorized vehicles [343, 13]. The sole infrastructure for pedestrians and cyclists is the crosswalk button, which only generates a binary output...is a pedestrian present? Using this method, obtaining the real-time quantity and type of non-motorized users, near the interaction, is impossible. This, again, most importantly includes 'vulnerable users'.

- **Acquisition obsolescence.** The crosswalk button is the only mechanism linking a non-motorized user with the signal control system. However, due to lack of regular maintenance, the reliability of these instruments is far from absolute [344]. Furthermore, some crosswalk buttons serve as placebo buttons in places like New York City [345]. While CV technologies continue to advance and become more ubiquitous in our traffic networks, non-motorized users are being left behind.

- **Dissemination inconsistency.** CV-related protocols and standards focus almost universally on communication with vehicles. There is no research or industry consensus on which channel should be used for sharing information with non-motorized users [342]. Since the previous CV systems mainly adopted the DSRC-based communication protocol on 5.9GHz band, it cannot directly match with the users' Personal Information Device (PID) (e.g., cellphones and wearable devices). If non-motorized travelers want to leverage the traffic information from connected infrastructures, special technical equipment

needs to be carried to transfer the DSRC information with personal device adaptable information protocols and formatting [346]. The added size and weight for this special technical equiments makes it an unrealistic solution.

- **Ability discrimination.** Due to the limitations discussed thus far, signal phasing for non-motorized users is often carried out by a predetermined or rigid pattern. Unique scenarios and circumstances are not considered, especially with regard to mobility challenged users [346]. For example, at present, vulnerable populations do not have an avenue to request necessary signal time extension based on their own special needs. Commonly, less-abled individuals are left to rely on specific signals (directional sounds, status prompts) or the assistance of other travelers to make the necessary judgment in deciding the right time to cross the intersection. When intersections are well-equipped with low-barrier pedestrian signals, many of these concerns are mitigated. However, in under-equipped intersections, noises and the influence of surrounding vehicles and traffic can cause significant safety concern. In unfamiliar areas with no effective prompt, individuals with lower mobility always have higher risk of harm and longer waiting times [347]. Unfortunately, for the Vulnerable Road Users (VRU), traffic safety is not in their hands but usually depends on the alertness and adaptability of the drivers of oncoming vehicles. This causes significant equity concern as these vulnerable populations bear a disproportionate burden for safety and mobility through intersections.

To overcome the aforementioned challenges, the work described in this paper firstly proposes a vision-based non-motorized user perception component that fully integrates with current signal control systems. With the rapid advancement of computer vision and edge computing technologies, customized solutions are quickly being made possible. Recent achievements in object

recognition [125, 348], detection and re-identification [191, 111], and human pose estimation [349, 350] can be used to build the necessary data collection procedure for non-motorized users at intersections. One-stage detectors [193, 191] can provide encouraging detection accuracy, and can also be processed in a real-time manner. Further, 2-D human pose estimation solutions have enabled promising research on pose direction estimation of non-motorized users [349]. Finally, edge computing presents the opportunity for legitimate real deployment in a cost-effective way [351]. Moving a significant amount of the data processing to the edge instead of streaming the original video to a central server, huge communication cost, bandwidth and the data storage capacity can be saved. In the proposed method, the perception result should include all safety and equity-related non-motorized users' characteristics, including user type, location, pose direction, mobility status, and even disability class. With this information, a realistic automatic directional crossing request generation can be achieved for various types of non-motorized users.

To over come information interaction inconsistency, non-motorized individuals in a connected environment need to be involved in the communication process through a convenient and cost-effective means. The architecture of connected environments needs to be consistent across vehicles and infrastructure, but there has been a missing component. Personal Information Devices (PIDs) should also be considered as part of this architecture [346, 352, 353]. The development of wireless communications and the near ubiquity of smartphones in today's culture make mobile phones an ideal bridge between infrastructures and active non-motorized users [354]. There is consensus in the research community that the convenience and flexibility of cell phones to assist pedestrians and other non-motorized users, especially those with disabilities, is the ideal technological choice to enable interaction with the connected environment.

[353, 346]. Further, to ensure safety for all road users at a specific intersection, the information and interaction messages must be broadcast to the PIDs carried by non-motorized users using an adaptable protocol, and not just to the On-Board Unit (OBU) installed on motorized vehicles. The dissemination improvements in Signal Phase and Timing (SPaT) information can significantly improve the traffic safety [355], equity and reduce the negative environmental impact of fossil-fuel-based vehicular transportation [356, 357].

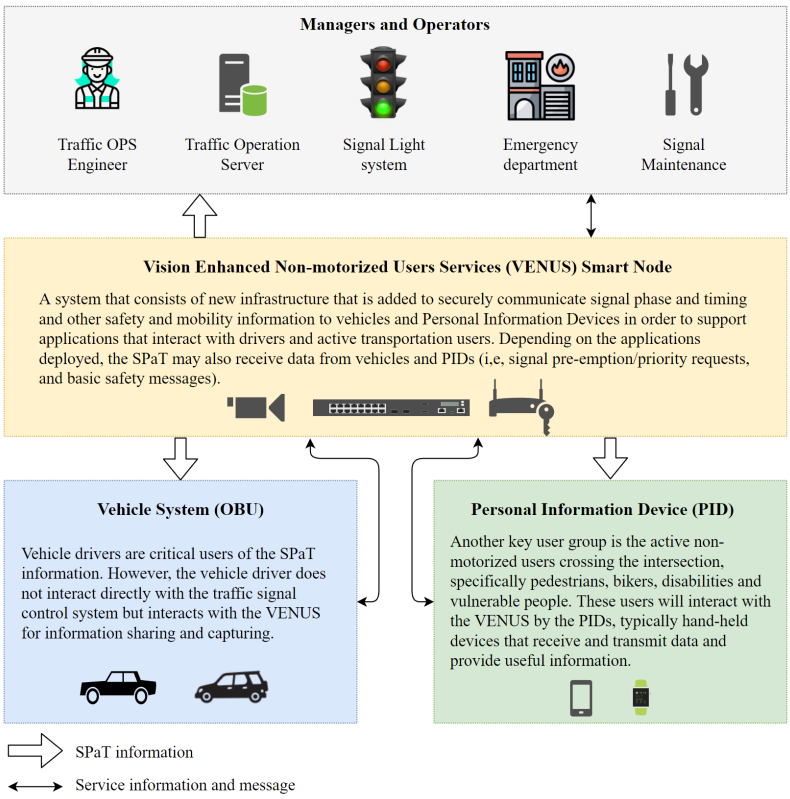


Figure 7.1: The overall system description of new connected and smart SPaT system architecture. Compared with previous architecture, the key change is the integration of the proposed VENUS Smart node and active users' PIDs.

In summary, the challenges facing non-motorized users' ability to interact with a connected environments are overcome and a new cooperative traffic signal assistance system is proposed,

called **V**ision **E**nhanced **N**on-motorized **U**sers **S**ervices **(VENUS)** smart node (as shown in Figure 7.1). The VENUS smart node is designed and deployed with powerful information perception pipelines. All signal-related non-motorized user information can be estimated in real-time. Information dissemination channels are well-established between signal controllers, vehicles and non-motorized users' PIDs. The VENUS smart node can deal with multiple communication protocols, including DSRC, and hand-held device friendly communication protocols such as 4G-LTE, 5G and Wi-Fi.

7.2 VENUS Smart Node System

7.2.1 Technical Objectives

The smart VENUS node is design to achieve six functionalities. These six functionalities can be divided into two categories: (1) Perception on non-motorized users, including those with disabilities; and (2) communication and interaction among users and other connected devices in the CV environment.

Sensing and perception of non-motorized users, including those with disabilities

1) Detect the non-motorized user type (i,e, pedestrian, biker) and, when applicable, disability types (i,e, person in wheelchair, pedestrian pushing a person in a wheelchair, person using crutches and person using a walking frame) automatically in real time;

2) Extract the potential directional cross/walk request for each user automatically in real time;

3) Extract the activities status (i,e, waiting, walking, sitting) of non-motorized users automatically in real time;

Communication and interaction among users, VENUS smart node, controller and TMC

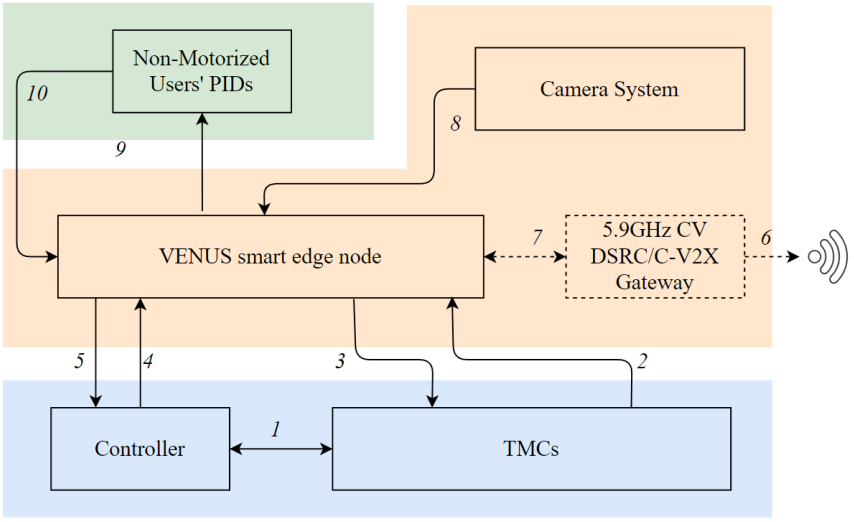
4) Capture the real-time SPaT and directional cross/walk request status information from the controllers, and synchronize the direction-aware cross/walk request generated by users to controllers;

5) Share and visualize the SPaT and directional cross/walk request status information to PIDs, and allow users to send cross/walk trigger through PIDs in case of missing detection;

6) Synchronize the real-time device operation and services status to TMCs, and allow TMC to send instructions back to PIDs;

7.2.2 System Design and Information Flow

As shown in Figure 7.2, the whole VENUS smart node consists of two different components, the VENUS smart node, and non-motorized users VENUS app. The signal controller and TMC serve as the control center and the SPaT information source. VENUS edge nodes capture real-time information and interact with the traffic operation component by using wired or wireless communication protocols based on [358]. In the VENUS smart node, three parts are integrated, including: (1) a camera, which serves to capture real-time intersection video information and send to the edge computing node; (2) the smart edge node, which is used for video processing



1 TMCs interaction with controllers, by Ethernet (NTCIP).

2 TMCs send signals to VEUNS edge node, by Ethernet (NTCIP).

3 VENUS edge node sends messgaes to TMCs, by Ethernet (NTCIP).

4 VENUS edge node requests information from the controller, by LAN (Wi-Fi or cable).

5 VENUS edge node sends requests to the controller, by LAN (Wi-Fi or cable).

6 VENUS edge node broadcasts information to CVs, by DSRC/C-V2X.

7 VENUS edge node shares information with the CV gateway, by cable.

8 VENUS camera transmits video to the edge computing node, by cable.

9 VENUS edge node broadcasts messages to non-motorized users' apps, by LAN or 4G LTE.

10 Non-motorized users send messages to VEUNS edge node by LAN or 4G LTE.

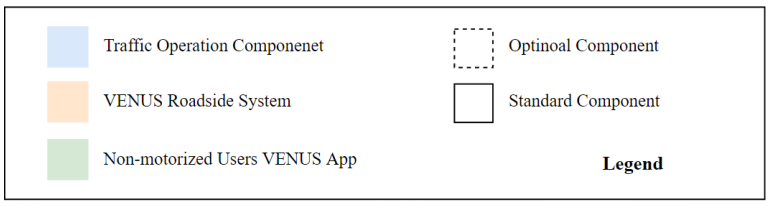


Figure 7.2: The architecture and information flow illustration of the proposed VENUS smart node.

of non-motorized users type classification, location, pose direction, and mobility status estimation; and (3) the user app, which enables the user to interact with the VENUS system, check real-time SPaT information and obtained the future cross direction waiting time. The VENUS

app can also visualize the real-time cross push-button's status. If the smart sensing nodes fail to capture the pedestrians waiting status, users can manually push the cross request through app user interface and obtain feedback. The detailed information flow and protocols amalgamated into VENUS are showed in Figure 7.2.

7.2.3 Sensing and Perception based on Computer Vision

The hierarchical sensing system developed in the VENUS edge node is paramount to this research. Figure 7.3. shows the workflow for the entire non-motorized-user-sensing-and-perception algorithm. As shown in figure 7.3, once the VENUS is installed near the target intersection area, two data sources would be programmed into the input data flow: the intersection physical alignment and the real-time streaming video. Then, the Region of Interest (RoI), including the waiting zone and cross zone, can be selected based on the camera view and the physical constraints of each intersection. The video inputs are processed by the user detection algorithm and filtered by the ROI constraints. After obtaining the location as well as the classification result of each non-motorized user using multi-object detection, the users located inside the ROI are processed by the human posed estimation and a 25 key points set of each objects is generated. Information is summarized about each object and used for two different purposes. First, in order to obtain the human pose direction and automatically trigger a push for users, the 2D image must be converted into a Bird's Eye View (BEV). To obtain the mobility status, another classification algorithm is implemented based on the user key points estimation result. In this step, four classifications of mobility status are defined: wait, walk, ride and sit. Finally, the streaming video inputs are analyzed to extract each user's information including users type, location, pose direction and mobility status. The mentioned four outputs are then leveraged to automatically

generating SPaT services for non-motorized users, especially those with disabilities. In the following two subsections, all the details regarding sensing and perception algorithms are explored.

User Detection

The first step necessary for sensing is detection and classification of the non-motorized users from the streaming frames. To achieve faster processing on the edge devices, the fourth version of the You Only Look Once (YOLOv4) [191] is chosen to detect the non-motorized users. YOLOv4 [153] is a single-stage detector with a CSPDarknet53 backbone, SPP neck [192] and the YOLO version three [193] feature head. This combination provides the perfect combination of hardware to balance the accuracy and processing speed. In the experiment, the detector is trained on two datasets: the mobility-aid dataset [359], and the manually labeled dataset. The output of the YOLOv4 detector is the location of bounding boxes (L_{bb}), class (C) and the confident score (R_c). Only the detected object whose R_c exceeds the predefined bottom range of confidence can be considered as a positive detection and then sent into the key-point estimation workflow.

User Representation Frame Selection

After the real-time detection, the next step is to filter the localization and classification result for estimating the users' pose direction and mobility status. Here, the selection rules are based on the following two factors: (1) the object classification confidence score (R_c), and (2) the object size ($\mathscr{S}$). A size normalization is integrated into the selection process by regarding the predefined frame of the object as a unit. Then, a confidence R_{ci} is used as the bottom value to filter low-confidence objects. The score of object is defined in a frame k as ρ^k. The score ρ^k is computed

following Equation 7.1:

$$\rho^k = \{R_c^k * \mathscr{S}^k | R_c^k > R_{ci}\} \tag{7.1}$$

Next, the candidate with the largest value of ρ^k is selected as the representative object frame to compose an object clip $\mathscr{C}$. $\mathscr{C}$ is then used as the input for the next several steps.

Pose Direction and Mobility Status Estimation

The second key component for vision sensing and perception is to estimate the user's pose (face) direction and concurrently their mobility status. For the pose direction estimation, two steps are necessary: (1) a 25 key-points of human pose estimation (as shows in Figure 7.4 and Figure 7.5.), and (2) a pose direction calculation by 2D to 3D projection (as shows in Figure 7.5). In the first step, the OpenPose 25 key points estimation proposed by the Carnegie Mellon University team [349] is used to obtain the individual level of key points information. The OpenPose system can achieve high accuracy and real-time performance, regardless of the number of users in the input. By designing the system with a non-parametric representation approach, OpenPose refers to Part Affinity Fields (PAFs) instead of body components directly. Then, it learns to associate different body components with individuals in the input images [349]. Combined body and foot key points detectors are integrated based on an annotated foot dataset that supports toe, foot and heel detection. Based on the PAF-only refinement process, the OpenPose shows promising results, both reducing the inference time maintaining the accuracy of human body components individually. The output of the OpenPose are the 25 key pose locations (L_{k_i}) and confidence scores (R_{k_i}).

After obtaining the key points, the next step is to obtain the pose direction of the human

Figure 7.3: The hierarchical vision-based perception component illustrations with examples (a) and key procedures (b). After the video clips are sent into the VENUS smart node, the first step is to finish the user detection. Then, the object with reliable detection results will be sent to finish the key points estimation for further pose direction and mobility status estimation. To obtain the human pose direction, a camera view projection from the default camera view to BEV is calculated by pre-defined reference points. In this chapter, all the mentioned procedures are illustrated and discussed in detail.

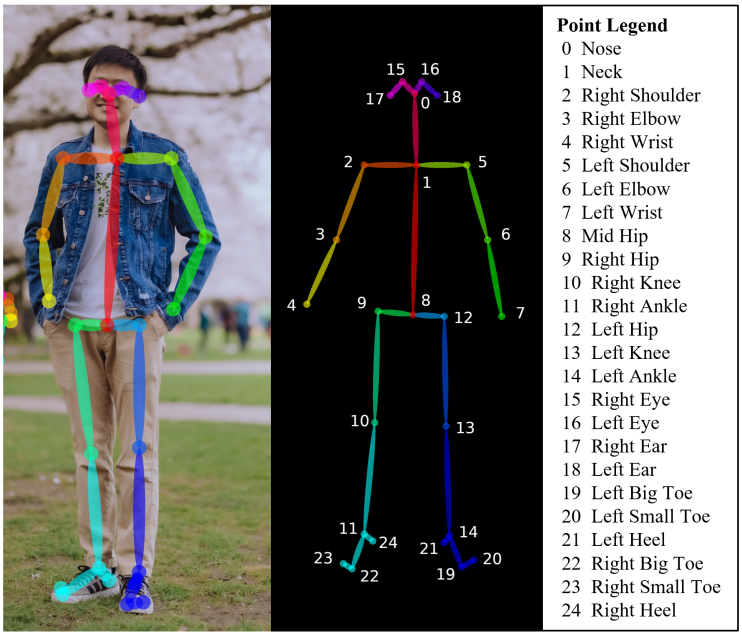


Figure 7.4: the illustration of 25 key points human post estimation and the reference body points.

body to estimate the cross direction. The process is showed in Figure 7.5. Firstly, the camera view projection algorithm is used to map the 2D information captured from the original video into the BEV plain $(\hat{p}_i, \hat{q}_i)$ with spatial-temporal information. To finish the transition, $\mathcal{N}$ twinning points (e.g., pedestrian crossing lines of the intersection on the BEV ground plane that are orthogonal with each other) are manually labelled, and then the real GPS information from Google map can be obtained. Based on the label points, two vanishing points can be derived on the ground plane. It has been proven that all camera parameters in a projection matrix $\mathcal{M}$ can be computed from, and with some constraints on, intrinsic camera parameters [360]. These constraints can be relaxed for more accurate estimation of view projection parameters. To do so, something more than bottom limitation of twinning points must be selected and then an optimization problem to minimize the re-projection error must be formulated as shown in equation

7.2:

$$f(p,q) = min \sum_{i=1}^{\mathcal{N}} |D(p_i,q_i) - D(\hat{p}_i,\hat{q}_i)| \tag{7.2}$$

In Equation 7.2, $D(p_i,q_i)$ are the node ground truth distance of select paring points and $D(\hat{p}_i,\hat{q}_i)$ denote the estimated segments distances that are back-projected to the BEV ground plane. By minimizing the Euclidean distances in both planes, the absolute differences between estimations and ground truths can be decreased.

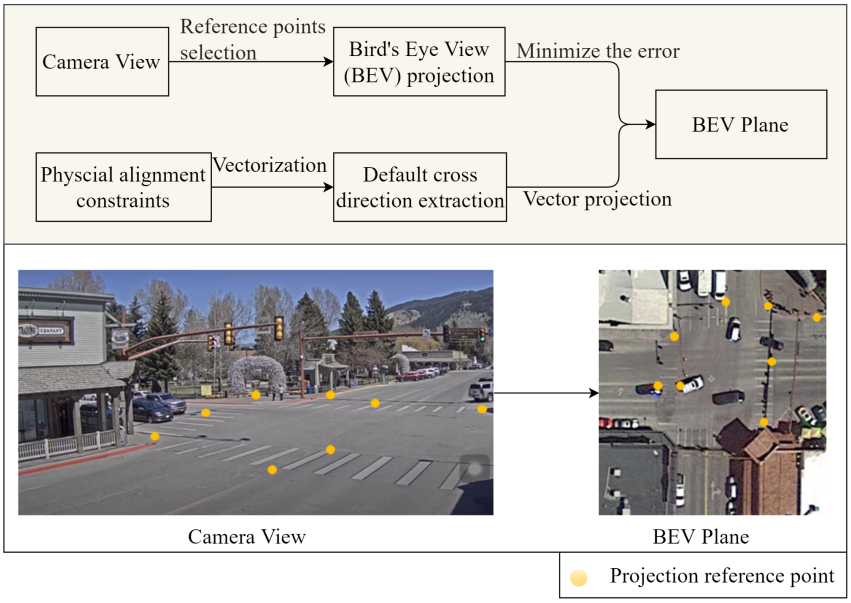


Figure 7.5: The illustration of camera view to BEV projection procedures.

After obtaining the projection matrix $\mathscr{M}$, the next step is to get the predefined default cross direction ($\mathbf{V_{dd}}$) in the BEV plane. As shown in Figure 7.5, a standard intersection includes eight cross direction (two at each corner). Then, the user body representation vector ($\mathbf{V_{ub_i}}$) for the detected user i is calculated by estimating a left-to-right body vector which represents

the pose direction. Here, the key points on the right foot of the person represented in Figure 7.4 (including right heel (24), right small toe (23), right big toe (22) and right ankle (11)), and points on the left foot (including left heel (21), left small toe (20), left big toe (19) and left ankle (14)) are used to generate a vector that is parallel to the BEV plane. In order to ensure the robustness of the system and avoid significant occlusion, the key points with the highest confidence are filtered to calculate the $\mathbf{V_{ub_i}}$. Finally, the pose direction is estimated using equation 7.3:

$$\beta_{ub_i} = min(\arccos(\mathbf{V_{dd_x}}, \mathbf{V_{ub_i}}), \arccos(\mathbf{V_{dd_y}}, \mathbf{V_{ub_i}})) \tag{7.3}$$

Where $\mathbf{V_{dd_x}}$ and $\mathbf{V_{dd_y}}$ are two potential default cross direction at each corner of an intersection.

For the mobility status estimation, a fully connected, two-layer neural network is implemented to post-processed the 25 key points estimation results. The neural network input is the OpenPose estimation result (dimension of 75×1) including the location and confidence score of each key point. The output is a 4×1 dimension which represents the confidence rate of four different mobility status, including waiting, walking, sitting and biking.

7.2.4 Communication

A vital aspect of VENUS's communication design is that, while its current iteration's communication protocol leverages Wifi, cellular network and DSRC, it can be expanded to other communication protocols, such as LORA and 5G. This allows VENUS to keep up with the constantly evolving communication landscape, maintain utility regardless of what technologies emerge. The communication system embedded in VENUS consists of two sub-components: (1) interaction with controllers, and (2) interaction with user PIDs.

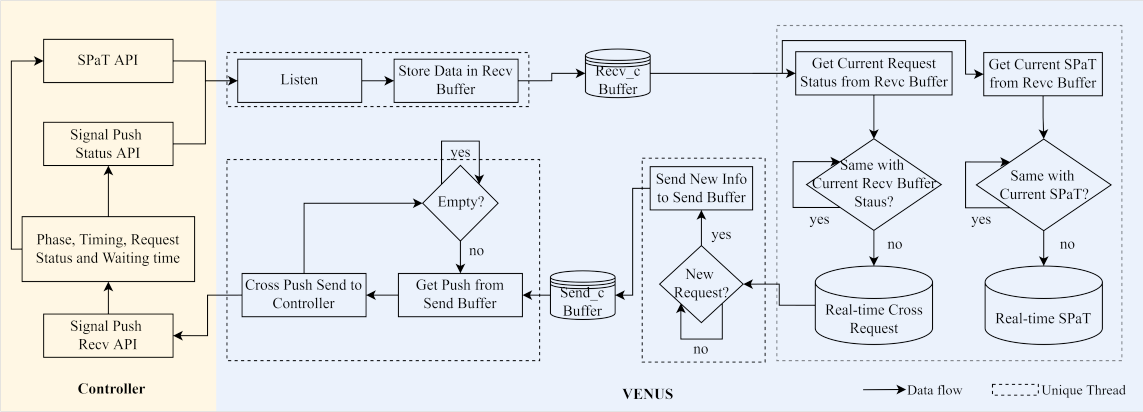


Figure 7.6: Communications between VENUS nodes and the signal controller

Figure 7.6 shows communication synchronization between VENUS nodes and controllers through NTCIP APIs. Communications between the VENUS node and controllers are handled asynchronously to save computing power. The VENUS node has four separate threads to handle sending and receiving SPaT information and storing it in receive buffers for controllers (Recv_c). These buffers hold the data until the main control threads are ready to work on it. For the communication between controllers and VENUS, a listen and store thread is used to capture the information from the controllers through the SPaT and Signal Push Status APIs. Then, the received data is processed by the yes/no conditions to renew the real-time, phase-aware SPaT and crossing request contents. Then, if a new cross request is captured by the VENUS sensing component or received by the user PIDs, the general receive buffer (Recv) can introduce the new request by adding the new information to the send-to-controller buffer and then transmit the cross push and matching phase to the controller push receive API.

VENUS Node with PIDs

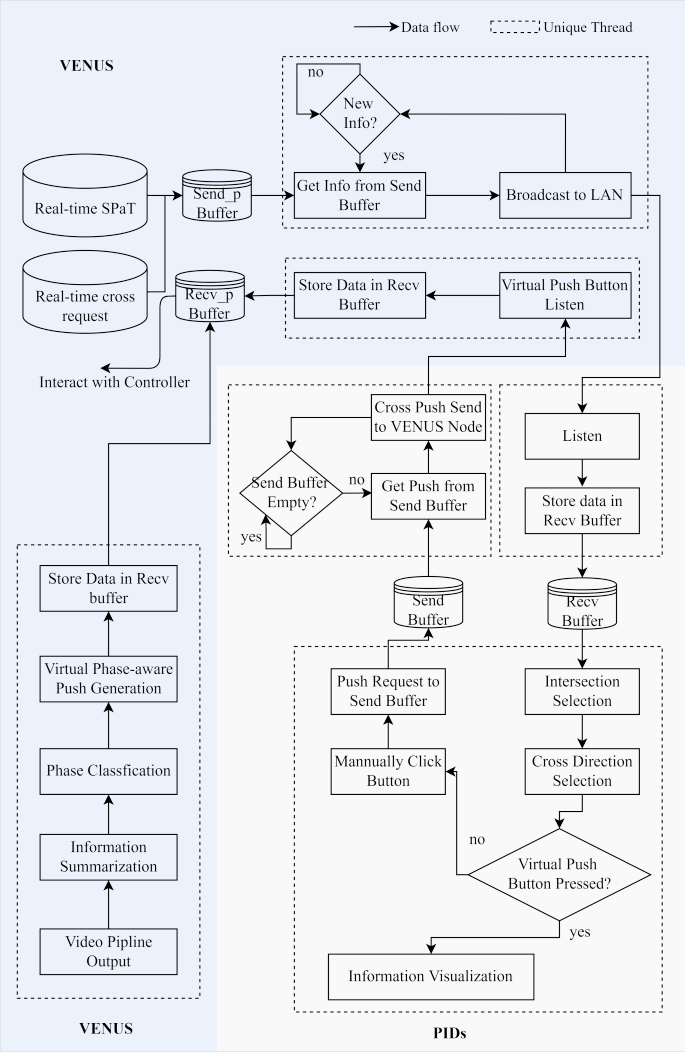


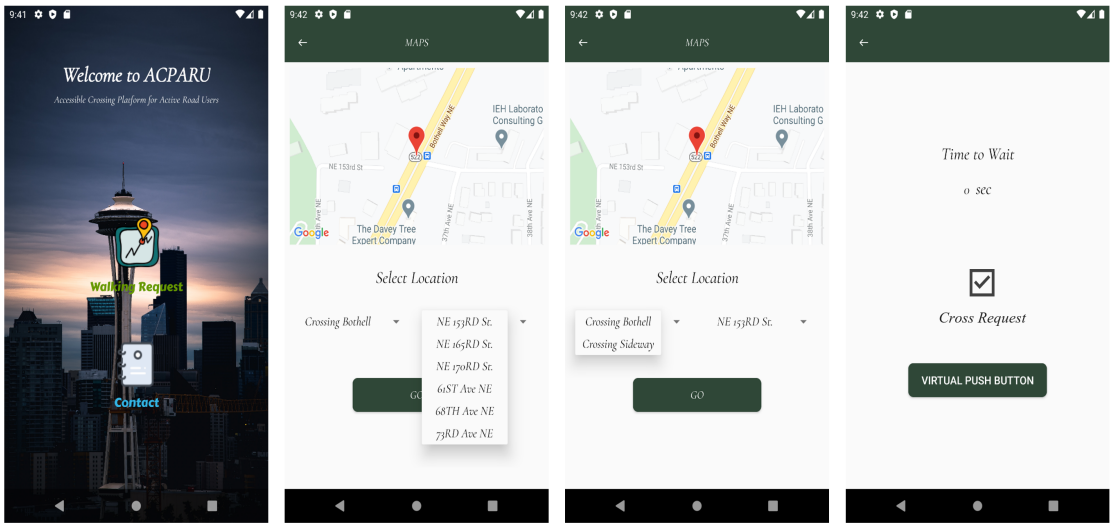
Figure 7.7: Communications between the VENUS node and the user PIDs

Figure 7.7 shows communication synchronization between the VENUS node and the user PID through WiFi or a cellular network. Similar to the architecture design between VENUS and a controller, the VENUS-and-PIDs communication process is handled by six different threads.

First and foremost, when a non-motorized user walks into the waiting zone, ideally, the VENUS sensing component will automatically generate a phase-aware virtual push and then transmit to the controller in real time. Then, the updated SPaT and crossing request data can be received by the VENUS node and stored in a real-time SPaT and crossing request file. Simultaneously, the VENUS node will obtain the new information from related files, transmit the data, and then broadcast to the pre-paired PID. Next, the integrated receiving thread can capture and store the data to the receive buffer on PIDs. The VENUS APP can assist the user in identifying the correct intersection, select the correct cross direction and and ultimately visualize the target phase wait time/push button status in real time. When necessary, the manually clicked data will be sent to the VENUS and captured by the virtual push button listener thread and synchronize to controllers through the communication subsystem between VENUS and controllers.

7.2.5 App

VENUS roadside edge nodes interact with users through an app, called Accessible Crossing Platform, for Active Road Users (ACPARU), which can be installed on cell phones (it is assumed that cell phones are the most common PIDs). Figure 7.8 shows the functionalities and User Interface (UI) of the ACPARU app. After the user starts the APP, the real-time data from VENUS roadside nodes can be automatically synchronized. Then, users need to select the target crossing interaction with the help of map-based information and internal GPS localization function support. Then, the crossing direction can be easily selected by orienting the phone head's direction or choosing the target crossing street name in the drop-down box. Both actions can directly trigger the necessity direction selection and visualize on the app UI. Next, the user-oriented direction information can be visualized on the app, including the time to wait as well as the current



(a) (b) (c) (d)

Figure 7.8: The VENUS user application on cell phone – the user interfaces illustration of proposed Accessible Crossing Platform for Active Road Users (ACPARU).

status of the cross button (triggered/not triggered). When the VENUS roadside sensing system does not successfully capture the crossing request, a virtual push button can be used and manually send a push through the ACPARU app. A pop-up window will show the feedback request "sent" status provided by VENUS edge nodes.

7.3 Experiment and Discussion

7.3.1 System Hardware Component

VENUS System mainly includes three key components: (1) traffic operation component (controllers and TMC server), (2) VENUS roadside system (VENUS smart edge node and on-board camera), and (3) non-motorized users app (visualized in the Figure 7.9). Here, the 2070 series (2070LDX) controller made by Q-free Inc. is used for testing. The key reason for using

2070LDX is that it meets and exceeds the current ATC, CalTrans, and NTCIP standards providing agencies with a robust, industry-leading, true open architecture hardware platform. VENUS can also be easily implemented with other types and brands of controllers based on the previous well-established architectures. Nvidia Jetson AGX Xavier uses an embedded edge platform for real-time sensing over streaming pixels for the smart edge nodes. The Jetson AGX Xavier has 8-core ARM CPU and 512-core Volta GPU. To fully stimulate the computing potential on edge nodes, the entire workflow running on the edge side of VENUS is combined with the reasonable Xavier resources without overlap in one cycle. For the communication component, the Collie (GL-X300B) made by GL.iNet. is used and integrated with the Jeston Xavier AGX and facilitates data transmission. Collie gateway is based on OpenWrt open-source embedded operating systems, which is compatible with 2.4GHz Wi-Fi and has full-band 4G communication ability. The system is agnostic of the brand and type of matching onboard camera. The app optimally runs on any Android-based phone (Android 8 or newer).

7.3.2 Sensing System Evaluation

Mobility Aids Dataset

The Mobility Aids dataset [359] is collected from a hospital in Frankfurt and the facilities of the Faculty of Engineering of the University of Freiburg in Germany, including over 17,000 annotated RGB-D images, containing five different categories according to the mobility aids people using: pedestrians, people in wheelchairs, people in wheelchairs with people pushing them, people with crutches and people using walking frames. This dataset was mainly used for the VENUS sensing detector training and evaluation for the user localization and classification.

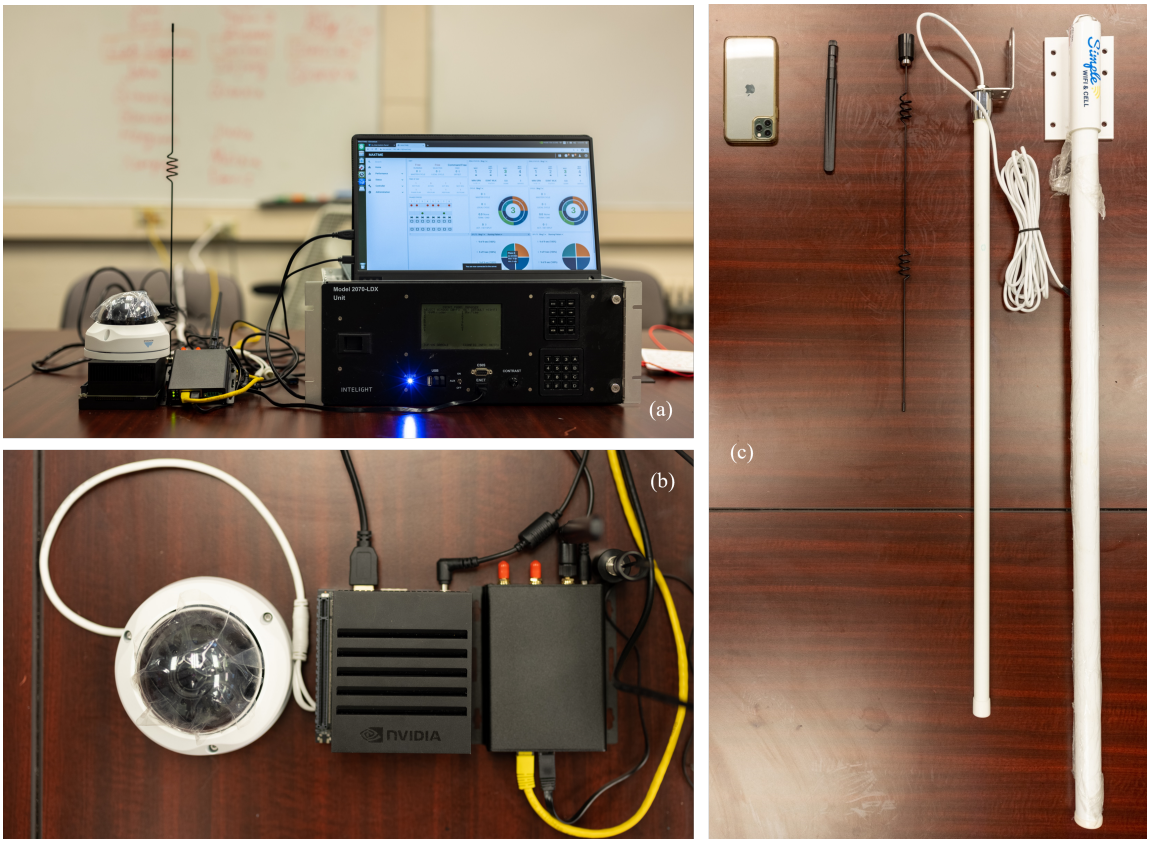


Figure 7.9: The hardware illustration of controllers, VENUS smart node, and attached external antennas. (a) is the overview of VENUS smart node system and controllers. The VENUS generated signals can be directly visualized by the controller and visualized on the screen. (b) is the VENUS smart node three key component (streaming camera, Nvidia Jetson AGX Xavier and the connected gateway, from left to right). (c) illustrated the various external antennas to cover the range from 50m-150m for information interaction.

VENUS Dataset

To thoroughly test the VENUS smart node, an additional dataset was collected. Video was collected from six intersections located in three different US states with varying designs, camera view and default crossing directions. This was summarized into a dataset called VENUS dataset. The total video sequences length was 19.95h, and each camera view included at least two default crossing directions. The default camera view, waiting and crossing zones of each video can be

Figure 7.10: The detailed illustration of camera view and pre-defined ROI zones of the collected VENUS dataset for testing the system.

found in Figure 7.10. From the 35,879 images, 4,560 images were selected and labelled, including six different non-motorized user classes: pedestrian, person in wheelchair, pedestrian pushing a person in a wheelchair, person using crutches, person using a walking frame and biker. For the sensing detection, 908 of the 4,560 images are used to test the detector and the rest are used

for model training and validation. For the pose direction estimation, the original video sequence is used as input and processed on a VENUS edge node equipped with the well trained detector. The selected testing set of 908 images included 1076 objects from six cameras was also used to evaluated the pose direction and mobility status estimation accuracy. Detailed object numbers and evaluation result can be found in the following section.

Edge Computing Optimization

Nvidia Jetson AGX Xavier works as the embedded edge platform for real-time detection and user pose direction and mobility status estimation. The CPUs are divided into seven groups for handling the operating system, input preparation, data processing for algorithms, output preparation and output control. The last two are use for communication receiving orders from TMC and controller and to transmit data to the servers or cell phones. The GPU is in charge of handling the detection script. The CPU and GPU share the information through the cache for each cycle. To speed up the whole edge framework, the detector and feature extractor use the TensorRT deep learning framework and perform asynchronous communication inference. In addition, all code was written in Python language and optimized using Numba.

Parameters Settings

The algorithms' parameters using on the VENUS smart edge nodes are as follows:

- In the user detection part, the minimum object confidence of R_c was set as 50%.
- In BEV view projection, I used a $\mathcal{N}$ value of 8.
- The two sensing threads (user detection, pose direction mobility status estimation) are

asynchronous. The default processing speed for the detection is in real time, and the pose direction mobility status estimation was set to 2fps.

- The pre-defined ideal size frame of the object was 300 pixels.
- Minimum object size ($\mathscr{S}$) was no less than 150 pixels.
- The activation function used in the mobility status estimation neural network was ReLU.

User Detection Evaluation

The VENUS edge node detection algorithm was trained and evaluated on the two aformentioned datasets: Mobiliy Aid dataset and VENUS dataset. For real-time multi-object detection, six classes were predefined as: pedestrian, person in wheelchair (PW), pedestrian pushing a person in a wheelchair (PPW), person using crutches (PC), person using a walking frame (PWF) and biker. For the mobility-aid dataset, only the top five categories were included and the average precision of detection were 82.29%, 84.64%, 74.56%, 73.11%, and 78.59% respectively. For the VENUS dataset, the average precision of the six categories were 88.31%, 85.12%, 74.77%, 72.14%, 72.32% and 89.46% respectively. The Intersection over Union (IoU) threshold was set to 50%, then the Area-Under-Curve for each unique recall mean average precision (mAP@0.50) on the mobility-aid dataset and VENUS dataset were 79.84% and 88.14%. The professing speed on the VENUS achieved an average of 21.09 fps when the input frame was 1080*720.

User Pose Direction and Status Estimation Evaluation

In order to test the potential of VENUS sensing algorithms, 908 separate frames were used, containing 1076 objects to test the user pose direction and mobility statues estimation. For

Figure 7.11: The pose direction estimation result visualization. All the input objects are calculated the β with four default cross directions under the related BEV plane. The visualization group is clustered to the belonging direction.

each testing image, at least one non-motorized user was included and used for estimatation and evaluation. When the user was present in the area of the intersection, the VENUS was used to calculate their β_x, and then referred to the potential crossing/waiting direction in the BEV plane. By comparing the obtained β_x with different default crossing directions, one of the default crossing directions (V_{dd}) closest to the vertical of the user body representation vector (V_{ub_i}) was chosen. This direction was assumed treated as the potential user crossing direction. Then, the calculated result were compared with ground truth data. For each camera view, all users' poses were ultimately classified and summarized into number of users on the four default crossing directions and visualized in Figure 7.11. The detailed users' pose direction estimation results are in Table 7.1.

In Table 7.1, the users' pose direction estimation results of camera views are summarized into different rows. The pose estimation results of different categories of users are distinguished. In general, among all types of users, the average accuracy of the pose direction estimation is 90.24%. The highest accuracy category is normal pedestrians, and the accuracy of potential crossing direction is 91.49%. Among all cameras, camera2 achieved best accuracy result for 95.59%. The key reason is that camera2 had the ideal installation location, height and low occlusion of users. For those users with disabilities, the person using crutches shows the lowest estimation precision due to limited data occurrences. Also, lifting one foot leads to a large error for the view transition, especially for the key points close to the feet. In summary, VENUS can successfully infer the direction of non-motorized users' crossing direction in a real time with high confidence, which is quite useful for pedestrians and cyclist sensing.

Based on the 25 key-points estimation result, a two-layer fully connected layer was designed for the mobility status estimation by a supervised learning procedure. The evaluation results are

Table 7.1: Summary of the users' pose direction estimation result. In each cell, the black word represent the total numbers in the evaluation dataset. The green numbers means the true positive inference and the red number represents the false negative inference.

CamID	Frames	Objects	Normal Users		Disabilities			
			Pedestrian	Biker	PW	PPW	PC	PWF
Camera1	76	92	82/76/6	5/4/1	3/2/1	–	2/2/0	–
Camera2	52	68	55/53/2	8/7/1	2/2/0	1/1/0	–	2/2/0
Camera3	312	365	336/311/25	14/12/2	5/4/1	2/1/1	7/5/2	1/1/0
Camera4	244	293	275/255/20	6/4/2	4/2/2	2/2/0	4/3/1	2/1/1
Camera5	129	150	133/115/18	11/10/1	2/2/0	1/1/0	3/1/2	–
Camera6	95	108	94/82/12	8/6/2	2/2/0	1/0/1	1/1/0	2/1/1
Total	908	1076	975/903/72	52/43/9	18/14/4	7/5/2	17/12/5	7/5/2
Overall Accuracy	–	**90.24%**	91.49%	82.69%	77.78%	71.43%	70.69%	71.43%

Table 7.2: Summary of the users' mobility status estimation result. In each cell, the black word represent the total numbers in the evaluation dataset. The green numbers means the true positive inference and the red number represents the false negative inference.

CamID	Frames	Objects	Mobility Status			
			Wait	Ride	Walk	Sit
Camera1	76	92	44/40/4	5/5/0	40/36/4	3/2/1
Camera2	52	68	26/25/1	8/8/0	32/28/4	2/2/0
Camera3	312	365	202/184/18	14/13/1	144/131/13	5/5/0
Camera4	244	293	151/129/22	6/6/0	132/122/10	4/3/1
Camera5	129	150	95/88/7	11/9/2	42/37/5	2/2/0
Camera6	95	108	65/55/10	8/8/0	33/27/6	2/2/0
Total	908	1076	583/521/62	52/49/3	423/381/42	18/16/2
Overall Accuracy	–	**89.87%**	89.37%	94.23%	90.07%	88.89%

summarized in Table 7.2. Among the four different mobility statuses, the riding status achieved the highest estimation accuracy to 94.23%. The walking and waiting status also achieved relatively high estimation accuracy at 90.07% and 89.37%, respectively. The sitting status was the most challenging status to estimate due to limited data occurrences and the inexact key points estimation result (especially when the user was backing towards the camera). Among all camera views, camera2 still led the accuracy by obtaining 92.65% precision. The camera6 had the lowest accuracy, which may have been due to the lower camera viewing angle and severe occlusion. Examples of Mobility status estimation results can be found in Figure 7.12.

Figure 7.12: The result visualization of user mobility status estimation for VENUS smart node.

7.3.3 Communication System Evaluation

The integrated gateway in VENUS can support the communications of VENUS to controllers and TMCs through both wired or wireless networks. In the VENUS smart edge node prototype, both the wired and wireless connections between the controller and VENUS node were tested by running the messages sending and receiving script every second. The test lasted for one week and all packages (604,800) were received by the target equipment; no package was lost. Further, the potential for complex, real world deployment environments were considered and two types of communications among VENUS node and user PIDs were implemented. In general, if the VENUS node was installed in an environment well served by reliable wireless, 4G information

sharing capability could be used as a default protocol for real-time information sharing. Meanwhile, if there were a necessity to install VENUS in the rural environment without 4G services provided by carriers, the internal self-organized network could be triggered and used for information sharing. To match the different intersection sizes, the modular design, programmable transmit power, customized antenna components were integrated with the VENUS edge node and serve for a variety of real conditions. This self-organized network is based on Wi-Fi protocol and fully supports the standard of IEEE 802.11ac. Customized communication scripts of the VENUS gateway were made to aptly support the connections between controllers, PIDs and VENUS nodes. The communication system is useful for two lanes, three lanes and even larger intersection areas with low latency and high sensitivity ability. At the same time, VENUS's smart gateway can easily adjust the software and hardware as required to achieve different sensitivity, and communication distance and price requirements. The detailed parameters and testing results can be found in Table 7.3.

Table 7.3: The communication performance of VENUS self-organized LAN based on Wi-Fi protocol with customized settings and antennas

Transmit power	Antenna	50m RSSI	80m RSSI	Max Range	Cost	Delay
17dBm (50mW)	6dB	-82dBm	–	59m	$115	<128ms
17dBm (50mW)	9dB	-77dBm	–	64m	$134	<128ms
17dBm (50mW)	12dB	-73dBm	–	75m	$215	<128ms
20dBm (100mW)	6dB	-66dBm	-80dBm	82m	$115	<128ms
20dBm (100mW)	9dB	-62dBm	-76dBm	90m	$134	<128ms
20dBm (100mW)	12dB	-59dBm	-71dBm	104m	$215	<128ms

7.3.4 System Evaluation

Innovations and Advantages

When real deployment is considered, scalability and cost become some of the most significant factors. Compared with traditional non-motorized users related SPaT and safety-related services system, and with previous central or edge & cloud-based architecture, such an edge-supported, vision-enhanced architecture has several obvious advantages. For the traditional video-based system, the implementer must pay for the hardware system, database fees, communication services, power supply, and periodical maintenance. However, for the VENUS smart edge system, the technology is truly plug-and-play. There are no requirements for the communication system and bandwidth, no severe requirements for the video data storage and processing, and no demand for the cell phone data service coverage (when only using the VENUS self-organized network). In summary, the VENUS system advantages can be summarized into following bullets:

- Due to the full edge architecture, VENUS can achieve significant financial savings on the server hardware.
- Venus is easily scalable.
- The communication requirement is modest and flexible. If communication with TMCs is necessary, the daily bandwidth can be controlled to less than 5GB per day.
- The system is more environmentally friendly due to reduced need for power consumption.

Considering the number of cameras currently installed at intersection surveillance systems for traffic monitoring, the VENUS ability to leverage any such system is advantageous.

Limitations and Future Works

Even thought the proposed VENUS smart node can significantly help non-motorized users, specifically those with disabilities, cross the intersection in a safer and more convenient way, some technical limitations and further work is still worth investigation.

- For the sensing and perception component, the appearance of disabled users only accounts for 0.5%-3% of daily users. Thus, it remains a challenge to achieve high-accuracy recognition precision in such a low-probability target group. Although the current detection true positive rate for the disabled category reaches more than 70%, there is still much room to improve the recognition precision. These are the users most vulnerable and thus in most need of the assistance provided by the VENUS smart node. Recently, there has been some promising work on unbalance data classification [52, 361], and open-set long-tail recognition [45]. Further, more advanced methodologies to improve the sensing accuracy, including the data augmentation techniques, data re-balancing and re-weighting methodologies will be investigated and tested for the disabled user data.

- In the proposed version of VENUS smart node system, sensing and perception of the non-motorized users, as well as the communication and interaction among signal controller and TMCs are two key achievements. Another important component of this subject is to propose a whole new signal phase and timing algorithm, which fully considers the demand of non-motorized users, especially those with disabilities. Currently, most of the control paradigms in CV environment are focused on improving the traffic network capacity, efficiency, and stability for vehicles [286, 362, 68], and generally ignores the impact of non-motorized users. Based on the current VENUS smart node, proposing and

evaluating an innovative cyber-physical cooperated control algorithm in consideration of equity and disability is the next step of the research.

- Finally, with VENUS's powerful sensing and information sharing ability, safety-related utility, collision warning, near-miss detection, disabilities crossing alert generation, can be further developed in the necessary scenarios to improve the traffic safety.

7.4 Chapter Summary and Future Works

In this research, a whole new road-side system, the VENUS smart node, is proposed to enhance traffic accessibility, equity and safety for non-motorized road users at signalized intersections. A customized approach to recognize non-motorized users at signalized intersections by video-based perception algorithms running on an edge device is developed. This enables the detection of active user classes, locations, crossing direction and mobility status for those attempting to cross a roadway and assists with actuating the signal request for them. Further, it can detect any disabled users crossing the roadway, alert controllers and nearby peers of their presence via an alert in a mobile application (app), and generate necessary trigger for potentially extending the crossing phase for them as needed. Based on the extensive experimentation conducted by collecting real surveillance video from six different intersections, the VENUS smart node can achieve precise non-motorized user sensing and perception, reliable communication and a touch-free crossing trigger generation in variety of intersection configurations. The VENUS smart node is slated for installation at nine intersection at the Washington State Route 522 to benefit the non-motorized users and improve intersection safety and equality.

8

Chapter 8. Enhancing-Minority Vehicle Recognition System Empowered by Vision-LiDAR Representation Fusion

This chapter is modified from the following work:

- H F. Yang, C. Liu, M. Tsai, X. Jiang and Y. Wang, "Trustworthy Cost-effective Vehicle Recognition System Empowered by Micro-Pulse LiDAR and Edge Artificial Intelligence." submitted to IEEE Transactions on Intelligent Transportation Systems.

Vehicle detection and recognition are the key tasks for almost all applications of modern intelligent transportation system (ITS), such as traffic signal control and optimization [363], traffic congestion prediction, logistics optimization [364, 269, 18], and dynamic traffic management [365]. Current sensing solutions used for counting and classifying vehicles fall into three main categories: loop detectors [366, 367, 368], video-based [369, 370] and others [371, 372].

Magnetic loop detectors, i.e., vehicle detection loops, are the most mature and widely applied. Being an energy-efficient [373], reliable, and scalable technology, loop detectors have already been widely integrated with active signal control systems in the US [374]. Despite this, due to its working mechanism, loop detectors are sensitive to metal materials, passing speed, skin effect, and nonuniform materials [374, 375]. Vehicle classification based on loop detectors thus suffers from low accuracies [376] and limited detectable vehicle types, not to mention the inconvenience of installing and maintaining the detectors underground. **Video-based solutions** are also popular for vehicle detection and classification. Object detection algorithms based on deep learning techniques have proved successful in capturing vehicle information such as appearance, color, type, and location via surveillance cameras [151, 29, 111]. However, costs for data storage, communication, and computational cost hinder actual implementations [369], and their performances are extremely limited in heavy occluded scenes (vehicle-vehicle, infrastructure-vehicle occlusions) [377, 378], changes in vehicle perception (appearance, camera placement, distortion, size, scale), and in sudden change in the environment (illumination, weather) [13, 379] and restricted lighting conditions (tunnel, night time) [13, 380]. Even though sensor fusion solutions (thermal cameras, radar) or external lights have been brought in to help, they make the system

much more expensive and complicated [381] and introduce new technical problems, e.g., limited thermal input resolution and heterogeneous signal transferring [381, 379].

The limitations of loop detectors and video sensing systems motivated the investigation of new sensing approaches, including (1) wireless sensors, i.e., Bluetooth or Wi-Fi probe counting [382, 383], (2) radio detection and ranging (Radar) [384, 385] and (3) light detection and ranging (LiDAR) [139, 141]. Among them, **LiDAR** is one of the most promising, characterized by its active sensing mechanism and high precision. In general, LiDAR uses the stimulated light/laser waves to measure the objects' shapes and distances to the sensor and then generates an accurate range of 3D information of the surroundings [135]. The active signal generation architecture makes LiDAR effective in capturing reliable and accurate information, especially in low light conditions. Despite this, current approaches leveraging LiDAR for vehicle sensing and classification still face significant challenges:

- **Trade-off between hardware cost and system performance.** Currently, to capture accurate object features, using the rotating LiDAR scanner or multiple fixed flash LiDAR simultaneously is necessary [139, 140, 142]. Even though good classification results can be obtained, the prohibitive costs and complicated post-processing workflow prevent such systems from being widely deployed.

- **Enhancing-minority classification in real-time.** Most of the previous research related to vehicle recognition with LiDAR heavily relies on the affluent output from the rotating scanning LiDAR and makes little methodological contribution [139, 140, 142, 143]. Their classification approaches are (1) out-of-date, e.g., Naïve Bayes, K-nearest neighbor, decision tree, support vector machine, (2) majority-dominated, always easily biased towards dominant classes and perform poorly on minority classes. i.e., as buses or trucks,

and (3) without customization and improvement from traffic-oriented scenarios. Most methods do not generalize well when they are tasked to distinguish fine features, i.e., sedan vs SUV, even though they perform well in common situations, e.g., truck vs sedan.

- **Raw data processing for retroreflective and transmissive surfaces captured under different traffic conditions.** In general, LiDAR generates unreliable data when object surfaces are highly transmissive or reflective, e.g., glass, light sources, and mirror [386]. A data correction procedure for such surfaces should be considered for further deployment.

The mentioned gaps motivate this study to investigate more advanced technologies. **A smaller, cheaper, and more durable LiDAR sensor should be considered.** For example, recently available modulated pulse LiDAR shows great potential [387, 132]. Instead of describing object surfaces via point clouds, the micro pulse LiDAR works like a rangefinder but at a very high scanning frequency (up to 10kHz per second). While typical LiDAR collects rich information about vehicle surface, size, height, and location by constructing 3D point clouds by rotating slowly (20Hz to 50Hz), most of the captured information is only environmental, i.e., foreground, background, and location, and not useful for vehicle classification [139, 140]. It is then natural to ask whether it is adequate to collect the vehicle's contour representations only, instead of the massive point cloud of the surroundings? **Cutting-edge deep learning classification models have demonstrated great potential in bridging this gap.** In the past few years, neural networks have achieved remarkable performances in the sequential data classification tasks, including text [388, 389], social network [390, 391], and spatio-temporal data classification [391, 392]. The combination of deep residual neural network [125], convolutional neural network (CNN) [393], fully convolutional network (FCN) [394, 391] and recurrent neural network (RNN) [395, 68] can effectively extract the inherited features and dependency

features of the sequence records. Furthermore, several promising approaches to improve the tail classes recognition precision in imbalance datasets are emerged, including the data re-sampling and augmentation [396, 45], multi-expert architecture [52], decoupled training and ensemble learning [397], which inspire us a lot to customize a minority-enhancing deep neural network.

In summary, to overcome the aforementioned challenges, this study proposes a pilot vehicle sensing framework, Compact LiDAR Empowered Vehicle Enhancing-minority Recognition (CLEVER) system, and its associated vehicle classification (VC) neural network, *CLEVER VC*. By leveraging and fusing high-speed compact pulse LiDAR and video synthesis data on edge computing nodes, the CLEVER system can successfully recognize ten different types of vehicles with average 91.70% true positive rate. The Chapter is organized into seven sections. Section two describes the CLEVER edge hardware system, and Section three introduces the workflow of the algorithm. Sections four and five present the CLEVER Vehicle Classification neural network framework and real-world experiment, where it is compared with other advanced solutions. The conclusions and future works are in the last section.

8.2 CLEVER Edge System Description

8.2.1 High-speed Compact (Micro) Pulse LiDAR Description

Based on the operating mechanisms, LiDAR can be generally divided into two types: continuous-wave and pulse-based [132, 132]. Continuous-wave LiDAR calculates the phase difference between the reflected light and the reflected light to calculate the target distance, achieving ultra-precise results, e.g., error less than 0.1mm [133]. To avoid multiple matches in one phase, measurement frequency needs to be low enough for phase distinguishing (always less than 50Hz) and thus this type of LiDAR is suitable to scan stationary or slow-moving objects [133, 134].

Pulse LiDAR sends out laser pulses at a high frequency and then detects corresponding reflective information [135, 132]. By contrast, pulse LiDAR controls pulse frequencies and the pulse energy is relatively concentrated in space. Therefore, pulse LiDAR is able to detect fast-moving objects even in challenging conditions (rainy or snowy conditions), reaching ultrahigh detection frequency (up to 10kHz), fast post-processing time and reliable accuracy (less than 1% error) [132].

"Time-of-flight" (ToF) measurement by pulsed laser is the fundamental concept for single beam high-speed compact pulse LiDAR. The basic principle is shown in Figure 8.1. Compared with other scanning or flash LiDAR sensor (i.e., VLP-16 LiDAR) mentioned earlier, the high-speed compact pulse LiDAR has a relatively limited field of view. Despite this, the advantages are still obvious and can be summarized as follows:

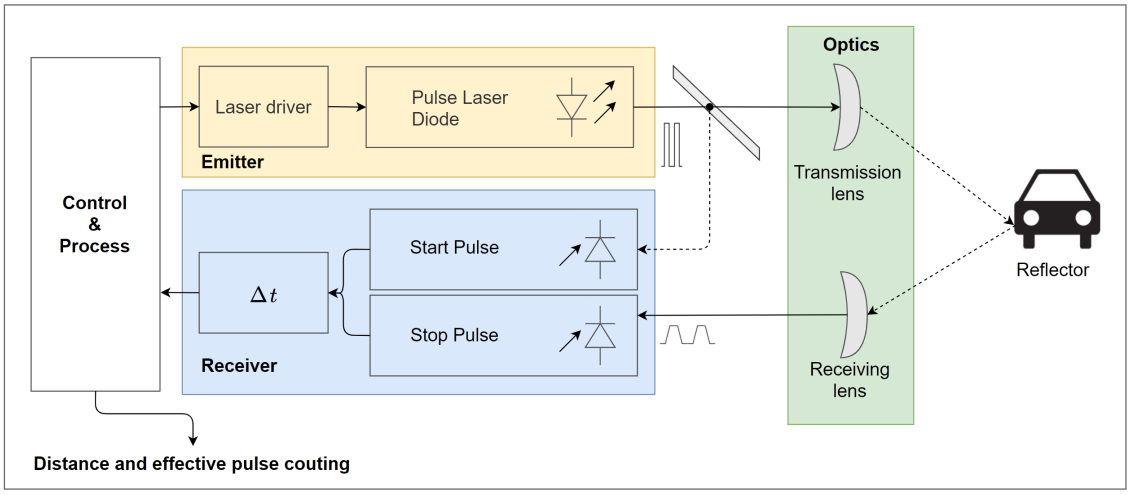


Figure 8.1: A pulsed laser-based 1D ToF system typically consists of a laser pulse transmitter, the necessary optics, two receiver channels, and a high-speed analog-to-digital converter (ADC). The laser pulse transmitter emits a continuous optical pulse to an optically visible target and simultaneously passes a starting signal to the receiver. In the same way, the optical pulse is reflected from the object and collected by the receiver detector of the stop receiver channel and then converted by ADC. The ADC uses its time base to convert the time interval of two signals, representing the target's distance.

- **Cost-effective.** The high-speed pulse LiDAR only costs 3% to 10% of the prices for the scanning LiDARs used in previous research.
- **High-speed.** The pulse measuring frequency can reach up to 5k-10kHz, which is hundreds or even thousands of times higher than scanning LiDAR.
- **High-reliability** Controlling only one or several beams, the LiDAR control algorithm can change dynamically in response to the external environment more quickly and then adjust the parameters of the laser beam to obtain a reliable detection result (especially in rain and snow conditions).
- **Lower Data Volume** The 3D data volume obtained by scanning LiDAR is huge (20-50 MB/s) and requires complicated processing and storage procedures. High-speed Compact Pulse LiDAR only has 2D sequence information, which is light-weight and friendly to data processing and storage terminals.
- **Lighter and Smaller** Since complex mechanical rotating units are no longer needed, the size and weight of the sensor can be greatly reduced.

8.2.2 Hardware System

The hardware system includes two key components: edge computing node and compact pulse LiDAR, as shown in Figure 8.2. This study uses Jetson Xavier NX as an embedded edge platform for real-time data processing streamed by the LiDAR sensor. The Jetson AGX Xavier has a 6-core ARM CPU and 384 CUDA cores, and 48 tensor cores GPU. To fully stimulate the computing potential on edge nodes, the entire workflow running on the edge side of the CLEVER system is combined with reasonable computing resources without overlap in one cycle. In this

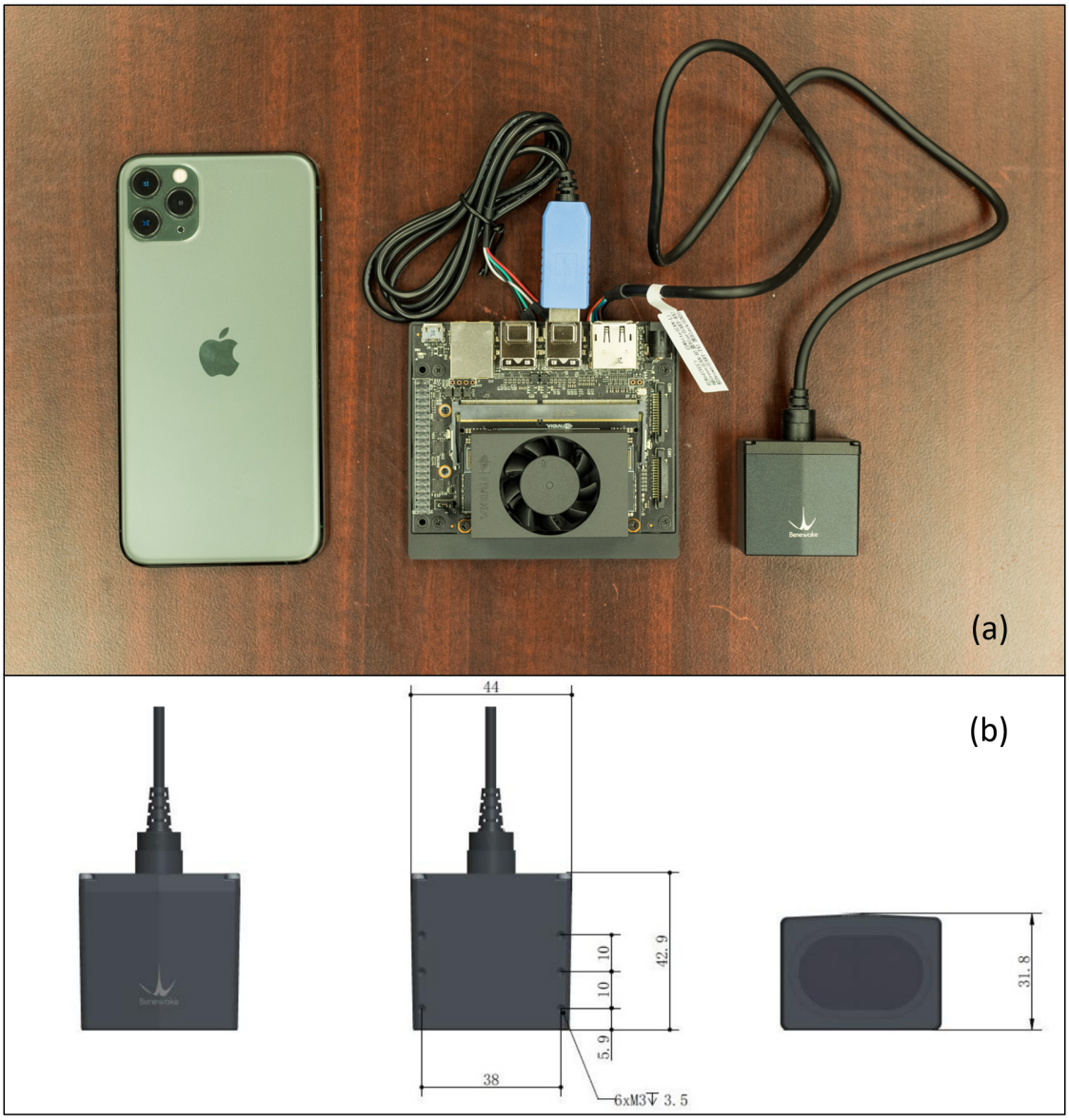


Figure 8.2: CLEVER system edge node, including the NVIDIA Jetson Xavier NX and Benewake TF03 pulse LiDAR. The subfigure (a) using an Apple iPhone 11 pro max (77.8mm*158.0mm*8.1mm) as a reference to show the system size. The sub figure (b) is the Benewake TF03 detail size.

research, the Benewake TF03 single beam pulse LiDAR, which measures the Pulse Time of Flight (PTOF) for range calculation, is fully integrated into the CLEVER hardware system.

8.2.3 Implementation Illustration

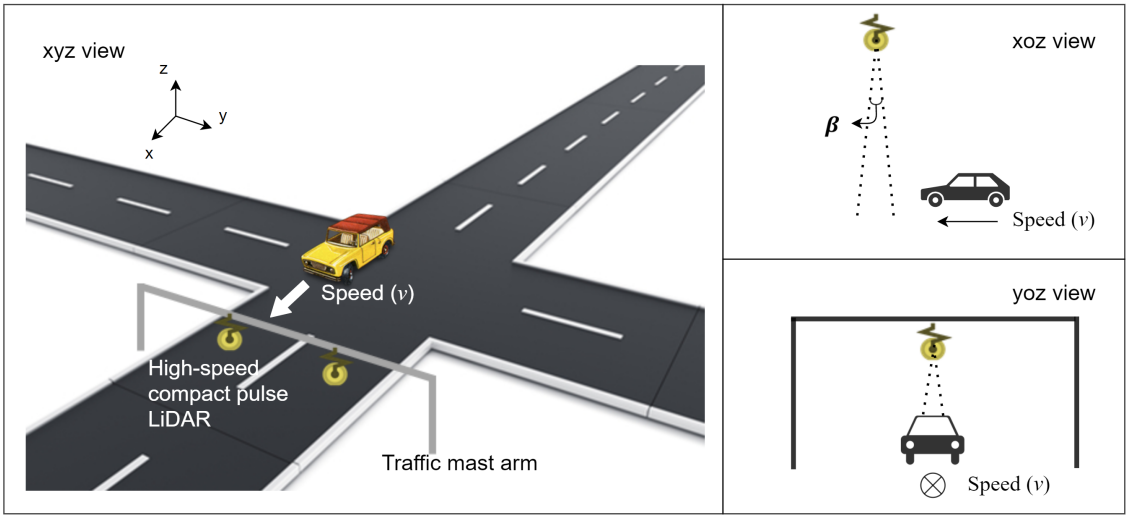


Figure 8.3: The installation illustration of the CLEVER system. Traffic mast is one of the facilities that can install the CLEVER edge nodes, besides, the overpass, tunnel top surface, L-shape utility pole, and other overhead facilities can be potential installation locations.

One CLEVER edge node can contain up to four TF03 LiDAR units to cover four separate travel lanes. The sensor can be deployed on the traffic mast arm, L-shape arm, overpass, or even signal light pole in an overhead configuration, as shown in Figure 8.3. The height is adjustable from two meters to eighty meters above the targets. Each LiDAR unit emits a single laser beam perpendicular to traffic on the XY plane and measures distances between the sensor and vehicles passing in front of the sensor continuously with a frequency from 100Hz to 10kHz measurements per second. The raw sensor readings (including distance and strength) and timestamp of each measurement are recorded.

8.3 Algorithms Workflow

8.3.1 System Input

As in Figure 8.4, the algorithms implemented for the CLEVER system can be divided into four parts: LiDAR data input, attributes input, the data pre-processing, and classification neural network. Two data sources are used as system input, including the captured distance and strength captured, as well as the environmental and object-level of attributes captured by sensors, e.g., speed limit, time of day, no object height, data record frequency, object contour length and the object passing duration. Both the LiDAR and attributes information are used for further data cleaning and pre-processing. Furthermore, a minority-enhancing customized neural network for vehicle classification is developed and implemented on the edge computing node.

8.3.2 Data Cleansing and Pre-processing

The data cleansing procedures are used to control and assure the real-time LiDAR features and attributes information quality. Due to the inherent characteristics of the LiDAR system, the generated pulse laser beam passes the high transmissive surface (i.e., glass) directly. The only limited reflective signal is obtained for these surfaces and can not support the range measurement algorithm. In such cases, obtained data need to be corrected based on the actual application scenarios. This study proposes a shape-aware data correction workflow. As shown in Figure 8.5, no meaningful data is received if the laser beam is shot on the vehicle's front or rear glass. If the reflective pulse is too weak to analyze, the output data records the range result as an outlier. Due to the vehicle's front and rear glass can be closely treated as linear change on the z-axis, a linear relationship is imputed based on the closest normal records and replacing the outlying LiDAR

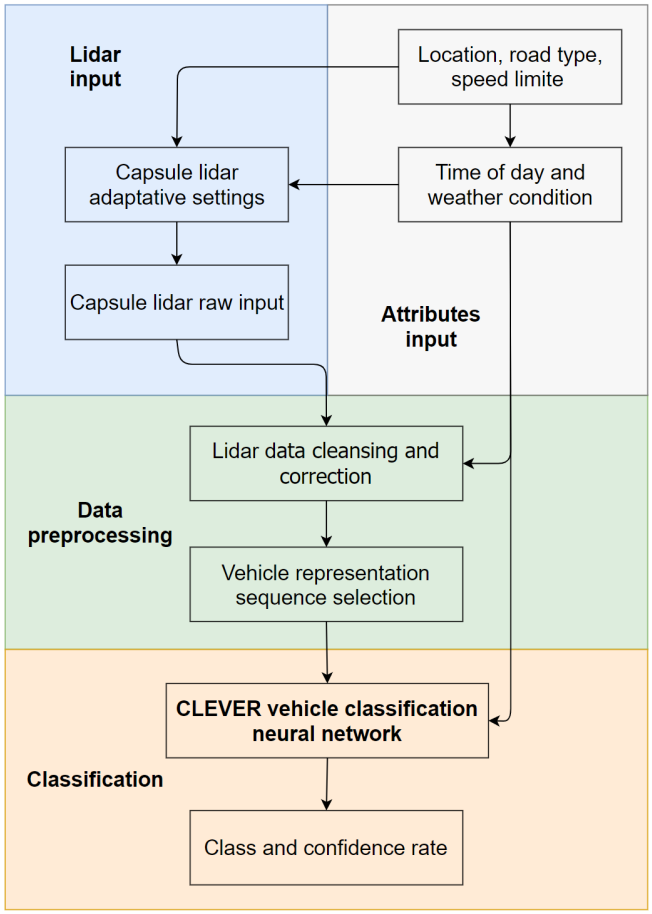


Figure 8.4: The algorithm workflow of CLEVER, including LiDAR and attributes input, data pre-processing component and classification neural network.

data.

After data correction, vehicle representation sequences are selected and fed to the neural network. To guarantee the vehicle contour is fully captured by the associated sequence, this study defines the following references:

- **Vehicle Starting Point (p_0^i).** To guarantee the laser detected a vehicle (i) instead of other disturbing objects, I defined that five continuous detection records above 30cm can be

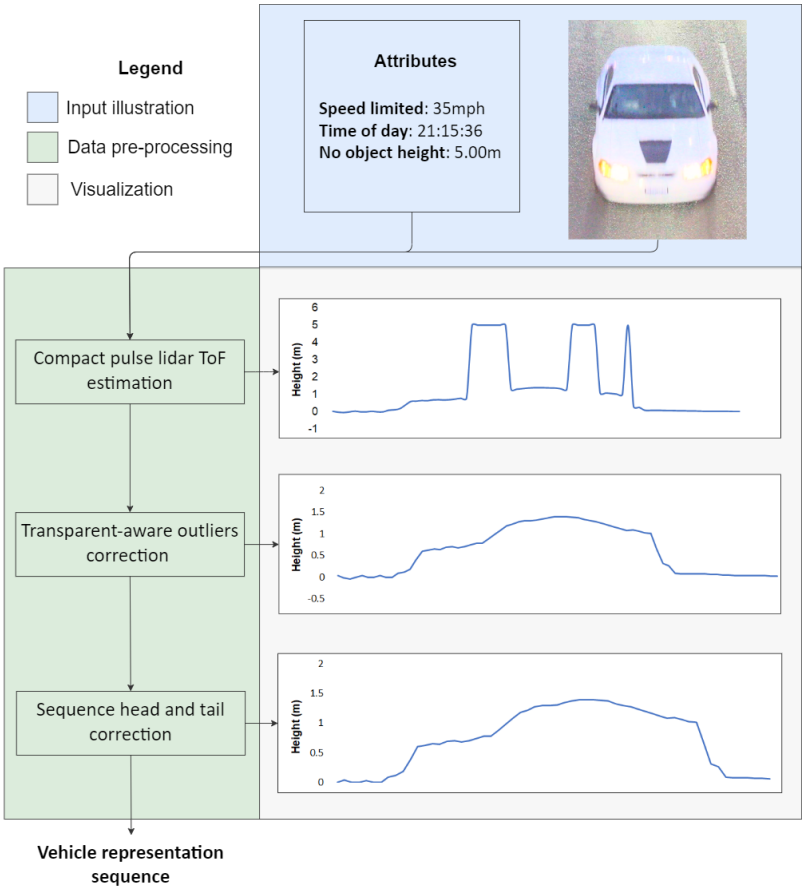


Figure 8.5: Illustration of data pre-processing procedures of CLEVER edge nodes. The top figure is the belonged white sedan captured by a video recorder.

treated as a vehicle head and used the first point as p_0^i.

- **Vehicle Representation Sequence Starting Point (s_0^i).** To make sure both the vehicle features and background conditions are fully covered by the sequence, p_0^i serves as the referring point, and all records with timestamp t_{head} before p_0^i are valid sequence points.

- **Vehicle Ending Point (p_{max}^i).** To guarantee the sequence recorded the vehicle completely, the vehicle ending points are defined as five continuous detection records below 20cm.

Denote the first record of these ending points as p^i_{max}.

- **Vehicle Representation Sequence Ending Point (s^i_{max}).** p^i_{max} is the referring point and all records with timestamp t_tail after p^i_0 are valid sequence points. t_{tail} is equal to t_{head}.

Finally, the vehicle contour representation sequence for vehicle i (S^i) is selected as the input of the CLEVER-VC. Typical representation sequences are visualized in Figure 8.6.

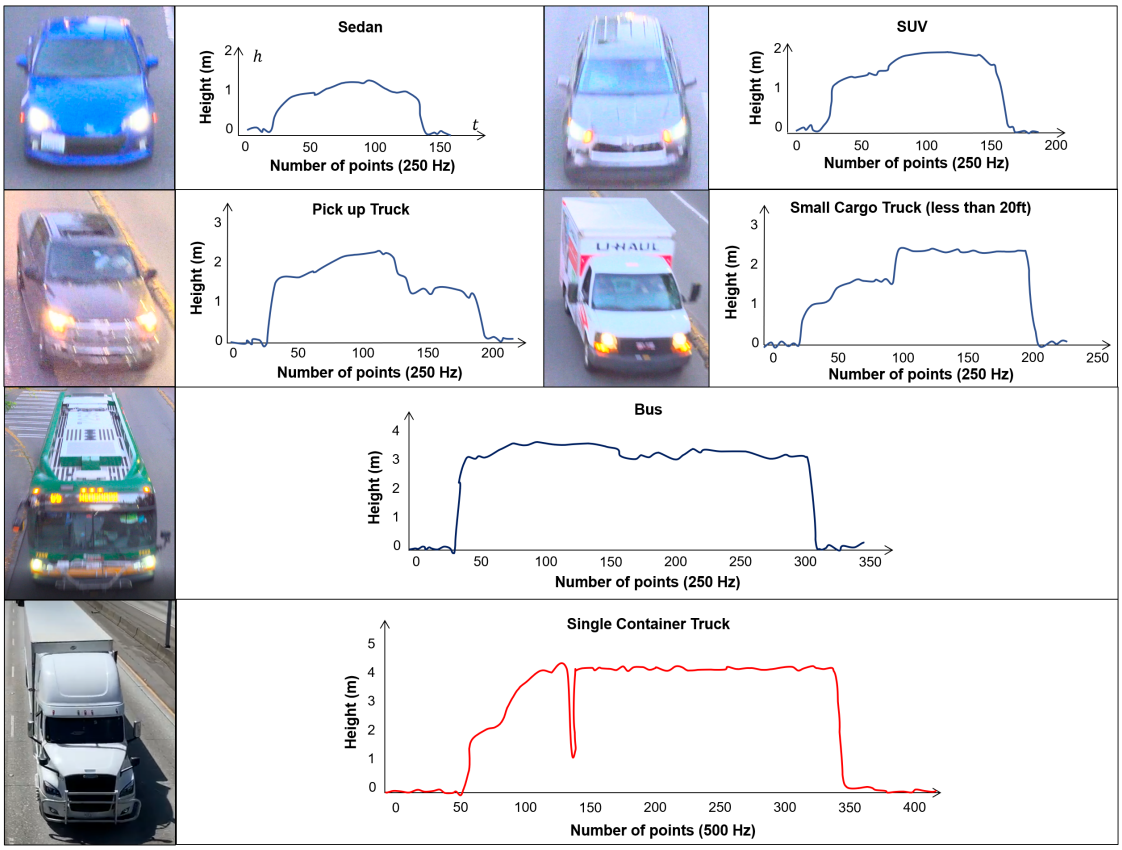


Figure 8.6: Vehicles representation sequences examples. The dark blue lines are collected in the urban area and the frequency is 250Hz. The single container truck sequence (red) is collected at the freeway using 500Hz frequency. Left figures are the belonged vehicles captured by a video recorder.

8.4 CLEVER Vehicle Classification neural network

To fully capture the vehicle contour features and classify vehicle type, a deep learning approach – CLEVER vehicle classification (CLEVER VC) neural network for sequential data recognition is proposed. Inspired by the text data classification models in natural language processing (NLP) [391, 392], two kinds of typical features, i.e., temporal dependency features and spatial distribution features, are fully learned and captured in the task. Furthermore, as vehicle representation sequence is also somehow related to multiple environmental and identity-level attributes, e.g., time of day, vehicles contour estimation deviation, the sensor installation location, and speed limit, building a learnable component to fully integrate the attributes impacts and further cooperate with spatial and temporal features is another target for the CLEVER VC. **In summary, CLEVER VC takes as inputs vehicle representation sequence** (S_i) **and attributes sequence** (A_i)**.**

8.4.1 Attributes Embedding

Attribute features are part of the classification tasks, and can be easily divided into two types: 1) the environmental features including the road speed limit (30mpg/60mph) and the time of day; 2) the sensing characteristics of each object, including the total contour representation sequence length, the data record frequency (250Hz/500Hz), the no object height and the object passing duration. Since the impact of these variables in the classification task cannot be fully quantified, and there may be overlap or superposition, it is crucial to integrate these variables into the framework. Instead of concatenating them with vehicle contour sequence, an embedding component is designed for these meta-features and is totally integrated into the CLEVER-VC. In general, the embedding process is to build a learnable procedure for transferring the previous input divi-

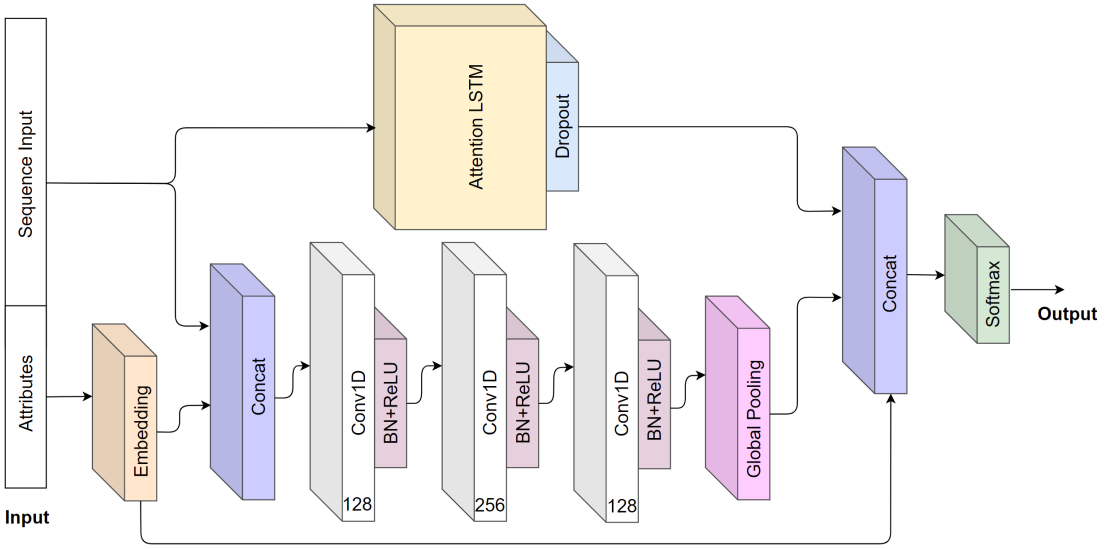


Figure 8.7: The CLEVER VC neural network architecture.

sions into a higher/lower-dimensional vector, which can capture the representation and enlarge the category feature difference [388, 68]. The mathematical process can be represented as:

$$Att_{a_j^i} = W^{a_j^i} \cdot a_j^i \tag{8.1}$$

In equation (1), a^i is the attribute information belonging to S^i. The j represents the sequence internal reference. W^{a^i} are the learnable weights, with dimension $\mathscr{R}^{\mathbf{A}*\mathbf{E}}$, which transforms the input attributes category values a^i ($a^i \in A$) from original space A to the embedding space $E^{a_j^i}$. Each attribute feature has its corresponding mapping matrix to ensure the feature can be mapped into a suitable space. After embedding, the next step is to concatenate the outputs E^{a^i} with S^i (using SE_i to represent the concatenating output). SE_i is then used as the input of the convolutional block.

8.4.2 Convolutional Block

Temporal convolutions (or 1D convolution) have proven to be an effective learning model for time series sequence classification [398, 399]. Fully Convolutional Networks (FCN) [400, 394], comprised of temporal convolutions, are typically used as feature extractors. Global average pooling is used to reduce the number of parameters in the model prior to classification. In the proposed model, the fully convolutional block is paralleled with an attention LSTM block for capturing the sequential features, as shown in Figure 8.7. The j^th dimension of its output is denoted as:

$$C_{SE_i^j}^{conv} = \sigma_{cnn} \cdot (W_{conv} * SE_i^{j:j+k-1} + \varepsilon) \tag{8.2}$$

In equation (2), the $C_{SE_i^j}^{conv}$ is the j^{th} record output after the 1D convolutional layer. And the W_{conv} represents the weight matrix that needs to be trained. σ_{cnn} is the activation function, and k is the kernel size of the filter. ε is the bias term. The fully convolutional block consists of three stacked temporal convolutional blocks with filter sizes of 128, 256, and 128, respectively. Each convolutional block is identical to the convolution block in the CNN architecture proposed by [391]. Each block consists of a temporal convolutional layer, which is accompanied by batch normalization followed by a Rectified Linear Unit (ReLU) activation function. Finally, global average pooling is applied after the final convolution block, where each channel has the average value.

8.4.3 Attention LSTM Block

To further capture the input sequence's temporal dependencies, I introduced the RNN module into the model. RNN is a widely-used artificial neural network for learning temporal features in sequences, especially the transportation data sequence. In CLEVER-VC, LSTM structure is leveraged to capture the sequence dependency features. The main contribution of the RNN block is that it can transfer and learn the features from the previous input to the future moments. However, in scenarios requiring long-range information dependence, it is challenging to train a well-perform RNN because of the gradient vanishing/exploding problems [1].

In each LSTM neuron, there are three gates are contained: input gate $i_{(t)}$, output gate $o_{(t)}$ and forget gate $f_{(t)}$ inside a neural. Each gate is controlled by their own weight $w^{(t)}$ and the previous neural output $h_{(t-1)}$. Also, the memory cascading process can be divided into two parts: new memory generation $\widetilde{c}_{(t)}$ and final memory generation $c_{(t)}$. After the last memory is generated, the new hidden state $h_{(t)}$ is raised by the control of output gate $o_{(t)}$. $o_{(t)}$ makes the assessment regarding what parts of the memory need to be shown in the $h_{(t)}$. Figure 8.7 shows the detailed structure of proposed neural and the mathematical formulations of (3)-(8) show the LSTM units working procedures in the following:

$$i_{(t)} = \sigma(W_{(i)}x_{(t)} + U_{(i)}h_{(t-1)}) \tag{8.3}$$

$$f_{(t)} = \sigma(W_{(f)}x_{(t)} + U_{(f)}h_{(t-1)}) \tag{8.4}$$

$$o_{(t)} = \sigma(W_{(o)}x_{(t)} + U_{(o)}h_{(t-1)}) \tag{8.5}$$

$$\widetilde{c}_{(t)} = tanh(W_{(c)}x_{(t)} + U_{(c)}h_{(t-1)}) \tag{8.6}$$

$$c_{(t)} = f_{(t)} \circ c_{(t-1)} + i_{(t)} \circ \widetilde{c}_{(t)} \tag{8.7}$$

$$h_{(t)} = o_{(t)} \circ tanh(c_{(t)}) \tag{8.8}$$

In fact, another challenge of vehicle classification based on sequential data is the key region dependency difference. In the LSTM model, each record of LSTM output using in h_i is treated equally while used as final input for dimensionality reduction; however, they are more like to contribute differently to the classification result. For example, the backside of a vehicle is the key region for distinguishing sedan and an SUV, and the length of the container is the critical difference between a 20ft or 40ft intermodal container truck. By introducing the attention mechanism, the sequential dependency can make a difference in the key regions. Specifically, the attention is essentially the weighted sum of the sequence h_i, where the weights are parameters learned by the model. In the CELVER VC neural network, one layer attention LSTM is integrated. Formally, I have the attention-aware LSTM $\mathscr{F}_{P_j}$ calculated by equations (9) and (10). The α_i is the weight for the i_{th} input historical occupancy sequence record. Moreover, the α_i is defined as (10).

$$\mathscr{F}_{P_j} = \sum_{i=1}^{L_{h_i}} \alpha_i * h_i \tag{8.9}$$

$$\alpha_i = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{i,k})} \tag{8.10}$$

In the equation (10), the $e_{ij} = fa(s_{i-1}, h_j)$ is an alignment model, the output scores how well the input around j^{th} record and the output at position i_{th} aligned. In this research, I use the alignment mechanism proposed by [401]. The integrated model directly computes a soft alignment,

which allows the gradient of the cost function to be backpropagated.

After the two blocks obtain the features, a concatenate layer is used to link both parts' features together, and then using a fully connected layer to transfer the original dimension of features to the number of classes. Finally, by passing a softmax operation layer, the probability of different classes can be obtained.

8.4.4 Loss Function

Considering the task for CLEVER VC is mainly sequential classifications, the cross-entropy ($\mathscr{L}_{Xent}$) [402] loss in training is defined as follows, where $\hat{y}_j$ is the estimated probability of the probe object that belongs to object j, and y_j denotes the ground truth vector while N denotes the number of identities in training data.

$$\mathscr{L}_{Xent} = -\frac{1}{N}\sum_{j=1}^{N} y_j \log(\hat{y}_j) \tag{8.11}$$

8.5 Experiment Results and Discussion

8.5.1 Dataset Description

Real-world Data Collection

To fully test the potential power of the CLEVER system, the data is collected by me in Seattle during both daytime and nighttime. As Figure 8.8 shows, LiDAR sensors are installed directly above the travel lane at overpasses (Figure 8.8 (a)): one site is urban road (speed limit 35 mph, original height 16.65ft, Figure 8.8 (b)), and the other is freeway (speed limit 60 mph, original height 24.89ft, Figure 8.8 (c)). A total number of 909 vehicles were captured in the daytime

Figure 8.8: The data collection illustration. (a) visualize the installation angle of CLEVER compact LiDAR for vehicle represent sequence collection. (b) and (c) are the two overpasses used for data collection, both located in the city of Seattle, Washington State, USA.

and 631 vehicles in the nighttime. Vehicles are divided into ten classes: sedan, SUV, minivan, pickup truck, standard city buses, bi-articulated buses, small trucks (less than 20ft), 20ft containers trucks, 40ft container trucks, and multi-trailer trucks. To validate the dataset, supporting

video clips were also recorded.

Except for the real-world dataset, I also collected vehicle sequence data by augmentation (as shown in Figure 8.9). A Mask R-CNN [150] (with ResNeXt-101-FPN backbone [125]) well-trained on COCO object detection and segmentation dataset [194] is then implemented to capture the mask of the vehicle. 100 different kinds of vehicles with real sizes were included: 15 types of sedans, 15 types of SUVs, 15 types of minivans, 10 types of pickup trucks, 5 types of standard city buses, 5 types of bi-articulated buses, 15 types of small trucks (less than 20ft), 10 types of 20ft container trucks, 5 types of 40ft container trucks, and 5 types of multi-trailer trucks.

Data Augmentation

After obtaining the masks for augmented data, pixels were matched with real vehicle height. The highest point on the mask is treated as the exact height (this study used the actual dimension size of the brand and model for the augmented vehicle). Then, the highest values on the y axis at every x pixel construct the raw vehicle contour. Furthermore, to simulate the sequence length at different traffic conditions with different speed limits, this study used the real record numbers in 75% distribution interval of each vehicle type captured from the real dataset as a pool and then randomly generated the vehicle sequences length for the augmented data. In this procedure, each vehicle used for augmentation was re-sampled five to ten times with vehicle length distribution. Ground truth sequence length is treated as a Gaussian distribution, and the re-sampled sequence length is located in the $((\mu - 2\sigma), (\mu - 2\sigma))$ interval. Furthermore, by randomly adding the sequence head and tail through the same procedures of data pre-possessing (I used the actual non-vehicle LiDAR records as head and tail and tried to introduce the real environment impact), the vehicle augmentation representation sequence is generated and then fed to training.

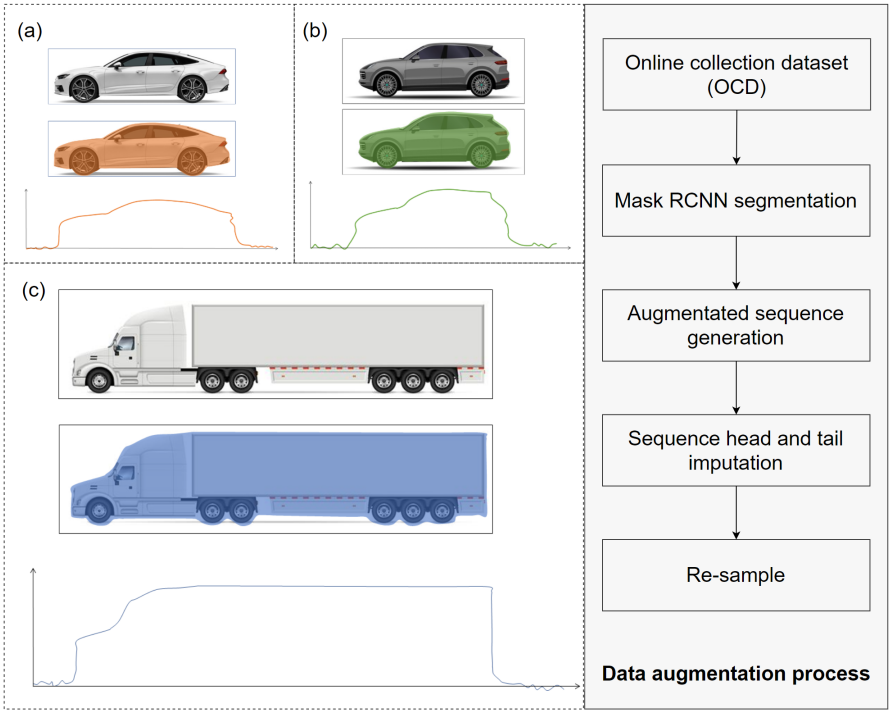


Figure 8.9: The data augmentation procedure for CLEVER System. (a), (b), (c) are three examples using in the data augmentation. Each of them includes original figures, figures passed the Mask R-CNN segmentation and the raw sequence generation visualization. Due to the raw sequence do not have real meanings except pixel location, the marks of x-axis and y-axis are both eliminated.

Finally, Table 8.1 summarizes class distribution in the dataset, sample volume, and augmented sequence volumes. Meanwhile, up-sampling was also conducted for the key original sequence to make sure the training was balanced.

8.5.2 CLEVER VC Neural Network Performance Evaluation

Baseline models

This study benchmarked CLEVER-VC against five different baselines: three traditional machine learning approaches and one deep learning approach, including SVM, k-NN, RF, and

Table 8.1: Detail information of both real collected and augmented datasets.

Class	Real Dataset				Augmented Dataset	Total
	Urban Road	Freeway	Daytime	Nighttime		
Sedan	234	274	294	216	150	658
SUV	167	221	214	174	150	538
Minivan	40	55	54	41	75	170
Pickup Truck	75	140	121	94	100	315
Standard city bus	45	24	50	19	75	144
Bi-articulated bus	22	8	25	5	40	70
Small truck	22	31	44	9	75	128
20ft containers truck	16	92	72	36	100	208
40ft containers truck	3	55	32	26	75	133
Multi-trailer truck	1	16	5	12	30	47
Total	**625**	**916**	**909**	**632**	**860**	**2401**

FCN. Each method is briefly described as follows:

- **SVM** [403] is a well-known linear classifier proposed by Cortes and Vapnik in 1995. By matching the given sequence data to a higher vector dimension, SVM maps training examples to points in a new space to maximize the width of the gap between the given categories. Then, the new examples are mapped to the same space and are predicted which category it belongs to by the gap size to befall in. The kernel used for optimization in SVM is a radial basis. In the experiment, the SVM is implemented based on the open-source Scikit-learn package [404].

- **k-NN** [405] classification algorithm is a well-known statistical method for pattern recognition. The basic idea is: according to each sample, select the k nearest neighbors to represent it, and then further distinguish and compare. Cover and Hart proposed the original proximity algorithm in 1968. The k-NN is part of the instance-based learning category, which does not have an explicit learning process. By using the input value as the classification feature, it can process directly after receiving the new sample. In this task, the k value is selected by the algorithm automatically based on the true positive rate (TPR), and the distance type is Euclidean distance. In this task, the k-NN is implemented based on the function 'KNNClassifier' available in the open-source package 'sequentia' [406].

- **RF** [407] is an ensemble learning method and is always used for classification and regression tasks, which operates by constructing multiple decision trees during the training phase and collecting the bagging weight of trees for result generation. When carrying the recognition task, the random forest output is the class selected by most trees. The RF shows the best performance in the previous LiDAR-based vehicle classification task

[139]. In this task, the number of trees used was 512 and the RF is implemented based on the open-source Scikit-learn package [404].

- **FCN** [393, 394] FCN is a special kind of CNN [394]. Usually, the FCN does not have fully connected layers (FC layers), which can extract more fine-grained information and reduce the loss of detailed features caused by dimension transition. This structure has been proved to have an excellent effect in the classification of sequence data than traditional CNN [408]. In the comparison, a three-layer FCN network with batch normalization and rectified linear unit is used (kernel). A global pooling layer is connected with the output for narrowing down the feature dimension. The cross-entropy loss ($\mathscr{L}_{Xent}$) is used to train the model.

Implementation and Comparison

The detailed experiment parameters used in the model comparison and research implementation are listed as follows:

- **Environment.** This study leverages a Linux machine (Ubuntu 18.04.2) with PyTorch v1.4.0, a Core-i7 7700K CPU. Two NVIDIA TITAN Xp GPUs are used for training the neural networks.

- **Data Split.** The total number of vehicles in the dataset (including both natural vehicle contour and augmented data) is 2401, and 1681 vehicles are selected as training (70%), 240 vehicles (10%) are selected as validation, and the rest part (480 vehicles, 20%) are used as testing set. To show the power of vision-augmented data, I also train the models by only using the real collected dataset (1541 vehicles), and the train, validation, and testing

split still follows the 70% (1078), 10% (154) and 20% (309) rule respectively.

- **Input for Vehicle Classification.** In the experiment, the input for the CLEVER VC and other baseline models includes two parts: the vehicle contour sequence (S_i), and the attributes information (A_i, including both environmental and object-level).

- **CLEVER VC Attributes Embedding.** In CLEVER-VC, the integrated **two environmental** attributes information and the belonged embedding vector dimension include: 1) Time of day to $\mathscr{R}^8$, 2) Road speed limit (mph) to $\mathscr{R}^4$. The total length for the environmental attributes embedding dimension is 12. Also, the integrated **four object-level** attributes information and the belonged embedding vector dimension are as follows: 1) Contour sequence length (Len_{S_i}) to $\mathscr{R}^{16}$, 2) No object height to $\mathscr{R}^2$, 3) Passing duration time (second) to $\mathscr{R}^{16}$, 4) Pulse signal frequency (Hz) to $\mathscr{R}^2$. The total length for the object-level attributes embedding dimension is 36.

- **Kernel information.** The kernel size of three 1D convolutional layers is set as 8, 5, and 3 respectively, which is the same as previous research settings [105, 391].

- **Number of LSTM neuron.** Based on the experiment, the number of LSTM neurons is 64.

- **Learning rate and optimizer.** I use the RMSprop as the optimizer, and the learning rate changes from $1e-3$ to $0.5e-4$. After every 50 epochs, the learning rate is reduced $1/\sqrt[3]{2}$.

- **Edge node deployment.** The well-trained model is deployed on Jeston Xavier NX, with PyTorch v1.4.0. The inference procedure consists of three parts: single sequence genera-

tion, sequence pre-processing, and classification model inference. All three procedures are performed on the edge nodes, which can make the model run in a plug-an-play manner.

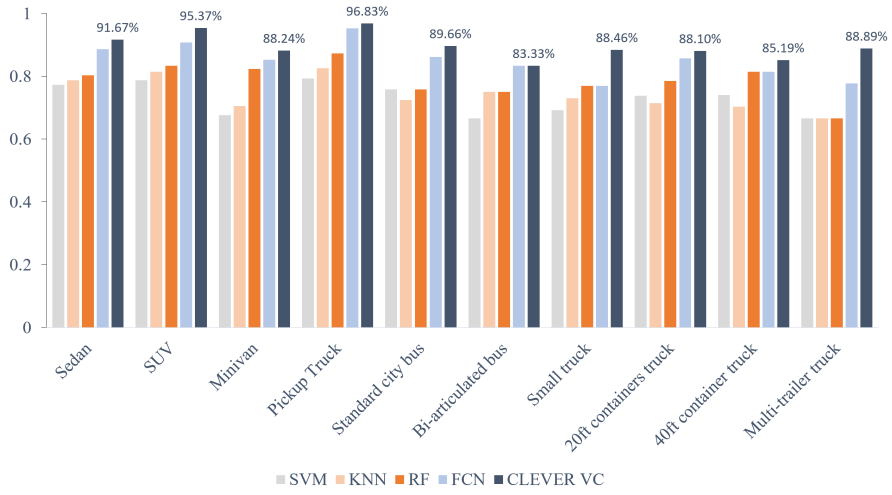


Figure 8.10: The performance comparison visualization of the baseline models and CLEVER VC (with the numeric label). It's clear to see that the classification precision of the minority-class significantly improved compared with other methods.

The detailed model comparison is presented in Table 8.2. Deep learning models outperformed traditional approaches both on class-level accuracies and on the overall accuracy. Based on experiment result, SVM always shows the worst performance; RF is the best traditional machine-learning model and shows better precision than k-NN and SVM. The same situation also can be found from the literature [139]. Furthermore, deep learning is proved to be a promising and encouraging approach in this classification task. The FCN is better than RF by 6% to 7%. The CLEVER VC shows the best performance in all classes, with the best accuracy on the sedan with 95.10% (no augmentation) and 96.83% (with augmentation). Furthermore, the comparison clearly shows all methods failed at the minority classes, such as buses and 40 ft container trucks without data augmentation. However, by including visual augmented data into the training and evaluation process, the performance on minority classes significantly im-

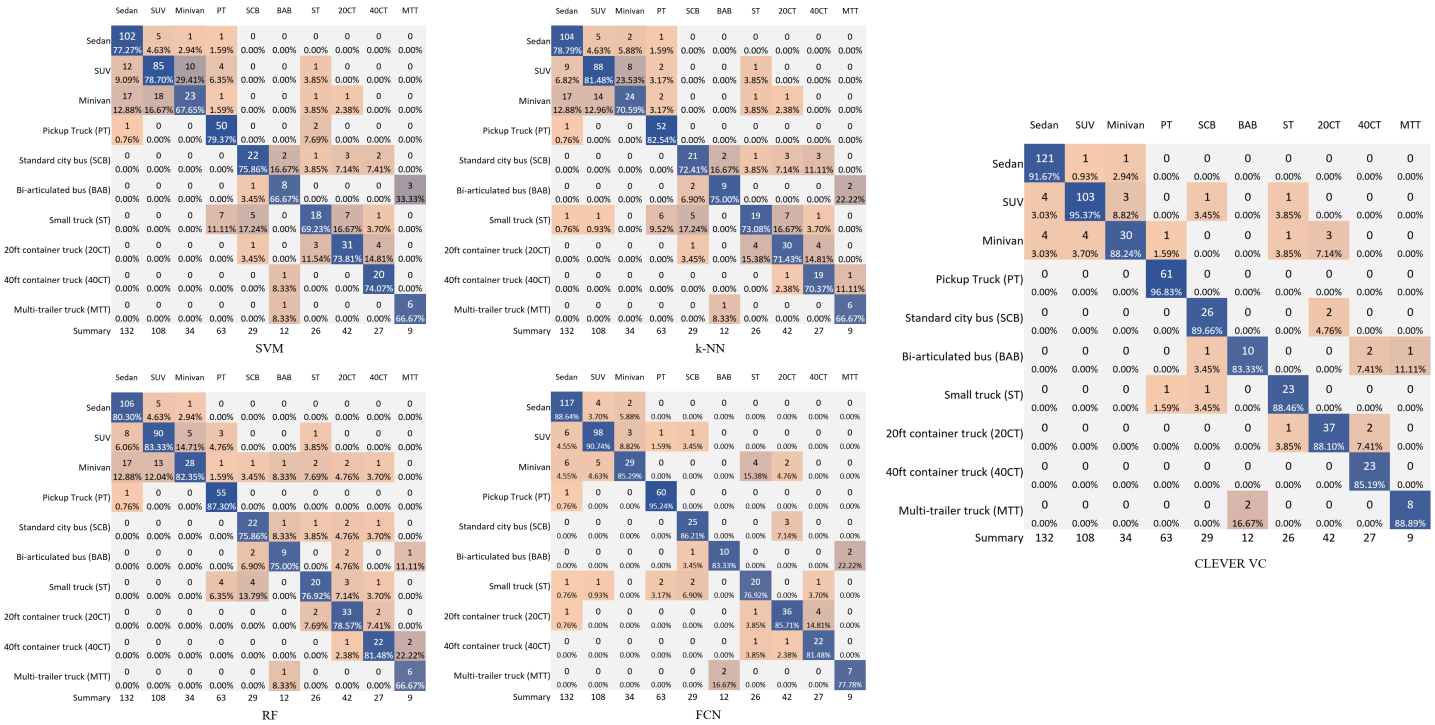


Figure 8.11: The classification confusion matrix of the baseline models and CLEVER VC.

Table 8.2: The TPR results comparison of CLEVER VC and baselines. The table is divided into two parts, the top half shows the result without data augmentation and low half shows the accuracy result with both real and augmented data.

Type	Class	Accuracy (TPR)				
		SVM	k-NN	RF	FCN	CLEVER VC
Without Data Augmentation	Sedan	74.51	75.49	78.43	88.24	**95.10**
	SUV	76.92	79.49	83.33	88.46	**92.31**
	Minivan	52.63	73.68	68.42	78.95	**89.47**
	Pickup Truck	79.07	81.40	88.37	93.02	**93.02**
	Standard city bus	64.29	71.43	71.43	78.57	**85.71**
	Bi-articulated bus	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**
	Small truck	72.73	72.73	**81.82**	63.64	72.73
	20ft containers truck	77.27	72.73	7.73	**86.36**	81.82
	40ft containers truck	58.33	58.33	58.33	58.33	**66.67**
	Multi-trailer truck	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**
	Overall Accuracy	73.65	76.25	79.17	85.33	**90.20**
With Data Augmentation	Sedan	77.27	78.79	80.30	88.64	**91.67** (↓ 3.43)
	SUV	78.70	81.48	83.33	90.74	**95.37** (↑ 3.06)
	Minivan	67.65	70.59	82.35	85.29	**88.24**(↓ 1.24)
	Pickup Truck	79.37	82.54	87.30	95.24	**96.83** (↑ 3.80)
	Standard city bus	75.86	72.41	75.56	86.21	**89.66** (↑ 3.94)
	Bi-articulated bus	66.67	75.00	75.00	**83.33**	**83.33** (↑ 16.67)
	Small truck	69.23	73.08	76.92	76.92	**88.46** (↑ 15.73)
	20ft containers truck	73.81	71.43	78.57	85.71	**88.10** (↑ 6.28)
	40ft containers truck	74.07	70.37	81.48	81.48	**85.19** (↑ 6.28)
	Multi-trailer truck	66.67	66.67	66.67	77.78	**88.89** (↑ 18.52)
	Overall Accuracy	75.73	77.18	81.12	87.97	**91.70** (↑ 1.5)

Table 8.3: The ablation study for the proposed CLEVER VC neural network.

Vehicle Contour Sequence	Object-level Attributes	Time of Day	Road Speed Limit	**TPR** ↑
✔	✗	✗	✗	88.23
✔	✔	✗	✗	91.44
✔	✔	✔	✗	91.50
✔	✔	✗	✔	91.57
✔	✔	✔	✔	**91.70**

proved, with a maximum 18.52% gain of accuracy on the multi-trailer truck. The customized data augmentation process successfully boosts the performances of minority-class and also lead to a 1.5% accuracy gain for overall accuracy. The detailed accuracy of each class, with or without data augmentation procedure, can be found in Table 8.2. The final performance visualization of each method, and the belong classification confusion matrix can be found in Figure 8.10 and Figure 8.11.

CLEVER VC Ablation Study

To thoroughly show the effectiveness of different inputs, including the vehicle representative contour sequence, the object-level sensing attributes, the environmental attributes including time of day and road speed limit, I devise a set of controlled experiments on the collected vehicle classification dataset. Besides the necessary vehicle representation contour sequence, I eliminate exactly one attribute with the same testing dataset for each experiment and test on the well-trained model. Then, I summarized the before and after true-positive rate, select the best models by five-fold cross-validation, and summarize the result into the following Table .

From the experiment, the CLEVER VC can reach to 88.23% TPR only using the vehicle

representation contour sequence. And the object-level attributes and the belonged customized feature embedding component shows significantly impact to the model precision improvement, which lead to a 3.21% TPR growth. Furthermore, even with the embedding operation, the time of day and road speed limit only show negligible contributions (only 0.13% and 0.11% respectively), and eliminating both only triggers the TRP drops 0.20%. Based on the before-after comparison, the CLEVER System and belong vehicle classification neural network can maintain the high-precision and stable classification results in various types of road segments and different time periods. However, the data was mainly collected in the Washington state, somehow with no extreme sunlight intensity conditions in the afternoon and evening. The time of day might be more beneficial for estimating the vehicle class for other states and countries.

8.6 System Evaluation

Table 8.4 compares the proposed new traffic sensing framework based on compact high-speed pulse LiDAR with the state-of-the-art research from both technical and practical aspects. Most of the previous research ignored system-level optimization. Among five systems, three of them used the VLP-16 LiDAR 16 beams LiDAR. Though they considered this LiDAR sensor as cost-effective, it is ten times higher than CLEVER system to cover two travel lanes. Considering the other hardware cost, including the communication facilities, laptops/server system, the CLEVER system cost is only around 10% of current cutting-edge solutions. Furthermore, previous solutions were all processed offline, which transmits collected data to the server or computational facilities where the post-processing was finished. However, CLEVER can work end-to-end with an average inference time from 212ms to 459ms based on the different lengths of input sequences. Besides affordability and easy-scalability architecture, compared with the cur-

Table 8.4: System level comparison of CLEVER System with other SOTA frameworks.

Comparable Item	Current State-of-the-art System				
	Wu's system [139]	Asborno's system [140]	Zhao's system [142]	Sahin's system [143]	**CLEVER System**
Proposed Year	2019	2019	2019	2020	2021
LiDAR Cost ($) (two lanes)	4000	unknown	4000	4000	400
System Cost ($) (two lanes)	6000 $\sim$ 8000	unknown	6000 $\sim$ 8000	6000 $\sim$ 8000	800 $\sim$ 1000
Power Consumption (watt)	80 $\sim$ 200w	unknown	80 $\sim$ 200w	80 $\sim$ 200w	12 $\sim$ 17w
Communication Support (Mbps)	17.2 $\sim$ 34.4	unknown	17.2 $\sim$ 34.4	17.2 $\sim$ 34.4	3.2 $\sim$ 5.4
Best Classification Method	RF	RF/MLP/SVM	1-layer NN	SVM	CLEVER VC
Installation Location	Upper side-fire	Upper side-fire	Upper side-fire	Side-fire	Overhead
Real-time Processing	✗	✗	✗	✗	✔
Online Classification	✗	✗	✗	✗	✔
Minority Class Friendly	✗	✗	✗	✗	✔
Plug-and-play	✗	✗	✗	✗	✔
Environmental Attributes Integration	✗	✗	✗	✗	✔

rent pioneer research solutions, CLEVER was also born with several inherent advantages from the system perspectives, including negligible power consumption (15% of other systems) and communication cost (20% of other systems), edge empowered plug-and-play skeleton, online data processing, and classifications, etc.

8.7 Chapter Summary and Future Works

This chapter presents a real-time plug-and-play vehicle counting and classification framework based on compact pulse LiDAR and edge artificial intelligence. A comprehensive LiDAR preprocessing workflow is proposed, automatically correcting the outlying values and providing reliable inputs for the classification algorithm. A customized minority-care vehicle classification methodology with environmental attribution integration component is customized and fully integrated into the CLEVER system and achieved significantly leading TPR to other advanced methodologies. Based on the benchmarking results, the CLEVER system has several advantages over previous solutions, including only about 10% of sensor and system costs, 15% of power consumption, and 20% of communication cost. Through the edge empowered end-to-end skeleton, the vehicle counting and classification algorithms can be finished in an online real-time manner. To build a more powerful and reliable vehicle classification system, future research will focus on improving algorithm accuracy, especially how to solve the vehicle recognition task under unbalanced data distributions.

9

Chapter 9. Final Remarks and Envisioning the Future

9.1 Research Contributions and Findings

Trustworthy ML and advanced computing in transportation systems merge dependable ML models with complex computational methodologies to enhance infrastructure cooperation and service adaptability. The efficacy of these models is tied directly to their training data and foundational algorithms. In transportation, the customization of trustworthy ML and advanced computing implies tailoring AI models that exhibit improved reliability, transparency, fairness

and robustness. These models have undergone comprehensive testing and validation, provide predictable results, can handle varied data inconsistencies, and are safeguarded against manipulation or deception. The dissertation's primary findings and research contributions, linked to the mentioned key topics, are organized into three distinct categories. These works have been thoroughly executed, documented, and discussed.

9.1.1 Part I: Contributions on Traffic Perception and Data Acquisition

As the saying goes, one cannot make bricks without straw. An accurate data-driven algorithm must be supported by a high-quality affluent data set. Thus, one part of my Ph.D. research work aims to explore novel sensing technologies and to improve the accuracy, efficiency and privacy of the perception algorithms.

Key Contribution 1. Cooperative Traffic Perception and Operation via Distributed IoT Systems: In the realm of data-driven solutions, abundant high-quality datasets are the backbone of effective algorithms. One key area of my doctoral research involves leveraging innovative sensing technologies to augment the accuracy, efficiency, and privacy of traffic perception and operation algorithms. Chapter 3 introduces the ECoMS system, an inventive algorithmic and edge-server collaborative structure that integrates edge computing and multi-camera re-identification to enhance traffic sensing via cooperative IoT architecture. ECoMS optimizes the utilization of computational resources at both the edges and traffic management centers, enabling efficient data transmission and the integration of road graph features. This results in an accurate representation of network-scale traffic information across various traffic modes—trucks, buses, motorcycles, bicycles, and regular vehicles—within a flexible, economical, and scalable workflow.

Key Contribution 2. Privacy-preserving Perception Methods of Non-motorized Users

with Edge AI: The rising tide of IoT devices is shifting the design of traffic perception systems from centralized cloud-based models to edge-of-network architectures. Edge computing is a distributed computing paradigm that situates computation and data storage closer to data sources, thereby enabling enhanced speed, reliability, security, and scalability. Furthermore, edge computing provides a platform for video-based pedestrian detection that is sensitive to privacy concerns. In Chapter 7, my research is thus devoted to the video-based perception of non-motorized users, with a focus on detection, classification, counting, pose estimation and interaction with traffic infrastructure systems based on customized Edge AI methods.

9.1.2 Part II: Contributions on Learning Multimodality Data Representations

To leverage the power of big data and traffic domain knowledge for establishing smarter urban traffic systems, customizing the machine learning methodologies and integrating the heterogeneous representations extracted from the deep neural networks to create physical-informed models for the traffic tasks has become one of my focuses.

Key Contribution 3. Fusing the Spatial-temporal Sequential Features with Language and Personal Attributes Features for More Robust, Precise and Interperable ML Models: In Chapter4 and Chapter 5, both proposed new methods incorporates multimodality data types, incldluing the historical sequential data, real-time sequential data, attributes categorical data, personal attributes data in the an end-to-end learning scheme. The integration is achieved via representation learning and heterogeneous feature embedding. Extensive testing shows that the customized ML models with multimodality data inputs outperform

Key Contribution 4. Integrating the Traffic Priors with Representation Leaning for More Robust, Precise and Interperable ML Models 6: Currently, city-wide application of

current congestion predictions methodologies presents significant challenges, including dealing with varied sensor data modalities, inadequate congestion fluctuation and propagation modeling, and poor adaptability to heterogeneous traffic network structures. My research addresses these issues by integrating urban planning domain priors into a general sequence prediction model, culminating in the TinT. The TinT model counters receptive field bias with a unique mix of long and short-range information routing mechanism and features an innovative anisotropic graph aggregation to capture uneven traffic flow propagation based on orientations. The TinT model's efficacy and versatility have been proven through extensive testing against other state-of-the-art models and across multiple data modalities in six major cities.

9.1.3 Part III: Contributions on Demonstrating Pilot Cooperative and Equitable Traffic Infrastructure Systems

I have been dedicated to enhancing the transportation needs of the elderly and disabled populations, as well as underserved and disadvantaged groups through the advanced technology customization. Currently, the traffic equity research is always closely related with safety and accessibility, and most applicable systems are in their infancy. To solve the challenges, my related research and contributions can be summarized into the following parts.

Key Contribution 5. Proposing the Cooperative Traffic Signal Assistance and Warning System for VRU: In the connected vehicle environment, most improvements primarily focus on the vehicular side, leaving a significant gap for non-motorized and disabled users. To address this, I introduced VENUS smart node, a novel infrastructure offering cooperative SPaT services and warnings. The smart node utilizes tailored computer vision algorithms and artificial intelligence pipelines at the edge to gather necessary user information in a privacy-preserving manner.

It generates real-time directional crossing requests for each pedestrian and cyclist. Furthermore, its enhanced communication system enables it to function as a reliable hub, sharing safety messages and interactions with signal controllers, connected vehicles, and users' personal devices. Compatible with the connected vehicles environment, the VENUS smart node improves the signal system cost-effectively due to its adaptability to existing infrastructures.

Key Contribution 6. Bias Mitigation for Learning-based Traffic Perception Models Through Vision-LiDAR Data Fusion: Given that real-world object distributions are often uneven and follow a long-tail pattern, deep learning networks may amplify biases and exhibit unacceptable accuracy for tail classes or uncommon objects with limited training data, leading to significant precision and equity issues. Therefore, quantifying and mitigating the biases of these networks in transportation data collection has become imperative. To enhance model fitness, I devised a specific multimodality data fusion mechanism that incorporates visual synthetic data to improve training for tail classes, yielding up to a 17% accuracy gain for uncommon vehicles, such as bi-articulated buses and multi-trailer trucks.

9.2 Future Research Directions

9.2.1 Customized Trustworthy Federated Learning and Advanced Computing Methods for Traffic Systems

With more and more edge-computing devices deployed in urban CPSs (cameras, LiDAR, communication devices, other smart sensors), compiling the data to a central server and training models using the data centrally pose increasing challenges, due to high latency, high communication cost, and privacy concerns. In my future research, I will keep investigating new distributed computing approaches, as well as novel machine learning approaches to collect urban data in

a collaborative and decentralized way (i.e., ubiquitous computing, federated learning). Federated learning is a promising technology that enables multiple devices to collaboratively learn a shared model while keeping all the training data locally on each device. This distributed learning framework has the potential to greatly improve model performances, while reducing the computational load (on servers) and better protecting the privacy of CPS users.

9.2.2 Equitable & Cooperative Traffic Infrastructure Systems

Transportation is a complex system-of-systems that needs to seamlessly and holistically integrate physical systems (roadways, bridges, devices), digital infrastructure (communication, data, software), and humans (users, workers, policy makers), with the central focus to fulfill the needs of people. This human-system cooperative approach is crucial to develop sensible transportation solutions that will be accepted and appropriately used. As part of this research direction, I will identify the factors that limit or facilitate system interactions with humans. I will foster infrastructure-vehicle cooperation technologies and research, including infrastructure/vehicle sensing and data collection, cooperative and infrastructure-enabled traffic-vehicle control, and related issues such as traffic equity and accessibility, cybersecurity (of both vehicles and the infrastructure) and privacy protection. Additionally, I will stay dedicated to enhancing the infrastructure needs of the elderly and disabled populations, as well as underserved and disadvantaged groups through cutting-edge technologies and novel methodologies.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[3] Mohammadreza Saeedmanesh and Nikolas Geroliminis. Dynamic clustering and propagation of congestion in heterogeneously congested urban traffic networks. *Transportation research procedia*, 23:962–979, 2017.

[4] Vinod Khandkar and Manjesh Hanawal. Fairnet: A measurement framework for traffic discrimination detection on the internet. *IEEE Transactions on Network and Service Management*, 2023.

[5] Kashif Ahmad, Majdi Maabreh, Mohamed Ghaly, Khalil Khan, Junaid Qadir, and Ala Al-Fuqaha. Developing future human-centered smart cities: Critical analysis of smart city security, data management, and ethical challenges. *Computer Science Review*, 43:100452, 2022.

[6] Wanxu Chen, Jiaojiao Bian, Jiale Liang, Sipei Pan, and Yuanyuan Zeng. Traffic accessibility and the coupling degree of ecosystem services supply and demand in the middle reaches of the yangtze river urban agglomeration, china. *Journal of Geographical Sciences*, 32(8):1471–1492, 2022.

[7] Yujie Guo, Zhiwei Chen, Amy Stuart, Xiaopeng Li, and Yu Zhang. A systematic overview of transportation equity in terms of accessibility, traffic emissions, and safety outcomes: From conventional to emerging technologies. *Transportation research interdisciplinary perspectives*, 4:100091, 2020.

[8] Andrés Camero and Enrique Alba. Smart city and information technology: A review. *cities*, 93:84–94, 2019.

[9] Adnan Mahmood, Sarah Ali Siddiqui, Quan Z Sheng, Wei Emma Zhang, Hajime Suzuki, and Wei Ni. Trust on wheels: Towards secure and resource efficient iov networks. *Computing*, 104(6):1337–1358, 2022.

[10] Lara Engelfriet and Eric Koomen. The impact of urban form on commuting in large chinese cities. *Transportation*, 45(5):1269–1295, 2018.

[11] Shafiza Ariffin Kashinath, Salama A Mostafa, Aida Mustapha, Hairulnizam Mahdin, David Lim, Moamin A Mahmoud, Mazin Abed Mohammed, Bander Ali Saleh Al-Rimy, Mohd Farhan Md Fudzee, and Tan Jhon Yang. Review of data fusion methods for real-time and multi-sensor traffic flow analysis. *IEEE Access*, 9:51258–51276, 2021.

[12] Mahima Nama, Ankita Nath, Nancy Bechra, Jitendra Bhatia, Sudeep Tanwar, Manish Chaturvedi, and Balqies Sadoun. Machine learning-based traffic scheduling techniques for intelligent transportation system: Opportunities and challenges. *International Journal of Communication Systems*, 34(9):e4814, 2021.

[13] Sokemi Rene Emmanuel Datondji, Yohan Dupuis, Peggy Subirats, and Pascal Vasseur. A survey of vision-based traffic monitoring of road intersections. *IEEE transactions on intelligent transportation systems*, 17(10):2681–2698, 2016.

[14] Andrea Benedetto, Fabio Tosti, Luca Bianchini Ciampoli, and Fabrizio D'amico. An overview of ground-penetrating radar signal processing techniques for road inspections. *Signal processing*, 132:201–209, 2017.

[15] Junxuan Zhao, Hao Xu, Yuan Tian, and Hongchao Liu. Towards application of light detection and ranging sensor to traffic detection: an investigation of its built-in features and installation techniques. *Journal of Intelligent Transportation Systems*, 26(2):213–234, 2022.

[16] Yang Wang. Survey on deep multi-modal data analytics: Collaboration, rivalry, and fusion. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1s):1–25, 2021.

[17] Wenqing Zheng, Hao Frank Yang, Jiarui Cai, Peihao Wang, Xuan Jiang, Simon Shaolei Du, Yinhai Wang, and Zhangyang Wang. Integrating the traffic science with representation learning for city-wide network congestion prediction. *Information Fusion*, 99:101837, 2023.

[18] Hehua Yan, Qingsong Hua, Daqiang Zhang, Jiafu Wan, Seungmin Rho, and Houbing Song. Cloud-assisted mobile crowd sensing for traffic congestion control. *Mobile Networks and Applications*, 22(6):1212–1218, 2017.

[19] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[20] Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E Kamal. Edge computing assisted autonomous driving using artificial intelligence. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 254–259. IEEE, 2021.

[21] Peter Corcoran and Soumya Kanti Datta. Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network. *IEEE Consumer Electronics Magazine*, 5(4):73–74, 2016.

[22] Hao Yang, Ruimin Ke, Zhiyong Cui, Yinhai Wang, and Karthik Murthy. Toward a real-time smart parking data management and prediction (spdmp) system by attributes representation learning. *International Journal of Intelligent Systems*, 37(8):4437–4470, 2022.

[23] Hao Frank Yang, Jiarui Cai, Chenxi Liu, Ruimin Ke, and Yinhai Wang. Cooperative multi-camera vehicle tracking and traffic surveillance with edge artificial intelligence and representation learning. *Transportation Research Part C: Emerging Technologies*, 148:103982, 2023.

[24] Nour-Eddin El Faouzi, Henry Leung, and Ajeesh Kurian. Data fusion in intelligent transportation systems: Progress and challenges–a survey. *Information Fusion*, 12(1):4–10, 2011.

[25] Matthew Veres and Medhat Moussa. Deep learning for intelligent transportation systems: A survey of emerging trends. *IEEE Transactions on Intelligent transportation systems*, 21(8):3152–3168, 2019.

[26] Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

[27] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Murtaza Choudhury, and Alex Kai Qin. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1544–1561, 2020.

[28] Ruimin Ke, Zhibin Li, Sung Kim, John Ash, Zhiyong Cui, and Yinhai Wang. Real-time bidirectional traffic flow parameter estimation from aerial videos. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):890–901, 2016.

[29] Guohui Zhang, Ryan P Avery, and Yinhai Wang. Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras. *Transportation research record*, 1993(1):138–147, 2007.

[30] Yegor Malinovskiy, Yao-Jan Wu, and Yinhai Wang. Video-based vehicle detection and tracking using spatiotemporal maps. *Transportation research record*, 2121(1):81–89, 2009.

[31] Álvaro González, Miguel Ángel Garrido, David Fernández Llorca, Miguel Gavilán, J Pablo Fernández, Pablo F Alcantarilla, Ignacio Parra, Fernando Herranz, Luis M Bergasa, Miguel Ángel Sotelo, et al. Automatic traffic signs and panels inspection system using computer vision. *IEEE Transactions on intelligent transportation systems*, 12(2):485–499, 2011.

[32] Xiaolei Di, Yu Xiao, Chao Zhu, Yang Deng, Qinpei Zhao, and Weixiong Rao. Traffic congestion prediction by spatiotemporal propagation patterns. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 298–303. IEEE, 2019.

[33] Mengting Bai, Yangxin Lin, Meng Ma, Ping Wang, and Lihua Duan. Prepct: Traffic congestion prediction in smart cities with relative position congestion tensor. *Neurocomputing*, 444:147–157, 2021.

[34] Ibai Lana, Javier Del Ser, Manuel Velez, and Eleni I Vlahogianni. Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine*, 10(2):93–109, 2018.

[35] Attila M Nagy and Vilmos Simon. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148–163, 2018.

[36] Shengnan Guo, Youfang Lin, Shijie Li, Zhaoming Chen, and Huaiyu Wan. Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3913–3926, 2019.

[37] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

[38] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[39] Zhiyong Cui, Longfei Lin, Ziyuan Pu, and Yinhai Wang. Graph markov network for traffic forecasting with missing data. *Transportation Research Part C: Emerging Technologies*, 117:102671, 2020.

[40] Jiarui Cai, Yizhou Wang, Hung-Min Hsu, Jenq-Neng Hwang, Kelsey Magrane, and Craig S Rose. Luna: Localizing unfamiliarity near acquaintance for open-set long-tailed recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 131–139, 2022.

[41] Zhongqi Miao, Ziwei Liu, Kaitlyn M Gaynor, Meredith S Palmer, Stella X Yu, and Wayne M Getz. Iterative human and automated identification of wildlife images. *Nature Machine Intelligence*, 3(10):885–895, 2021.

[42] Zhiming Luo, Frederic Branchaud-Charron, Carl Lemaire, Janusz Konrad, Shaozi Li, Akshaya Mishra, Andrew Achkar, Justin Eichel, and Pierre-Marc Jodoin. Mio-tcd: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, 27(10):5129–5141, 2018.

[43] Nadiya Shvai, Abul Hasnat, Antoine Meicler, and Amir Nakib. Accurate classification for automatic vehicle-type recognition based on ensemble classifiers. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1288–1297, 2019.

[44] Amir Mukhtar, Michael J Cree, Jonathan B Scott, and Lee Streeter. Mobility aids detection using convolution neural network (cnn). In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–5. IEEE, 2018.

[45] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019.

[46] Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Long-tailed recognition via weight balancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6897–6907, 2022.

[47] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 36(4):25–34, 2020.

[48] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5310–5319, 2019.

[49] Angelina Wang and Olga Russakovsky. Directional bias amplification. In *International Conference on Machine Learning*, pages 10882–10893. PMLR, 2021.

[50] Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8919–8928, 2020.

[51] Lu Yang, He Jiang, Qing Song, and Jun Guo. A survey on long-tailed visual recognition. *International Journal of Computer Vision*, pages 1–36, 2022.

[52] Jiarui Cai, Yizhou Wang, and Jenq-Neng Hwang. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 112–121, 2021.

[53] Olcay Sahin, Reza Vatani Nezafat, and Mecit Cetin. Methods for classification of truck trailers using side-fire light detection and ranging (lidar) data. *Journal of Intelligent Transportation Systems*, 26(1):1–13, 2022.

[54] Swadesh Kumar Maurya and Ayesha Choudhary. Deep learning based vulnerable road user detection and collision avoidance. In *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–6. IEEE, 2018.

[55] Birhanu Eshete. Making machine learning trustworthy. *Science*, 373(6556):743–744, 2021.

[56] Bhavani Thuraisingham. Trustworthy machine learning. *IEEE Intelligent Systems*, 37(1):21–24, 2022.

[57] Ke Wang and Ping Guo. An ensemble classification model with unsupervised representation learning for driving stress recognition using physiological signals. *IEEE transactions on intelligent transportation systems*, 22(6):3303–3315, 2020.

[58] Hailong Zhang, Yuankai Wu, Huachun Tan, Hanxuan Dong, Fan Ding, and Bin Ran. Understanding and modeling urban mobility dynamics via disentangled representation learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2010–2020, 2020.

[59] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[60] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28, 2018.

[61] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. Robust road network representation learning: When traffic patterns meet traveling semantics. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 211–220, 2021.

[62] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1695–1704, 2018.

[63] Xinyan Liu. An image classification network for network traffic representation learning. In *2021 6th International Symposium on Computer and Information Processing Technology (ISCIPT)*, pages 358–361. IEEE, 2021.

[64] Lingbo Liu, Jiajie Zhen, Guanbin Li, Geng Zhan, Zhaocheng He, Bowen Du, and Liang Lin. Dynamic spatial-temporal representation learning for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(11):7169–7183, 2020.

[65] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

[66] Neema Davis, Gaurav Raina, and Krishna Jagannathan. A framework for end-to-end deep learning-based anomaly detection in transportation networks. *Transportation research interdisciplinary perspectives*, 5:100112, 2020.

[67] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*, pages 2181–2191, 2019.

[68] Hao Yang, Chenxi Liu, Meixin Zhu, Xuegang Ban, and Yinhai Wang. How fast you will drive? predicting speed of customized paths by deep neural network. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[69] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1):63–85, 2021.

[70] Maryam Shaygan, Collin Meese, Wanxin Li, Xiaoliang George Zhao, and Mark Nejad. Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities. *Transportation research part C: emerging technologies*, 145:103921, 2022.

[71] Boris Medina-Salgado, Eddy Sanchez-DelaCruz, Pilar Pozos-Parra, and Javier E Sierra. Urban traffic flow prediction techniques: A review. *Sustainable Computing: Informatics and Systems*, page 100739, 2022.

[72] Alireza Ermagun and David Levinson. Spatiotemporal traffic forecasting: review and proposed directions. *Transport Reviews*, 38(6):786–814, 2018.

[73] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[74] Qianru Zhang, Meng Zhang, Tinghuan Chen, Zhifei Sun, Yuzhe Ma, and Bei Yu. Recent advances in convolutional neural network acceleration. *Neurocomputing*, 323:37–51, 2019.

[75] Shuaichao Zhang, Lingxiao Zhou, Xiqun Chen, Lei Zhang, Li Li, and Meng Li. Network-wide traffic speed forecasting: 3d convolutional neural network with ensemble empirical mode decomposition. *Computer-Aided Civil and Infrastructure Engineering*, 35(10):1132–1147, 2020.

[76] Toon Bogaerts, Antonio D Masegosa, Juan S Angarita-Zapata, Enrique Onieva, and Peter Hellinckx. A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C: Emerging Technologies*, 112:62–77, 2020.

[77] Di Yang, Songjiang Li, Zhou Peng, Peng Wang, Junhui Wang, and Huamin Yang. Mf-cnn: Traffic flow prediction using convolutional neural network and multi-features fusion. *IEICE TRANSACTIONS on Information and Systems*, 102(8):1526–1536, 2019.

[78] Qiang Gao, Fan Zhou, Ting Zhong, Goce Trajcevski, Xin Yang, and Tianrui Li. Contextual spatio-temporal graph representation learning for reinforced human mobility mining. *Information Sciences*, 606:230–249, 2022.

[79] Yingming Li, Ming Yang, and Zhongfei Zhang. A survey of multi-view representation learning. *IEEE transactions on knowledge and data engineering*, 31(10):1863–1883, 2018.

[80] Kan Guo, Yongli Hu, Zhen Qian, Hao Liu, Ke Zhang, Yanfeng Sun, Junbin Gao, and Baocai Yin. Optimized graph convolution recurrent neural network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):1138–1149, 2020.

[81] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[82] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[83] Hao-Fan Yang, Tharam S Dillon, and Yi-Ping Phoebe Chen. Optimized structure of the traffic flow forecasting model with a deep learning approach. *IEEE transactions on neural networks and learning systems*, 28(10):2371–2381, 2016.

[84] Guillem Boquet, Antoni Morell, Javier Serrano, and Jose Lopez Vicario. A variational autoencoder solution for road traffic forecasting systems: Missing data imputation, dimension reduction, model selection and anomaly detection. *Transportation Research Part C: Emerging Technologies*, 115:102622, 2020.

[85] Alessio Sacco, Flavio Esposito, and Guido Marchetto. Restoring application traffic of latency-sensitive networked systems using adversarial autoencoders. *IEEE Transactions on Network and Service Management*, 19(3):2521–2535, 2022.

[86] Yuxuan Zhang, Senzhang Wang, Bing Chen, Jiannong Cao, and Zhiqiu Huang. Trafficgan: Network-scale deep traffic prediction with generative adversarial nets. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):219–230, 2019.

[87] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.

[88] Yi Wang, Changfeng Jing, Shishuo Xu, and Tao Guo. Attention based spatiotemporal graph attention networks for traffic flow forecasting. *Information Sciences*, 607:869–883, 2022.

[89] Binbing Liao, Jingqing Zhang, Ming Cai, Siliang Tang, Yifan Gao, Chao Wu, Shengwen Yang, Wenwu Zhu, Yike Guo, and Fei Wu. Dest-resnet: A deep spatiotemporal residual network for hotspot traffic speed prediction. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1883–1891, 2018.

[90] Weijia Zhang, Hao Liu, Yanchi Liu, Jingbo Zhou, and Hui Xiong. Semi-supervised hierarchical recurrent graph neural network for city-wide parking availability prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1186–1193, 2020.

[91] Soumya Suvra Ghosal, Abderrahman Bani, Amine Amrouss, and Issmail El Hallaoui. A deep learning approach to predict parking occupancy using cluster augmented learning method. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 581–586. IEEE, 2019.

[92] Jamie Arjona, MªPaz Linares, Josep Casanovas-Garcia, and Juan José Vázquez. Improving parking availability information using deep learning techniques. *Transportation Research Procedia*, 47:385–392, 2020.

[93] Yuecheng Rong, Zhimian Xu, Ruibo Yan, and Xu Ma. Du-parking: Spatio-temporal big data tells you realtime parking availability. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 646–654, 2018.

[94] Shuguan Yang, Wei Ma, Xidong Pi, and Sean Qian. A deep learning approach to real-time parking occupancy prediction in transportation networks incorporating multiple spatio-temporal data sources. *Transportation Research Part C: Emerging Technologies*, 107:248–265, 2019.

[95] J Arjona, Mari Paz Linares, and Josep Casanovas. A deep learning approach to real-time parking availability prediction for smart cities. In *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems*, pages 1–7, 2019.

[96] Neha Arora, James Cook, Ravi Kumar, Ivan Kuznetsov, Yechen Li, Huai-Jen Liang, Andrew Miller, Andrew Tomkins, Iveel Tsogsuren, and Yi Wang. Hard to park? estimating parking difficulty at scale. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2296–2304, 2019.

[97] Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. Attention based lstm for target dependent sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[98] Gang Liu and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[99] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han, and Yao Zhao. Ea-lstm: Evolutionary attention-based lstm for time series prediction. *Knowledge-Based Systems*, 181:104785, 2019.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[101] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[102] Yi Bin, Yang Yang, Fumin Shen, Ning Xie, Heng Tao Shen, and Xuelong Li. Describing video with attention-based bidirectional lstm. *IEEE transactions on cybernetics*, 49(7):2631–2641, 2018.

[103] Liang Zhang, Guangming Zhu, Lin Mei, Peiyi Shen, Syed Afaq Ali Shah, and Mohammed Bennamoun. Attention in convolutional lstm for gesture recognition. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1957–1966, 2018.

[104] Feng Zhang, Ningxuan Feng, Yani Liu, Cheng Yang, Jidong Zhai, Shuhao Zhang, Bingsheng He, Jiazao Lin, and Xiaoyong Du. Pewlstm: Periodic lstm with weather-aware gating mechanism for parking behavior prediction. In *IJCAI*, pages 4424–4430, 2020.

[105] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[106] Fan Liu, Xingshe Zhou, Jinli Cao, Zhu Wang, Tianben Wang, Hua Wang, and Yanchun Zhang. Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional lstm-cnn. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[107] Zhiwen Xiao, Xin Xu, Huanlai Xing, Shouxi Luo, Penglin Dai, and Dawei Zhan. Rtfn: A robust temporal feature network for time series classification. *Information Sciences*, 571:65–86, 2021.

[108] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[109] Wei Sun, Ethan Stoop, and Scott S Washburn. Evaluation of commercial truck parking detection for rest areas. *Transportation Research Record*, 2672(9):141–151, 2018.

[110] Filipe de Almeida Araujo Vital, Petros Ioannou, and Arti Gupta. Survey on intelligent truck parking: Issues and approaches. *IEEE Intelligent Transportation Systems Magazine*, 2020.

[111] Hao Yang, Jiarui Cai, Meixin Zhu, Chenxi Liu, and Yinhai Wang. Traffic-informed multi-camera sensing (tims) system based on vehicle re-identification. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[112] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[113] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018.

[114] Kyoungseok Han, Mooryong Choi, and Seibum B Choi. Estimation of the tire cornering stiffness as a road surface classification indicator using understeering characteristics. *IEEE Transactions on Vehicular Technology*, 67(8):6851–6860, 2018.

[115] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[116] Lalla Meriem Zouhal and Thierry Denoeux. An evidence-theoretic k-nn rule with parameter optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(2):263–271, 1998.

[117] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[118] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[119] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[120] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8(1):1–37, 2021.

[121] Phan Thanh Noi and Martin Kappas. Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery. *Sensors*, 18(1):18, 2017.

[122] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[123] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[124] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[126] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[127] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[128] Khaled F Hussain, Mahmoud Afifi, and Ghada Moussa. A comprehensive study of the effect of spatial resolution and color of digital images on vehicle classification. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1181–1190, 2018.

[129] Hossein Gholamalinejad and Hossein Khosravi. Vehicle classification using a real-time convolutional structure based on dwt pooling layer and se blocks. *Expert Systems with Applications*, 183:115420, 2021.

[130] Jian-Gang Wang and Lu-Bing Zhou. Traffic light recognition with high dynamic range imaging and deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(4):1341–1352, 2018.

[131] Shichao Zhou, Chenwei Deng, Zhengquan Piao, and Baojun Zhao. Few-shot traffic sign recognition with clustering inductive bias and random neural network. *Pattern Recognition*, 100:107160, 2020.

[132] Behnam Behroozpour, Phillip AM Sandborn, Ming C Wu, and Bernhard E Boser. Lidar system architectures and circuits. *IEEE Communications Magazine*, 55(10):135–142, 2017.

[133] Liang Mei and Mikkel Brydegaard. Continuous-wave differential absorption lidar. *Laser & Photonics Reviews*, 9(6):629–636, 2015.

[134] Rasul Torun, Mustafa M Bayer, Imam U Zaman, Jose E Velazco, and Ozdal Boyraz. Realization of multitone continuous wave lidar. *IEEE Photonics Journal*, 11(4):1–10, 2019.

[135] James D Spinhirne. Micro pulse lidar. *IEEE transactions on geoscience and remote sensing*, 31(1):48–55, 1993.

[136] Charles Harlow and Shiquan Peng. Automatic vehicle classification system with range sensors. *Transportation Research Part C: Emerging Technologies*, 9(4):231–247, 2001.

[137] Ho Lee and Benjamin Coifman. Side-fire lidar-based vehicle classification. *Transportation Research Record*, 2308(1):173–183, 2012.

[138] Ho Lee and Benjamin Coifman. Using lidar to validate the performance of vehicle classification stations. *Journal of Intelligent Transportation Systems*, 19(4):355–369, 2015.

[139] Jianqing Wu, Hao Xu, Yichen Zheng, Yongsheng Zhang, Bin Lv, and Zong Tian. Automatic vehicle classification using roadside lidar data. *Transportation Research Record*, 2673(6):153–164, 2019.

[140] Magdalena I Asborno, Collin G Burris, and Sarah Hernandez. Truck body-type classification using single-beam lidar sensors. *Transportation Research Record*, 2673(1):26–40, 2019.

[141] Zhenyao Zhang, Jianying Zheng, Hao Xu, and Xiang Wang. Vehicle detection and tracking in complex traffic circumstances with roadside lidar. *Transportation research record*, 2673(9):62–71, 2019.

[142] Junxuan Zhao, Hao Xu, Hongchao Liu, Jianqing Wu, Yichen Zheng, and Dayong Wu. Detection and tracking of pedestrians and vehicles using roadside lidar sensors. *Transportation research part C: emerging technologies*, 100:68–87, 2019.

[143] Olcay Sahin, Reza Vatani Nezafat, and Mecit Cetin. Methods for classification of truck trailers using side-fire light detection and ranging (lidar) data. *Journal of Intelligent Transportation Systems*, pages 1–13, 2020.

[144] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo De La Escalera. Birdnet: a 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3517–3523. IEEE, 2018.

[145] Takeo Miyasaka, Yoshihiro Ohama, and Yoshiki Ninomiya. Ego-motion estimation and moving object tracking using multi-layer lidar. In *2009 IEEE intelligent vehicles symposium*, pages 151–156. IEEE, 2009.

[146] Amara Dinesh Kumar, R Karthika, and KP Soman. Stereo camera and lidar sensor fusion-based collision warning system. *Advances in Computational Intelligence Techniques*, page 239, 2020.

[147] Pouria Babahajiani, Lixin Fan, Joni-Kristian Kämäräinen, and Moncef Gabbouj. Urban 3d segmentation and modelling from street view images and lidar point clouds. *Machine Vision and Applications*, 28(7):679–694, 2017.

[148] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[149] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[150] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[151] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[152] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[153] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 3645–3649. IEEE, 2017.

[154] Zheng Tang and Jenq-Neng Hwang. Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d. *IEEE Access*, 7:31934–31945, 2019.

[155] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490, 2019.

[156] Sultan Daud Khan and Habib Ullah. A survey of advances in vision-based vehicle re-identification. *Computer Vision and Image Understanding*, 182:50–63, 2019.

[157] Rene O Sanchez, Christopher Flores, Roberto Horowitz, Ram Rajagopal, and Pravin Varaiya. Vehicle re-identification using wireless magnetic sensors: Algorithm revision, modifications and performance analysis. In *Proceedings of 2011 IEEE International Conference on Vehicular Electronics and Safety*, pages 226–231. IEEE, 2011.

[158] Yegor Malinovskiy, Yao-Jan Wu, Yinhai Wang, and Un Kun Lee. Field experiments on bluetooth-based travel time data collection. Technical report, 2010.

[159] Yegor Malinovskiy, Un-Kun Lee, Yao-Jan Wu, and Yinhai Wang. Investigation of bluetooth-based travel time estimation error on a short corridor. Technical report, 2011.

[160] Michael Abbott-Jard, Harpal Shah, and Ashish Bhaskar. Empirical evaluation of bluetooth and wifi scanning for road transport. In *Australasian Transport Research Forum (ATRF), 36th*, page 14, 2013.

[161] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[162] Ryan A Kerekes, Thomas P Karnowski, Mike Kuhn, Michael R Moore, Brad Stinson, Ryan Tokola, Adam Anderson, and Jason M Vann. Vehicle classification and identification using multi-modal sensing and signal learning. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2017.

[163] Xinchen Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2016.

[164] Dominik Zapletal and Adam Herout. Vehicle re-identification for automatic video traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–31, 2016.

[165] Qi Zheng, Chao Liang, Wenhua Fang, Da Xiang, Xin Zhao, Chengping Ren, and Jun Chen. Car re-identification from large scale images using semantic attributes. In *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5. IEEE, 2015.

[166] Rogerio Schmidt Feris, Behjat Siddiquie, James Petterson, Yun Zhai, Ankur Datta, Lisa M Brown, and Sharath Pankanti. Large-scale vehicle detection, indexing, and search in urban surveillance videos. *IEEE Transactions on Multimedia*, 14(1):28–42, 2011.

[167] Zhongdao Wang, Luming Tang, Xihui Liu, Zhuliang Yao, Shuai Yi, Jing Shao, Junjie Yan, Shengjin Wang, Hongsheng Li, and Xiaogang Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 379–387, 2017.

[168] Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.

[169] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *European Conference on Computer Vision*, pages 869–884. Springer, 2016.

[170] Yantao Shen, Tong Xiao, Hongsheng Li, Shuai Yi, and Xiaogang Wang. Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1900–1909, 2017.

[171] Yi Zhou, Li Liu, and Ling Shao. Vehicle re-identification by deep hidden multi-view inference. *IEEE Transactions on Image Processing*, 27(7):3275–3287, 2018.

[172] Tsung-Wei Huang, Jiarui Cai, Hao Yang, Hung-Min Hsu, and Jenq-Neng Hwang. Multi-view vehicle re-identification using temporal attention model and metadata re-ranking. In *AI City Challenge Workshop, IEEE/CVF Computer Vision and Pattern Recognition (CVPR) Conference, Long Beach, California*, 2019.

[173] Kai Lv, Weijian Deng, Yunzhong Hou, Heming Du, Hao Sheng, Jianbin Jiao, and Liang Zheng. Vehicle reidentification with the location and time stamp. In *Proc. CVPR Workshops*, 2019.

[174] Hao Chen, Benoit Lagadec, and Francois Bremond. Partition and reunion: A two-branch neural network for vehicle re-identification. In *Proc. CVPR Workshops*, pages 184–192, 2019.

[175] Ming-Ching Chang, Jiayi Wei, Zheng-An Zhu, Yan-Ming Chen, Chan-Shuo Hu, Ming-Xiu Jiang, and Chen-Kuo Chiang. Ai city challenge 2019–city-scale video analytics for smart transportation. In *Proc. CVPR Workshops*, pages 99–108, 2019.

[176] Hung-Min Hsu, Tsung-Wei Huang, Gaoang Wang, Jiarui Cai, Zhichao Lei, and Jenq-Neng Hwang. Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In *AI City Challenge Workshop, IEEE/CVF Computer Vision and Pattern Recognition (CVPR) Conference, Long Beach, California*, 2019.

[177] Xiao Tan, Zhigang Wang, Minyue Jiang, Xipeng Yang, Jian Wang, Yuan Gao, Xiangbo Su, Xiaoqing Ye, Yuchen Yuan, Dongliang He, Shilei Wen, and Errui Ding. Multi-camera vehicle tracking and re-identification based on visual and spatial-temporal features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[178] Sen Qiu, Hongkai Zhao, Nan Jiang, Zhelong Wang, Long Liu, Yi An, Hongyu Zhao, Xin Miao, Ruichen Liu, and Giancarlo Fortino. Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges. *Information Fusion*, 80:241–265, 2022.

[179] Zishan Ahmad, Raghav Jindal, NS Mukuntha, Asif Ekbal, and Pushpak Bhattachharyya. Multi-modality helps in crisis management: An attention-based deep learning approach of leveraging text for image classification. *Expert Systems with Applications*, 195:116626, 2022.

[180] Kejing Xiao, Zhaopeng Qian, and Biao Qin. A survey of data representation for multi-modality event detection and evolution. *Applied Sciences*, 12(4):2204, 2022.

[181] Tongxue Zhou, Su Ruan, and Stéphane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3:100004, 2019.

[182] Xiangyun Liao, Yinling Qian, Yilong Chen, Xueying Xiong, Qiong Wang, and Pheng-Ann Heng. Mmtlnet: Multi-modality transfer learning network with adversarial training for 3d whole heart segmentation. *Computerized Medical Imaging and Graphics*, 85:101785, 2020.

[183] Jing Xu, Jiarui Ou, Chen Li, Zheng Zhu, Jian Li, Hailun Zhang, Junchen Chen, Bin Yi, Wu Zhu, Weiru Zhang, et al. Multi-modality data-driven analysis of diagnosis and treatment of psoriatic arthritis. *npj Digital Medicine*, 6(1):13, 2023.

[184] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[185] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[186] Suvarna Kadam and Vinay Vaidya. Review and analysis of zero, one and few shot learning approaches. In *International Conference on Intelligent Systems Design and Applications*, pages 100–112. Springer, 2018.

[187] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things journal*, 1(2):112–121, 2014.

[188] Wenjia Li, Houbing Song, and Feng Zeng. Policy-based secure and trustworthy sensing for internet of things in smart cities. *IEEE Internet of Things Journal*, 5(2):716–723, 2017.

[189] Christopher Neff, Matías Mendieta, Shrey Mohan, Mohammadreza Baharani, Samuel Rogers, and Hamed Tabkhi. Revamp 2 t: Real-time edge video analytics for multicamera privacy-aware pedestrian tracking. *IEEE Internet of Things Journal*, 7(4):2591–2602, 2019.

[190] Ruimin Ke, Yifan Zhuang, Ziyuan Pu, and Yinhai Wang. A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[191] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[192] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[193] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[194] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[195] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3038–3046, 2017.

[196] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 941–951, 2019.

[197] GeekAlexis. Fast mot, 2020.

[198] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.

[199] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3702–3712, 2019.

[200] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[201] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

[202] Junaid Ahmed Ansari, Sarthak Sharma, Anshuman Majumdar, J Krishna Murthy, and K Madhava Krishna. The earth ain't flat: Monocular reconstruction of vehicles on steep and graded roads from a moving camera. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8404–8410. IEEE, 2018.

[203] David Schrank, Tim Lomax, and Bill Eisele. 2017 urban mobility report. *Texas Transportation Institute,[ONLINE]. Available: http://mobility. tamu. edu/ums/report*, 2019.

[204] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[205] Ratnesh Kuma, Edwin Weill, Farzin Aghdasi, and Parthasarathy Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2019.

[206] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018.

[207] Meiping Yun and Wenwen Qin. Minimum sampling size of floating cars for urban link travel time distribution estimation. *Transportation Research Record*, 2673(3):24–43, 2019.

[208] Anis Koubaa, Adel Ammar, Anas Kanhouch, and Yasser AlHabashi. Cloud versus edge deployment strategies of real-time face recognition inference. *IEEE Transactions on Network Science and Engineering*, 9(1):143–160, 2021.

[209] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[210] Zhiyong Cui, Shen Zhang, Kristian C Henrickson, and Yinhai Wang. New progress of drive net: An e-science transportation platform for data sharing, visualization, modeling, and analysis. In *2016 IEEE International Smart Cities Conference (ISC2)*, pages 1–2. IEEE, 2016.

[211] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[212] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. Fastreid: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631*, 6(7):8, 2020.

[213] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification: A survey and outlook. *arXiv preprint arXiv:2001.04193*, 2020.

[214] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018.

[215] Dinh-Van Nguyen, Fawzi Nashashibi, Trung-Kien Dao, and Eric Castelli. Improving poor gps area localization for intelligent vehicles. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 417–421. IEEE, 2017.

[216] Moein Shakeri and Hong Zhang. Moving object detection in time-lapse or motion trigger image sequences using low-rank and invariant sparse decomposition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[217] Zi Yang and Lilian SC Pun-Cheng. Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image and Vision Computing*, 69:143–154, 2018.

[218] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2018.

[219] Gordon F. Newell. A simplified theory of kinematic waves in highway traffic, part I: General theory. *Transportation Research Part B: Methodological*, 27(4):281–287, 1993.

[220] Moshe Ben-Akiva, Michel Bierlaire, Haris Koutsopoulos, and Rabi Mishalani. Dynamit: a simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop*, pages 1–12. Delft The Netherlands, 1998.

[221] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

[222] J. W. C. Van Lint, S. P. Hoogendoorn, and Henk J. van Zuylen. Accurate freeway travel time prediction with state-space neural networks under missing data. *Transportation Research Part C: Emerging Technologies*, 13(5-6):347–369, 2005.

[223] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. In *6th International Workshop on Urban Computing (UrbComp 2017)*, 2016.

[224] Jiawei Wang, Ruixiang Chen, and Zhaocheng He. Traffic speed prediction for urban transportation network: A path based deep learning approach. *Transportation Research Part C: Emerging Technologies*, 100:372–385, 2019.

[225] Bingnan Jiang and Yunsi Fei. Vehicle speed prediction by two-level data driven models in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1793–1801, 2016.

[226] Lu Lin, Jianxin Li, Feng Chen, Jieping Ye, and Jinpeng Huai. Road traffic speed prediction: a probabilistic model fusing multi-source data. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1310–1323, 2018.

[227] Jane Macfarlane. When apps rule the road: The proliferation of navigation apps is causing traffic chaos. it's time to restore order. *IEEE Spectrum*, 56(10):22–27, 2019.

[228] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. AAAI, 2018.

[229] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[230] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[231] Seungjae Lee, Young-Ihn Lee, and Bumcheol Cho. Short-term travel speed prediction models in car navigation systems. *Journal of advanced transportation*, 40(2):122–139, 2006.

[232] Di Zang, Jiawei Ling, Zhihua Wei, Keshuang Tang, and Jiujun Cheng. Long-term traffic speed prediction based on multiscale spatio-temporal feature learning network. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3700–3709, 2018.

[233] Wei Liu, Hai Yang, and Yafeng Yin. Expirable parking reservations for managing morning commute with parking space constraints. *Transportation Research Part C: Emerging Technologies*, 44:185–201, 2014.

[234] Fangni Zhang, Wei Liu, Xiaolei Wang, and Hai Yang. Parking sharing problem with spatially distributed parking supplies. *Transportation Research Part C: Emerging Technologies*, 117:102676, 2020.

[235] Jacob Bertish, Austin Jarrett, William Krause, and Chetan Jaiswal. Truparking: Smart parking and the internet of things. In *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 203–209. IEEE, 2018.

[236] American Trucking Association et al. American trucking trends 2017. *Transport Topics*, 2017.

[237] Carol A Wrenn. *Can Autonomous Technology Reduce the Driver Shortage in the Commercial Trucking Industry*. PhD thesis, Doctoral dissertation, California Southern University, 2017.

[238] Michael J Sprung et al. Freight facts and figures 2017. 2018.

[239] Gongjun Yan, Weiming Yang, Danda B Rawat, and Stephan Olariu. Smartparking: A secure and intelligent parking system. *IEEE intelligent transportation systems magazine*, 3(1):18–30, 2011.

[240] Hongwei Wang and Wenbo He. A reservation-based smart parking system. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 690–695. IEEE, 2011.

[241] Yanfeng Geng and Christos G Cassandras. New "smart parking" system based on resource allocation and reservations. *IEEE Transactions on intelligent transportation systems*, 14(3):1129–1139, 2013.

[242] Yang Cheng, Steve Rau, Anupam Srivastava, Shen Li, Steven T Parker, Ernie Perry, Soyoung Ahn, and David A Noyce. Data archiving and performance measurement for a multistate truck parking information management system (tpims). In *International Conference on Transportation and Development 2020*, pages 251–260. American Society of Civil Engineers Reston, VA, 2020.

[243] Mehmet Emre Bayraktar, Farrukh Arif, Halit Ozen, and Gorm Tuxen. Smart parking-management system for commercial vehicle parking at public rest areas. *Journal of Transportation Engineering*, 141(5):04014094, 2015.

[244] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253, 2017.

[245] Bassel A Sadek, Elliot W Martin, and Susan A Shaheen. Forecasting truck parking using fourier transformations. *Journal of Transportation Engineering, Part A: Systems*, 146(8):05020006, 2020.

[246] Lili Zheng, Xue Xiao, Baofeng Sun, Duo Mei, and Bo Peng. Short-term parking demand prediction method based on variable prediction interval. *IEEE Access*, 8:58594–58602, 2020.

[247] Khademul Haque, Sabyasachee Mishra, Rajesh Paleti, Mihalis M Golias, Afrid A Sarker, and Karlis Pujats. Truck parking utilization analysis using gps data. *Journal of Transportation Engineering, Part A: Systems*, 143(9):04017045, 2017.

[248] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[249] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[250] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

[251] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. *arXiv preprint arXiv:1801.02143*, 2018.

[252] Hao Yang, Chenxi Liu, Christopher Gottsacker, Xuegang Ban, Chao Zhang, and Yinhai Wang. Cell-speed prediction neural network (cpnn): A deep learning approach for trip-based speed prediction. Technical report, 2019.

[253] Ray J Frank, Neil Davey, and Stephen P Hunt. Time series prediction and neural networks. *Journal of intelligent and robotic systems*, 31(1-3):91–103, 2001.

[254] Steven CH Hoi, Wei Liu, Michael R Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2072–2078. IEEE, 2006.

[255] Steven CH Hoi, Wei Liu, and Shih-Fu Chang. Semi-supervised distance metric learning for collaborative image retrieval and clustering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):1–26, 2010.

[256] Youru Li, Zhenfeng Zhu, Deqiang Kong, Meixiang Xu, and Yao Zhao. Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1004–1011, 2019.

[257] Meng Chen, Xiaohui Yu, and Yang Liu. Mpe: A mobility pattern embedding model for predicting next locations. *World Wide Web*, 22(6):2901–2920, 2019.

[258] Jianan Li, Yunchao Wei, Xiaodan Liang, Fang Zhao, Jianshu Li, Tingfa Xu, and Jiashi Feng. Deep attribute-preserving metric learning for natural language object retrieval. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 181–189, 2017.

[259] Brian Kulis et al. Metric learning: A survey. *Foundations and trends in machine learning*, 5(4):287–364, 2012.

[260] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019.

[261] Bo Geng, Dacheng Tao, and Chao Xu. Daml: Domain adaptation metric learning. *IEEE Transactions on Image Processing*, 20(10):2980–2989, 2011.

[262] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering*, pages 245–273. Springer, 2014.

[263] Aminu Da'u and Naomie Salim. Recommendation system based on deep learning methods: a systematic review and new directions. *Artificial Intelligence Review*, pages 1–40, 2019.

[264] Jiawei Zhang and S Yu Philip. *Broad Learning Through Fusions*. Springer, 2019.

[265] Jia Liu, Tianrui Li, Peng Xie, Shengdong Du, Fei Teng, and Xin Yang. Urban big data fusion based on deep learning: An overview. *Information Fusion*, 53:123–133, 2020.

[266] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. In *AAAI*, volume 18, pages 1–8, 2018.

[267] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[268] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[269] Hao Yang, Chenxi Liu, Yifan Zhuang, Wei Sun, Karthik Murthy, Ziyuan Pu, and Yinhai Wang. Truck parking pattern aggregation and availability prediction by deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[270] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[271] Zoubin Ghahramani and Geoffrey E Hinton. Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, 1996.

[272] Jianhua Guo, Wei Huang, and Billy M Williams. Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43:50–64, 2014.

[273] Anthony Stathopoulos and Matthew G Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies*, 11(2):121–135, 2003.

[274] Faraz Malik Awan, Yasir Saleem, Roberto Minerva, and Noel Crespi. A comparative analysis of machine/deep learning models for parking space availability prediction. *Sensors*, 20(1):322, 2020.

[275] Wei Shao, Yu Zhang, Bin Guo, Kai Qin, Jeffrey Chan, and Flora D Salim. Parking availability prediction with long short term memory model. In *International Conference on Green, Pervasive, and Cloud Computing*, pages 124–137. Springer, 2018.

[276] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Transferable representation learning with deep adaptation networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):3071–3085, 2018.

[277] Ruolin Zhang and Neda Masoud. A distributed algorithm for operating large-scale ridesourcing systems. *Transportation Research Part E: Logistics and Transportation Review*, 156:102487, 2021.

[278] Mingyu Pi, Hanbyul Yeon, Hyesook Son, and Yun Jang. Visual cause analytics for traffic congestion. *IEEE transactions on visualization and computer graphics*, 27(3):2186–2201, 2019.

[279] Nishant Kumar and Martin Raubal. Applications of deep learning in congestion detection, prediction and alleviation: A survey. *Transportation Research Part C: Emerging Technologies*, 133:103432, 2021.

[280] Daniel Stokols, Raymond W Novaco, Jeannette Stokols, and Joan Campbell. Traffic congestion, type a behavior, and stress. *Journal of Applied Psychology*, 63(4):467, 1978.

[281] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[282] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[283] Hyeong-Seok Jeon, Dong-Suk Kum, and Woo-Yeol Jeong. Traffic scene prediction via deep learning: Introduction of multi-channel occupancy grid map as a scene representation. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1496–1501. IEEE, 2018.

[284] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1234–1241, 2020.

[285] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, Xiaolei Ma, and Yinhai Wang. Learning traffic as a graph: A gated graph wavelet recurrent neural network for network-scale traffic prediction. *Transportation Research Part C: Emerging Technologies*, 115:102620, 2020.

[286] Wenqing Zheng, Qiangqiang Guo, Hao Yang, Peihao Wang, and Zhangyang Wang. Delayed propagation transformer: A universal computation engine towards practical control in cyber-physical systems. *Advances in Neural Information Processing Systems*, 34:12141–12153, 2021.

[287] Ting-Kuei Hu, Fernando Gama, Tianlong Chen, Wenqing Zheng, Atlas Wang, Alejandro R Ribeiro, and Brian M Sadler. Scalable perception-action-communication loops with convolutional and graph neural networks. *IEEE Transactions on Signal and Information Processing over Networks*, 2021.

[288] Rich Taylor and Richard Margiotta. Traffic congestion and reliability: Making the connection with operations: Part 2: Operations strategies. *Institute of Transportation Engineers. ITE Journal*, 76(3):30, 2006.

[289] Ranwa Al Mallah, Alejandro Quintero, and Bilal Farooq. Cooperative evaluation of the cause of urban traffic congestion via connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):59–67, 2019.

[290] Cambridge Systematics. Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation. Technical report, United States. Federal Highway Administration, 2005.

[291] Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Generative video transformer: Can objects be the words? In *International Conference on Machine Learning*, pages 11307–11318. PMLR, 2021.

[292] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[293] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[294] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[295] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[296] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021.

[297] Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *International Conference on Learning Representations*, 2021.

[298] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[299] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[300] Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7464–7471, 2020.

[301] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021.

[302] Jeevana Priya Inala, Yichen Yang, James Paulos, Yewen Pu, Osbert Bastani, Vijay Kumar, Martin Rinard, and Armando Solar-Lezama. Neurosymbolic transformers for multi-agent communication. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 13597–13608, 2020.

[303] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[304] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[305] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[306] Ling Cai, Krzysztof Janowicz, Gengchen Mai, Bo Yan, and Rui Zhu. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3):736–755, 2020.

[307] Balsam Alkouz and Zaher Al Aghbari. Snsjam: Road traffic analysis and prediction by fusing data from multiple social networks. *Information Processing & Management*, 57(1):102139, 2020.

[308] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[309] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[310] Hao Yang, Ruimin Ke, Zhiyong Cui, Yinhai Wang, and Karthik Murthy. Toward a real-time smart parking data management and prediction (spdmp) system by attributes representation learning. *International Journal of Intelligent Systems*, 2021.

[311] KyoHoon Jin, JeongA Wi, EunJu Lee, ShinJin Kang, SooKyun Kim, and YoungBin Kim. Trafficbert: Pre-trained model with large-scale data for long-range traffic flow forecasting. *Expert Systems with Applications*, 186:115738, 2021.

[312] Shuo Feng, Xingmin Wang, Haowei Sun, Yi Zhang, and Li Li. A better understanding of long-range temporal dependence of traffic flow time series. *Physica A: Statistical Mechanics and its Applications*, 492:639–650, 2018.

[313] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. Multi-stage attention spatial-temporal graph networks for traffic prediction. *Neurocomputing*, 428:42–53, 2021.

[314] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[315] Yipeng Liu, Haifeng Zheng, Xinxin Feng, and Zhonghui Chen. Short-term traffic flow prediction with conv-lstm. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2017.

[316] Sungbin Choi. Traffic map prediction using unet based deep convolutional neural network. *arXiv preprint arXiv:1912.05288*, 2019.

[317] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[318] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[319] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[320] Xu Chen, Yuanxing Zhang, Lun Du, Zheng Fang, Yi Ren, Kaigui Bian, and Kunqing Xie. Tssrgcn: Temporal spectral spatial retrieval graph convolutional network for traffic flow forecasting. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 954–959. IEEE, 2020.

[321] L Bai, L Yao, C Li, X Wang, and C Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *34th Conference on Neural Information Processing Systems*, 2020.

[322] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2020.

[323] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

[324] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[325] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[326] Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In *International Conference on Learning Representations*, 2021.

[327] Ning Lu, Nan Cheng, Ning Zhang, Xuemin Shen, and Jon W Mark. Connected vehicles: Solutions and challenges. *IEEE internet of things journal*, 1(4):289–299, 2014.

[328] Elisabeth Uhlemann. Time for autonomous vehicles to connect [connected vehicles]. *IEEE vehicular technology magazine*, 13(3):10–13, 2018.

[329] Radovan Miucic. *Connected vehicles: Intelligent transportation systems*. Springer, 2018.

[330] Roy Sumner, Bruce Eisenhart, John Baker, et al. Sae j2735 standard: applying the systems engineering process. Technical report, United States. Department of Transportation. Intelligent Transportation ..., 2013.

[331] Turner-Fairbank Highway Research Center Federal Highway Administration. Intersection safety, 2022.

[332] Biao Xu, Xuegang Jeff Ban, Yougang Bian, Wan Li, Jianqiang Wang, Shengbo Eben Li, and Keqiang Li. Cooperative method of traffic signal optimization and speed control of connected vehicles at isolated intersections. *IEEE Transactions on Intelligent Transportation Systems*, 20(4):1390–1403, 2018.

[333] Nengchao Lyu, Jiaqiang Wen, Zhicheng Duan, and Chaozhong Wu. Vehicle trajectory prediction and cut-in collision warning model in a connected vehicle environment. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[334] Meixin Zhu, Hao Frank Yang, Chenxi Liu, Ziyuan Pu, and Yinhai Wang. Real-time crash identification using connected electric vehicle operation data. *Accident Analysis & Prevention*, 173:106708, 2022.

[335] Yasir Ali, Michiel CJ Bliemer, Zuduo Zheng, and Md Mazharul Haque. Comparing the usefulness of real-time driving aids in a connected environment during mandatory and discretionary lane-changing manoeuvres. *Transportation research part C: emerging technologies*, 121:102871, 2020.

[336] Yasir Ali, Md Mazharul Haque, Zuduo Zheng, and Michiel CJ Bliemer. Stop or go decisions at the onset of yellow light in a connected environment: A hybrid approach of decision tree and panel mixed logit model. *Analytic Methods in Accident Research*, 31:100165, 2021.

[337] Nianfeng Wan, Ardalan Vahidi, and Andre Luckow. Optimal speed advisory for connected vehicles in arterial roads and the impact on mixed traffic. *Transportation Research Part C: Emerging Technologies*, 69:548–563, 2016.

[338] Evangelos Mintsis, Eleni I Vlahogianni, Evangelos Mitsakis, and Seckin Ozkul. Enhanced speed advice for connected vehicles in the proximity of signalized intersections. *European Transport Research Review*, 13(1):1–14, 2021.

[339] Eddie Curtis. Adaptive signal control technology. *US Federal Highway Administration*, 2017.

[340] Wade Odell, Katherine F Turnbull, et al. Automated and connected vehicle (av/cv) test bed to improve transit, bicycle, and pedestrian safety: concept of operations plan. 2017.

[341] Qiangqiang Guo, Li Li, and Xuegang Jeff Ban. Urban traffic signal control with connected and automated vehicles: A survey. *Transportation research part C: emerging technologies*, 101:313–334, 2019.

[342] Mashrur Chowdhury, Mizanur Rahman, Anjan Rayamajhi, Sakib Mahmud Khan, Mhafuzul Islam, Zadid Khan, and James Martin. Lessons learned from the real-world deployment of a connected vehicle testbed. *Transportation Research Record*, 2672(22):10–23, 2018.

[343] Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):485–494, 2011.

[344] Michael Luo. For exercise in new york futility, push button. *The New York Times*, 2004.

[345] Tom de Castella. Does pressing the pedestrian crossing button actually do anything? In *BBC News Magazine*. BBC, 2013.

[346] Govindarajan Vadakpat, Stephen F Smith, Zachary B Rubinstein, and M Bernardine Dias. Technology to make signalized intersections safer for pedestrians with disabilities. *Public Roads*, 84(4), 2021.

[347] Abigail L Cochran. Understanding the role of transportation-related social interaction in travel behavior and health: A qualitative study of adults with disabilities. *Journal of Transport & Health*, 19:100948, 2020.

[348] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1284–1293, 2019.

[349] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[350] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: the body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015.

[351] Xuan Zhou, Ruimin Ke, Hao Yang, and Chenxi Liu. When intelligent transportation systems sensing meets edge computing: Vision and challenges. *Applied Sciences*, 11(20):9680, 2021.

[352] Ahmed Hussein, Fernando Garcia, Jose Maria Armingol, and Cristina Olaverri-Monreal. P2v and v2p communication for pedestrian warning on the basis of autonomous vehicles. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2034–2039. IEEE, 2016.

[353] Sara Khosravi, Byungho Beak, K Larry Head, and Faisal Saleem. Assistive system to improve pedestrians' safety and mobility in a connected vehicle technology environment. *Transportation research record*, 2672(19):145–156, 2018.

[354] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[355] Yasir Ali, Anshuman Sharma, Md Mazharul Haque, Zuduo Zheng, and Mohammad Saifuzzaman. The impact of the connected environment on driving behavior and safety: A driving simulator study. *Accident Analysis & Prevention*, 144:105643, 2020.

[356] Ardalan Vahidi and Antonio Sciarretta. Energy saving potentials of connected and automated vehicles. *Transportation Research Part C: Emerging Technologies*, 95:822–843, 2018.

[357] Evangelos Mintsis, Eleni I Vlahogianni, and Evangelos Mitsakis. Dynamic eco-driving near signalized intersections: Systematic review and future research directions. *Journal of Transportation Engineering, Part A: Systems*, 146(4):04020018, 2020.

[358] G Curtis Herrick. The ntcip guide: National transportation communications for its protocol (version 2 draft). Technical report, 1999.

[359] Marina Kollmitz, Andreas Eitel, Andres Vasquez, and Wolfram Burgard. Deep 3D perception of people and their mobility aids. *Robotics and Autonomous Systems*, 114:29–40, 2019.

[360] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International journal of computer vision*, 4(2):127–139, 1990.

[361] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition. *arXiv preprint arXiv:2202.11233*, 2022.

[362] Qiangqiang Guo, Ohay Angah, Zhijun Liu, and Xuegang Jeff Ban. Hybrid deep reinforcement learning based eco-driving for low-level connected and automated vehicles along signalized corridors. *Transportation Research Part C: Emerging Technologies*, 124:102980, 2021.

[363] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[364] Grigorios D Konstantakopoulos, Sotiris P Gayialis, and Evripidis P Kechagias. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, pages 1–30, 2020.

[365] Kapileswar Nellore and Gerhard P Hancke. A survey on urban traffic management system using wireless sensor networks. *Sensors*, 16(2):157, 2016.

[366] Seri Oh, Stephen G Ritchie, and Cheol Oh. Real-time traffic measurement from single loop inductive signatures. *Transportation Research Record*, 1804(1):98–106, 2002.

[367] Hesham Rakha and Wang Zhang. Estimating traffic stream space mean speed and reliability from dual-and single-loop detectors. *Transportation Research Record*, 1925(1):38–47, 2005.

[368] Carlos Sun, Stephen G Ritchie, and Kevin Tsai. Algorithm development for derivation of section-related measures of traffic system performance using inductive loop detectors. *Transportation Research Record*, 1643(1):171–180, 1998.

[369] Savera Tanwir and Harry Perros. A survey of vbr video traffic models. *IEEE Communications Surveys & Tutorials*, 15(4):1778–1802, 2013.

[370] Hao Frank Yang. *Novel Traffic Sensing Using Multi-Camera Car Tracking and Re-Identification (MCCTRI)*. PhD thesis, 2020.

[371] Jie Guo, Bin Song, Ying He, Fei Richard Yu, and Mehdi Sookhak. A survey on compressed sensing in vehicular infotainment systems. *IEEE Communications Surveys & Tutorials*, 19(4):2662–2680, 2017.

[372] Joseph Mathew and PM Xavier. A survey on using wireless signals for road traffic detection. *International Journal of Research in Engineering and Technology*, 3(1), 2014.

[373] Luz-Elena Y Mimbela, Lawrence A Klein, et al. Summary of vehicle detection and surveillance technologies used in intelligent transportation systems. 2007.

[374] Dan Middleton and Ricky Parker. Vehicle detector evaluation. Technical report, Texas Transportation Institute, Texas A & M University System, 2002.

[375] Jose J Lamas-Seco, Paula M Castro, Adriana Dapena, and Francisco J Vazquez-Araujo. Multi-loop inductive sensor model for vehicle traffic applications. *Sensors and Actuators A: Physical*, 263:580–592, 2017.

[376] Dan R Middleton, Ricky Parker, and Ryan Longmire. Investigation of vehicle detector performance and atms interface. Technical report, Texas Transportation Institute, Texas A & M University System College ..., 2007.

[377] Amirali Jazayeri, Hongyuan Cai, Jiang Yu Zheng, and Mihran Tuceryan. Vehicle detection and tracking in car video based on motion model. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):583–595, 2011.

[378] Bin Tian, Ming Tang, and Fei-Yue Wang. Vehicle detection grammars with partial occlusion handling for traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 56:80–93, 2015.

[379] Honghai Liu, Shengyong Chen, and Naoyuki Kubota. Intelligent video systems and analytics: A survey. *IEEE Transactions on Industrial Informatics*, 9(3):1222–1233, 2013.

[380] Ciprian Adrian Corneanu, Marc Oliu Simón, Jeffrey F Cohn, and Sergio Escalera Guerrero. Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1548–1568, 2016.

[381] Ting Fu, Luis Miranda-Moreno, and Nicolas Saunier. Pedestrian crosswalk safety at nonsignalized crossings during nighttime: use of thermal video data and surrogate safety measures. *Transportation research record*, 2586(1):90–99, 2016.

[382] Malik Tubaishat, Peng Zhuang, Qi Qi, and Yi Shang. Wireless sensor networks in intelligent transportation systems. *Wireless communications and mobile computing*, 9(3):287–302, 2009.

[383] Marco Bottero, Bruno Dalla Chiara, and Francesco Paolo Deflorio. Wireless sensor networks for traffic monitoring in a logistic centre. *Transportation Research Part C: Emerging Technologies*, 26:99–124, 2013.

[384] Jesús Sánchez-Oro, David Fernández-López, Raúl Cabido, Antonio S Montemayor, and Juan José Pantrigo. Radar-based road-traffic monitoring in urban environments. *Digital Signal Processing*, 23(1):364–374, 2013.

[385] Erik Minge, Jerry Kotzenmacher, and Scott Peterson. Evaluation of non-intrusive technologies for traffic detection. Technical report, Minnesota Department of Transportation, Research Services Section Saint Paul, MN, 2010.

[386] Paul Foster, Zhenghong Sun, Jong Jin Park, and Benjamin Kuipers. Visagge: Visible angle grid for glass environments. In *2013 IEEE International Conference on Robotics and Automation*, pages 2213–2220. IEEE, 2013.

[387] Brandon Cochenour, Linda Mullen, and John Muth. A modulated pulse laser for underwater detection, ranging, imaging, and communications. In *Ocean Sensing and Monitoring IV*, volume 8372, page 83720S. International Society for Optics and Photonics, 2012.

[388] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[389] Chengqing Zong, Rui Xia, and Jiajun Zhang. *Text Data Mining*. Springer, 2021.

[390] Alireza Souri, Shafigheh Hosseinpour, and Amir Masoud Rahmani. Personality classification based on profiles of social networks' users and the five-factor model of personality. *Human-centric Computing and Information Sciences*, 8(1):1–15, 2018.

[391] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[392] Fazle Karim, Somshubra Majumdar, and Houshang Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:67718–67725, 2019.

[393] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[394] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[395] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[396] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021.

[397] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Junhao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. The devil is in classification: A simple framework for long-tail instance segmentation. In *European conference on computer vision*, pages 728–744. Springer, 2020.

[398] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48, 2010.

[399] Matthew Dixon. Sequence classification of the limit order book using recurrent neural networks. *Journal of computational science*, 24:277–286, 2018.

[400] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[401] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[402] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

[403] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[404] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[405] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[406] Renjie Wu and Eamonn J Keogh. Fastdtw is approximate and generally slower than the algorithm it approximates. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[407] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[408] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.