2021

Modeling, Analysis, and Application of Open Traffic Data for Train Delay Prediction

Jianqing Wu

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Modeling, Analysis, and Application of Open Traffic Data for Train Delay Prediction

Jianqing Wu

Supervisors:
Associate Professor Jun Shen
Dr. Luping Zhou
Dr. Chen Cai

This thesis is presented as part of the requirement for the conferral of the degree:
Doctor of Philosophy

University of Wollongong
School of Computing and Information Technology

March 2021

Abstract

Train delays are among the most complained events by the public communities in urban cities. Train delay prediction is critical for advanced traveler information systems (ATIS), which provides valuable information for enhancing the efficiency and effectiveness of intelligent transportation systems (ITS). However, the train delay prediction problem cannot be easily solved by modeling historical/static data from a single data source. A large amount of data is collected from sensor devices across the cyber-physical networks in the big data era. Multimodal transport management systems offer greater availability of various open data sources, such as General Transit Feed Specification (GTFS) static and real-time feeds. With the development of advanced machine learning techniques, a growing number of open data sources are playing more and more critical roles in planning and operation of transportation services. Recently**,** very few existing 'big data' methods meet the specific needs in railways.

This thesis emphasizes open traffic data modeling, analysis, and application for train delay prediction. More specifically, GTFS, with standard open-source data in both static and real-time formats, is being widely used in public transport planning and operation management. However, compared to other extensively studied data sources such as smart card data and GPS trajectory data, the GTFS data lacks proper investigation yet. Utilization of the GTFS data is challenging for both transport planners and researchers due to its difficulty and complexity of understanding, processing, and leveraging the raw data. This thesis proposes a GTFS data acquisition and processing framework to offer an efficient and effective benchmark tool for converting and fusing the GTFS data to a ready-to-use format. The contribution of this new framework will render great potential for wider applications and deeper researches. Secondly, we demonstrate a novel data-driven Primary Delay Prediction System (PDPS) framework,

which combines General Transit Feed Specification (GTFS), Critical Point Search (CPS), and deep learning models to leverage the data fusion. Different from existing researches, we present a hybrid deep learning solution for predicting multi-step train delays. Our solution uses Long Short-Term Memory (LSTM) to generate the forecasts for train delays based on the delay causes, run-time delay, and dwell time delay. The LSTM tackles the tasks for long-term predictions of running time and dwell time with univariate and multivariate time series data, respectively. We present the performance of the standard LSTM and its variants applied in a novel architecture. Experimental results indicate that the proposed method has superior accuracy for long-term delay prediction.

Lastly, as the first work in this area in the world, we apply a real entropy for measuring the time series regularity and find approximated potential predictability on train delays. Different from the existing train delay studies that had strived to explore sophisticated algorithms, this study focuses on finding the bound of improvements on predicting multi-scenario train delays with different machine learning methods. Motivated by the observation of deep learning methods failing to improve the prediction performance if the delay occurs rarely, we present a novel augmented machine learning approach to improve the overall prediction accuracy further. Our solution proposes a rule-driven automation (RDA) method, including a delay status labeling (DSL) algorithm, and the resilience of section (RSE) and resilience of station (RST) indicators to generate the forecast for train delays. The experiment results demonstrate that the Random Forest based implementation of our RDA method (RF-RDA) can significantly improve the generalization ability of multivariate multi-step forecast models for multi-scenario train delay prediction. The proposed solution surpasses state-of-art baselines based on real-world traffic datasets, which treat various real-time delays differently. Even when the predictability of conventional deep learning methods decreases, the performance of our

method is still acceptable for practical use to provide accurate forecasts.

Acknowledgments

I would like to say that pursuing a Ph.D. was an unforgettable journey. I wish to thank lots of people very much for the help from my supervisor and colleagues, as well as the support of my family. Finally, I was able to complete this research.

I would like to express the deepest appreciation to the principal supervisor, A/Prof. Jun Shen, during my years at Wollongong. During the Ph.D. process, he always encouraged me to express my own ideas and put the ideas down in writing. I feel extremely fortunate to have Jun, who mentored me and kept me on the right track, providing support throughout my Ph.D. He is one of the most influential people in my life and has helped me tremendously through my Ph.D. journey.

My appreciation extends to my co-supervisor, Dr. Luping Zhou, who is at the University of Sydney, and my associate supervisor, Dr. Chen Cai, who is at Data61 of CSIRO, for their guidance and great assistance to my research.

I would also like to thank A/Prof. Yihui Wang (BJTU), Dr. Ruimin Li (Sydney Trains), and Dr. Bo Du (SMART/CME at UOW) for offering continuous support and valuable insights. I wish to express my thanks to my peers in the Ph.D. study, Dr. Geng Sun, Dr. Huaming Chen, Dr. Jiayi Yang, Mr. Fucun Li, Mr. Jiayi Lin, Miss. Yunshu Zhu, Mr. Ting Song, Mr. Fei Xie, Mr. Zhexuan Zhou, Mr. Bo Wang, Mr. Zengyang Gong, Dr. Qiang Wu, Dr. Zhongyuan Yao, Dr. Yang Li, and Dr. Lei Qi, for their great support and friendships during my study.

I gratefully acknowledge the funding received towards my Ph.D. from the China Scholarship Council (CSC) and the University of Wollongong (UOW**)** Joint Postgraduate Scholarships.

I would like to thank my family: my grandpa, Xiaoting Wu, and grandma, Yueting Li. Special thanks to my mother, Pinfang Lin, as a single mother who gave her child

the best education.

Thanks to my lovely 5-month-old daughter, Jiaxin Wu. Her smile and laugh always cheer me up. Thanks to my beautiful wife, Dandan Xu. She has been taking care of the baby during the pandemic. And thanks for always supporting and understanding.

Jianqing Wu

Wollongong, Australia

March 2021

Certification

I, Jianqing Wu, declare that this thesis submitted in fulfilment of the requirements for the conferral of the degree Doctor of Philosophy, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

_____

Jianqing Wu

4th March 2021

List of Names or Abbreviations

ITS	Intelligent Transportation System
ATIS	Advanced traveler information systems
IoT	Internet of Things
MaaS	Mobility as a Service
GTFS	General Transit Feed Specification
NSW	New South Wales
AVL	Automatic Vehicle Location
OD	Origin Destination
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GNN	Graph Neural Networks
CPS	Critical Point Search
TfNSW	Transport for NSW
RDA	Rule-Driven Automation
DSL	Delay Status Labeling
RSE	Resilience of a Section
RST	Resilience of a Station
R^2	R-squared
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
ME	Maximum Residual Error
MAPE	Mean Absolute Percentage Error
SMAPE	Symmetric Mean Absolute Percentage Error
RRSE	Root Relative Squared Error

Table of Contents

List of Figures

List of Tables

List of Publication

Journal Papers

J-1. **J. Wu**, Y. Wang, B. Du, Q. Wu, Y. Zhai, J. Shen, L. Zhou, C. Cai, W. Wei, and Q. Zhou. "The Bounds of Improvements Towards Real-time Forecast of Multi-Scenario Train Delays," *Accepted by IEEE Transactions on Intelligent Transportation Systems*, 2021, doi: 10.1109/TITS.2021.3099031.

J-2. **J. Wu**, B. Du, Z. Gong, Q. Wu, J. Shen, L. Zhou, C. Cai. "A GTFS Data Acquisition and Processing Framework and its Application to Train Delay Prediction," *Submitted to Traval Behaviour and Society Journal*.

J-3. **J. Wu**, Q. Wu, J. Shen, and C. Cai, "Towards Attention-Based Convolutional Long Short-Term Memory for Travel Time Prediction of Bus Journeys," *Sensors*, vol. 20, no. 12, p. 3354, 2020, doi: 10.3390/s20123354.

Conference Papers

C-1. **J. Wu**, Q. Wu, L. Zhou, C. Cai, B. Du, Y. Zhai, W. Wei, J. Shen, and Q. Zhou. "Multivariate Multi-Step Train Delay Forecasting: A Hybrid LSTM-CPS Solution," *Transportation Research Board (TRB) 100th Annual Meeting*, 2021. https://trid.trb.org/view/1759773

C-2. **J. Wu**, L. Zhou, C. Cai, F. Dong, J. Shen, and G. Sun, "Towards a General Prediction System for the Primary Delay in Urban Railways," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019: IEEE, pp. 3482-3487.

C-3. **J. Wu**, L. Zhou, C. Cai, J. Shen, S. K. Lau, and J. Yong, "Data Fusion for MaaS: Opportunities and Challenges," in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2018: IEEE, pp. 642-647.

Chapter 1

Introduction

1.1 Research Background

The world has been urbanizing rapidly. The urbanized population is expected to reach 68% by 2050 [1]. This trend has led to an increasing number of people living in urban areas. As traffic congestions are getting worse, the current road infrastructure is no longer a cost-effective solution. Besides, the enhancement of the existing infrastructure may bring many expenses in terms of finance and labor, and at the same time, it requires more time and may cause more traffic congestions. One of the more promising methods of relieving congestion is through the design and implementation of new technology in the Intelligent Transportation System (ITS). Advanced traveler information systems (ATIS) and advanced traffic management systems (ATMS) are two significant parts of the ITS.

Moreover, high-capacity public transport plays an essential role in meeting the growing demand for urban transport. Relying on single-mode transportation, we cannot use the existing transport capacity effectively. Multimodal transport or combined transport is an efficient way to access destinations, activities, services, and goods. It is a combination of two or more transport modes for a trip, and one transfer or more transfers must be made among different types of transport modes, such as private transport and public transport [2]. However, the multimodal transport system exhibits increased complexity in the modern world. In order to seek multimodal transport innovation solutions that improve services and operations, the government, transport service planners, and operators in each city release a wide range of traffic data via an open data

platform. Developers and researchers can easily collect the data, and then develop tools and conduct analysis to compare outcomes and results from different studies.

In today's world, the railway plays a vital role in multimodal transport systems. Train delays adversely affect punctuality, reliability, and quality of experiences (QoE). They also increase the travel time of passengers. Once a train delay occurs, various strategies may be applied for overcoming the subsequent delays, such as adjusting the operational speed, dwell time, or even schedule. To prevent delays, a traffic control center relies on various models to predict potential delays in the system. For example, Stockholmstag in Sweden has used a prediction model to visualize the entire commuter train system two hours ahead [3]. Train dispatchers have an increasing demand for using data-driven models to acquire helpful information supporting both long-term and short-term decision-makings. Academic researchers and practical operators or modelers in the rail transport industry have become increasingly interested in developing decision support tools by embracing the latest computation methods.

Accurate estimation of travel information is crucial to ITS. The approaches to studying the impact of various travel information in multimodal transport systems (MTS) can be divided into knowledge-driven, model-driven, and data-driven methods [4]. With the smart city concept development, a wide range of Internet of Things *(*IoT) sensor devices generates massive and complex traffic data every day. Cities can develop and implement intelligent analysis systems to monitor public transit performance by modeling, analyzing, and interpreting real-world data. Unlike simulated data, real-world data is observational data from varied sources.

In the Internet of Things (IoT) paradigm, consumers and service providers can easily connect and communicate through smart devices, intelligent software services, and scalable cloud computing systems. Physical objects surrounding us have access to the

network via wireless telecommunication technologies [5, 6]. With the rapid development of the Internet of Things (IoT), cloud and edge computing, Big Data analytics (BDA), and artificial intelligence (AI) technologies, an increasing number of AI-based systems have been implemented to solve practical problems in various fields, such as business, healthcare, biology, education, and transportation. A large volume of data is generated through IoT devices, and they are stored on pervasive cloud platforms. However, such volume, velocity, and data variety cannot be processed by conventional data processing algorithms and tools. BDA and AI applications play vital roles in handling the IoT-based sensor data to provide better services for human production activities and daily life needs. Currently, BDA and AI have increasingly attracted the attention of practitioners and researchers in aspects of rail transportation engineering [7]. For instance, studying and analyzing delay propagation behavior is essential for developing such practical applications.

1.2 Research Motivation

1.2.1 Fusion of Open Traffic Data

Over the last decade, multimodal transport has become an increasingly popular and influential topic in smart cities. With the wide adoption of multimodal transport systems, transport planners can integrate various mobility services into one unified platform with effective coordination and cooperation among multiple modes of transport, such as Mobility-as-a-Service (MaaS) [8]. It is an innovative solution to allow all transport providers to run as a single virtual organization, which offers available mobility services based on the travelers' actual and real-time needs. Travelers can use a mobile app or a smart card to pay for their trips at once. Such integrated transport services will generate massive demand for the pre-acquisition of traffic information, such as travel time, waiting time, and service delays.

As one of the crucial challenges in public transport operation, service delay affects people's travel experience and the level of service of public transport system, which causes additional costs, such as the economic cost related to the increased passenger travel time [9]. Facing significant delays in public transport services, travelers may switch to alternative transport modes to continue the trip, and such travel behavior change may decrease the patronage of public transport services. As one of the essential components of Intelligent Transportation Systems (ITS), Advanced Traveler Information System (ATIS) provides real-time traffic information to travelers. Evidence indicated that good quality real-time information leads to increased rider satisfaction and ridership [10-12].

In recent years, open traffic data is playing an increasingly important role. The open traffic data refers to that traffic data is published on an open-source platform. Developers, entrepreneurs, and data analysts can collect and use the data to create innovative solutions for travelers. For example, the multimodal transport ecosystem of New South Wales (NSW) in Australia provides General Transit Feed Specification (GTFS) static and real-time data to transport planners and operators [13]. The GTFS data can offer real observations as multivariate time-series data. Based on the real-time traffic information, travelers can plan their trips easily, and likewise, transport operators can schedule and coordinate transport services to offer better services.

Nowadays, with the broad deployment of smart city infrastructures, a large number of Internet of Things (IoT) devices also generate massive data. To facilitate data-driven research, various data sources such as automatic fare collection (AFC) data, automatic vehicle location (AVL), and automatic passenger counting (APC) are widely used. For instance, the AFC data collected from smart cards can be employed to derive an accurate origin-destination (OD) matrix to represent travel demand. The AVL data shows a detailed representation of supply on corresponding trips with planned time and vehicle

location. The APC data provides accurate counts of ridership on vehicles. However, limited access to those datasets becomes a major obstacle for widely usage of the data.

Different from aforementioned data sources, GTFS data received rare investigation in literature studies. Moreover, due to the large amount of GTFS data, it is very difficult and complex to process the data, and fuse the static and real-time data for further usage. Therefore, as the first step, how to acquire and process the large-amount and complicated GTFS raw data to support further development of data-driven models and algorithms remains an essential but critical issue. To the best of our knowledge, limited existing research had ever made maneuvers to tackle this foundation issue.

1.2.2 Primary Delay Prediction System Framework for Long-Term Prediction

In today's world, train transport has played a leading role in public transportation as an essential component of mobility-as-a-service (MaaS) for travelers from one city location to another [8]. The train delay causes an increase in travel time to complete passenger journeys. It is not only frustrating but also disruptive. In the 5G era, cloud computing platforms are offering timely analytics over big data to meet the needs of clients for high-quality analysis and prediction. Train dispatchers have an increasing demand for using data-driven models to acquire useful information in long-term, short-term, and real-time decision making. Researchers and experts in the field of rail transport have become increasingly interested in developing decision support tools by embracing the latest computation methods.

Moreover, existing models applied to train delay analysis have three categories: delay distributions, delay propagation, and timetable rescheduling. Delay prediction and recovery are two significant and challenging issues in delay propagation [14]. In rail networks, delay propagation refers to that once a delay occurs at one station or one line, it often causes consequent delays in multiple stations or multiple lines and even leads to

the interruption of the entire railway network. If we predict the single primary delay, we can prevent delays in advance. For example, let us assume that a train departs from Station A and passes through Stations B and C. When Station A has a 120-second delay, followed by Station B with a 130-second delay, Station C has a 140-second delay. It is worth noting that if Station A's delay is alleviated or avoided, Station B and C may produce a delay of less than 30 seconds due to the nature of train delay propagation. Under such a circumstance, it is said that the train passes through stations A, B, and C on time since the 30-second delay is allowed for on-time performance.

Current train delay prediction systems still use static rules, which are built and operated by domain experts based on classical statistics. Establishing a practical and accurate delay prediction system could provide useful information to significantly improve traffic management and dispatching processes underlying passenger information systems, freight tracking systems, nominal timetable planning, delay management [15]. However, most of the delay and prediction information obtained from the data could be useless for adjusting timetables to schedule real-time trains. This is because if a train arrives at or departs from a station more than 30 seconds or 60 seconds later than the scheduled time, it is considered as a delay. The often-occurred small delays need to be studied by data analysts again, which is very time-consuming. Additionally, there is a lack of traceback for the causality of predicted data. Thus, to establish an automated train delay prediction system, it should contain two major components: an AI-based component to deal with big data and an expert system-based component to emulate the ability of human experts to reason the data causality.

As railway IoT systems generate a large amount of data every day, it is feasible to apply the concepts of machine learning and deep learning to establish data-driven models of train delay prediction. Yaghini et al. developed an artificial neural network

(ANN) model to estimate train delay based on historical data [16]. Pongnumkul et al. proposed two algorithms to predict train arrival times at three train stations. The experiment was based on a moving average of historical travel times and the travel times of k-nearest neighbors (k-NN) of the last known arrival time [17]. Oneto et al. implemented shallow and deep Extreme Learning Machines (ELM) for forecasting train delays of a large-scale network with weather information on the Apache Spark [18]. In the follow-up work, Oneto et al.evaluated the system on six months of train movement data from the entire Italian railway network [15].

Train delay prediction has been explored more and more, along with open data becoming increasingly available. Transit agencies have published open datasets to remove barriers for information-sharing among developers, researchers, and data analytic organizations. For example, GTFS provided detailed schedules and associated geographic information in an open data format [19]. Even though the initial aim of GTFS is to offer a unified data format for developing user-focused route and schedule planning software, it has also become a critical data source for researches on intelligent railway systems [20]. However, there are still many issues with the direct use of these data for the prediction of a train delay, such as a large amount of data duplication, inconsistent information, missing data, and lack of practical information integration.

In practice, to make a set of data-driven dispatching decisions to minimize the total delay of the trains, a train dispatcher needs to understand delay distributions and delay propagation patterns. For a railway network, decision-making in the train timetable adjustment is another thought-provoking task. Although model-based methods have an excellent performance on train delays in academic experiments, in most of the real scenarios, it has been indicated that completing a rescheduling process tends to be very labor-intensive and time-consuming because rail operators have to perform the analysis

of a massive amount of data collected from IoT devices. Thus, we need to move towards a computer-aided forecast system that emulates the knowledge and expertise of dispatchers to analyze such big data automatically.

For implementing a delay prediction application, current studies always strive to find an algorithm that can reach maximum prediction accuracy on the dataset. However, train delays consist of a certain amount of randomness (or irregularity, e.g., unexpected event) and a certain degree of regularity (e.g., morning and evening peaks). The single-handed predictive algorithm cannot predict the outcome of a random event.

1.2.3 Multi-Scenario Real-Time Train Delay Forecast

The existing methods and models being applied to train delay analysis can be classified into three categories: delay distributions, delay propagation, and train rescheduling. Delay prediction is one of the most significant and challenging research problems in delay propagation [14]. To develop a set of data-driven dispatching decisions to minimize the total delay of train services, train dispatchers need to understand delay distributions and delay propagation patterns. Although model-based prediction methods have shown excellent performances on train delays in various experiments, evidences have also showed that train rescheduling in real-life practice was very labor-intensive and time-consuming due to the massive amount of data collected from various Internet of Things (IoT) devices [7, 15], [21, 22]. Thus, we need to move towards a computer-aided forecast system that emulates the knowledge and expertise of dispatchers to analyze such big data automatically.

In the 5G era, advanced IoT technology can capture data in real-time, and cloud computing platforms offer timely analytics over big data to meet clients' needs for high-quality analysis and prediction. In the meanwhile, the punctuality performance of train services has been improved significantly in the past years. For example, the punctuality

performance of the Sydney rail network is between 88.7% and 93.3% from July 2017 to June 2018, within just one year [23]. Most of the historical data is smooth, which is utilized to build a data-driven predictive model. The model can be used for on-time forecast, but it will easily fail in the train delay forecast (see **Appendix 1**). Furthermore, when the train will be delayed or when it will be on time in real-time forecast is still unknown. Therefore, an effective manner to extract and structure data on demand is needed.

Current research strives to find an algorithm that can reach the maximum accuracy of delay prediction. However, train delays consist of a certain amount of randomness (or irregularity, e.g., unexpected event) and a certain degree of regularity (e.g., morning and evening peaks). The single-handed predictive algorithm cannot predict the outcome of a random event. Consequently, the data-driven predictive models entirely rely on the availability of 'good-quality' historical data. An effective manner is needed to extract and structure data on demand and to query the data required by the models to deal with various scenarios. For example, if there is a five-minute delay when a train departs from the current station, to predict the running time to or dwell time at the next station, the historical observations at the current station are needed, which include three types of data, the delays affected by the previous trip, the delays affected by the previous station, and the delays affected by both the previous trip and the previous station.

Moreover, the resilience of the railway transportation system refers to the ability of a railway system to resist, absorb, accommodate, and even quickly recover from disruptions or disasters [24]. In order to forecast train delays under various situations, a comprehensive and reliable predictor should consider the resilience of a given trip, such as the ex-post effects of train delays.

1.3 Research Objectives and Contributions

From the background and motivation of this research, the thesis focuses on the following research issues: 1) Fusion of Open Traffic Data, 2) Primary Delay Prediction System Framework for Long-Term Prediction of Train Delays, and 3) Multi-Scenario Real-Time Train Delay Forecasting. In the following sections, we will elaborate on the research issues and their corresponding contributions:

For the first research issue, we aim to reinvigorate the GTFS data by proposing a general data acquisition and preparation (DAP) framework as a foundation to support a diversity of data-driven studies across a broad research scope. In short, the main contributions are highlighted herein:

- A GTFS DAP framework is designed to acquire, process and fuse GTFS static and real-time data from cloud-based live feeds;
- Based on the GTFS DAP framework, a data cleaning and aggregation tool is developed to generate benchmark datasets for real-time delay prediction and long-term delay prediction, respectively;
- The proposed GTFS DAP framework and generated benchmark dataset are utilized for time series prediction using multivariate multistep Long Short-Term Memory (MM-LSTM). Numerical experiments are conducted based on a case study of Sydney trains with promising results, which validate effectiveness and benefits of the proposed framework.

To address the second research issue, we target bridging the aforementioned research gaps and propose a data-driven system framework. It includes the GTFS data pre-processing tool, the critical point search (CPS) algorithm, and deep learning models. The combination is not only to deal with big data in railways but also to achieve causality for delay event classifications. The main contributions are demonstrated as follows.

- A data-driven Primary Delay Prediction System (PDPS) framework is designed to predict primary delays using GTFS static and real-time data.
- A critical point search algorithm is proposed to classify data efficiently and reasonably.
- For evaluation, serval state-of-the-art deep learning models are applied to an open dataset from the GTFS data pre-processing tool, and the experimental results illustrate the effectiveness of the proposed system framework.

To address the third research issue considered in this thesis, we propose a comprehensive architecture of deep learning methodology to predict train delays based on temporal-correlated entropy, delay causes, resilience factors, and data label algorithm. The contributions are summarized as follows.

- A real entropy is proposed to measure the degree of predictability on the train delay time-series datasets.
- Interpretations of run-time delays and dwell delays are accurately distinguished as the causes of the train delay. Both running times and dwell times are predicted first instead of the delay prediction directly.
- The deep learning models are performed on train time series data, and the experimental results illustrate the effectiveness of the proposed network structure. These outputs are applied to classify predicted delay data.
- Based on the foundations laid in this thesis, we also discuss how to build a robust long-term forecasting delay system.
- A novel method, namely rule-driven automation (RDA), is implemented to reconstruct a dataset, including a delay status labeling (DSL) and resilience factors. The DSL algorithm is applied to labeling historical data. The resilience of

section (RSE) and resilience of station (RST) are introduced in the multivariate multi-step forecasting models.

- The detailed steps of feature generation for multivariate regression are presented. Most conventional and baseline predictive models, with or without RDA, are performed on train time series data, and the comprehensive experimental results illustrate the effectiveness of the proposed RDA-based improvements for solving the train delay forecasting in real-time scenarios.

Due to our main purpose of evaluating train delay prediction models, we leave an attention-LSTM study in Appendix 2 for a record of how to learn travel time, which related to the train delay.

Figure 1.1 shows the structure of this thesis and offers an overview of the above-described original contributions to knowledge from the research. The thesis has six chapters, including this introduction. Chapter 2 presents a survey of related works, including multimodal transport, data fusion, entropy, train delay prediction, machine learning predictive model. Chapter 3 develops a novel DAP framework for the fusion of GTFS static and real-time data. Chapter 4 proposes a general prediction system for primary delays using univariate and multivariate analysis. Chapter 5 focuses on finding the bound of improvements in the real-time forecast of multi-scenario train delays with various machine learning methods. There are three components in the prediction system, namely real entropy, DSL algorithm, and RSE, and RST factors. Chapter 6 demonstrates a summary of main findings and contributions, implications, and future works.

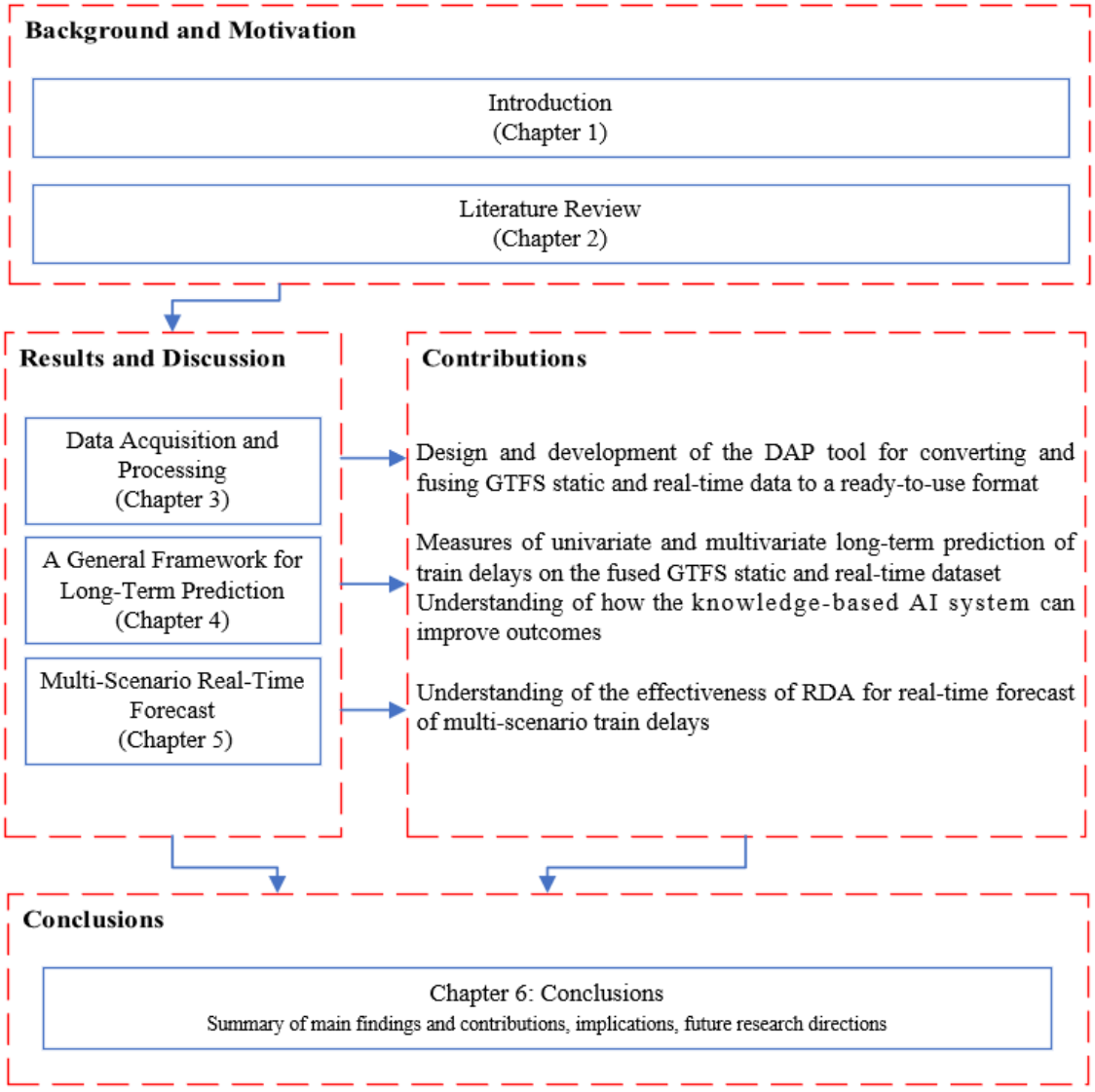


Figure 1.1 Thesis Structure

Chapter 2

Literature Review

This chapter is demonstrated to review related works to this thesis, including works on multimodal transport, data fusion on GTSF static and real-time, entropy for measuring uncertainty, train delay prediction, and machine learning predictive model.

2.1 Open Traffic Data

The sustainable development of smart cities requires reliable and efficient transportation systems [25]. Internet of Things (IoT) can be applied with the existing infrastructure and service networks for the design of the transportation systems, such as software-defined networks and communication technologies [26-28]. IoT-based Intelligent transportation system (IoT-ITS) can be classified into four main fields: Advanced Traveler Information System (ATIS), Advanced Public Transportation System (APTS), Advanced Traffic Management System (ATMS), and Emergency Management System (EMS) [28]. Transportation systems are shifting from conventional technology-driven systems to more powerful multifunctional data-driven ITS [29-31]. Massive traffic sensor data gathered by various sensors is vital for informed, scientific decision-making processes in traffic operation, pavement design, and transportation planning [32]. Data analytics in ITS consider essential factors that influence decision-making processes, such as travel time or traffic congestion of public transport services [33-34]. The fusion of traffic data from multiple sources produces a better understanding of the observations for reaching a better inference in ITS [35-38].

Multimodality is essential to reduce dependence on cars and more sustainable

transportation behavior [39]. Most adult populations use multimodal transport during weekly trips, while few people travel unimodal, such as only a single mode of transport for a trip [40]. A multimodal transport management system, such as MaaS, is an innovative solution to achieving collaboration and integration among transport providers [41]. The key concept behind MaaS is to provide users with mobility solutions to meet their diverse travel needs [42]. Accordingly, public and private sectors should have an integrated view on the future of mobility to understand the impacts of transportation mode changes [43]. All transport providers run as a single unified organization to offer a single mobility service based on traveler's needs [44]. However, MaaS is not adequately defined yet. It is currently reasonably understood as, "Mobility as a Service is a user-centric, intelligent mobility distribution model in which all mobility service providers offerings are aggregated by a sole mobility provider, the MaaS provider, and supplied to users through a single digital platform." as specified in [45]. An extended version of MaaS uses SaaS (Software as a Service) and the operator interface and management system, Collaboration-as-a-Service (CaaS), to support the ecosystem [46].

Furthermore, MaaS providers typically share high-quality multimodal transport data through an open data platform. For example, the MaaS ecosystem of New South Wales (NSW) of Australia contains GTFS static, GTFS real-time, General Bikeshare Feed Specification (GBFS), and real-time Vehicle information [13]. Open data standards such as GTFS static and GTFS real-time provide common formats for multimodal transport data, which offers real observations as multivariate time-series data.

Smart card data are collected by the AFC system that implies valuable knowledge about human travel patterns [47, 48]. The availability of smart card data is of vital importance for answering various research questions in ITS, such as the estimations of the OD distribution, the demand forecasting, the citywide crowd flows forecast, and the

bus bunching identification and prediction [47] [49-52]. Specifically, travel demand obtained in these studies comes from archived smart card transaction records with a data provision based on GTFS schedules, which generate highly detailed representations for bus network microsimulation [53]. AFC data with GTFS instead of AVL data can accelerate the search for generating O-D matrices [54, 55]. Smart card transactional data from APC systems and GTFS can be fused to visualize public transit use [56]. Since GTFS is more accessible to obtain than AVL data, a robust trip chaining method can use probability distributions to infer the most likely trajectory of individual transit passengers.

GTFS data provides a list of candidate stops with scheduled time to find the closest stop to the current tag location from AFC data [57]. Smart card data can be utilized with GTFS static and other data sources to reduce the use of the expensive and time-consuming Household Travel Surveys (HTS) for inferring passengers' trip purpose [58]. Advanced deep learning algorithms learn from the fusion of GTFS static and real-time data to solve time series forecasting tasks, such as train delay prediction or travel time forecasting of bus journeys [4, 59]. Such strong evidence has shown that GTFS plays a critical role in supplementing other data sources. Compared with AVL, APC, and AFC, GTFS static and real-time data comply with industry standards for data formats and allow for data sharing over all countries in the world. However, the GTFS raw data cannot be used directly, and an effective and efficient data cleaning solution needs to be developed.

2.2 Data Fusion

Cities can develop and implement intelligent analysis systems to monitor public transit performance by integrating, analyzing, and modeling real-world data rather than depending on simulations [60]. In recent years, it is a common practice that deep learning models, as the classifier or predictor, can frequently enhance the accuracy in a significant

number of ITS applications [61]. Relying on the completeness of datasets, deep learning models can easily derive patterns and models from large amounts of data [33]. Data fusion is widely used to form labeled training samples, such as combining mobility data with survey data, or subway smartcard data, and taxi GPS data with mobile phone signaling data [62-64].

2.2.1 Data Cleaning

In general, data cleaning activities consist of error detection and error repair [65]. Outlier detection techniques for error detection of time series data can be divided into two main types to deal with two kinds of issues from a computational perspective, including outliers crossing over a time-series database and outliers within a single time series [66]. Given an available time series, it also contains two cases, namely point outliers (particular elements) and subsequence outliers. GTFS real-time Trip Update APIs provide daily delay information, which includes abnormal delays. Furthermore, the abnormal event is usually unforeseen or unpredictable. Such events usually have features of suddenness and uncertainty, therefore, various methodologies can be used to find point outliers for a time series. The most applicable techniques can be divided into three types: the prediction model, the profile-based model, and the information-theoretic compression-based model.

For multivariate time series data, prediction models can directly compute outliers for all constituent time series [66], such as multilayer perceptron (MLP) [67], mixture transition distribution (MTD) [68], and autoregressive integrated moving average (ARIMA) [69]. However, Bayesian structural time series (BSTS) can flexibly adapt to various assumptions on the latent states of the observed data, including local trends and seasonality [70]. It also uses a Bayesian method to model the temporal evolution of observation data and utilizes a regression to avoid overfitting. BSTS is a statistical technique that can be applied for feature selection, time series forecasting, nowcasting,

and inferring causal impact [70-73]. BSTS can also be extended to deal with inference and prediction for multiple correlated time series [74]. Training a Bayesian model requires building a posterior that factors over model parameters, such as Markov Chain Monte Carlo (MCMC) and variational inference (VI) [71, 75].

Furthermore, data imputation is another common task in data analysis, which fixes missing or empty values. For a GTFS dataset, these values invalidate the record, and they can be the exact string *NaN* or the number 0. No matter how these values appear in the dataset, understanding what to expect, and checking consistency whether the data matches that expectation, will reduce potential issues in using the information later on [76].

2.2.2 Data Fusion Methodologies

Data fusion, information fusion, and knowledge fusion can be considered three levels of abstraction [77]. Nonetheless, they are tightly related. In data fusion, several sources of raw data are extracted to generate more useful information in order to remove noisy and redundant data; with information fusion, several sources of information are combined to create knowledge; for knowledge fusion, several heterogeneous sources of information and/or knowledge are merged to create a complete knowledge [78]. Both information and knowledge can be employed to support decision-making [79]. Additionally, the knowledge fusion problem is regarded as using multiple knowledge extractors to extract values from each dataset and then decide the degree of correctness of the extracted knowledge to generate a knowledge base [80].

The availability of multimodal transport data provides more useful information for understanding the mobility changes in urban areas. With multi-source datasets, computational models are constructed for uncovering and optimizing urban mobility patterns. Matrix factorization decomposes an observed matrix M into a product of two matrices, which present the latent factors of user-feature and item-feature, respectively

[81]. Since the production of the two matrices can approximate the observed matrix *M*, matrix factorization can predict the missing or incomplete data for effectively fusing the information or knowledge from multiple heterogeneous data sources. On the other hand, Singular Value Decomposition (SVD) and non-negative matrix factorization (NMF) are two main methods to deal with missing values and dimensionality reduction [81, 82]. Moreover, a data fusion approach with penalized matrix tri-factorization (DFMF) is proposed to decompose data matrices reveal hidden associations simultaneously. This approach identifies that matrix factorization-based data fusion achieves a high accuracy result and time response in a particular scenario [83]. To tackle the issues with sparse data, a context-aware tensor factorization (CATF) model has employed high-order singular value decomposition (HOSVD) to integrate with contextual features (e.g., features of gas stations and weather conditions) [84].

Multi-view learning is a learning paradigm that uses one function to model each view and jointly optimizes all the functions to exploit redundant views of the same input data [85, 86]. For example, for predicting weekday ridership of a rail station, there are often two views representing the given rail station: its local temporal information (e.g., weather conditions) and spatial information (e.g., distribution of Points of Interest). According to [86], the existing multi-view learning algorithms can be divided into three groups: co-training style algorithms, co-regularization style algorithms, and margin consistency style algorithms. Meanwhile, multi-view learning algorithms can also be grouped into co-training, multiple kernel learning, and subspace learning [81, 85]. The above algorithms can be mutually integrated; we only review subspace methods as the main solutions for the multimodal transport scenario. The main idea of subspace methods is to exploit the latent subspace for multi-view data [81].

Particularly, Principal Component Analysis (PCA) is a powerful method to

exploit the subspace for single-view data [81]. For example, PCA can be utilized to explore the spatial and temporal structure of aggregated human mobility; the predictions of most pixel population variations (PPVs) of the PCA model can achieve superior performance than the auto-regression moving average (ARMA) model [87]. The constraint or assumption of statistical independence on the sources helps to achieve essential uniqueness or diversity in data-driven models [88]. The use of the ICA method for data fusion can fix the indeterminacy of factor analysis (FA).

Canonical correlation analysis (CCA) is an efficient and powerful approach to find the correlation between two sets of variables. As CCA is concerned with seeking a pair of linear transformations associated with the two sets of variables, the projected variables on each view are maximally correlated [89].

Nevertheless, for capturing the nonlinear correlation among data, Kernel CCA is proposed to map each data point to a higher space [81, 90]. Moreover, Tensor CCA is developed to generalize CCA to handle many views in a straightforward and natural way [91]. Mainly, CCA, Kernel CCA, and Tensor CCA exploit the subspace in an unsupervised way. Since the generalization performances of unsupervised learning approaches may not be good enough for prediction tasks, the Bayesian multi-view dimensionality reduction (BMDR) method is proposed to project data points into a unified subspace [92].

A probabilistic graphical model (PGM) encodes probability distributions or expresses conditional dependencies among large numbers of random variables. Generally, a PGM is a declarative representation (or a graphical representation) that consists of nodes and edges, where nodes correspond to a group of random variables and edges express interactions among variables [93]. Additionally, there are two typical models of PGMs: directed graphical models (Bayesian Networks) and undirected graphical models

(Markov Networks). Bayesian Networks (BNs) represent causality between variables; however, Markov Networks represent mutual relationships between variables [94].

Inference using PGMs is the process of predicting the status of latent variables from the probabilistic model. It can be regarded as an optimization problem. Approximate inference algorithms are derived to implement the optimization. There are two major methods in approximate inference algorithms: 1) variational methods are deterministic, and 2) particle-based inference methods use stochastic numerical sampling from distributions such as forward sampling, importance sampling, and Gibbs sampling.

Particularly, Gibbs sampling can be employed equally well to both BNs and MNs [93]. Moreover, a topic model based on Latent Dirichlet Allocation (LDA) and Dirichlet Multinomial Regression (DMR) can infer the functional regions in a city by utilizing Gibbs sampling [95]. Inspired by PGMs in text mining, a Human Mobility Representation model (HuMoR) is proposed to infer latent patterns from anonymized sequences of user locations using Collapsed Gibbs sampling. Gibbs sampling is not an efficient parameter inference method [96].

Furthermore, the Gaussian Bayesian Network (GBN) based graphical model was applied to generate the casual spatiotemporal pathways for air pollutants by combing pattern mining and Bayesian learning. The experiments identified that the model outperforms ARMA, linear regression model, and support vector machine for regression with a Gaussian radial basis function kernel (SVM-R) in both efficiency and inference accuracy [97].

According to [94], both BN and MN are used to develop a hybrid network for estimating multivariate Gaussian distribution so that computational complexity can be reduced and the probability of finding the best solution can be increased. As PGMs provide an approximation of exact distributions, large-scale heterogeneous data sets can

describe complex relations using statistical inference [93]. Additionally, the information in multiple heterogeneous information networks (HINs) can be fused to obtain a more comprehensive and consistent knowledge [98].

2.3 Entropy

For exploring the degree of predictability in individual mobility, the entropy measure is one of the most promising methods to characterize the limits of predictability $\prod^{max}$ in one user mobility. To be specific, if a user with $\prod^{max}$ = 0.25, it means that at least 75% of the time, the user's location appears to be random. Only in the remaining 25% of the time we can predict the location that the user appears at. In other words, no matter how good the predictive algorithm is, the study cannot predict with better than 25% accuracy. Thus, $\prod^{max}$ presents the fundamental theoretical limit for the potential predictability in user mobility [99]. Correspondingly, in [99], the study of actual prediction algorithms on 500,000 users was conducted to show how close they were to the maximum potential predictability. Markov chain (MC) based models were implemented to forecast the actual location visited by each user. A comparison of the results reveals that a higher-order MC-based model does not significantly improve prediction accuracy when approaching the maximum predictability [100].

As aforementioned, train delay prediction can be regarded as a time series prediction task, which has been studied by interdisciplinary researchers [15]. A certain level of randomness concerns most of the time-series datasets, which is unpredictable [99, 101]. [99] used entropy to demonstrate the fundamental theoretical limit for the predictability in analyzing user mobility based on historical time series. The entropy-based measure is one of the most effective approaches to characterize both the randomness and the temporal correlation. It utilizes a value between 0 and 1 to illustrate the regularity of the model inputs [101]. Generally, there are two main methods to solve

the prediction problem: statistical model-based prediction and machine learning-based prediction. Autoregressive-moving average (ARIMA), Kalman Filtering (KF), and fixed-interval smoothing have been extensively used for time-series forecasting [102]. Compared to model-based approaches, deep learning has brought a breakthrough in tasks involving sequential inputs [103].

2.5 Train Delay Prediction

In literature, train fare design can significantly reduce passenger demands of congested train stations and spread peak demand across multiple stations, such as Sydney train systems [104, 105]. However, many factors cause train delays; for example, primary congestion predictive factors have the most significant impacts on congestion delay, including meets, passes, and overtakes [106]. Hence, although the train delay can be reduced, it cannot be avoided. Moreover, several studies have investigated train delay prediction. [107] indicated a comparison study of least-trimmed squares (LTS) robust linear regression model, regression trees, and random forests for the accuracy of the dwell time and running time predictions. The obtained results showed an error within ten percent in running time estimation. However, the selected predictor variables could not fully explain the dwell time variability; furthermore, the prediction error of the dwell time was significantly large for real-time estimation. A certain degree of uncertainty remains unresolved regarding the dwell time prediction.

Extreme Learning Machines (ELM) establish a repeated learning process to predict train delays with weather information in a large-scale network. The models are updated incrementally with new daily input parameters [18, 108]. Bayesian networks with dynamic stochastic prediction provide another approach to update the probability distribution of train delays and present their evolution over time. The model used parent nodes as prior probabilities of calibrating the resulting Bayesian network [109]. The state-

of-the-art online traffic models mostly rely on conditional probability distributions and regression coefficients for adjacent stations, such as the Bayesian network [109, 110]. A Bayesian network consists of an acyclic directed graph and conditional probabilities [111]. The predictions can be sufficiently reliable for short horizons only, for example, up to 15 min or 30 min.

Nair et al. developed a linear ensemble forecasting method combining RF with kernel regression for train delays at a mesoscopic level [22]. [112] presented a gradient-boosted regression tree model to predict train delay time and showed the trend of train delays based on the fusion of a three-month weather dataset, a train delay dataset, and a train schedule dataset. On the other hand, [59] proposed a generic framework that leveraged forward chaining algorithm for incorporating expert knowledge with Long Short-Term Memory (LSTM) and variants. [113] presented a bi-level random forest model to predict train delays in the Netherlands. The model was composed of a classification forest for determining the follow-up changes of a current delay at the primary level, and several regression forests for quantifying the amount of delay at the secondary level.

2.6 Machine Learning Predictive Model

The autoregressive integrated moving average (ARIMA) model, as one of the most popular statistics-based time series models, can find the best fit of the model to the past observations using the Box–Jenkins approach [102]. For many years, the ARIMA model has been combined with artificial neural networks as hybrid solutions, which improve forecasting accuracy in linear and nonlinear modeling [114, 115]. A random forest is an ensemble learning that fits a group of decision trees on training data and uses voting or averaging to predict performance better [116].

More recently, deep neural network models such as Convolution Neural Networks, Recurrent Neural Networks (RNN), and Graph Neural Networks (GNN) achieved great success for time series forecasting. CNN is a widely used tool to process image, speech, and time series since it was introduced by LeCun et al. [117]. In traffic status prediction, CNN models often use one or more convolutional layers and activation functions, max-pooling layers, and thoroughly connected (dense) layers. Besides, adding dropout layers can effectively avoid over-fitting, where the model fits the training data too well but fails to fit additional data or accurately predict future observations. Moreover, one-dimensional (1D) convolution can learn from the two-dimensional (2D) time-series data directly. The structure of 2D convolution can be performed to capture spatio-temporal features. Three-dimensional (3D) convolution not only extracts features in both spatial and temporal dimensions but also considers the temporal properties of data, cyclical patterns, and trends, for instance, closeness (local patterns) and weekly period (long-term patterns) [118]. However, the disadvantage of 3D CNNs is that it requires multiple layers to learn long-term dependencies at each time step. It also needs to calculate and update the optimal weights of outputs repeatedly and then obtain the final result using element-wise multiplication and sum operators [118].

Feed-forward neural networks, like CNN, cannot directly capture temporal dependencies between successive time-steps. To overcome the limitation, a recurrent neural network (RNN) was proposed to study temporal correlations without learning trends and seasonality. RNNs include multiple variants, for instance, Long Short-Term Memory (LSTM) [119] and Gated Recurrent Unit (GRU) [120]. Multiple CNNs and LSTMs can be stacked and combined in various ways to form a more complex architecture. In some cases, they (such as GRU-LSTM and Bi-directional LSTM) lead to good outcomes where the models show performance improvement than a simple structure

for discovering trends, seasonal, and cyclical patterns of data [52], [121-123]. Applying the attention mechanism can further learn local and global dependencies between input and output to produce more interpretable models, when deep learning is concerned [124].

Nevertheless, very few of the existing work has considered using entropy to calculate the maximum predictability on train delay data priorly or applying multi-step predictive models with delay status and resilience indicators to tackle the multi-scenario delay prediction tasks.

Chapter 3
Chapter 3

A GTFS Data Acquisition and Processing Framework

With advanced artificial intelligence and deep learning techniques, a growing number of data sources are playing more and more critical roles in planning and operating transportation services. The General Transit Feed Specification (GTFS), with standard open-source data in both static and real-time formats, is being widely used in public transport planning and operation management. However, compared to other extensively studied data sources such as smart card data and GPS trajectory data, the GTFS data lacks proper investigation yet. Utilization of the GTFS data is challenging for both transport planners and researchers due to its difficulty and complexity of understanding, processing, and leveraging the raw data.

This chapter proposes a general GTFS data acquisition and preparation (DAP) framework in which we consider a comprehensive data cleansing process. The framework is developed to convert and fuse the GTFS data to a ready-to-use format. To validate and test the proposed framework, a multivariate multistep Long Short-Term Memory is also developed to predict train delay with minor anomaly in Sydney as a case study. The contribution of this new framework will render great potential for wider applications and deeper researches.

The rest of this chapter is organized as follows. In Section 3.1, we first introduce GTFS static and real-time data. The merging and fusion of GTFS static and GTFS real-time is demonstrated in Section 3.2. We illustrate the proposed DAP framework, and the

BSTS with a rule-based inference engine in Section 3.3 and Section 3.4. We present the experiment results using a train line in Sydney as a case study in Section 3.5. Finally, Section 3.6 concludes briefly.

3.1 GTFS Static and Real-Time Data

GTFS is developed for transit agencies to publish detailed transit schedules in an open data format; GTFS and GTFS real-time specifications enable transit agencies and operators to exchange both static and real-time public transit information [19]. Using GTFS data can conduct accessibility analysis, discover schedule padding, perform single or multiple transit system analysis, and investigate social equity in transportation planning [125-128]. However, GTFS real-time needs to be collected through the Application Programming Interface (API). The downloaded raw data needs to be preprocessed by data duplication, sorting, and so on. It is merged with the information of GTFS-Static. Additionally, the new datasets are backed up and stored on the cloud as research resources.

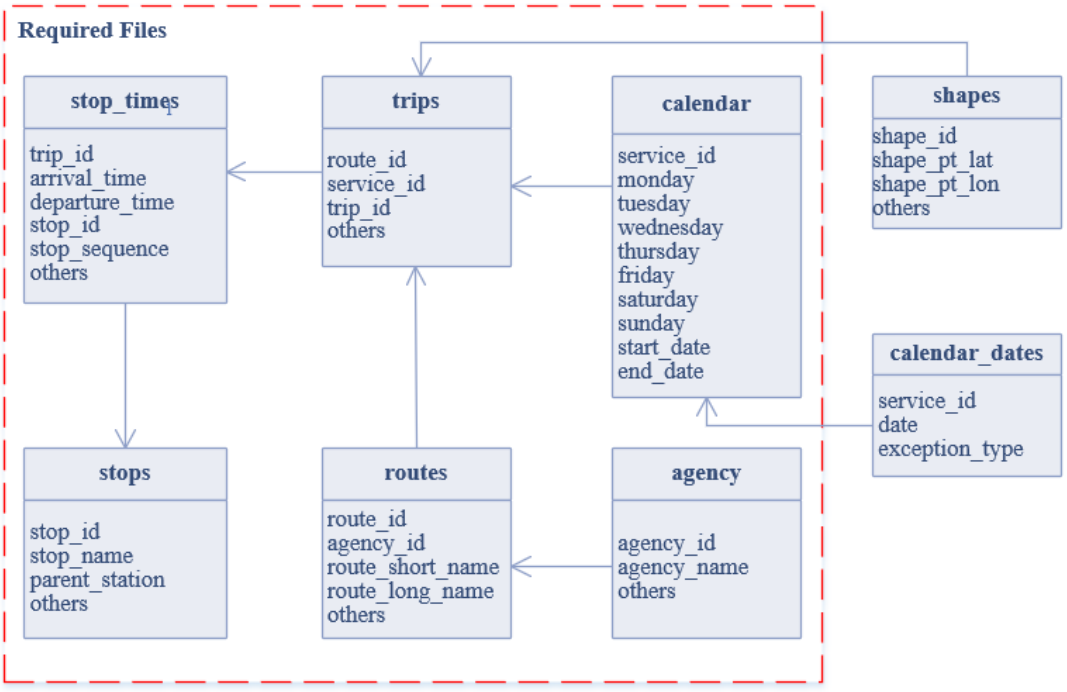


Figure 3.1 GTFS File Structure and Entity Relationship

GTFS data is entirely open and free, which consists of static and real-time formats. GTFS static data defines a common format for public transport schedules with associated geographic information; while GTFS real-time data provides real-time information of public transport services, such as vehicle location and road congestion level. On the one hand, further investigation of GTFS static and real-time data creates opportunities to explore valuable information from daily operation data; on the other hand, such work serves as a foundation to develop and validate various models and algorithms to support planning and operation of public transport services. Figure 3.1 depicts an example of a GTFS static about the relationship between files and entities. First, the GTFS static has two types of files: required files and optional files. The required files contain six txt files, namely 'agency.txt', 'routes.txt', 'trips.txt', 'stop_times.txt', 'stops.txt', and 'calendar.txt'. All the other files are optional, such as 'shapes.txt' and 'calendar_dates.txt'.

GTFS real-time contains vehicle positions, service alerts, and trip updates. This study employed the trip updates feed to obtain delay information. A trip update for trip_id '600D.1384.124.128.T.8.57243995' that presents a train arriving at a station (stop_id: '2205114') 183 seconds late and leaving 194 seconds late, would look as follow:

```
entity {
    id:" 600D.1384.124.128.T.8.57243995"
    trip_update{
        trip{
            trip_id: "600D.1384.124.128.T.8.57243995"
            schedule_relationship: SCHEDULED
            route_id: ""
        }
        stop_time_update{
            arrival{
                delay: 183
            }
            departure{
                delay: 194
```

```
                    }
                    stop_id: "2205114"
                    schedule_relationship: SCHEDULED
                }
            }
        }
```

3.2 GTFS Static and Real-Time Data Merging and Fusion

Table 3.1 GTFS Static Data Merging with Real-time Data

Overlapped			**Static**			**Real-time**	
trip_id	stop_id	scheduled_date	arrival_time	departure_time	stop_name	arrival_delay	departure_delay
600D	202291	2019/4/15	8:25:01	8:29:01	Bondi Junction Station	115	46
600D	202761	2019/4/15	8:32:00	8:32:30	Edgecliff Station	71	77
600D	201171	2019/4/15	8:34:30	8:35:00	Kings Cross Station	70	107
600D	2000361	2019/4/15	8:37:00	8:37:30	Martin Place Station	92	117
600D	2000394	2019/4/15	8:39:00	8:40:00	Town Hall Station	106	97
600D	2000345	2019/4/15	8:42:00	8:43:00	Central Station	75	67
600D	2015142	2019/4/15	8:44:36	8:45:06	Redfern Station	60	81
600D	2205114	2019/4/15	8:53:30	8:54:00	Wolli Creek Station	84	87
600D	2220364	2019/4/15	9:02:00	9:06:00	Hurstville Station	20	0

As a starting point, the ‘calendar_dates’ text file is used as an example to create a usable GTFS timetable in comma-separated values (CSV) format. However, it is conditionally required that most GTFS data do not provide date information when a service exception does not occur, like all GTFS bundles from the Transport for NSW [13].

In the 'calendar.txt' file, '1' denotes an available service, and '0' implies that a service is not available for Monday to Sunday in the date range. We fill in missing information by using 'service_id' to create new data with regard to the date when the regular running of service occurs, namely 'scheduled_date'.

As seen in Table 3.1 , we consider a relational schema R with attributes $atr(R)$, a functional dependency (FD) is defined as $X -> A$, wherein $X \subseteq atr(R)$ and $A \subseteq atr(R)$. We have X the left-hand side (LHS) and A the right-hand side (RHS). As the RHS of each FD is a subset of its LHS, the FD is trivial. Thus, agency_id $->$ route_id and route_id, service_id $->$ trip_id, stop_id are valid concerning a GTFS dataset instance.

Algorithm 3.1: Online Collection of GTFS Real-time Data

Input: entities E // collecting JSON objects with the API
Output: S // a set of entries is returned
Repeat
For E **in** Feed_Entity
If $E.hasField("TU")$ // TU: trip update; VP: vehicle positions; SA: service alerts
For *count* in range (length ($E.stop_1, \dots E.stop_n$))
If $E.stop.trip\_id$ and $E.stop.stop\_id$ exist
$S.add(E.stop_{current})$
$S.remove(E.stop_{previous})$
Else
$S.add(E.stop_{current})$
End for
time.sleep(10)
End for

A Trip Update API is called once every 10-30 seconds for extracting GTFS real-time data. All collected data are stored in the CSV file. Firstly, the dataset contains a large

number of repetitions, which are cleaned and filtered in the deduplication step. As the collected data is incremented based on the time stamp, 'trip_id' and 'stop_id' are used to keep the last one and remove the others for identifying and dropping duplicates, as shown in Table 3.1. Also, the GTFS real-time data online collection is demonstrated in **Algorithm 3.1**.

Table 3.2 GTFS Static and Real-time Data Fusion

Trip_Number	**Scheduled_Date**	**Arrival_Delay**	**Departure_Delay**	**Dwell_Time**	**Stop_Sequence**	**Running_Time**
600D	2019/4/15	115	46	171	1	0
600D	2019/4/15	71	77	36	2	204
600D	2019/4/15	70	107	67	3	113
600D	2019/4/15	92	117	55	4	105
600D	2019/4/15	106	97	51	5	79
600D	2019/4/15	75	67	52	6	98
600D	2019/4/15	60	81	51	7	89
600D	2019/4/15	84	87	33	8	507
600D	2019/4/15	20	0	220	9	413

Secondly, GTFS static (DB-S) and GTFS real-time (DB-R) can be fused as a unified dataset by mapping 'trip_id,' 'stop_id,' 'date' at the schema-based data fusion step. Hence, the schema matches should be DB-S.trip_id = DB-R.trip_id, DB-S.stop_id = DB-R.stop_id, and DB-S.scheduled_date = DB-R.Scheduled_Date. Thirdly, an outlier detection model expects the input data to be in the standard format, such as time and date, that need to be converted into the same unit [65]. Herein, data transformation is applied

to standardizing data format. An example of a GTFS static and real-time fusion dataset containing the main attributes is shown in Table 3.2.

3.3 A GTFS Data Acquisition and Preparation (DAP) Framework

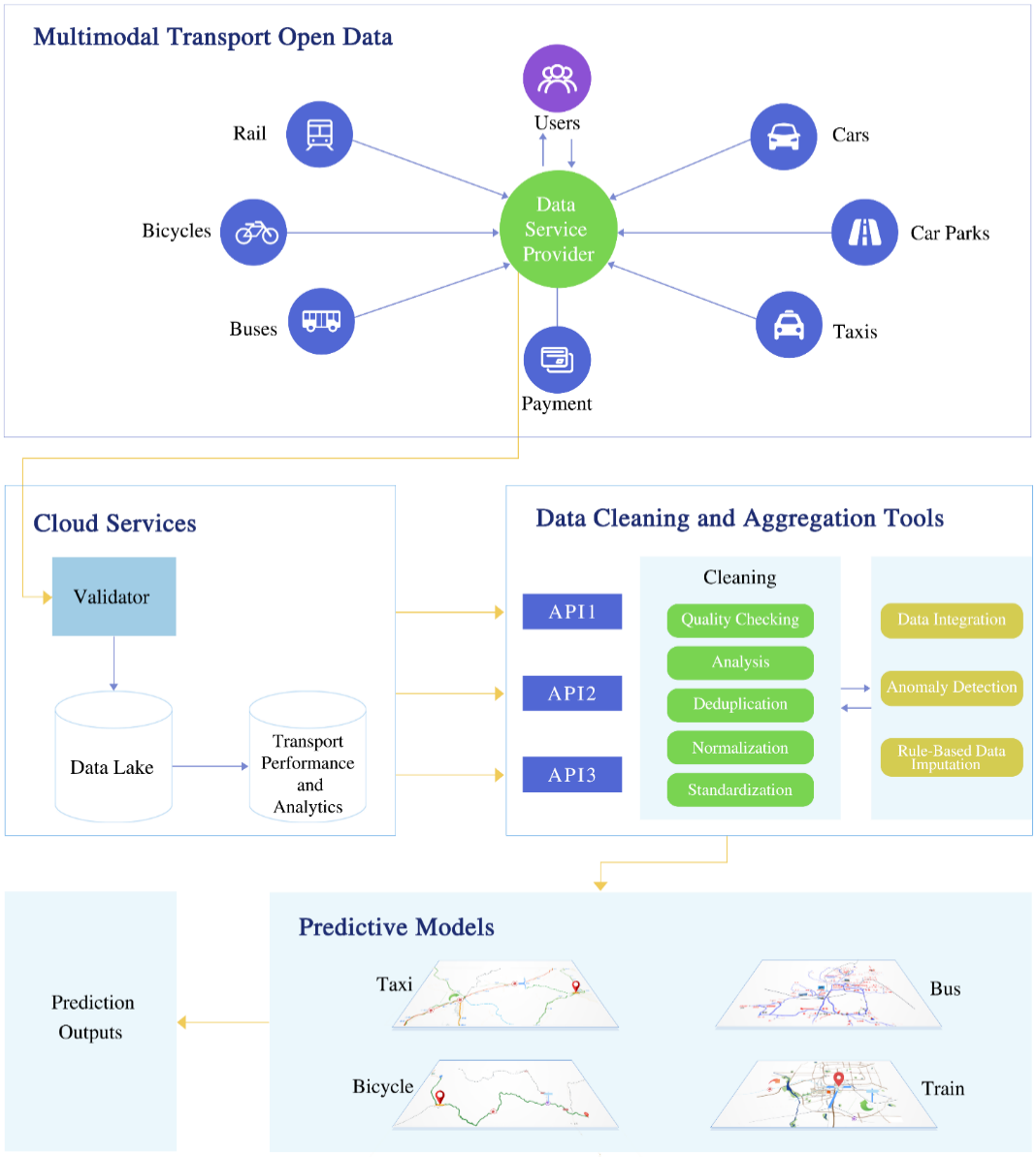


Figure 3.2 A Data Acquisition and Preparation Framework

The proposed GTFS DAP framework includes data cleaning and aggregation tools, which can provide an interface to connect with a multimodal transport system and cloud services, as shown in Figure 3.2. In a multimodal transport management system such as MaaS, an integrated data platform is used by data service providers, which is an

intermediate layer between users and transport operators. It offers a data service bringing together all the modes of existing transportation. The provision of real-time information is shared by cloud computing services.

A validator validates data to ensure that there are no errors or issues before the data is loaded in the data lake of the cloud service. Moreover, data are uploaded or transferred from the validator into the data lake before any data is transferred to the Transport Performance and Analytics (TPA) database. The TPA converts credible transport data to the JSON format, which is easily shared and reanalyzed. In this study, data cleaning and aggregation tools are developed to collect data by using standard application programming interfaces (APIs).

Following data validation, data deduplication is implemented to remove incorrect or undesirable observation data. Then, the preprocessed GTFS real-time dataset is combined with GTFS static data to generate an integrated dataset, and the whole process can be regarded as data fusion. Furthermore, an anomaly detection method obtains predicted values, which are used to calculate the Z-scores for standardization to find abnormal data. Finally, the application of proper data imputation technologies can fill in missing values. Normalization is an essential data preprocessing step to scale features before training a prediction model. This process ensures that the scaling variables are within the same range of values to provide appropriate inputs to the predictive models. As a result, the proposed GTFS DAP framework can obtain multivariate time series datasets from the multimodal transport system platform.

As shown in Figure 3.3 , the data cleaning and aggregation tool is described as a workflow to depict the components of offline GTFS static, online GTFS real-time, data deduplication, schema-based data fusion, data transformation, outlier detection, and rule-based data imputation.

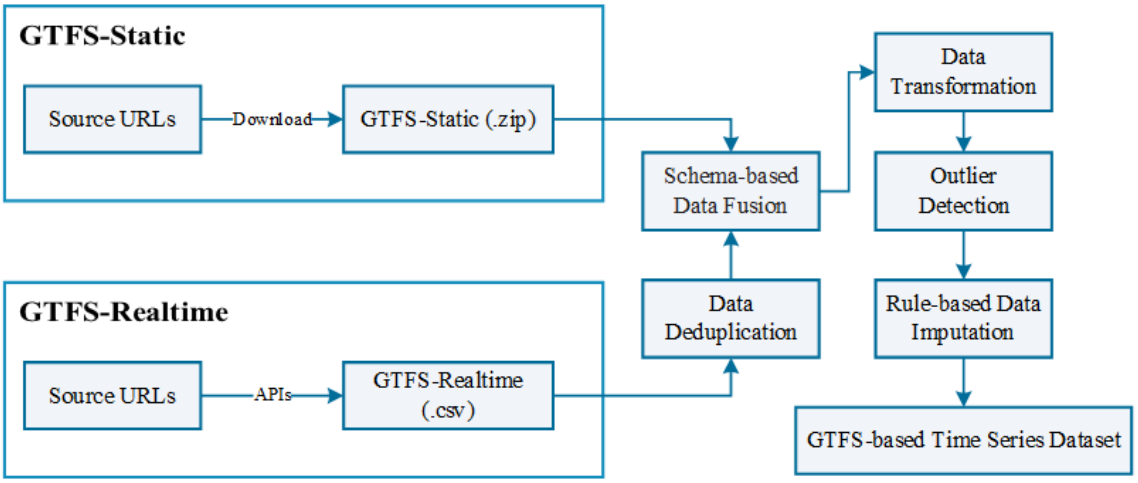


Figure 3.3 Data Cleaning Workflow

3.4 Outlier Detection and Imputation

The traffic information prediction can be regarded as the problem of time series analysis, which involves univariate and multivariate variants. As the multivariate method can automatically contain the univariate approach, traffic information prediction studies are conventionally referred to as multivariate time series analysis. For time series forecasting, it is essential to filter out outliers to ensure that the observations are accurate and useful for modeling the subsequent prediction problem. Anomaly detection methods detect outliers, which are outside the scope of defining normal data. In this study, we apply a statistics-based outlier detection technique, the BSTS model [70]. It is a state-space model (SSM) for time series data. The model can utilize posterior predictive samples to compute the posterior distribution of cumulative impact. Expected data points appear in high probability regions of the model, whereas outliers occur in the low probability regions of the model. Standardized residuals can be employed to detect outliers by comparing them to a Z-score. The formulation of the problem as a pair of equations is given by:

$$y_t = Z_t^T a_t + \varepsilon_t \,,\ \varepsilon_t \sim N(0, \sigma^2) \tag{3.1}$$

$$a_{t+1} = T_t a_t + R_t \eta_t \, , \eta_t \sim N(0, Q_t) \tag{3.2}$$

where equation (3.1) is the observation model, and equation (3.2) is the transition model (state equation). The transition model and observation model are both linear Gaussians. A linear Gaussian model is a Bayesian network, which defines multivariate Gaussian distributions. $\varepsilon \sim N(0, \sigma^2)$ is an observation noise term. $\eta_t \sim N(0, Q_t)$ is a transition noise term. y_t is observed data of the model at time *t*. a_t is a state vector. The transition model describes the evolution of the state vector from time step *t* to timestep *t*+1. Z_t^T is the observation matrix. T_t is a transition matrix. R_t is a control matrix. σ^2 and Q_t are covariance matrices.

Structural time series (STS) models contain three types of critical components: seasonality, regression, and local linear trends [70]. For seasonality, we utilize date-time to generate a seasonal feature, which captures seasonal effects. The regression component is the essential state component of the model. For the local linear trend, an autoregressive (AR) model is applied to balance short-term information with previous steps, in which the value of each step is a noisy linear combination of the previous steps. The model can use the additive or the multiplicative form [129]. An STS model represents an observed time series as the additive form:

$$f(t) = \sum_{i=1}^{N} f_i(t) + \varepsilon_t \tag{3.3}$$

Thus, the observed time series can be decomposed in the following form: $Observed\ Series = Seasonal + Linear\ Regression + Autoregressive + error$

(3.4)

For a given STS, each component is treated as a Bayesian model, which contains the set of model parameters θ and the observation vector $y = \{y_1, \dots, y_t\}$. The posterior density $P(\theta|y)$ is computed by the Bayes' theorem:

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)} \tag{3.5}$$

To learn the set of model parameters θ, the Kalman filter algorithm is utilized to derive the likelihood $P(y|\theta)$. Applying the Bayes' theorem to calculate the $P(\theta|y)$ is referred to as an inference problem. Moreover, we can turn it into an optimization problem by minimizing the Kullback-Leibler (KL) divergence from $Q(\theta\ ;\ \lambda)$ to $P(y|\theta)$.

$$Q(\theta\ ;\ \lambda)^* = \arg\min_{\lambda} \mathrm{KL}\big(Q(\theta\ ;\ \lambda)||P(\theta|y)\big) \tag{3.6}$$

where variational inference is applied to approximating the posterior distribution of the model parameters $P(\theta|y)$, by employing variational distribution $Q(\theta)$ with variational parameters λ from the given observation y. After the data y is observed, the Bayesian model has the marginal likelihood (model evidence):

$$P(y) = \int P(y,\theta)d\theta \tag{3.7}$$

As the marginal likelihood is a constant to λ , we can maximize the Evidence Lower Bound (ELBO) instead of minimizing the KL divergence:

$$Q(\theta\ ;\ \lambda)^* = \arg\max_{\lambda} ELBO\big(Q(\theta)\big) \tag{3.8}$$

Furthermore, the gradient descent, as an optimization algorithm, is used to minimize $-ELBO\big(Q(\theta)\big)$ regarding the variational parameters:

$$\nabla_\lambda ELBO\big(Q(\theta)\big) = \nabla_\lambda \int Q(\theta) \log \frac{P(y,\theta)}{Q(\theta)} d\theta \tag{3.9}$$

To compute integration, sampling is often exploited in the process. Hence, we incorporate the observations directly into the model by applying a reparameterization gradient [130, 131]. Moreover, we firstly use multiple attributes as inputs. Then, the model is trained to maximize $ELBO\big(Q(\theta)\big)$. Training a BSTS with rule-based inference engine for data preprocessing is indicated in **Algorithm 3.1**. Although advanced deep learning models can derive approximate values very well, they fail to preprocess GTFS

data. As there are causal relationships among various GTFS variables, we propose a rule-based inference engine as a rule-based data imputation method to derive more missing variables accurately by using the scheduled arrival time A_{t_s}, scheduled departure time D_{t_s}, actual arrival delay AAD_{t_a}, and actual departure delay ADD_{t_a} at time step t.

Algorithm 3.2: Training a BSTS with Rule-Based Inference Engine for Data Preprocessing

---

Require: type of tasks *TS*, spatiotemporal attributes y_t^N, the number of attributes *N*, random variables *x*, variational distribution $Q(\theta)$, actual arrival time A_{t_a}, actual departure time D_{t_a}, actual running time R_{t_a}, and actual dwell time W_{t_a}.

If *TS* = Preprocessing a data set is used for a real-time forecast:

For $\boldsymbol{n = 1 \rightarrow i}$ do

$A_{t_a} = A_{t_s} + AAD_{t_a}$

$D_{t_a} = D_{t_s} + ADD_{t_a}$

If $\left(A_{t_a} - A_{(t-1)_a}\right) \leq 0$ or $\left(D_{t_a} - D_{(t-1)_a}\right) \leq 0$:

set anomalies to *NaN*

Applying a real-time delay prediction model

Else if *TS* = Preprocessing a data set is used for a long-term forecast:

Initialize λ, *N*, *t* =1.

Output: λ

Repeat

Draw *S* samples from the variational approximation

For s = 1 **to** *S* **do**

$x_s \sim Q(\theta)$

End for

Estimate the gradient $\nabla_\lambda ELBO\big(Q(\theta)\big) = \nabla_\lambda \int Q(\theta) \log \frac{P(y_t^N, \theta)}{Q(\theta)} d\theta$ as in equation (3.9)

Set the learning rate

Update the parameter λ

$t = t + 1$

Until the convergence conditions are satisfied

If z-score > threshold value

set anomalies to *NaN*

Applying a optimal data imputation algorithm

Applying a rule-based inference engine to search more missing variables, including A_{t_a}, D_{t_a}, R_{t_a} , and W_{t_a}

For each stop $es = (A_{t_a},\ D_{t_a},\ R_{t_a}, W_{t_a})$ **do**

$A_{t_a} = A_{t_s} + AAD_{t_a}$

$D_{t_a} = D_{t_s} + ADD_{t_a}$

$R_{t_a} = A_{t_a} - D_{t-1_a}$

$W_{t_a} = D_{t_a} - A_{t_a}$

End for

3.5 Multivariate Multistep Delay Prediction

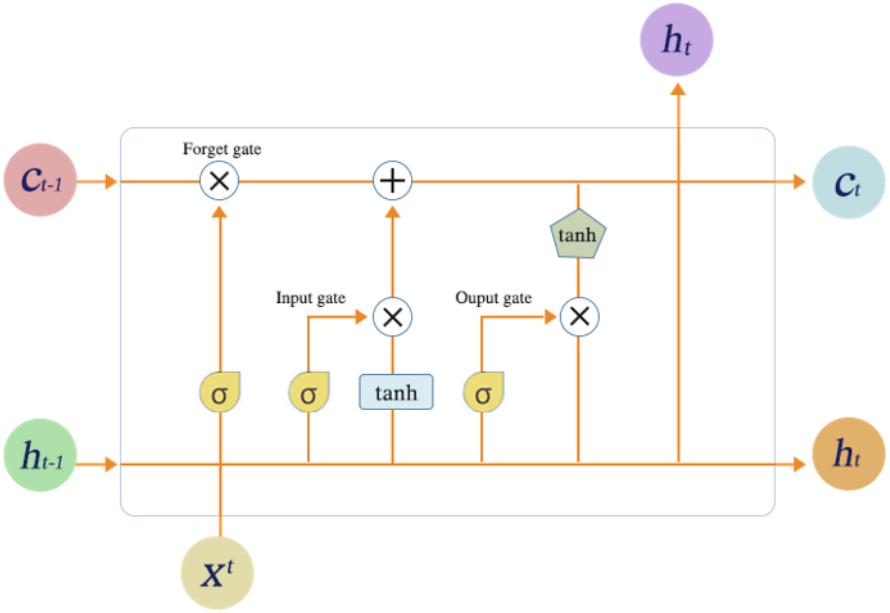


Figure 3.4 Illustration of an LSTM Network

Delay prediction can be expressed as a task extract information from historical data to accurately estimate future delay times for a corresponding mode of transport. In this study, a multivariate multistep LSTM (MM-LSTM) model is trained to verify the effectiveness of the proposed GTFS DAP framework. As indicated in Figure 3.4, an

LSTM has the short-term state h_{t-1} and the long-term state from c_{t-1} to c_t. Additionally, the input X^t is fed to sigmoid functions σ and an activation function $tanh$.

Firstly, the LSTM drops some memories in the forget gate and adds some new memories to the input gate by applying the addition operation. Secondly, the long-term state is passed through the activation function. Lastly, the output gate filters the results to generate h_t. W_{xi} , W_{xf} , W_{xc} , W_{xo}, W_{hi} , W_{hf} , W_{hc} , and W_{ho} denote the weight matrices that connect the input and the hidden vectors to the corresponding gates, respectively. b_i , b_f, b_c, and b_o express bias vectors. The equations of LSTM are demonstrated in equation (3.10), wherein the operator '$\circ$' is the Hadamard product:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}\, X^t + W_{hi} h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma\big(W_{xf}\, X^t + W_{hf} h_{t-1} + W_{cf} \circ c_{t-1} + b_f\big) \\
c_t &= f_t \circ c_{t-1} + i_t \circ tanh(W_{xc} X^t + W_{hc} h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}\, X^t + W_{ho} h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
\tag{3.10}
$$

3.6 Experiments

3.6.1 Data Description

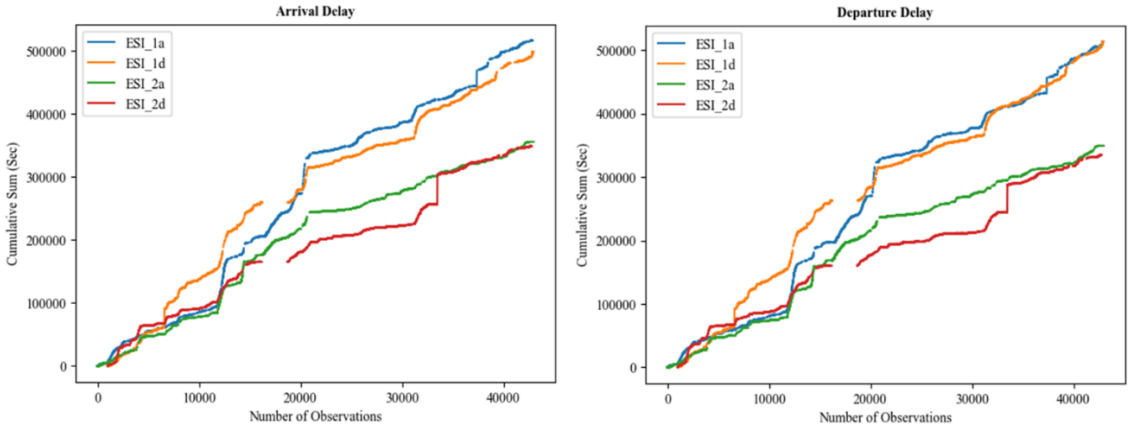


Figure 3.5 Cumulative Train Delay on T4 ESI train line

To validate the proposed DAP framework, we choose train services in Sydney as a case study. For operational data, we utilize Sydney Trains' GTFS static and real-time data obtained from the Transport for NSW's open data portal, which publishes its data regularly [13]. Spatial-temporal feature analysis is performed on a fused dataset of the GTFS static and real-time data, which consists of 150-day data observations of T4 Eastern Suburbs and Illawarra (ESI) train line from April 15 to November 8 in 2019.

In spatial dimension, the T4 line has four routes, including ESI_1a (Bondi Junction to Waterfall), ESI_1d (Bondi Junction to Cronulla), ESI_2a (Waterfall to Bondi Junction), and ESI_2d (Cronulla to Bondi Junction). Figure 3.5 shows cumulative sums of train delay of the four routes on the T4 line based on 42946 observations at discrete time series points. For departure delays, ESI_1a and ESI_1d have no noticeable difference. However, ESI_1a has the most significant arrival delays. In addition, the top ten arrival delay platforms and top ten departure delay platforms are listed as shown in Figure 3.6. It indicates that Wolli Creek station platform 4 has the most significant delays on arrival and departure trains.

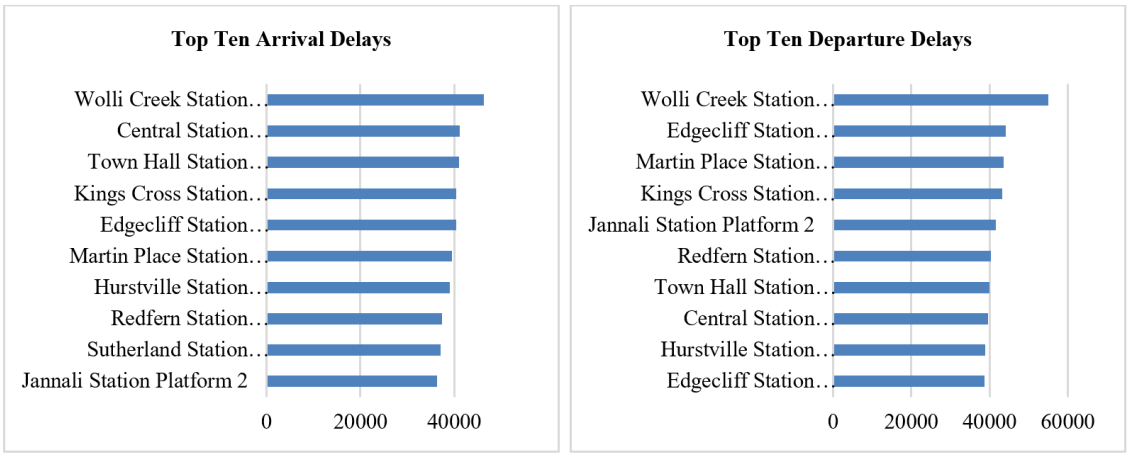


Figure 3.6 Top Ten Arrival and Departure Delays

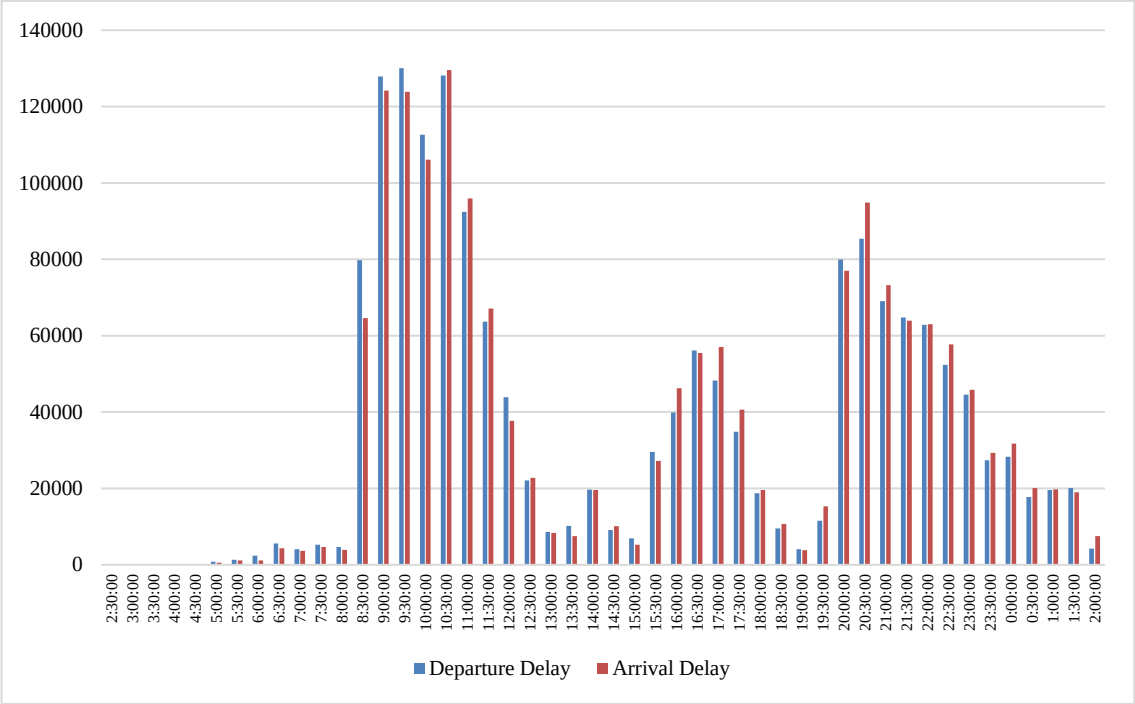


Figure 3.7 Distribution of Train Delay at Different Time Segments

As observed from Figure 3.7, for time-interval data in the temporal dimension, it is clear that delay peaks occur at three different periods: A.M. peak from 8:30 am to 11:30 am, P.M. peak from 3:30 pm to 5:30 pm, and Night Inter-peak from 8:00 pm to 11:00 pm. In addition, the A.M. peak period has more than double the delay time of the P.M. peak period. In this study, we aim to investigate the highest delay trips from T4. Thus, we choose trips after 8:30 am and passing through the Wolli Creek Station Platform 4. Furthermore, it can be found that a trip number '600D' of the ESI_1a meets the corresponding conditions as a qualified sample dataset. The selected experimental dataset consists of 150-day data observations at seven stations. Its daily scheduled arrival time is 8:34:30 am, and the expected departure time is 8:35:00 am at Bondi Junction Station from Monday to Friday morning.

For details of trip '600D', a specific train line connecting Bondi Junction Station and Hurstville Station, namely BJS-HS, is selected for numerical experiments. The BJS-HS line is one of the busiest urban rail lines in Sydney, linking CBD and some of the most

populous suburbs and also localities around the airport. Among them, Hurstville station has the highest number of daily transit patrons outside Sydney's CBD, while Bondi Junction connects to the most popular beach in the southern hemisphere. Trains operating on this line pass through seven major stations, including Edgecliff Station (ES), Kings Cross Station (KCS), Martin Place Station (MPS), Town Hall Station (THS), Central Station (CS), Redfern Station (RS), and Wolli Creek Station (WCS), as shown in Figure 3.8.

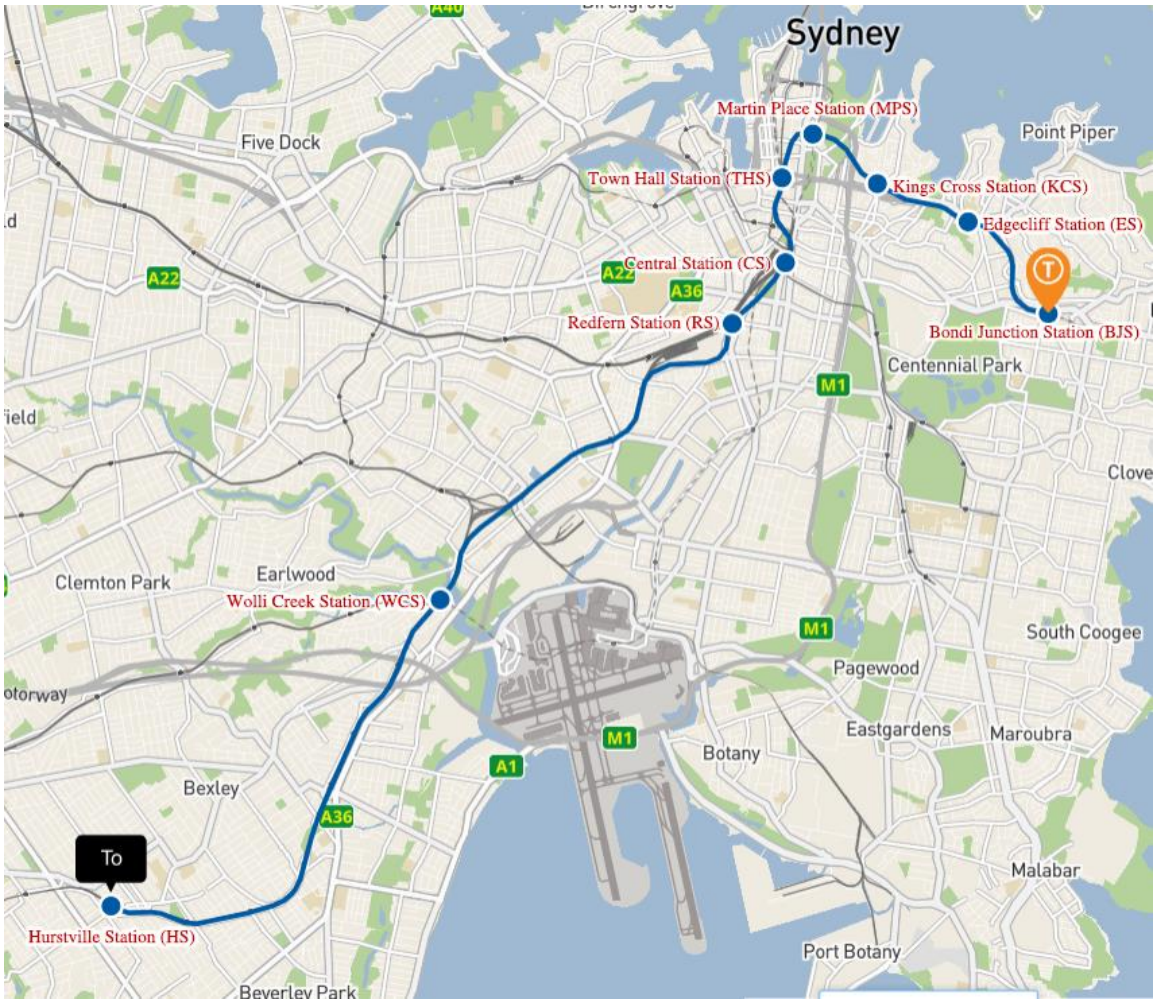


Figure 3.8 A Train Line (BJS-HS) in Sydney

3.6.2 Experimental Setup

According to **Algorithm 3.2**, we have two ways to deal with anomalies: One is to use a delay prediction model (non-imputed), and the other is to use a data imputation

algorithm (imputed). This study conducts a comparison between a non-imputed strategy and an imputed strategy for the long-term forecasting of train delays. The imputed strategy is to apply the mean of observed data for replacing missing values for arrival delays and departure delays in the GTFS real-time data, before they are used for modeling traffic information prediction tasks. Moreover, the BSTS model constructs one-step-ahead predictive distributions for all timesteps using samples from the variational posterior. To detect anomalous time steps in time series data, we use Z-scores to classify the observed values into two subgroups to identify whether a value is within a defined predictive interval.

$$Z = \frac{|x-\bar{x}|}{s} \tag{3.11}$$

wherein Z represents a standard score, x is the observed value, $\bar{x}$ denotes the mean of the predictive distributions, and s is the standard deviation of the predictive distributions. A time-series data point would be considered abnormal if its Z-score is greater than a threshold value. As a result, according to the Z-score (the standard normal) table, we identify the corresponding data points, which contain regular delay patterns. The LSTM uses the first 105 days of historical records as the training set, the next 30 days as the validation set, and the last 15 days as the testing set, respectively.

3.6.3 Evaluation Metrics

In the experiments, five metrics are applied to evaluating the prediction performance of the different algorithms combined with the proposed method, namely, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), and Symmetric Mean Absolute Percentage Error (SMAPE). SMAPE provides a result between 0% and 200%. $\hat{y}_i$ and y_i denote the predicted and actual values, respectively. The definitions of the five metrics are written as follows.

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}|\,\hat{y}_i - y_i\,| \tag{3.12}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2} \tag{3.13}$$

$$\mathrm{MedAE} = median(|\,\hat{y}_1 - y_1\,|, \ldots, |\,\hat{y}_i - y_i\,|) \tag{3.14}$$

$$\mathrm{SMAPE} = \frac{200\%}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{|\hat{y}_i| + |y_i|} \tag{3.15}$$

3.6.3 Model Comparison

With detailed numerical results, Table 3.3 shows the performance of our proposed framework in predicting traffic information over the entire data set with the chosen testing LSTM. An Imputed dataset can achieve the best performance in RMSE, MAE, MedAE, and SMAPE across nearly all stations. When the Imputed dataset is used for prediction, the performance is better than the Non-Imputed dataset. According to the prediction results, the rule-based data imputation is an effective strategy for pre-processing the entire data set.

Table 3.3 Arrival Delay and Departure Delay Prediction Performances

Model	**Station**	**RMSE (sec)**		**MAE (sec)**		**MedAE(sec)**		**SMAPE (%)**	
		Arrival	Departure	Arrival	Departure	Arrival	Departure	Arrival	Departure
Non-Imputed	ES	48.4	53	42	45	88.4	46.1	169.2	131.2
	KCS	54.5	54.9	47.3	47.2	45.2	48.3	168.9	114.6
	MPS	63.8	77.7	53.9	63.6	47.2	65	167	121.3
	THS	65.7	76.6	55.5	56.9	147.6	43.5	137.8	102.6
	CS	65.6	61.8	51.7	41	46.9	32.4	153.2	151.7
	RS	63.5	77.3	46.5	55.7	28.7	45.7	167.3	156.5
	WCS	51.9	55.5	45.8	48.8	47.6	57.5	128	118.5
Imputed	ES	23.6	48	13.4	38.8	1.8	25.4	24.5	57.3
	KCS	21.7	39.5	15.6	27.9	7.9	18.4	27.9	40.9
	MPS	35.8	51.4	27.3	40.6	22.4	39.2	35.3	47.4
	THS	47.1	53.3	34.3	37.6	24.5	22.8	36.1	55.6
	CS	38.1	50.1	24.6	33.7	13.2	16.3	33.6	39
	RS	40.9	39.8	25.8	23.9	11.4	12.1	35	23.8
	WCS	49.1	55.9	32.1	36.9	12.1	12.3	51.4	49

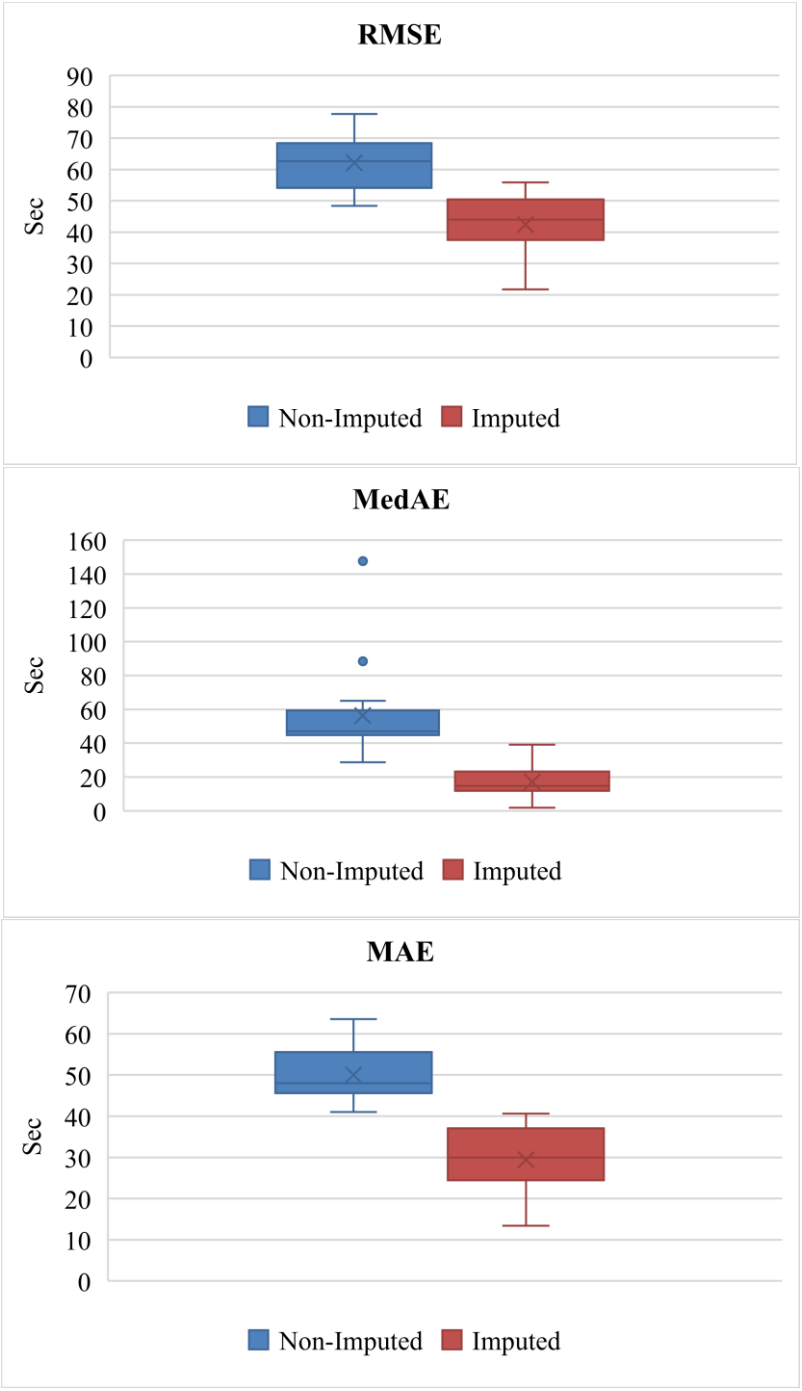


Figure 3.9 RMSE, MAE, and MedAE for Arrival and Departure Delays

Figure 3.9 presents a comparison of RMSE, MAE, and MedAE errors produced by Non-Imputed and Imputed datasets for arrival and departure delay predictions. Due to the complexity and noise in the GTFS real-time data, the data pre-processing can identify and remove most of the impact of outliers from the raw data. Additionally, the data pre-processing can also reduce the impact of missing values more effectively. It can be seen

from the Figure 3.9 that the Non-Imputed has two prediction anomalies (blue points) at MedAE. As a result, the Non-Imputed indicates large prediction errors among the three error metrics. Furthermore, the imputed strategy indicates better performance than other models in terms of the maximum, minimum, and median errors. The error distribution of the Imputed is more concentrated in the distribution of errors. In contrast, the Non-Imputed has relatively larger errors.

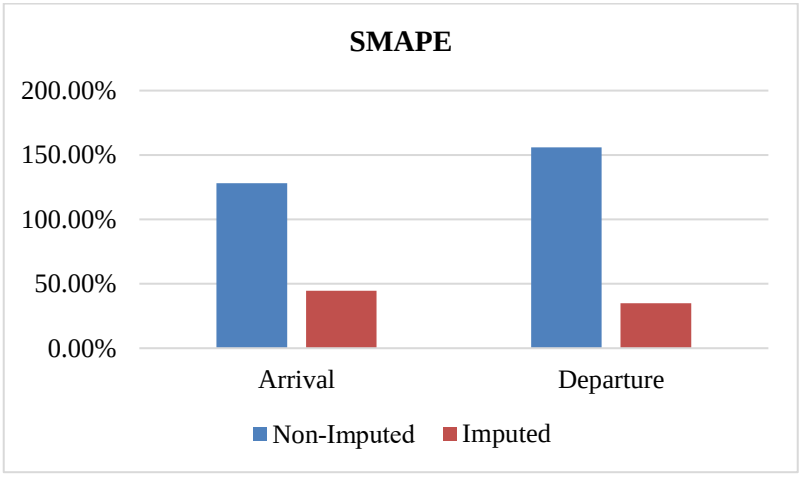


Figure 3.10 SMAPE of the Predictions

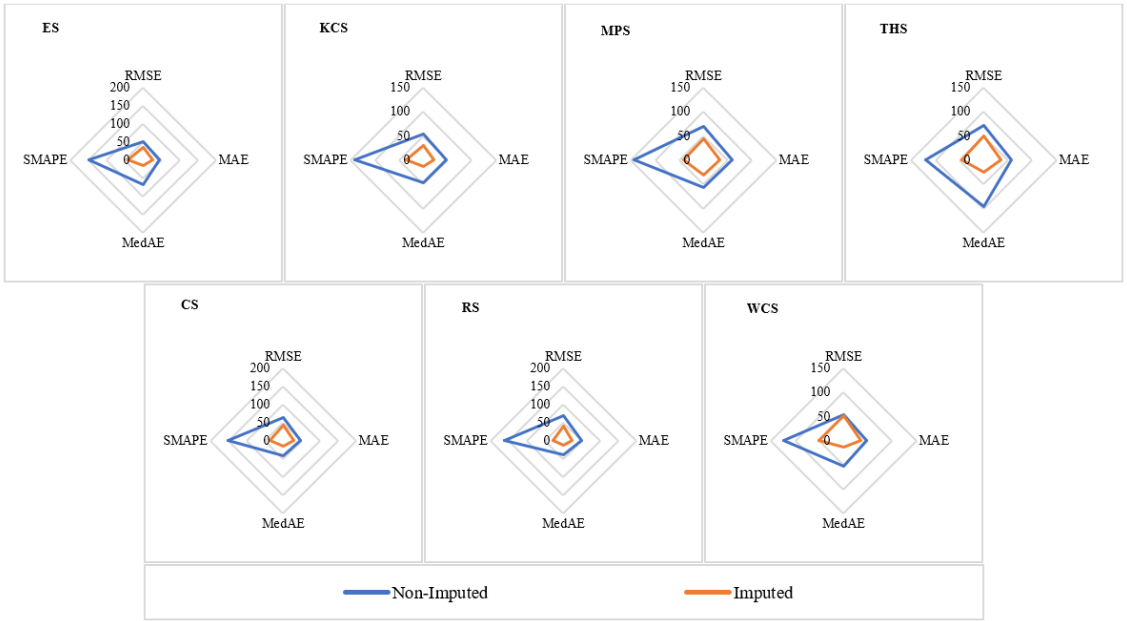


Figure 3.11 Performance Evaluation at Selected Train Stations

Figure 3.10 shows that the Imputed has the lowest percentage of errors for the arrival delay and departure delay at SMAPE. Additionally, Non-Imputed has the lowest accuracy in terms of SMAPE. In summary, it can be seen that the Imputed dataset provides a sufficiently good prediction performance in terms of validation metrics.

To validate the robustness of the proposed framework, we further compare the performance of the proposed model on Non-Imputed datasets and Imputed datasets for selected train stations, as shown in Figure 3.11. In general, the proposed method produces less errors and shows a more stable prediction ability among four error indexes. According to the results, using the imputed strategy can effectively develop a DAP framework to produce a GTFS integrated dataset. After identifying the best combination, all missing values can be replaced by the imputed strategy for the arrival delay and departure delay. With minimized anomaly, these datasets can be used to feed a diversity of deep learning models.

3.7 Summary

This chapter proposed and implemented a DAP framework to process the GTFS data in both static and real-time formats. A comprehensive data cleaning workflow and a state-of-the-art BSTS based time-series outlier detection method combined with a rule-based data imputation algorithm were proposed. An MM-LSTM model was applied to exploring the Non-imputed and Imputed datasets for predicting delay information in the numerical experiments.

Overall, even though the GTFS data would be complex and noise-intensive, our proposed method could create a high-quality time series dataset for traffic information forecasting with deep learning algorithms. The new method and tool can also be used to solve the real-time delay prediction tasks, including regular and irregular delays. It is challenging to obtain delay records caused by abnormal events. GTFS data provides a

unique opportunity to obtain such valuable information. Data quality is the basis of data-driven modeling. However, such a work was overlooked in literature and the proposed GTFS DAP framework in this chapter aimed to fill the gap.

The fused GTFS data is useful for traffic information forecasting and generates great potential to complement other data sources, such as APC and AFC. Furthermore, GTFS real-time Vehicle Position and GTFS real-time Service Alerts can be fused by modifying our proposed DAP framework. Since the primary purpose of the chapter is to design a practical and ready-to-use DAP framework, some other methods, such as Hamiltonian Monte Carlo for outlier detection and Generative Adversarial Network for data imputation, have not been involved. These models should be evaluated and compared in future studies.

Chapter 4

A General Prediction System Framework for Primary Delays

This chapter designs a comprehensive and general data-driven primary delay prediction framework in which we consider the advantages of expert systems and machine learning. Our solution uses LSTM and CPS to generate forecasts for train delays. The LSTM tackles the tasks for long-term predictions of running time and dwell time. The CPS utilizes past (observed) or future (projected) values with a nominal timetable to identify past or future primary and secondary delays based on the delay causes, run-time delay, and dwell time delay. Based on this framework, we have also used an open-source data collection and processing tool from the DAP that reduces the barrier to using the different open data sources. Finally, we demonstrate an advanced deep learning model, the novel ConvLSTM Encoder-Decoder model with CPS, for better primary delay predictions. Finally, we demonstrate the performance of the standard LSTM and its variants applied in a novel multivariate architecture.

The rest of the chapter is organized as follows. Section 4.1 introduces the description of the primary delay prediction problem. Section 4.2 describes the proposed train primary delay prediction system framework. Section 4.3 presents how to use the GTFS static and the GTFS real-time data to build and test the proposed models for univariate and multivariate time-series predictions. Section 4.4 summarizes our experimental results, and finally, Section 4.5 concludes briefly.

4.1 Primary Train Delay Prediction Problem

The train delays are divided into two categories: primary delays and flow-on delays. The flow-on delay, which also is referred to as the secondary delay, is caused by the primary delay [132, 133]. From a system perspective, there are two approaches to prevent delays from spreading out by either making a more robust timetable or avoiding the occurrence of primary delays [134]. During the peak hours in urban railways, trains are operated quite densely. Once a delay occurs, it could be easily propagated to the succeeding trains. Thus, if we can predict and reduce the primary delays, the propagated delays can be reduced or avoided accordingly. This leads to great alleviation of humans' effects on the traffic management system.

According to [15] and [134], a railway network is considered as a graph where nodes indicate a series of checkpoints $\mathrm{C} = \{\mathrm{C1}, \mathrm{C2}, \cdots, \mathrm{Cn}\}$ successively connected. For any checkpoint C, a train arrives at the time $s_{t_A}^{C}$ and departs at a time $s_{t_D}^{C}$ in the scheduled timetable, where *t* denotes a timestamp. The actual arrival and departure times of a train are denoted as $a_{t_A}^{C}$ and $a_{t_D}^{C}$. The differences of $(a_{t_A}^{C} - s_{t_A}^{C})$ and $(a_{t_D}^{C} - s_{t_D}^{C})$ are defined as the arrival and departure delays, respectively. A train is delayed if its delay is greater than the 30s (or 1 min), generally. Additionally, a dwell time is obtained by calculating the difference between the arrival and departure time $(a_{t_D}^{C} - a_{t_A}^{C})$, while a running time is gained by calculating the difference between the departure time of the current checkpoint and the arrival time of the next checkpoint $(a_{t_A}^{C+1} - a_{t_D}^{C})$.

The primary delay detection problem is to predict the checkpoints that will have the first delay, which will cause delays in succeeding checkpoints. If the delay occurs, we can quickly predict which stations will also have a primary delay in the future. Traffic operators, based on the information, reschedule the train network in a timely and accurate manner, thereby reducing the number of stations that are delayed or even avoiding the train network failure. Therefore, the primary delay prediction is a crucial task in the field

of the railway management system.

4.2 Primary Delay Prediction Framework

Typically, an expert system includes knowledge bases, an inference engine, and user interfaces. An Inference Engine mainly contains two types of algorithms: Forward Chaining Algorithm (FCA) and Backward Chaining Algorithms (BCA) [135]. Inspired by Spring's work [135], a knowledge-based AI system in intelligent transportation systems (ITS) is derived, as shown in Figure 4.1. An expert system is to mimic the intelligence and function of domain experts.

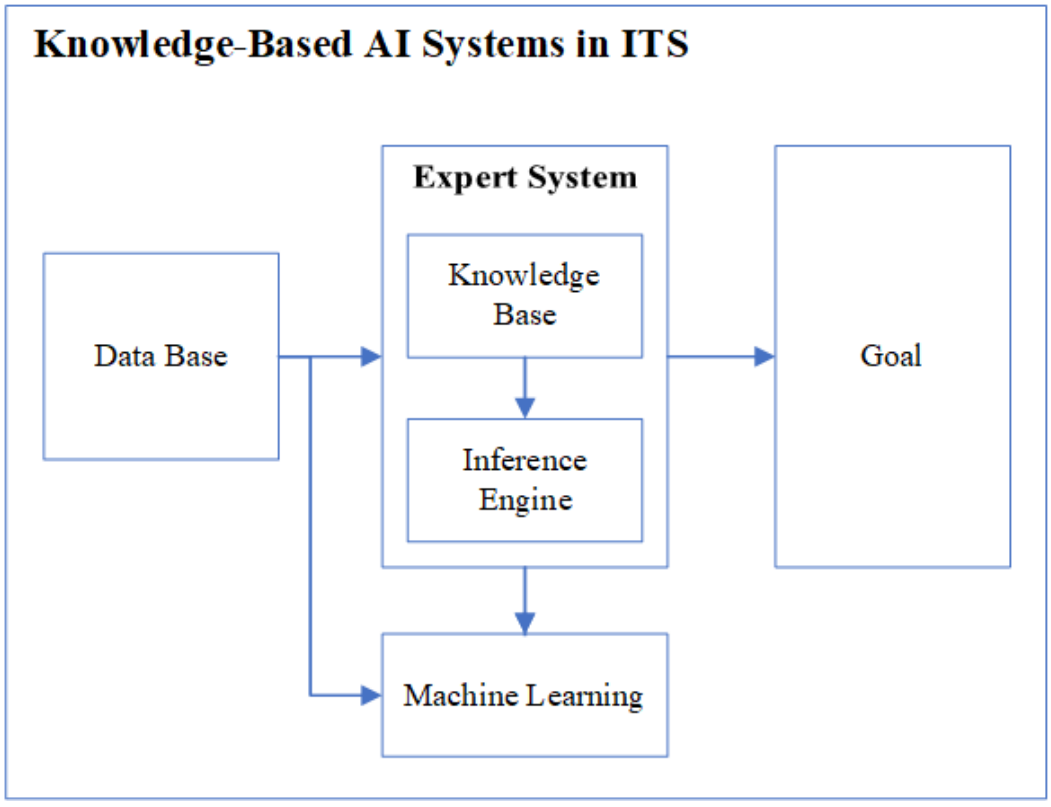


Figure 4.1 Knowledge-Based Artificial Intelligence System

On the other hand, machine learning methods aim to apply complex mathematical calculation-based algorithms to explore the relationships among large-scale data. To build a practical PDPS, the advantages of the expert system and machine learning should be integrated to achieve successful and scientifically useful predictions. The entire PDPS

framework is roughly divided into four main modules: database, knowledge base, inference engine, and machine learning component.

As depicted in Figure 4.2, each component is composed of multiple corresponding subcomponents. Firstly, we develop a data collector to collect real-time train data to establish a database. Secondly, for having a knowledge base, we implement a data preprocessing tool to fuse the data from two data sources, namely train schedules and associated geographic information. As structured information is created efficiently, we deploy the knowledge base on the cloud server for long-term data storage. Additionally, our model only uses data from the knowledge base to predict train delays; therefore, the overall calculation time of the entire system is greatly reduced. Thirdly, we propose a critical point search algorithm to integrate domain knowledge as an inference engine to categorize the data and find the primary delays. Finally, deep learning models are applied to achieve accurate predictions. As a result, the system extracts valuable information, which is directly visualized to the system users for the planning and control rail services at the operational level. To the best of our knowledge, this is the first work that provides a comprehensive and conceptual framework for the design of the combination of expert systems in PDPS and deep learning. The system that performs causal reasoning in the delay prediction task is illustrated in Figure 4.2.

On the other hand, machine learning methods aim to apply complex mathematical calculation-based algorithms to explore the relationships among large-scale data. To build a practical PDPS, the advantages of the expert system and machine learning should be integrated to achieve successful and scientifically useful predictions. The entire PDPS framework is roughly divided into four main modules: database, knowledge base, inference engine, and machine learning component.

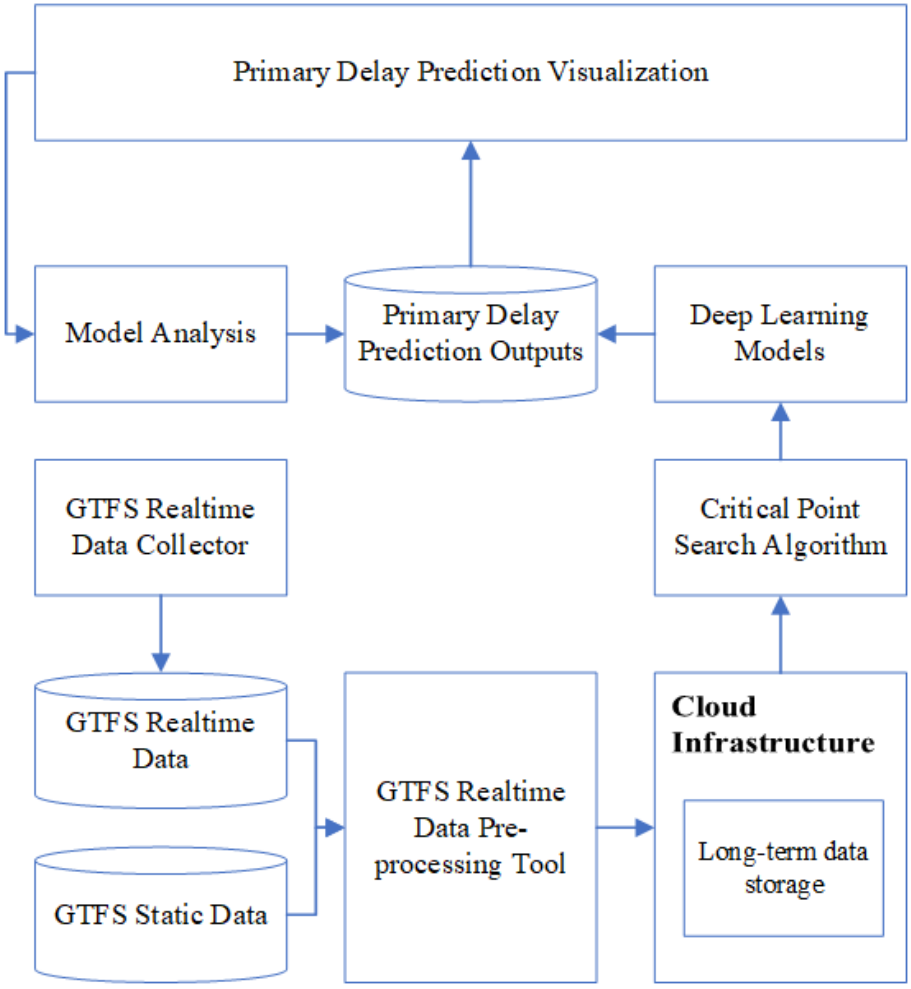


Figure 4.2 Framework of PDPS

4.2.1 Critical Point Search Algorithm

The motivation to use the critical point search algorithm is to identify primary delays and secondary delays. Our proposed algorithm is employed in the time series forecasting models to improve the prediction of the primary train delays, which are the causes of a lot of secondary delays due to tracing causality. Since no existing studies, to the best of our knowledge, analyze the primary delay scenario with machine learning approaches and conduct delay classifications, this is a novel design in the train delay prediction field.

In order to predict the primary and secondary delays, firstly, we need to calculate the difference among the actual departure time $a_{t_D}^C$, the actual arrival time $a_{t_A}^C$, the

scheduled departure time $s_{t_D}^C$ and the scheduled arrival time $s_{t_A}^C$. Subsequently, a few difference values are calculated from the following equations.

(a) The difference D_1 between the actual arrival time and the scheduled arrival time:

$$D_1 = a_{t_A}^C - s_{t_A}^C \tag{4.1}$$

(b) The difference D_2 between the actual arrival time at a timestep $\boldsymbol{t}$ and the actual departure time at a timestep $t-1$:

$$D_2 = a_{t_A}^C - a_{t_D}^C \tag{4.2}$$

(c) The difference D_3 between the scheduled arrival time at a timestep $\boldsymbol{t}$ and the scheduled departure time at a timestep $t-1$:

$$D_3 = s_{t_A}^C - s_{t_D}^C \tag{4.3}$$

(d) The difference D_4 between D_2 and D_3:

$$D_4 = D_2 - D_3 \tag{4.4}$$

Algorithm 4.1 Critical Point Search Algorithm

Require: Input all train data $\boldsymbol{R_t} = (\boldsymbol{R_1^t}, \boldsymbol{R_2^t} \ldots \boldsymbol{R_N^t})$**,** and pre-defined thresholds $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$

Output: $\boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}$

for each train $\boldsymbol{R} = (\boldsymbol{a_{t_A}^C}, \boldsymbol{a_{t_D}^C}, \boldsymbol{s_{t_A}^C}, \boldsymbol{s_{t_D}^C})$ **do**

  $\mathbf{D_1} = \boldsymbol{a_{t_A}^C} - \boldsymbol{s_{t_A}^C}$

  if $\mathbf{D_1} >= \boldsymbol{V_1}$:

    $\boldsymbol{D_4} = \mathbf{D_2} - \mathbf{D_3}$

    if $\boldsymbol{D_4} >= \boldsymbol{V_2}$:

      $\boldsymbol{W_1}$

    else:

      $\boldsymbol{W_2}$

  else:

    $\boldsymbol{W_3}$

end for

To find the primary points, an inference engine using forward chaining searches the critical points until it finds the points where $D_1 \geq$ the first threshold value V_1 and $D_4 \geq$ the second threshold value V_2. For an initial checkpoint, only D_1 is used to find the primary points. The Critical Point Search Algorithm (**Algorithm 4.1**) is summarized as follows. By calculating differences of respective train arrival or departure times, the different category of delay or on-time points are added to the corresponding lists, W_1 ,W_2 , W_3. The output W_1 denotes a list of primary delay points, W_2 a list of secondary delay points, and W_3 a list of running on-time points.

4.2.2 LSTM Neural Networks for Multi-Step Time Series Forecasting

Feed-forward neural networks, such as CNN, cannot directly capture temporal dependencies between successive timesteps. To overcome the limitation of temporal dependency capture, a recurrent neural network was proposed to study temporal correlations without learning trends and seasonality. It is a commonly used and effective tool for sequence prediction problems. LSTM networks are capable of solving many tasks of the time series by using fixed-length time windows [136]. They have stacked to accurately model complex patterns of multivariate sequences [137].

In this chapter, we follow the version of FC-LSTM from [138]. A standard LSTM contains two types of states: the long-term state and the short-term state. Specifically, the short-term state h_{t-1} and the current input X^d are fed to three Sigmoid functions and a $tanh$ activation function. Moreover, the long-term state c_{t-1} traverses the network to c_t. Firstly, it drops some memories in the forget gate. Secondly, some new memories are added to the input gate by using the addition operation. And then, the long-term state is passed through the $tanh$ activation function. Finally, the output gate filters the results, generating the output y_t and the short-term state h_t (y_t is equal to h_t at time step).

Shi et al. introduced a convolutional LSTM (ConvLSTM) architecture, which is a

combination of convolutional and LSTM layers [139]. Based on the state-of-the-art encoder and decoder design, Gehring et al. proposed a fully convolutional model structure for sequence-to-sequence learning, which achieved superior performance over the strong recurrent models on machine translation tasks [140]. According to Shi et al.'s work [139]. The ConvLSTM included the convolution operator *. It used convolution structures directly in both the input-to-state and state-to-state transitions. Thus, the model is suitable to encode information for spatiotemporal data.

The input-to-state filters determine the output (W_{xi}, W_{xf} , W_{XC}, W_{xo}) and state-to-state filters (W_{hi}, W_{hf},W_{hc}, W_{ho}). The input X^d at the time step t is the historical arrival or departure delay time. The final output is the predicted arrival or departure delay time, respectively. ConvLSTM does not have negative number predictions by using nonlinear activation function at each of ConvLSTM layers and Rectified Linear Unit (ReLU) activation function at the fully connected (FC) layer. It is vital for delay forecast models to predict positive times from the positive times of historical data.

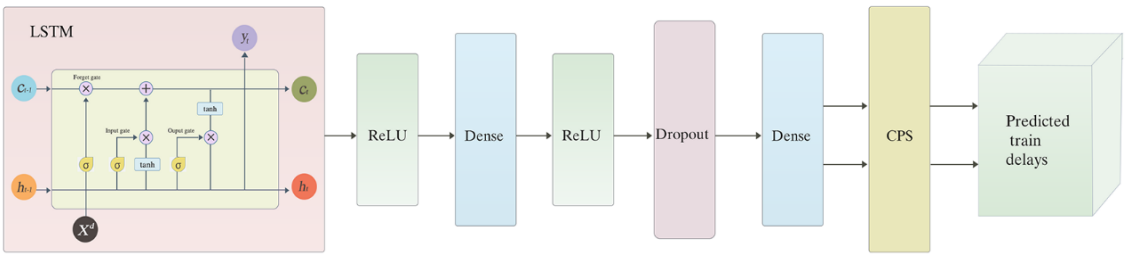


Figure 4.3 LSTM-CPS Network Topology

As depicted in Figure 4.3, the design of a comprehensive deep predictive learning architecture is demonstrated for multivariate train delay predictions. A multivariate multi-step forecasting model is composed of an LSTM, and two rectified linear units (ReLU), two dense layers, a dropout layer, and a CPS component. The last dense layer outputs a range of future values. The CPS classifies the predicted values, which not only find the

primary delays, secondary delays, and on-time running of corresponding stations but also show the status of the whole trip in the future from a given history. Furthermore, different machine learning methods and model hyperparameters are included in our studies.

4.3 Data Preparation

For evaluation, the proposed models are applied to a Sydney Train GTFS dataset from the NSW open data hub, which unlocks its data to share with developers, researchers, and data analytic organizations, and offers exciting opportunities for them to create an innovative solution for diverse stakeholders [13]. The raw data with a frequency range of 10 to 30 sec is extracted from the real-time GTFS that has a large amount of data every day. For example, collecting GTFS trip updates of Sydney Trains with a 10-sec frequency generates a dataset between 2 and 4 GB per day, which is preprocessed into a data set between approximately 3 and 6 MB dataset. Such open-source data have great potential to be preprocessed to carry out a longitudinal study in rail transportation.

4.3.1 Univariate Analysis

After removing duplicated data, we pre-processed the dataset as follows. As shown in Figure 4.4, the means of the daily arrival delay and departure delay data for the entire railway network can be calculated. Specifically, according to Figure 4.4, we sorted the March 26 data and found many delays longer than 30 minutes. Moreover, ConvLSTM's future state of a cell in the grid is determined by the input and past state of its local neighbors [139]. Based on our experiments, the prediction error could increase significantly when an outlier is used as an input at a timestep close to the timestep of the output. Hence, such data cannot be harnessed to predict the next day's delay times.

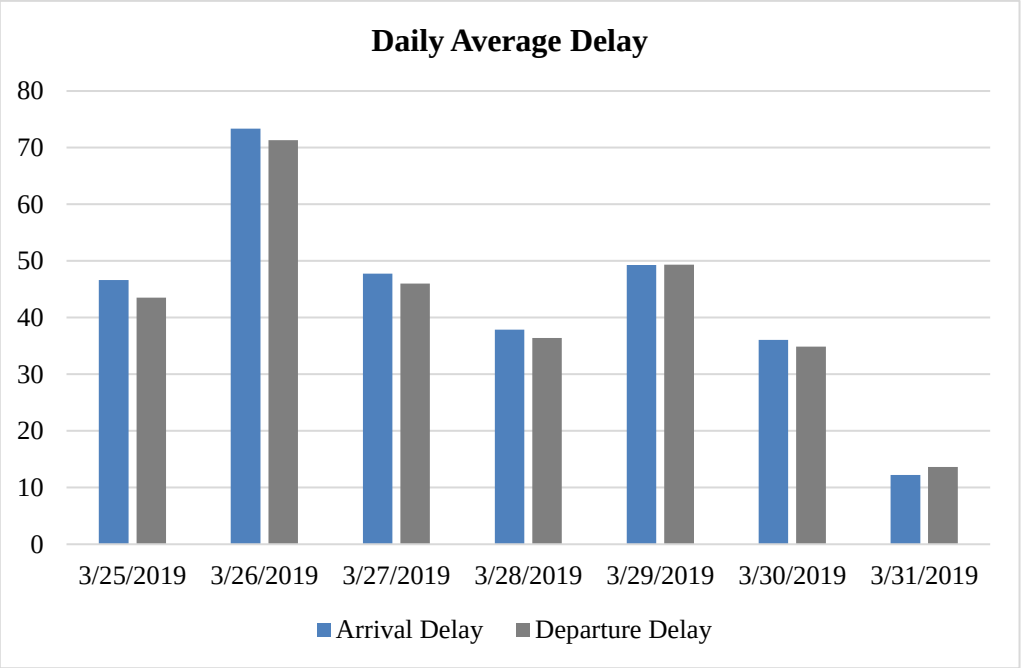


Figure 4.4 Daily Average Delay (sec)

We proposed critical point search rules that can classify data efficiently and reasonably. The algorithm (**Algorithm 4.1**) limits the upper and lower bounds of the data through a set of rules to split the dataset into three lists (W_1, W_2, W_3). It can be easily modified and extended to generate more categories of the list for the actual prediction, for instance, a list for special events. Furthermore, the entire train network consists of 8 lines, namely T1, T2,⋯T8, where each line has multiple routes, and each route has multiple trips per day; and also, the total number of nodes (stations) are different among the trips. Since each trip has a unique reference number, and there are no obvious systematic time-dependent patterns among the difference trips, if we simply split the dataset into the training and the test sets and then apply the deep learning model to predict the delays by using the datasets, the predicted results could be erroneous and not convincing. Besides, to utilize LSTM models for supervised learning in sequence data, we need to predefine the number of subsequences and the length of subsequences to determine the number of nodes we expected to forecast. Thus, for train delay prediction, the model only predicts a trip of one checkpoint at a time. If we would like to complete

calculations for all checkpoints quickly, parallel computing can be used to perform all calculations simultaneously.

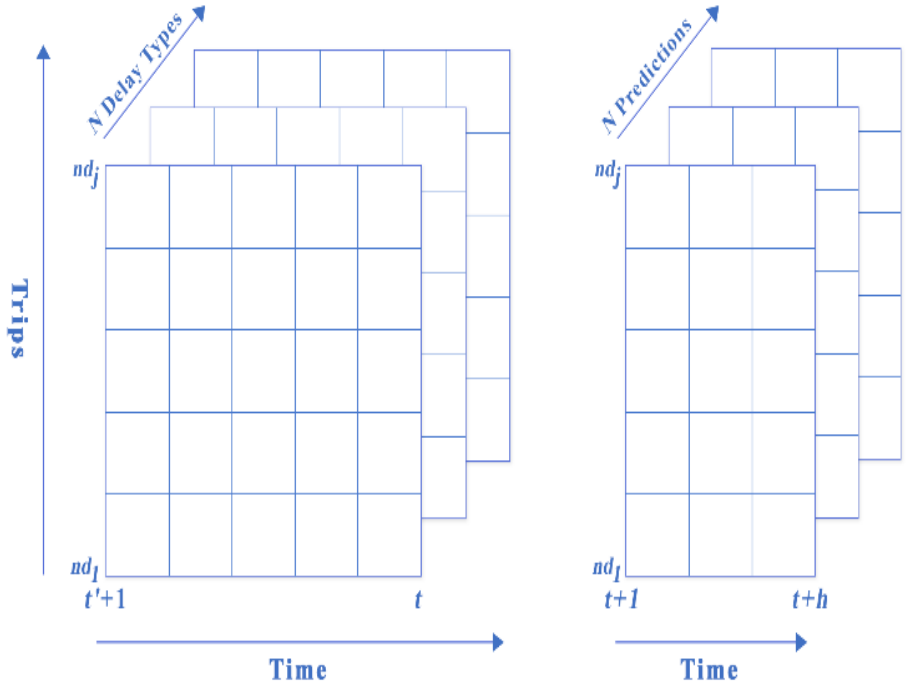


Figure 4.5 Input and Output Shapes

It is worth mentioning that the model we proposed is a generic model, which does not depend on a specific data set. The ConvLSTM model can learn long-term correlation in a sequence and capture the spatiotemporal patterns by using good quality input data. Figure 4.5 shows that *N* delay categories of samples, where t' is an initial time, and each trip with a window of the historical time steps from $t' + 1$ to t. nd_1 to nd_j indicate the number of trips. The outputs include *n* delay predictions at *h* time-steps ahead, $t + 1, \cdots, t + h$. For multiple trips, the input samples are sampled at non-fixed time resolutions to predict the outputs. Therefore, the data is transformed into a two-dimensional format oriented to supervised learning (train and test data in a tabular form). Specifically, our design is to split the trips into three tabular forms by using CPS. For further study, when a checkpoint occurs a primary delay, we use Bayesian Learning to calculate the probability of a trip at subsequent checkpoints, at which events occur (the primary delay,

secondary delay, or on-time running). The framework from this chapter can be applied to generate a delay prediction model to estimate the arrival delay time or departure delay time for each type of events.

4.3.2 Multivariate Analysis

For multivariate analysis, the dataset consists of 161-day data observations for the trip number '600D' in the period Apr. 11, 2019 to Nov. 21, 2019. The train line is from Sydney's Bondi Junction Station, Platform 1 to Hurstville Station, Platform 4 in the morning peak hours from Monday to Friday.

The available data has spatial data from multiple train stations and temporal data for each station. The train delay data is interpreted as a time-series format, which can be processed and transformed into many sequential matrices. The sequential model learns the spatio-temporal features by using the arrays as input shapes. Consequently, the correlations of independent variables for train delays among different stations are utilized during the predictive model training.

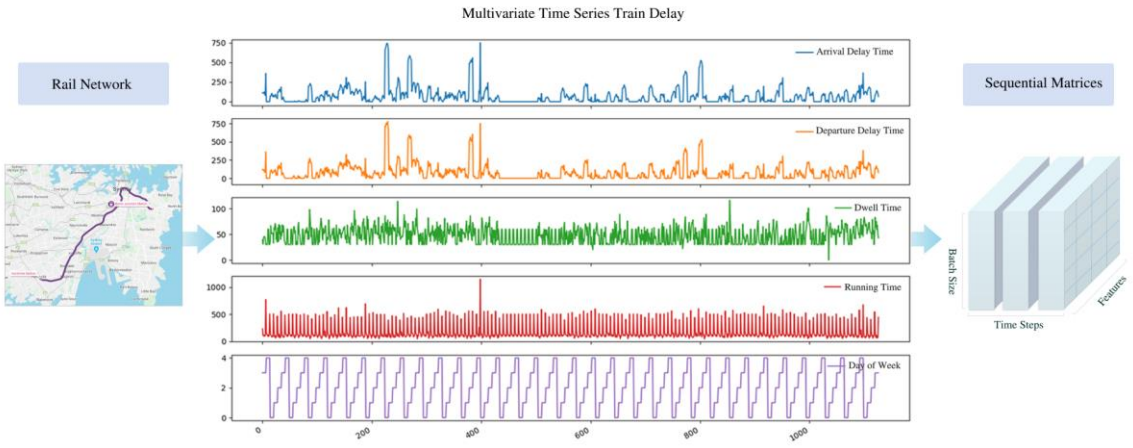


Figure 4.6 Graphical Representation of Feature Generation Process

Figure 4.6 demonstrates our architecture of data visualization for the feature generation process. Firstly, a pre-planned train trip with possible delays is included. The available data has spatial data from multiple train stations and temporal data for each station. The train delay data is interpreted as a time-series format, which can be processed

and transformed into many sequential matrices. The sequential model learns the spatio-temporal features by using the arrays as input shapes. Consequently, the correlations of independent variables for train delays among different stations are utilized during the predictive model training.

For a sequence-to-sequence model, the relevant time series as inputs are denoted by X^d,where $d = 1,2,\cdots,D$. The illustration of X^d is a set of sequential matrices. We have a sequence of train delays, X^d, which can be denoted in vector form as $X^d = [X_t^1, X_{t+1}^2, X_{t+2}^3, \cdots, X_{t+n-1}^d]$. Each matrix with a time-step is from t to $t+n-1$. For example, in X_t^1, the time steps of the lag w are from $t-w+1$ to t, which can be expressed explicitly in matrix form as

$$X_t^1 = \begin{bmatrix} x_t^1 & x_t^2 & \cdots & x_t^d \\ x_{t-1}^1 & x_{t-1}^2 & \cdots & x_{t-1}^d \\ \vdots & \vdots & \ddots & \vdots \\ x_{t-w+1}^1 & x_{t-w+1}^2 & \cdots & x_{t-w+1}^d \end{bmatrix} \text{.....(5.6)}$$

All features are standardized to the range $[0, 1]$ before being passed to the predictive learning model.

4.4 Model Comparison

RMSE, MAE, mean absolute percentage error (MAPE), coefficient of determination (R^2), and Adjusted R-squared (Adjusted R^2) are applicable measures to evaluate the efficiency of the proposed prediction models. They have been defined as indicated in Equation (4.7) – Equation (4.9), where y_t is the actual time for sample *t* and $\hat{y}_t$ is the predicted time. As the multi-time-step model predicts train delays for all *r* trips for the next *n* time-steps, both y_t and $\hat{y}_t$ have the dimensionality $h \times r$ on the univariate analysis and have the dimensionality $h \times r \times d$ on the multivariate analysis.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_t - \hat{y}_t}{y_t}\right) \quad (4.7)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{i=1}^{n}(y_t - \bar{y})^2} \tag{4.8}$$

$$Adjusted\ R^2 = 1 - \frac{n-1}{n-(k+1)}(1 - R^2) \tag{4.9}$$

4.4.1 Univariate Prediction Results

To obtain better predictions, we repeat the evaluation of the same model configuration on the same GTFS dataset and then estimate the average performance of the prediction models. For this experiment, we explored the patterns of train delays on weekdays. Table 4.1 shows the results of the trip number "146U" at the "Seven Hills Station Platform 2" delay forecast, using GTFS data between January 29, 2019, and April 2, 2019, and GTFS-Static data. The advantage of integrating GTFS data is that we have more information about each station, such as station name, coordinates, node number, route name, and so on. As evidenced by the results, except for the slight difference in the performance of CNN, the performance of three types of LSTM does not have much different. Our results are consistent with Greff et al.'s findings as well [141].

Table 4.1 Results of the Models without CPS

Model	MAE (sec)		RMSE (sec)	
	Mean	SD	Mean	SD
CNN	84.51	4.52	136.13	3.83
Pure LSTM	80.61	1.25	133.17	1.26
CNN-LSTM	79.64	1.90	134.08	1.84
ConvLSTM	79.49	0.83	133.61	0.60

After applying CPS to find higher than 40-sec primary delays for the trip number "146U" at an initial station, "Emu Plains Station Platform 2", the results of the proposed models indicate the different results. The main reason for using the different stations is

that CPS can remove the outliers at the same station, which means that the non-primary delayed data is not considered for primary delay prediction. Hence, the forecast result is improved.

Although Pure LSTM performs well on the given dataset, we found CNN, Pure LSTM, and CNN-LSTM perform negative values for delay predictions, which are abnormal values. Additionally, ConvLSTM's mean and standard deviation (SD) is higher than Pure LSTM's in Table 4.2, whereas it has the smallest SD in Table 4.1. To sum up, ConvLSTM is more stable than other models to make predictions based on data with large residuals. Notably, it also performs accurate forecasts that are closer to the ground truth.

Table 4.2 Results of the Proposed Models

Model	MAE (sec)		RMSE (sec)	
	Mean	SD	Mean	SD
CNN	48.63	14.08	53.65	14.75
Pure LSTM	16.82	1.19	18.66	1.18
CNN-LSTM	34.97	4.53	37.62	5.29
ConvLSTM	37.56	3.48	42.63	3.89

In predicted primary train delay results, the algorithm assumes that all the predicted train running is the same as the actual train running time, and this assumption is unrealistic. The recommended algorithm would be sensitive to the values of V_1 and V_2 . Normally, V_1 or V_2 should be greater than 30 or 60 seconds. Our proposed model could capture long-term correlation in sequence learning. Inspired by Yamamura's work [132] , as the prediction error exists, to accurately find the primary delay, the value of an offset needs to be calculated and be involved with the predicted output data. Therefore, to

develop a primary delay prediction system, V_1 , V_2, offset should be suggested by the domain experts, who can estimate the values based on reality.

4.4.2 Multivariate Prediction Results

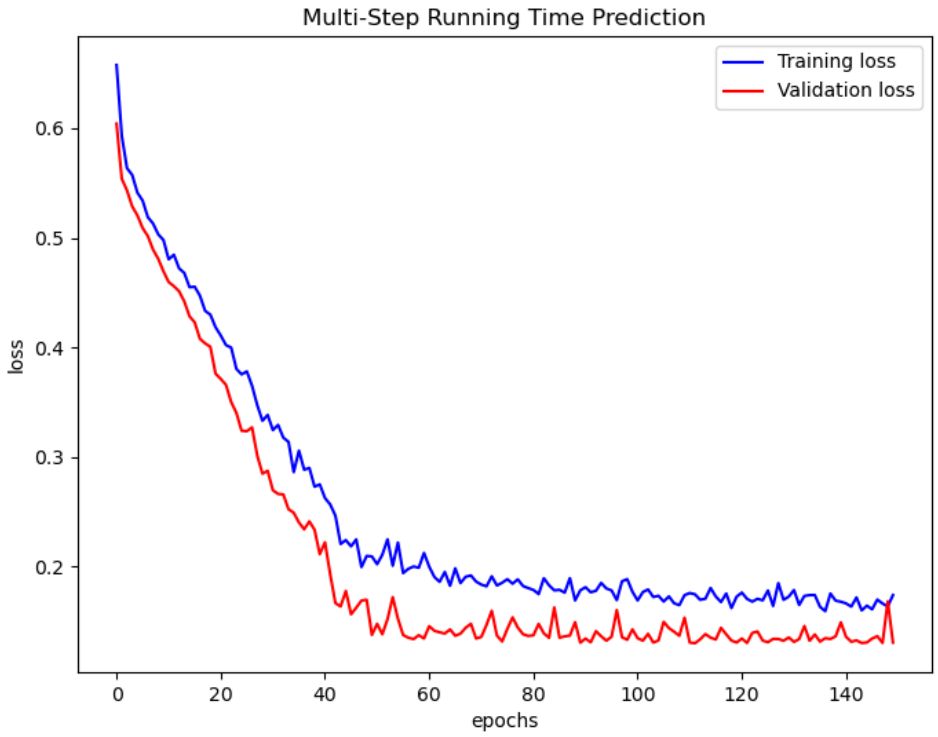


Figure 4.7 Training and Validation Loss for a Multi-Step Running Time Prediction with MAE as Loss Function

For training a multivariate deep learning model, the first 100 days of the 161-day dataset is compiled as the training dataset and the remaining data as the validation dataset, that roughly 70/30 split. According to the network structure of Figure 4.3, we select a batch size of 32 and 150 epochs as the parameters of the training model. illustrates the loss during the training and validation procedures. Both the training loss and validation loss converge after approximately 60 epochs. MAE is exploited as a loss function in the training process. Since the performance difference between training and validation is acceptable, the model does not overfit the training data.

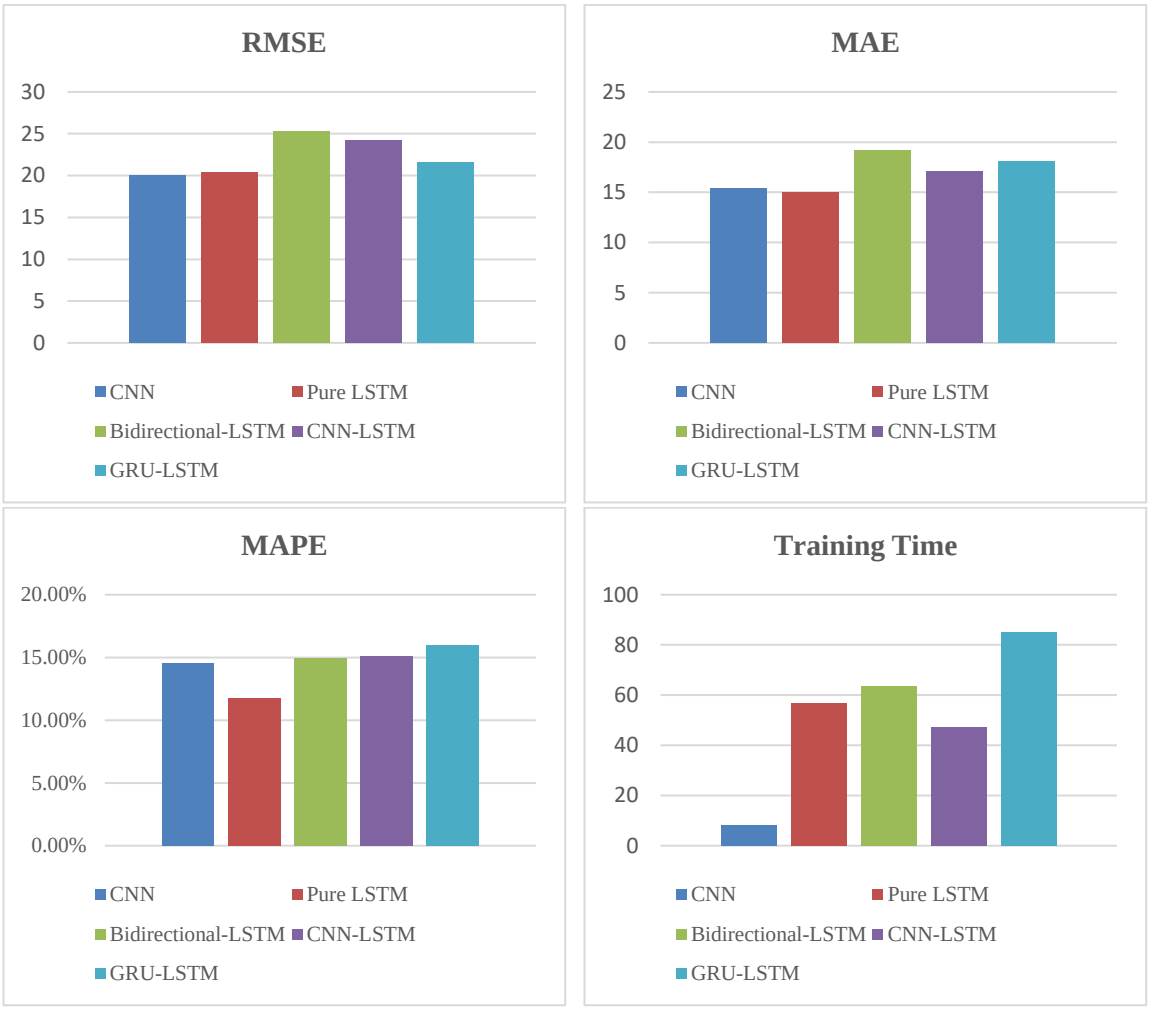


Figure 4.8 RMSE, MAE, MAPE, and Training Time for Running Time Prediction

Figure 4.8 reports the performance of our proposed models with different variants or combinations of CNN, LSTM, and GRU for predicting running time from one day ahead (t+7). All *x*-axes express the corresponding models. The *y*-axis of RMSE and MAE represents the error in seconds; the *y*-axis of MAPE denotes the proportion of the error; the *y*-axis at the lower right of the figure represents the training time (seconds) of the corresponding models. All models do not have significant performance differences. The standard LSTM has the lowest percentage error at MAPE. CNN reaches minimum training time. In summary, the standard LSTM has a sufficiently good prediction performance in terms of all validation metrics.

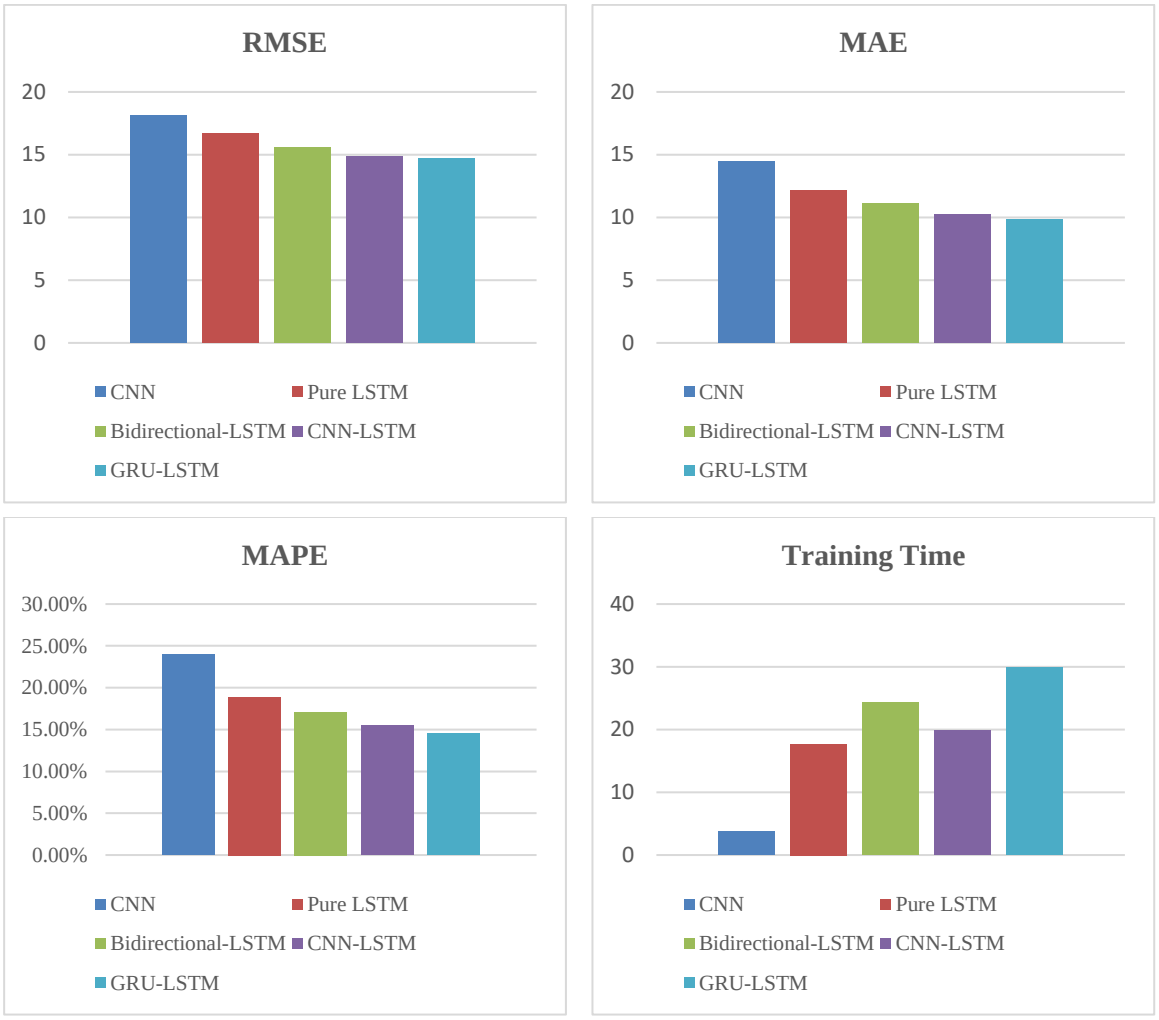


Figure 4.9 RMSE, MAE, MAPE, and Training Time for Dwell Time Prediction

The expressions of the *x*-axis and *y*-axis in Figure 4.9 are the same as in Figure 4.8, but the performance of the proposed models for dwell time prediction and estimation accuracies in terms of all validation metrics with one day ahead are reported instead. For RMSE, MAE, and MAPE, GRU-LSTM achieves the highest prediction accuracy; however, the time cost of the model is the highest. To sum up, compared with the other variants, the standard LSTM and CNN have more substantial errors and less training time. Variants do not have significant performance differences.

Table 4.3 and Table 4.4 demonstrate the results of predictive models. In Table 4.3, it is easy to observe that the training time used by CNN is the lowest. For 7-steps ahead prediction, Pure LSTM achieves lower MAE and MAPE compared to others. However,

in Table 4.4, GRU-LSTM performs well for 7-steps ahead prediction. Also, the differences between all variants in performance are slight.

Table 4.3 Running Time Prediction Performances

Model	RMSE (sec)	MAE (sec)	MAPE (%)	Training Time (sec)
CNN	20.0	15.411	14.583	8.3
Pure LSTM	20.419	14.973	11.779	56.6
Bidirectional-LSTM	25.256	19.136	14.907	63.7
CNN-LSTM	24.218	17.076	15.066	47.1
GRU-LSTM	21.527	18.086	15.960	85.2

Table 4.4 Dwell Time Prediction Performances

Model	RMSE (sec)	MAE (sec)	MAPE (%)	Training Time (sec)
CNN	18.128	14.491	24.014	3.8
Pure LSTM	16.670	12.125	18.897	17.7
Bidirectional-LSTM	15.586	11.147	17.107	24.3
CNN-LSTM	14.837	10.229	15.498	19.8
GRU-LSTM	14.717	9.844	14.480	29.9

For building a real-world deep learning application, we need to balance or even trade-off accuracy and training time, also prevent overfitting of the model. Time series data have a certain degree of randomness. In some cases, certain model approaches a high accuracy, but overfitting occurs. It dramatically reduces the portability of the model. In addition to accurately assess the maximum predictability of the dataset, in further studies, we need to explore automated model selection techniques to identify an appropriate

predictor for the corresponding inputs.

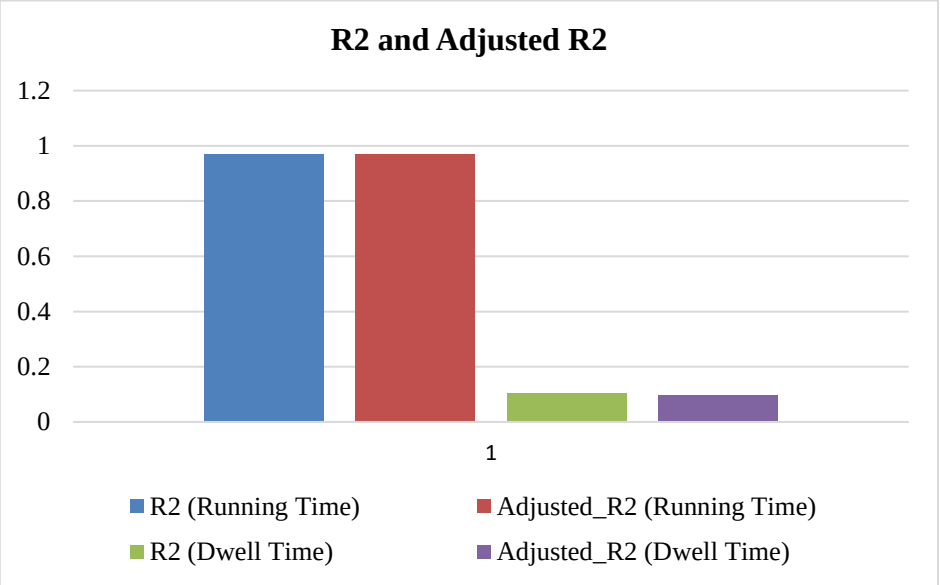


Figure 4.10 R-Squared and Adjusted R-Squared

Finally, we compared R-squared and Adjusted R-squared values resulting from the proposed models. Figure 4.10 reports a standard LSTM for running time prediction and best performance variants for dwell time prediction from one day ahead. Moreover, a baseline model predicts $\bar{y}$, which has default values $R^2 = 0$ and $R^2_{adj} = 0$. The chapter compares the proposed standard LSTM and variants against the baseline model. The result shows that the running time has a small variation with high accuracy, which is to a great extent explained by predictors in the standard LSTM. The proposed architecture presents reliable predictive power. Additionally, the running time shows a weak dependence on train delays. The impact of running time on the peak hours of workdays is also weak. Therefore, dwell time is more closely related to arrival delays.

Figure 4.10 shows the R-squared and Adjusted R-squared values for one day ahead. There is an amount of unexplainable uncertainty, which is represented by R-squared and Adjusted R-squared. Our model chooses the appropriate numbers of epochs to achieve the smallest value $\hat{h}_n$ through the method of Figure 4.7. If we increase the

number of epochs to reach the higher values of R-squared and Adjusted R-squared, it will cause the model to be overfitting or underfitting. Furthermore, the selected predictor variables cannot fully explain the variability of dwell times. It is a common consensus that using more relevant factors as an input of the model can reduce prediction errors and derive more accurate predictions of dwell time. For example, simply the dwell time is sensitive to the number of people waiting for the corresponding trains. However, accurately calculating the number of people getting on and off at the corresponding station is still an unresolved issue though some trials with IoT technologies are underway.

4.5 Summary

This chapter proposed a PDPS system framework for train delay prediction and identified the feasibility of using GTFS data for such studies. The system framework includes the GTFS data pre-processing tool, the critical point search algorithm, and deep learning models. The combination is to deal with big data in railways and achieve causality for delay event classifications. Moreover, this chapter established a hybrid deep learning architecture for long-term train delay prediction on real-world data collected from different sources. The presented value indicated that the LSTM could reach high accuracy by discovering the long-term temporal dependency patterns. Several deep learning models, including CNN, LSTM, and their variants, had been investigated to predict the running time and dwell time based on multivariate inputs. Moreover, CPS utilizes experimental results to implement the predicted primary delays and the predicted secondary delays.

The solution can be directly applied for long-term decision support in urban railway systems. It is a critical component for MaaS applications, and it assists in real-time forecasting. Meanwhile, the large amount of computation cost caused by deep learning models makes it difficult to quickly analyze and simulate the entire railway

network by using high-frequency real-time data based on long-term prediction performance. Our experiments classify the data of a single train line and forecast the corresponding stops of a single line. We will extend and apply the CPS to implement the data classification for the entire train network. We can use a stochastic probability model, such as a conditional Bayesian model, to effectively adjust the delay information as meaningful future work.

Chapter 5

Real-Time Forecast of Multi-Scenario Train Delays

In many big cities, train delays are among the most complained events by the public. Different from literature studies, this chapter focuses on finding the bound of improvements in predicting multi-scenario train delays with various machine learning methods in real-time.

The rest of the chapter is organized as follows: preliminary investigations of train delay are described in Section 5.1, including train delay problem and train delay prediction. Real entropy for maximum predictability, resilience inference formulations, a delay status labeling algorithm, and real-time multi-scenario delay predictions are introduced in Section 5.2. The performance of our proposed model is validated and further be applied to investigate real-time multi-scenario delay predictions in Section 5.3. Finally, Section 5.4 concludes briefly.

5.1 Preliminary Investigations

5.1.1 Train Delay Problem

A passenger train usually runs along a railroad track based on a regular schedule to serve a series of train stations successively in a trip. It follows an itinerary characterized by an original station s_1, a destination station s_k, and some intermediate stations $s_i, i = 2,3, \dots, k-1$. Train delay is described as the difference between the actual time t and the scheduled time $\hat{t}$, as shown in Figure 5.1. If the difference of departure times (i.e., $d^{s_1} -$

$\hat{d}^{s_1}$) or arrival times (i.e., $a^{s_1} - \hat{a}^{s_1}$) is greater than 30 sec or 60 sec, and then a train is considered to have a departure delay or an arrival delay, respectively. Running time between two stations, e.g., s_1 and s_2, is the difference between the arrival time at the current station s_2 and the departure time at the previous station s_1, such as $R_1 = a^{s_2} - d^{s_1}$. Dwell time is the difference between the departure time and the arrival time at the same station, e.g., s_2, such as $D_2 = d^{s_2} - a^{s_2}$. By computing the differences in the running time and the dwell time between the scheduled time and the actual time, train delay can be identified when a train departs from or arrives at a station.

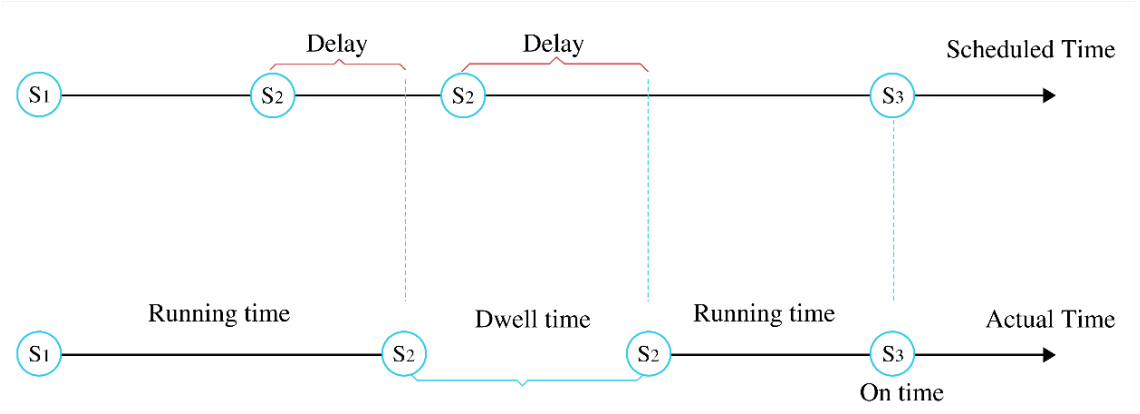


Figure 5.1 Graphical Representation of Train Delays

5.1.2 Train Delay Prediction

In general, train delay prediction can be divided into three categories: short-term, medium-term, and long-term. The short-term delay prediction is usually defined as the prediction of delay time for trains from the current time *t* to future periods (e.g., one hour or a few hours later). If the train delay is predicted 24 hours in advance, it is usually referred as a medium-term prediction. The prediction of train delays over a certain period (e.g., one day or one week) in advance is classified as a long-term prediction.

Train delay prediction can be treated as a multivariate regression problem. As it is in a time-series context, we need to use the date-time index. More precisely, it can also be formed as a multivariate time series analysis problem. According to data acquisition time

and computation time constraints, most of the studies can be divided into short-term forecast (or real-time) and long-term (or medium-term) forecast. As a train can arrive at or depart from a station early, on time, or late, the train delay observations over time and space are known as a stochastic process. The most natural and elegant way is to use probability models to describe the evolution of discrete-time observations based on a continuous-time process. Consequently, probabilistic models mainly solve short-term forecast with high-frequency real-time data. Data-driven models are mainly used for medium-term or long-term forecast. Probabilistic graphical models and deep learning methods have complementary characteristics. To implement a practical delay prediction model, the combination of both methods can integrate their advantages.

Train delay prediction can be treated as a multivariate regression problem. As it is in a time-series context, we need to use the date-time index. More precisely, it can also be formulated as a multivariate time series analysis problem. According to data acquisition time and computation time constraints, most of the studies can be divided into short-term forecast (or real-time) and medium-/long-term forecast. As a train can arrive at or depart from a station early, on time, or late, the train delay observations over time and space are known as a stochastic process. The most inherent and elegant method is to use probability models to describe the evolution of discrete-time observations based on a continuous-time process. Consequently, probabilistic models mainly solve short-term forecast with high-frequency real-time data. Data-driven models are mainly used for medium-term or long-term forecast. Probabilistic graphical models and deep learning methods have complementary characteristics. To implement a practical delay prediction model, the combination of both methods can leverage their advantages.

As shown in Figure 5.1, the actual dwell time and running time are two main factors determining the train delay. If predictors can predict the dwell time and the

running time correctly, the train delay time can be estimated accurately. In this chapter, we focus on the short-term prediction of train delay. However, the methodology and learning architecture proposed in this chapter can be easily extended to cope with the long-term prediction.

5.2 Enhancing Multivariate Prediction with Rule-Driven Automation Method

5.2.1 Maximum Predictability of Running Time and Dwell Time

Entropy is a measure of uncertainty of the state in time series analysis. It can be used to represent the degree of predictability from a time series dataset. For running time or dwell time data, low entropy implies high predictability, and vice versa. The real entropy estimates the probabilities of the values in the historical time series and temporal regularity of the sequences [101]. For example, given two sequences of the train delay $D1 = [1,2,1,1]$ and $D2 = [2,1,1,1]$, we have different values of real entropy. The real entropy obtains an approximation value that converges rapidly by applying the Lempel-Ziv algorithm [101, 142]. Accordingly, the approximation of real entropy for historical running time or dwell time can be calculated by

$$H(S) \approx \frac{w \ln w}{\sum_t sub_t^j} \tag{5.1}$$

where $H(S)$ denotes real entropy of the historical time series sequence. Its exponential time complexity is $O(2^n)$. sub_t^j is the length of the shortest subsequence from time step t that has not appeared before t in a trip j. w is the length of the list of running time or dwell time. Our work explores the upward bound on the predictability in the train delay for selecting an appropriate predictor. According to [99, 101], the following equation (5.2) computes the maximum predictability $\prod^{max}$ of the train delay dataset.

$$f(\prod^{max}) = -\prod^{max} log_2(\prod^{max}) - (1 - \prod^{max}) log_2(1 - \prod^{max}) + (1 -$$

$$\prod^{max}) log_2(N-1) - H(S) \quad (5.2)$$

where $H(S)$ represents the given value of real entropy, as explained above. N represents the number of distinct values in train delay time series in a trip j. The whole calculation process of the maximum predictability for the train delay dataset is presented in **Algorithm 5.1**. We herein apply equation (5.2), the first derivative, and the second derivative. The accuracy has a 16-bit floating-point, which is set as 0.0001.

Algorithm 5.1: Calculating the maximum predictability

Require: The value of $H(S)$ and N

Output: $\prod^{max}$

Maximum_predictability (N, $H(S)$):

$f(\prod^{max}) = f(\prod^{max}, N, H(S))$ //using equation (5.2)

$first\ derivative = log_2(1-\prod^{max}) - log_2(\prod^{max}) - log_2(N-1)$

$second\ derivative = \frac{1}{(\prod^{max}-1)\prod^{max}}$

if $H(S) \leq log_2(N)\ and\ H(S) \leq 0.01$:

return 0.999

else if $H(S) \leq log_2(N)\ and\ H(S) > 0.01$:

$\prod^{max} = \frac{(N+1)}{2N}$

While abs ($f(\prod^{max})$) > 0.0001：

$$\prod^{max} = \prod^{max} - \frac{f(\prod^{max})}{first\ derivative - \frac{f(\prod^{max}) * second\ derivative}{2 * first\ derivative}}$$

end while

return $\prod^{max}$

5.2.2 Calculating Resilience Indicators and Labeling Delay Status

With the advancement of data collection and cloud storage, real-time delay data

is widely used. Once it is fused with schedule data (timetable), some critical predictor variables can be inferred from measurements of the observable variables. The travel time of a train trip consists of two parts, running time and dwell time. When a train runs in a section, the evolution of train delays is closely related to RSE by excluding the disturbance of external factors, such as abnormal lousy weather and traffic accidents.

Furthermore, the mutual influence between trains that occur at the same station and successive delays begins with the increase of train dwell time. The evolution of train delays is closely related to RST. They are calculated as,

$$RSE = \left(\hat{a}^{s_i} - \hat{d}^{s_{i-1}}\right) - \left(a^{s_i} - d^{s_{i-1}}\right) \tag{5.3}$$

$$RST = \left(\hat{d}^{s_i} - \hat{a}^{s_i}\right) - \left(d^{s_i} - a^{s_i}\right) \tag{5.4}$$

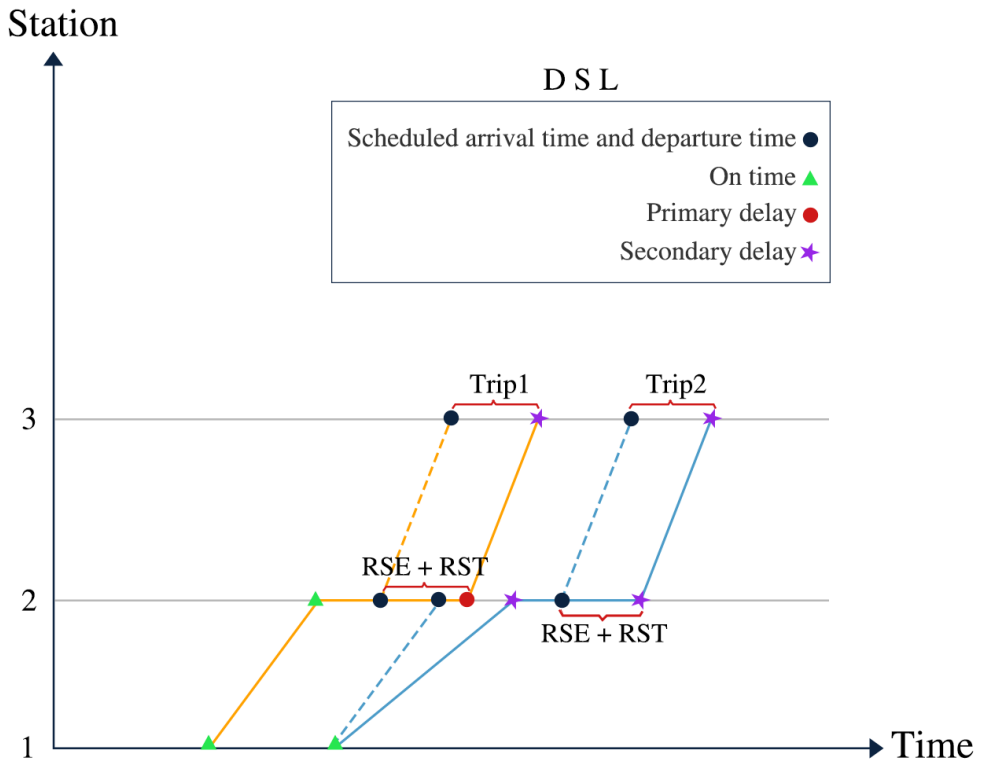


Figure 5.2 RSE and RST in Delay Status Labeling

As illustrated in Figure 5.2, black points denote scheduled arrival time and scheduled departure time. DSL is a rule-based method to label data for searching primary, secondary, and on-time points. The technique can be extended to provide more

data categories or identify primary delay lines by connecting points into lines. Furthermore, the influence of RSE and RST from Trip 1 results in a primary delay for departure time, which occurs at station B. Additionally, they cause a secondary delay for the arrival time of Trip 2 and a secondary delay for the departure time of Trip 2 at station B. Furthermore, it causes changes in the arrival time of Trip 2 at station B and the departure time of Trip 2 from station B. To sum up, a train's arrival time at a station is affected by the departure time on the previous trip at the same station and the departure time at the previous station on the same trip. Hence, RSE and RST play essential roles in assessing the impact of disruptions on the railway.

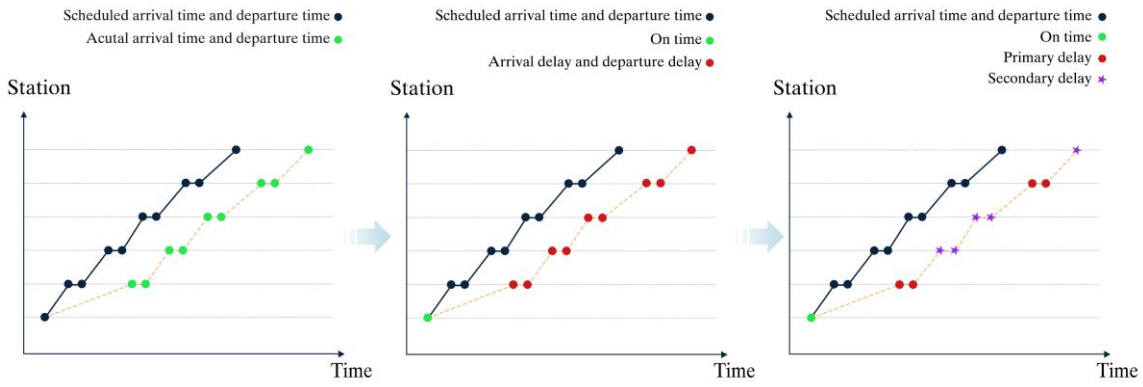


Figure 5.3 Illustration of Delay Status Labeling

Train delays can be divided into primary and secondary delays [59][132]. Due to the delay propagation impacts, an initial delay (primary delay) occurs at a current station, which often causes secondary delays at the successive stations. Delay classification can be implemented to help us understand the model inputs and outputs by determining the upper and lower bounds of data and labeled delay status. DSL incorporates domain knowledge to extract the information from the preprocessed dataset, as shown in Figure 5.3. Specifically, black points are scheduled arrival time or scheduled departure time, and green points are actual arrival time and departure time. Those points are from raw data. When the actual arrival time subtracts the scheduled arrival time, red points can be

obtained, representing the arrival delays and departure delays. There are two ways to prevent delays from spreading out from a system perspective, making a more robust timetable or preventing primary delays. Therefore, if we can predict and reduce the primary delays, the secondary delays can be reduced or avoided. The idea is to use domain knowledge to classify data for finding useful insights, namely the primary and secondary delay points.

Algorithm 5.2: Delay Status Labeling (DSL algorithm)

Require: Input all train data, arrival delay or departure delay at i train station D_i , arrival delay or departure delay for a previous trip D_i^{pre}, an initial station of a trip D_0, and pre-defined thresholds V.

Output: arrival delay status or departure delay status L_i

If D_i^{pre} exist, $\forall D_i^{pre} \geq V$, and $D_i \geq V$, then

$L_i.add(2)$

For $q = (\forall i + 1) \rightarrow i$ do

if $D_q < V$ then

$L_i.add(1)$

else if $(RSE + RST) \geq V$ then

$L_i.add(3)$

else

$L_i.add(2)$

else

if $D_0 \geq V$ then

$L_i.add(3)$

else

$L_i.add(1)$

for $q = 1 \rightarrow i$ do

if $D_q < V$ then

$L_i.add(1)$

else if $(RSE + RST) \geq V$ then

$L_i.add(3)$

else

$L_i.add(2)$

To find the primary and secondary delays at the corresponding arrival stations, the DSL algorithm (**Algorithm 5.2**) traverses all the arrival delays in a trip list. By calculating differences of arrival delay of the current station and arrival delay of the previous station, the different types of delay points or on-time points are added to a corresponding list, L_i. "1" denotes an on-time running point status, "2" denotes a secondary delay point status, and "3" denotes a primary delay point status. The algorithm is used to calculate the departure delays as well. According to DSL, RSE and RST, we can define five common scenarios with train delays and one scenario without delay,

$$\begin{cases} \text{C1}: d_{s_i} < V \; and \; L_i = \text{"1"} \\ \text{C2}: V \leq d_{s_i} < B \; and \; L_i = \text{"2"} \\ \text{C3}: V \leq d_{s_i} < B \; and \; L_i = \text{"3"} \\ \text{C4}: d_{s_i} \geq B \; and \; L_i = \text{"2"} \\ \text{C5}: d_{s_i} \geq B \; and \; L_i = \text{"3"} \\ \text{C6}: no \; delay \; occurs \end{cases} \tag{5.5}$$

wherein d_{s_i} denotes departure delay and B denotes buffer time. C1, C2, C3, C4, C5, and C6 show six types of conditions. C1 expresses no delay occurs at the current station. C2 represents a secondary delay that occurs at the current station. C3 shows a primary delay occurs at the current station. C4 indicates that a secondary delay occurs at the current

station and is affected by the delay of the previous station and the previous trip. C5 denotes a primary delay at the current station and is affected by the delay of the previous station and the previous trip. C6 demonstrates that no delay occurs on a trip.

5.2.3 Multi-Scenario Real-Time Delay Forecast

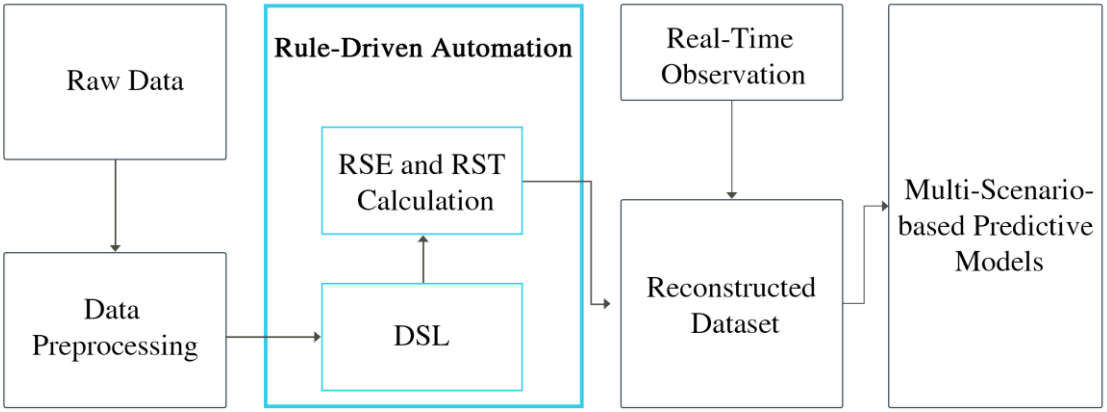


Figure 5.4 Flowchart of Real-Time Delay Predictions

This section illustrates the design and development of a comprehensive real-time predictive learning framework for multivariate train delay predictions. As depicted in Figure 5.4, the input of a multi-scenario based predictive model is composed of data pre-processing, RDA, real-time observation, and reconstructed dataset. The DSL labels the observed values, which find the primary delays, secondary delays, and on-time running of corresponding stations and show the status of the whole trip from a given history. Furthermore, the influence of RSE and RST on delay predictions and the relationship between running time and dwell time are included via primary and secondary delays in our studies.

Firstly, a pre-planned train trip with possible delays is included. The available data consists of spatial data from multiple train stations and temporal data at each station. The train delay data is interpreted as a time-series format, which can be processed and transformed into many sequential matrices. The sequential model learns the spatio-temporal features by using the arrays as input shapes. Consequently, the correlations of

independent variables for train delays among different stations are utilized during the predictive model training.

We apply multivariate regression as a supervised machine learning algorithm with arrival delay status, departure delay status, RSE and RST variables, and other variables to investigating the relationship between adjacent train delays, which is defined as

$$h_\theta(X)_i = [(\theta^T x)^d]_i = \sum_{m=i}^{n} \theta_m x_m + b \tag{5.6}$$

where $h_\theta(X)_i$ is the dependent variable, called the hypothesis. θ is (or weights), and it is what the models try to learn. x is the independent variable. b is the error term. For a multivariate multi-step time series forecast model, the relevant time series inputs are denoted by X. We have a sequence of train delays, X, which can be denoted in vector form as $X = [X_t, X_{t+1}, X_{t+2} \dots, X_{t+n-1}]$. The matrix with time-steps is from t to $t + n - 1$. All features are standardized to the range $[0, 1]$ before being passed to the predictive learning model. Hence, for a tree-based method or a deep learning-based method, the objective function of multivariate multi-step train delay prediction can be written as,

$$\mu = arg\ min \frac{1}{n} \sum_{i=1}^{n} \| y_i - h_\theta(X)_i \| \tag{5.7}$$

where μ is the minimum value of the objective function in the model; y_i is the ground truth and $h_\theta(X)_i$ denotes the prediction of our model.

We herein again summarize the main characteristics of our machine learning solution in the following points.

- Real entropy and the temporal correlation of data sequences are applied to measure the train delay uncertainty on the given trip.
- The DSL component can extract multiple primary delays and secondary delays on a single trip, presenting critical causal relationships in the railway network.
- Multivariate multi-step models are implemented to estimate regression coefficients for multiple stations.

- Our proposed machine learning solution adds DSL, RSE, and RST to the multivariable regression to explain observed data and generate high accurate predictions about future observations. It can deal with multi-input and multi-output prediction and estimation issues. Notably, this solution allows tree-based models to handle large-scale railway networks, even with higher efficiency and accuracy in various prediction scenarios.

5.3 Experiments

5.3.1 Data Description

To validate the proposed methodology and evaluate the performance of predictive models with the proposed impact factors, we choose train services in Sydney as a case study, which was the same as the observations we obtained and reported in the Appendix, where random forest could perform extremely better than the latest deep learning methods. The experiments have been performed on the scheduled data and real-time train data from the NSW's (one Australian state) open data hub. The dataset consists of 161-day data observations for the trip number '600D' (8:29 am – 9:02 am) from 11 April to 21 November in 2019. The train line is from Sydney's Bondi Junction Station to Hurstville Station (BJS-HS) in morning peak hours in working days (Monday to Friday), as shown in Figure 3.8. To consider the impact of RSE and RST from the previous station , we observed the real-time delays of the second station ES. We selected data on November 20 as a C1 test dataset, which contained delays in subsequent stations. Meanwhile, we selected data on November 15 as a C3 test dataset, which has the successive delays caused by the primary delay. The data before the corresponding dates were used as historical data for model training.

1)Train Schedule Data

The schedule data and geographical information are obtained from the Transport

for NSW (TfNSW) open data hub in the same period [13]. The datasets contain schedule arrival times, scheduled departure times, station names, longitudes, and latitudes.

2)Train Delay Data

GTFS provides detailed schedules and associated geographic information in an open data format [19]. We developed a Web data extraction and pre-processing tool from a real-time GTFS application programming interface (APIs) of TfNSW [13]. The raw data with a 10-sec frequency is extracted from the real-time GTFS that generates a dataset between 2 and 4 GB each day. The dataset contains arrival delays, departure delays, and station IDs. After data cleansing, we fuse the dataset with schedule data and finally obtain complete data of the trip '600D'.

Figure 5.5 Rule-based Train Delay Categories

According to (5.5), we can obtain six sub-datasets from C1 to C6 containing different information types. C2 and C4 have not enough data for training, making it difficult to estimate the generalizability of the proposed method. C5 relies on the RSE and RST of the previous trip. C6 stands for that all stations are on time and can be removed as invalid data. Hence, we choose C1 and C3 as the input data for predictive models, as illustrated in Figure 5.5.

5.3.2 Implementation and Training

The proposed solution is implemented in Python using the TensorFlow Framework [143] and Scikit-learn [144]. Prior to measuring the effectiveness of the proposed RDA, the maximum predictability $\prod^{max}$ should be calculated based on the dataset of the trip '600D'. The results have been calculated by equation (5.1) and equation (5.2), as demonstrated in Table 5.1, which indicates the fundamental limit for predictability of train delays in the given trip. In other words, the train delay can be correctly forecasted with certain accuracy. It also indicates that delay prediction with spatio-temporal correlation has its bottleneck. Furthermore, a continuous-time series is discretized as a discrete-time series via using the RDA. The $\prod^{max}$ value has dropped significantly. The datasets with RDA as inputs have negative impacts on spatio-temporal modeling.

Table 5.1 Predictability of Different Scenarios for 600D

Input dataset	$\prod^{max}$ for **Running time**	$\prod^{max}$ for **Dwell time**
C1 Dataset	0.872	0.888
C1 Dataset with RDA	0.732	0.783
C3 Dataset	0.873	0.889
C3 Dataset with RDA	0.637	0.739

5.3.3 Evaluation Metrics

In our experiments, the evaluation of predictive models is based on the following five standard metrics: RMSE, MAE, SMAPE, maximum residual error (ME), and root relative squared error (RRSE). ME and RRSE have been formalized in (5.8) and (5.9), where y_t is the actual value at time step t , $\hat{y}_t$ is the predicted value, $\bar{y}$ is the mean of the observed values of the dependent variable, *n* is the total sample size.

$$\mathrm{ME} = \max(|y_t - \hat{y}_t| \tag{5.8}$$

$$\mathrm{RRSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{i=1}^{n}(y_t - \bar{y})^2}} \tag{5.9}$$

To achieve a fair way for training the best predictive models, in addition to using the same settings, we also leverage the early callback to stop training when a monitored metric has stopped improving. As the performance difference between training and validation is acceptable, the model does not overfit or underfit the training data.

5.3.4 Prediction Results

We compare the performance of the RF model with RDA to the following baseline methods: RF, CNN, LSTM, GRU-LSTM, Bidirectional LSTM with Attention (BiLSTM-A), CNN-RDA, LSTM-RDA, GRU-LSTM-RDA (GL-RDA), and Bidirectional LSTM with Attention-RDA (BiLSTM-A-RDA). Table 5.2 and Table 5.3 demonstrate the results of predictive models for running time and dwell time, respectively. It is observed that the training time used by RF-RDA is the lowest. For the running time and dwell time in the two prediction scenarios, RF-RDA achieves the lowest errors in RMSE, ME, MAE, and SMAPE, compared to others. Furthermore, RRSE indicates that the RF-RDA models have perfect fits between the observed and predicted data. By contrast, the differences between all deep learning models and variants in performance are slight or even not noticeable.

Figure 5.6 reports the performance of RF, CNN, LSTM, and different variants of LSTM with RDA and without it for predicting running time and dwell time. All *x*-axes express the corresponding models. The *y*-axis of RMSE, ME, and MAE represents the error in seconds. In addition to RF-RDA, other models do not have significant performance differences. Table 5.1 shows that using RDA causes decreases in the maximum predictability $\prod^{max}$ of train delay dataset. As seasonality and weekly data

patterns are disrupted by RDA, the negative impact occurs in deep learning models for spatio-temporal correlation analysis.

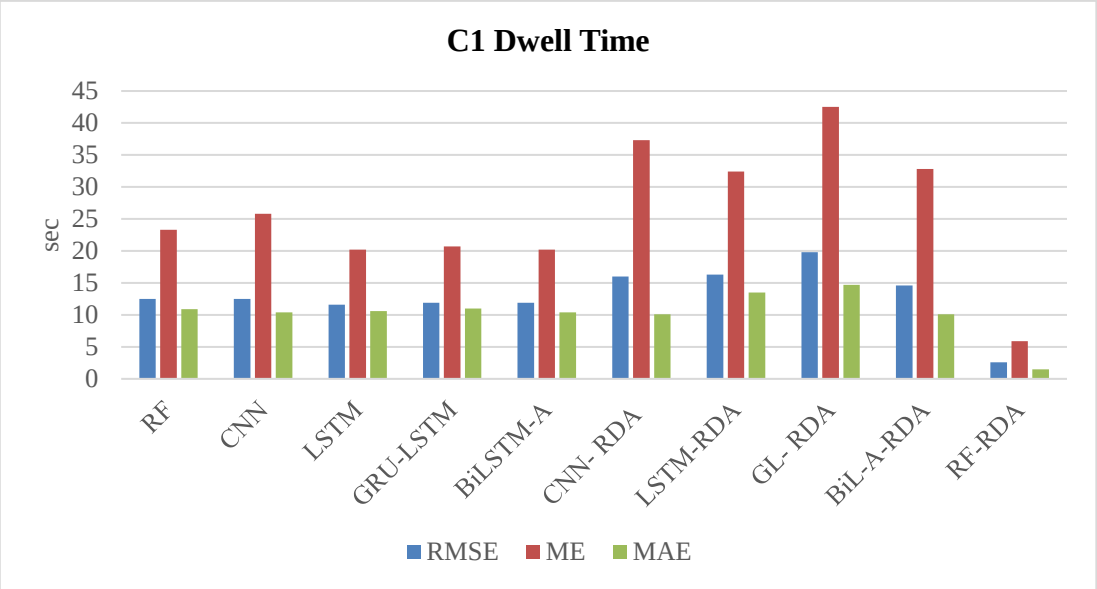
However, with the help of the RDA, random forest can leverage delay status, RSE, and RST variables for making decisions. Hence RDA provides a more effective way to take into consideration lower and upper bounds as features to the algorithm, instead of verifying the range on predicted values and making some post-processing. Thus, RF-RDA achieves the best prediction accuracy among all the models.

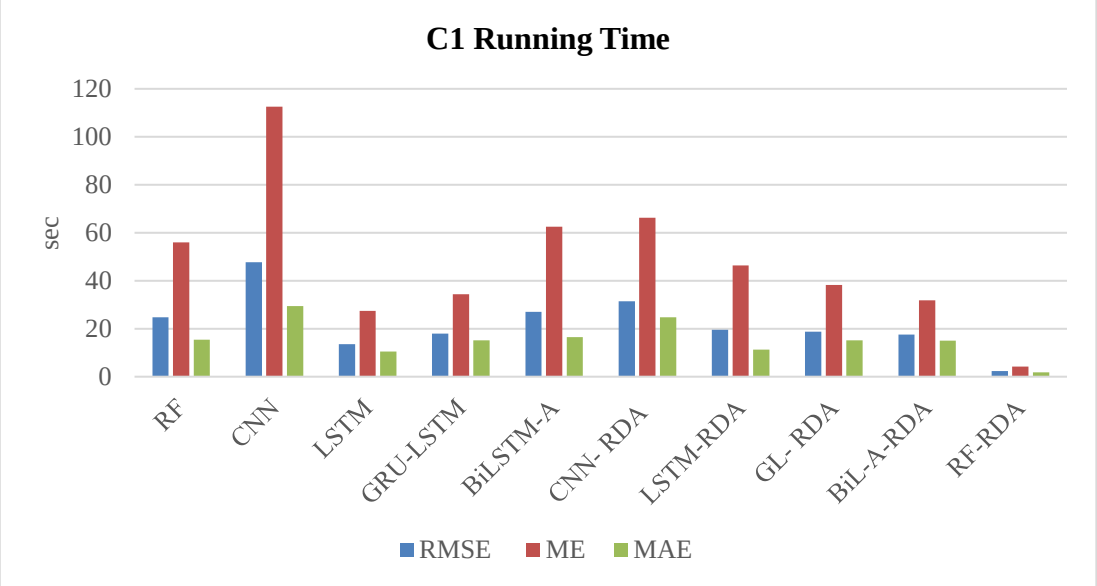
Table 5.2 Running Time Prediction Performance

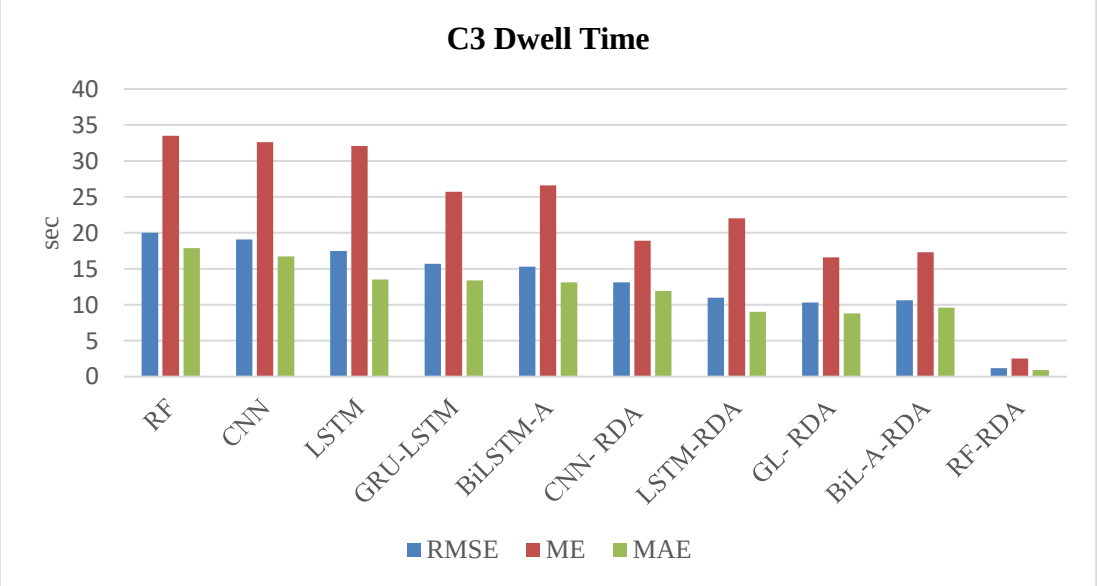
C	Model	RMSE	ME	MAE	SMAPE	RRSE	Time
C1	**RF**	24.7	56.0	15.4	9.3	0.147	4.0
	CNN	47.7	112.6	29.4	15.3	0.283	6.9
	LSTM	13.6	27.5	10.5	6.7	0.081	13.5
	GRU-LSTM	17.9	34.4	15.1	11.8	0.106	13.6
	BiLSTM-A	27.1	62.5	16.5	9.4	0.161	12.4
	CNN- RDA	31.5	66.3	24.7	16.4	0.187	6.7
	LSTM-RDA	19.5	46.4	11.3	5.9	0.116	18.6
	GL- RDA	18.8	38.2	15.2	11.0	0.111	18.5
	BiL-A-RDA	17.6	31.8	15.	11.7	0.105	13.1
	RF-RDA	**2.4**	**4.2**	**1.8**	**1.4**	**0.014**	**1.9**
C3	**RF**	70.9	147.1	49.2	23.8	0.333	3.9
	CNN	88.0	195.0	58.9	27.0	0.413	5.7
	LSTM	54.4	93.7	39.5	21.3	0.256	11.2
	GRU-LSTM	66.3	139.4	47.4	23.7	0.311	12.2
	BiLSTM-A	69.9	143.6	47.5	22.5	0.328	12.7
	CNN- RDA	102.1	232.3	66.0	29.9	0.479	10.3
	LSTM-RDA	87.7	183.0	56.5	26.5	0.412	16.0
	GL- RDA	111.4	250.1	68.1	28.9	0.523	22.7
	BiL-A-RDA	88.0	191.0	53.4	21.7	0.099	29.0
	RF-RDA	**21.1**	**39.5**	**14.3**	**6.7**	**0.099**	**1.9**

Table 5.3 Dwell Time Prediction Performance

C	Model	RMSE	ME	MAE	SMAPE	RRSE	Time
C1	**RF**	12.5	23.3	10.9	21.7	0.827	3.3
	CNN	12.5	25.8	10.4	20.1	0.830	6.2
	LSTM	11.6	20.2	10.6	20.9	0.769	14.5
	GRU-LSTM	11.9	20.7	11.0	21.5	0.792	12.6
	BiLSTM-A	11.9	20.2	10.4	20.8	0.789	15.2
	CNN- RDA	16.0	37.3	10.1	19.2	1.060	11.2
	LSTM-RDA	16.3	32.4	13.5	27.2	1.083	6.8
	GL- RDA	19.8	42.5	14.7	27.3	1.315	29.2
	BiL-A-RDA	14.6	32.8	10.1	20.0	0.972	12.3
	RF-RDA	**2.6**	**5.9**	**1.5**	**2.9**	**0.171**	**1.8**
C3	**RF**	20.0	33.5	17.9	32.3	1.740	3.4
	CNN	19.1	32.6	16.7	29.7	1.657	4.4
	LSTM	17.5	32.1	13.5	23.1	1.519	7.0
	GRU-LSTM	15.7	25.7	13.4	23.1	1.363	10.1
	BiLSTM-A	15.3	26.6	13.1	22.3	1.330	10.5
	CNN- RDA	13.1	18.9	11.9	20.2	1.139	4.9
	LSTM-RDA	11.0	22.0	9.0	14.6	0.955	8.1
	GL- RDA	10.3	16.6	8.8	14.5	0.893	10.6
	BiL-A-RDA	10.6	17.3	9.6	16.0	0.920	16.8
	RF-RDA	**1.2**	**2.5**	**0.9**	**1.5**	**0.107**	**1.6**


C1 Dwell Time
sec
45
40
35
30
25
20
15
10
5
0
RF
CNN
LSTM
GRU-LSTM
BiLSTM-A
CNN- RDA
LSTM-RDA
GL- RDA
BiL-A-RDA
RF-RDA
RMSE
ME
MAE


C1 Running Time
sec
120
100
80
60
40
20
0
RF
CNN
LSTM
GRU-LSTM
BiLSTM-A
CNN- RDA
LSTM-RDA
GL- RDA
BiL-A-RDA
RF-RDA
RMSE
ME
MAE


C3 Dwell Time
sec
40
35
30
25
20
15
10
5
0
RF
CNN
LSTM
GRU-LSTM
BiLSTM-A
CNN- RDA
LSTM-RDA
GL- RDA
BiL-A-RDA
RF-RDA
RMSE
ME
MAE

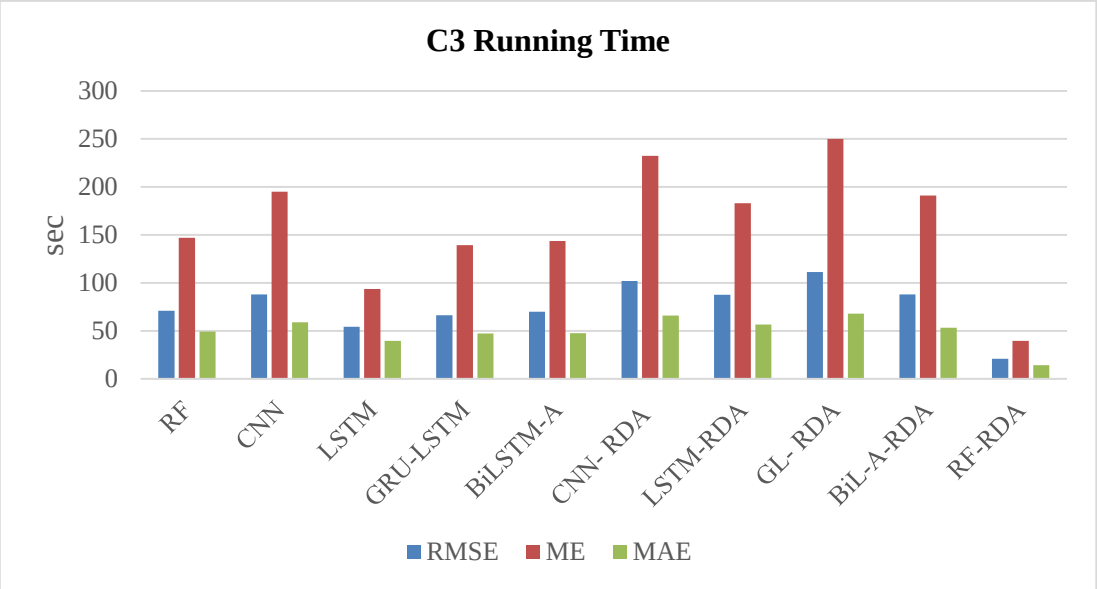


Figure 5.6 RMSE, ME, and MAE for Running Time and Dwell Time Prediction

We observe that, among all these predictive models, RF, CNN, LSTM, and its variants cannot better predict the running time and dwell time when the maximum predictability is low. It has the lowest percentage error at SMAPE, as indicated in Figure 5.7.

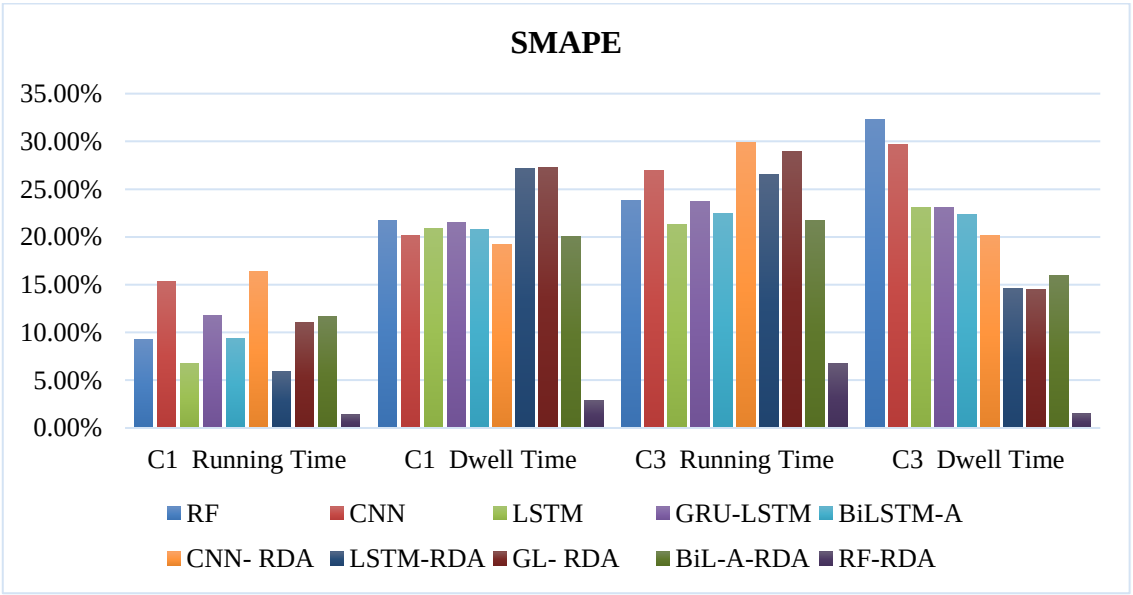


Figure 5.7 SMAPE for Running Time and Dwell Time Prediction

In Figure 5.8, we compare RRSE values resulting from all predictive models. The RRSE can ignore the impact of data scale and identify outlier prediction results. A model

with RRSE over 1 indicates poor applicability for prediction. The experiments show that RF cannot be well applied for multivariate multi-step prediction, specifically, dwell time prediction. Deep learning models have not shown significant performance differences either. For running time prediction, a simpler neural network model can quickly achieve the best performance in both C1 and C3 scenarios. Nevertheless, for dwell time prediction, we need a more complex neural network to capture hidden patterns in data for the C3 scenario. The complex neural network causes more significant errors than the simpler neural network model in the C1 scenario.

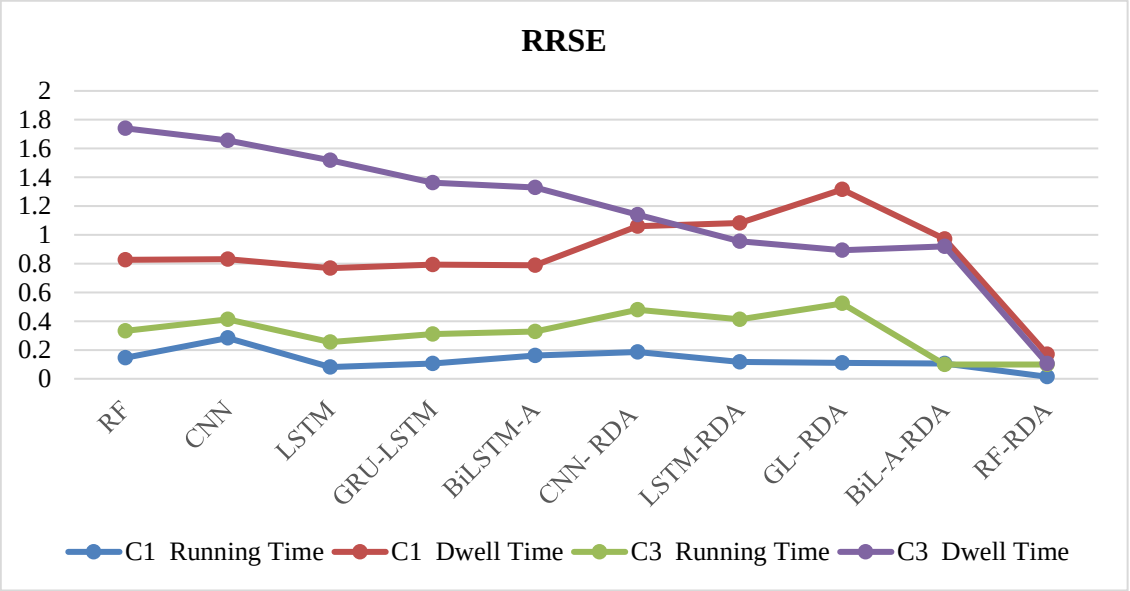
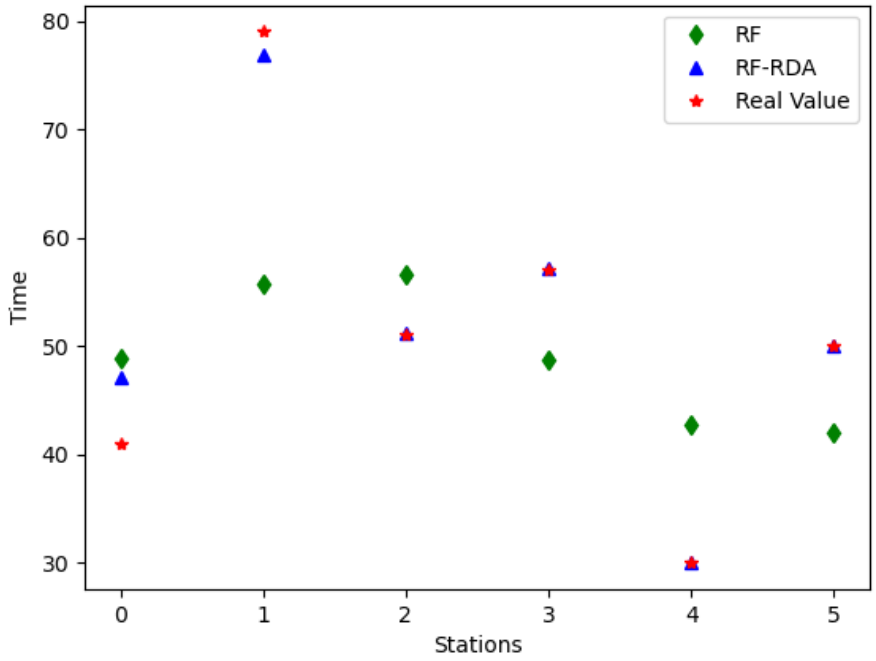


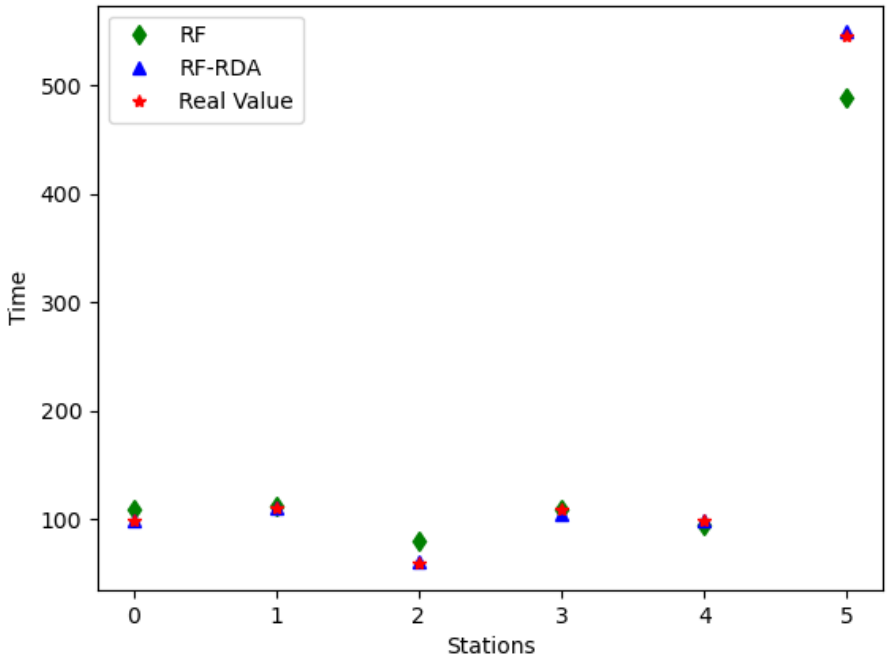
Figure 5.8 RRSE for Running Time and Dwell Time Prediction

Furthermore, the selected predictor variables cannot fully explain the variability of dwell times. It is common to use more relevant impact factors as inputs of the model to reduce prediction errors and derive more accurate dwell time. For example, the dwell time is simply sensitive to the number of passengers waiting for the corresponding trains. However, how to get the accurate number of passengers boarding and alighting at the corresponding station is still an unresolved issue though some trials with IoT technologies are underway. Luckily, RF-RDA shows superior performance and remarkable

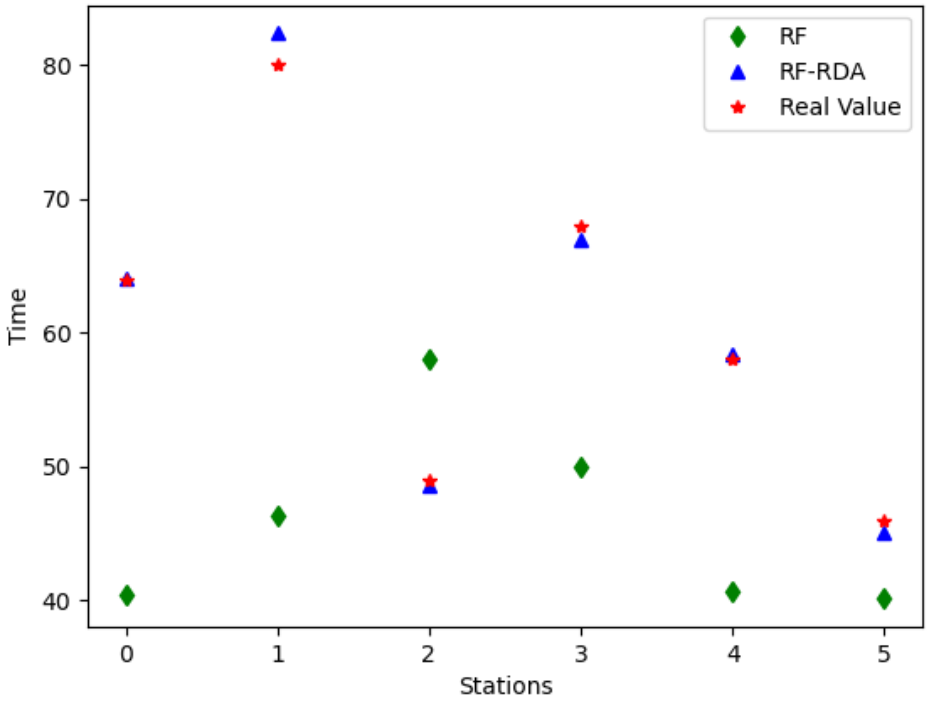
improvements against other models even though there is no stable method to count passengers.



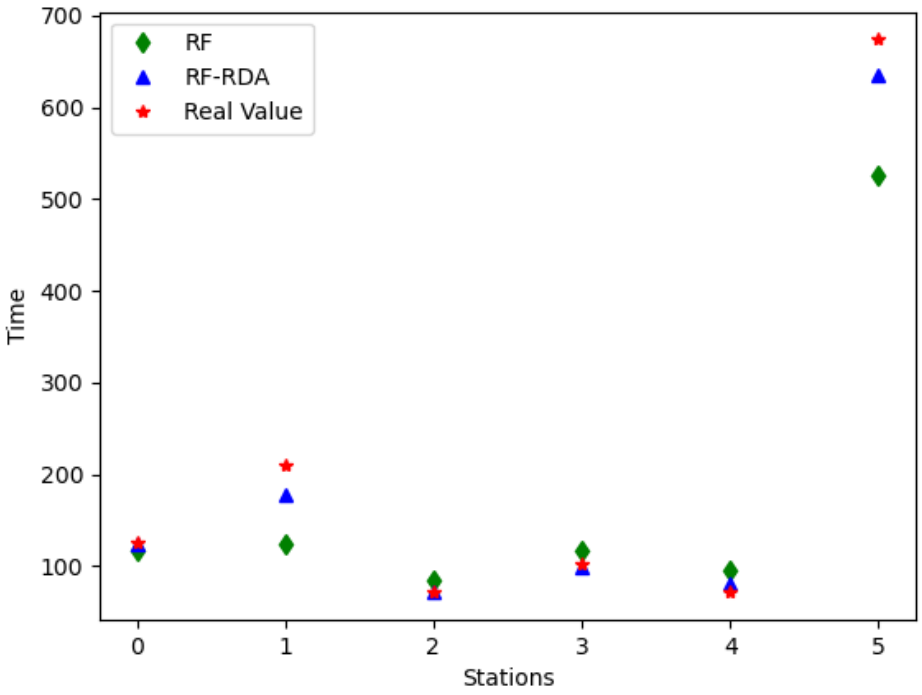
(a)



(b)



(c)



(d)

Figure 5.9 Comparison of Predicted Accuracies under RF and RF-RDA for C1 and C3

Scenarios.

(a) predicted value of C1 dwell time; (b) predicted value of C1 running time; (c) predicted value of C3 dwell time; (d) predicted value of C3 running time

The prediction results of C1 and C3 scenarios are indicated in Figure 5.9. The x-axis indicates that the consecutive stations that need to be predicted given a trip, and the y-axis represents running time or dwell time. If we can accurately predict the running time and dwell time, we can calculate the departure delay and arrival delay accurately based on the running and dwell time. It can be seen that, when RDA is utilized, the prediction accuracies are improved significantly, while the prediction errors are decreased. For illustration purpose, only C1 and C3 scenarios are selected in the experiments. However, our method is applicable to all scenarios. Because RDA reconstructs the dataset, the difference between the upper and lower bounds will be effectively limited to have high-quality data for tree-based models, which dramatically reduces the number of abnormal data (data that does not meet the conditions from the corresponding scenario).

To further assess the effectiveness of the proposed method, time series cross-validation is used to compare and evaluate the performance of RF and RF-RDA. Figure 5.10 demonstrates the comparison of the prediction errors produced by RF only and RF-RDA using C1 and C3 scenarios. We employed an 8-split time-series cross-validation on the data sets. The Scikit-learn library provides an implementation for splitting given data sets [144]. A lower RRSE value indicates a higher R-squared value and a larger effect size. Moreover, considering the complexity and noise of real-time data, even RF-RDA cannot completely remove anomalies when making predictions, the proposed method can reduce the effect of the outlier value more effectively.

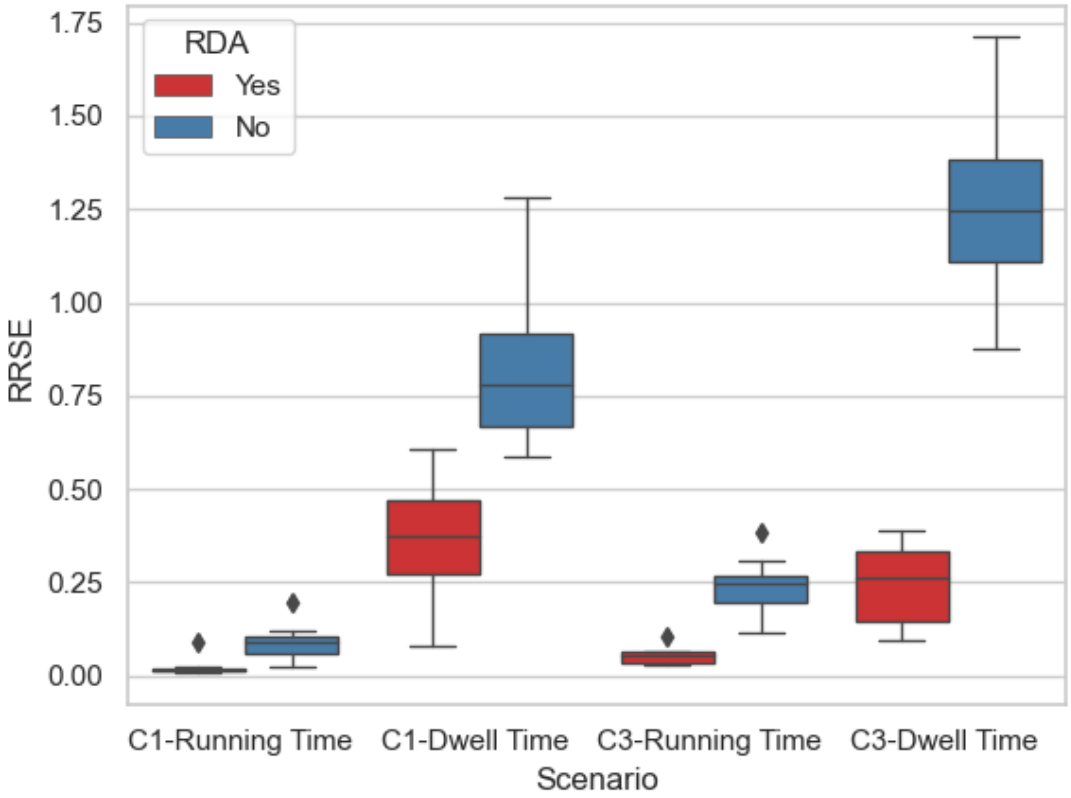


Figure 5.10 Time-based cross-validation results for C1 and C3

Additionally, RF-RDA also shows more satisfactory performance than RF in terms of RRSE errors. It can be seen in the Figure 5.10 Time-based cross-validation results for C1 and C3 that the RF-RDA has a more stable prediction ability than RF. The proposed method produces a larger effect size, which indicates a stronger relationship among variables.

In summary, the results of the experiment demonstrate that the capability of deep learning in time series prediction heavily relies on the availability of high-quality data. It is challenging to obtain such data in most cases, unfortunately. On the other hand, training a deep learning model is very time-consuming and requires significant computational power. It should be noted that, although using RDA increases the computational complexity of the proposed method, it mainly occurs during an offline phase. Thus, RF-RDA with reduced size of the inputs achieves multi-step prediction in a real-time environment and makes a faster update of the real-time prediction results based on current

observations more evidently. Compared with other models, our model shows better prediction performance in terms of all validation metrics and different scenarios. The time complexity of online computation with the RF-RDA model is exceptionally low.

5.4 Summary

This chapter established a rule-driven automation method for improving multi-scenario real-time delay predictions on real-world data, which were collected from different sources in a public transport agency. In particular, we found the bound of performance improvement (with RDA) for tree-based methods. Furthermore, the proposed solution explored the real entropy and probability $\prod^{max}$ of the datasets. The presented value indicated that the RF-RDA could reach high accuracy by discovering the average of sub decision trees. Several deep learning models, including CNN, LSTM, and their variants, have been investigated to predict the running time and dwell time based on multivariate inputs.

Moreover, the proposed approach can be directly applied for real-time decision support in the railway system. The large amount of computation cost caused by deep learning models makes it difficult to quickly analyze and simulate the entire railway network using high-frequency real-time data. Instead, with our method, we can use regression tree-based methods with the proposed solution to offer a set of simple, common-sense rules that make accurate predictions of future values. Our experiments show that the tree-based regression method based on observed and real-time operational data is more effective for real-time train delay prediction than the deep learning methods.

We will conduct experiments supporting further comparisons with more robust tree-based methods, such as AdaBoost, Gradient Boosting, and deep random forest as meaningful future work.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

This thesis had demonstrated studies on modeling, analysis, and application of open traffic data for train delay prediction, including developing a novel open data and preparation tool framework, a novel general delay prediction system framework, and an effective real-time multi-scenario delay prediction method. In the following text, the significant results and experimental findings of this thesis are summarized.

Chapter 3, "A GTFS Data Acquisition and Processing Framework," has developed a data acquisition and preparation framework, namely DAP, to convert and fuse the GTFS static and real-time data to a ready-to-use format for a diversity of usages. The fused dataset is an open-source alternative to automatic vehicle location (AVL) data, one of the three commonly used datasets in transportation, automatic fare collection (AFC) data, AVL, and automatic passenger counting (APC). A data cleaning and aggregation tool is proposed to detect real-time outliers and handle missing data via a Bayesian Structural Time Series (BSTS) with Rule-Based Inference Engine. Compared with AVL (historical data), our proposed framework can provide accurate real-time observations in transportation modeling. The predictive models can make short-term or long-term multi-scenario delay predictions based on current observations.

Chapter 4, "A General Prediction System Framework for the Primary Delays," proposed a comprehensive and general data-driven PDPS framework, which combines GTFS, CPS, and deep learning models to leverage the data fusion. Based on this framework, we have also used the DAP to pre-process the data. Finally, we demonstrated

advanced deep learning models for univariate and multivariate time series forecast. We demonstrated a novel ConvLSTM Encoder-Decoder model with CPS to obtain better primary delay predictions for univariate time series forecast. For multivariate time series forecast, The LSTM tackles the tasks for long-term predictions of running time and dwell time. The CPS utilizes the predicted values with a nominal timetable to identify the primary and secondary delays based on the delay causes, run-time delay, and dwell time delay. We demonstrated the performance of the standard LSTM and its variants applied in a novel architecture. The results show that the variants can improve upon the standard LSTM significantly when compared through predicting time steps of dwell time feature. The experiments also show historical trend volatility with many irregularities, which prompts further studies needed to tackle them.

Chapter 5, “Real-Time Forecast of Multi-Scenario Train Delays,” has developed a novel augmented machine learning approach to improve the overall prediction accuracy further. Firstly, we applied a real entropy for measuring the time series regularity and found approximated potential predictability on train delays. Motivated by the observation that deep learning methods cannot further improve the prediction performance if the delay occurs rarely, our solution proposed a rule-driven automation (RDA) method, including a delay status labeling (DSL) algorithm and the resilience of section (RSE) and resilience of station (RST) factors to generate the forecast for train delays. The DSL utilizes the historical values to label the primary and secondary delays based on the delay causes, run-time delay, and dwell time delay. Finally, we compared the performance of tree-based methods and deep learning-based methods for multivariate regression with and without additional variables from our RDA, respectively. The experiment results demonstrate that the Random Forest based implementation of our RDA method (RF-RDA) can significantly improve the generalization ability of multivariate multi-step forecast models

for multi-scenario train delay problems. Our proposed solution surpasses state-of-art baselines on real-world traffic datasets, which treat various real-time delays differently. The prediction performance is still acceptable for practical use to provide high-accurate forecasts when the predictability of conventional deep learning methods decreases.

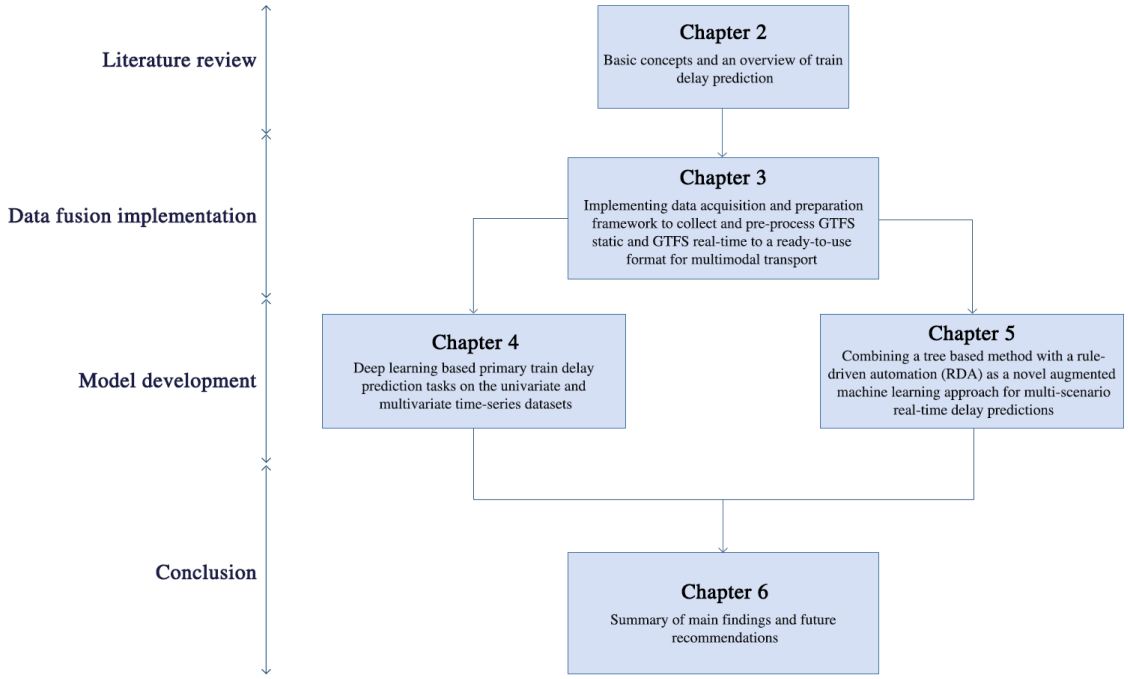


Figure 6.1 An Overview of the Thesis Chapters

To sum up, as shown in Figure 6.1, this thesis has been divided into six chapters. Besides the Introduction chapter, we conducted a systematic experimental investigation to design a general-purpose prediction model for the train delay predictions in various scenarios. Firstly, a data collection and preprocessing tool was developed to obtain real-time multimodal traffic data, fused with static schedule data to provide rich information about multimodal trips. Secondly, based on univariate and multivariate data, we have proposed a general prediction framework to target primary delay forecast. Finally, we found that advanced deep learning still has a performance bottleneck in real-time multi-scenarios train delay forecast tasks. A real-time predictive learning framework was

proposed to provide a novel augmented machine learning approach to enhance overall prediction accuracy, including a tree-based method and an RDA.

6.2 Limitations and Recommendations for Future Research

In addition to the above results and findings, some research issues still need to be further studied. Thus, in this section, we briefly list some limitations and corresponding recommendations for future work.

First, although our proposed hybrid deep learning architecture can be directly applied for long-term decision support in urban railway systems, the large amount of computation cost caused by deep learning models makes it difficult to quickly analyze and simulate the entire railway network by using high-frequency real-time data, based on the performance of the long-term prediction in the proposed architecture. Despite we can extend and apply the CPS or DSL to implement the data classification of the entire train network, setting the appropriate threshold value for CPS and DSL is still a challenging problem. Deep learning is incredibly powerful for multivariate time series regression tasks. However, it is an unresolved task to include rail domain knowledge when building robust and scalable models with deep learning.

Second, it is still challenging to obtain more useful data to integrate with the fused GTFS dataset. For example, smart card data is not open source. Without such data, the number of waiting passengers at the corresponding station will not be known. The performance of the delay prediction model will be affected, especially the long-term prediction. In addition, the use of smart card data contains potential security risks and privacy issues. Meanwhile, the relevant weather condition data in real-time cannot be used directly to combine with the delay data. Therefore, multi-source data brings opportunities and challenges to GTFS-based data fusion.

Moreover, there are still some errors in collecting real-time GTFS data from sensors, and we are still facing the need for higher quality physical hardware and timely and stable cloud systems for obtaining more efficient data. Furthermore, more comparisons should be implemented to determine the best combination of long-term or short-term delay prediction methods in multi-scenarios, such as Bayesian learning and more robust tree-based methods. In reality, forecasting should consider the impact of multimodal transport, such as train stations and bus stops. Therefore, the development of a multimodal transport forecast model should be meaningful future work.

Appendix 1

Preliminary Experiment for Chapter 6

As shown in Table 5.4, we have conducted preliminary experiments on historical data. Based on the same observation (which contains more on-time data than delay data), we apply a tree-based machine learning algorithm (RF) and a deep learning algorithm (LSTM) to predict a trip ahead of two different dates. Herein ‘HP’ expresses high punctuality (more on-time running), and ‘HD’ expresses high-frequency delays (more delays). From the results, we can find that two algorithms could not learn temporal and spatial correlations well, and both show high errors in HD. It also shows that the train delays in the dataset form a discrete-time series, consisting of data points separated by time intervals.

Table 5.4 Results of The Preliminary Experiment

S	**Model**	**RMSE**	**ME**	**MAE**	**SMAPE**	**RRSE**
HP	**RF**	2.0	5.5	0.9	0.7	0.076
	LSTM	4.8	15.6	2.8	2.2	0.085
HD	**RF**	165.3	555.0	84.7	45.8	1.015
	LSTM	165.8	555.4	86.2	45.2	1.018

Although historical data can provide sufficient information, the model uses a large amount of on-time data or a large amount of delayed data, which can only provide good prediction performance in some specific scenarios, but not across multi-scenarios. Spatio-temporal modeling can have a good performance on a spatio-temporal dataset. However, the occurrence of train delays is discrete. To perform spatio-temporal sequence learning,

we have to keep on-time data as a continuous-time series dataset. However, the on-time data is “abnormal” for delay prediction and vice versa. It is the reason why the predictive models fail in real-time delay prediction. Therefore, a practical method is needed to select various inputs to deal with different scenarios, whether a delay occurs, and to calculate potential prediction values to approach the true value.

Appendix 2

Towards Attention-Based Convolutional Long Short-Term Memory for Travel Time Prediction of Bus Journeys [1]

Abstract

travel time prediction is critical for advanced traveler information systems (ATIS), which provides valuable information for enhancing the efficiency and effectiveness of the urban transportation systems. However, in the area of bus trips, existing studies have focused on directly using the structured data to predict travel time for a single bus trip. For state-of-the-art public transportation information systems, a bus journey generally has multiple bus trips. Additionally, due to the lack of study on data fusion, it is even inadequate for the development of underlying intelligent transportation systems. In this paper, we propose a novel framework for a hybrid data-driven travel time prediction model for bus journeys based on open data. We explore a convolutional long short-term memory (ConvLSTM) model with a self-attention mechanism that accurately predicts the running time of each segment of the trips and the waiting time at each station. The model is more robust to capture long-range dependence in time series data as well.

[1]This section has been published as J. Wu, Q. Wu, J. Shen, and C. Cai, "Towards Attention-Based Convolutional Long Short-Term Memory for Travel Time Prediction of Bus Journeys," *Sensors*, vol. 20, no. 12, p. 3354, 2020, doi: 10.3390/s20123354.

1 Introduction

The usage of Intelligent transportation systems (ITS) is motivated in significant part by passenger increase and sustainable development [145, 146]. The ITS has a direct impact on energy consumption, personal living expenses, public health, and safety. Seamless integration of vehicles and sensing devices has made it possible to capture and collect large amounts of sensor data from various data sources in real-time. Developing sustainable and intelligent transportation applications operate and manage real-time and historical data efficiently that has become an increasingly important yet challenging task. It also plays a vital role in achieving the main objectives of ITS, which include accessibility and mobility, environmental sustainability, and economic development [147, 148]. With the advent of artificial intelligence (AI), machine learning and expert systems-based paradigms have driven the development of society and the steady growth of the economy. Besides, deep learning can discover patterns in complex data sets, which could not be found via conventional methods. Merging machine learning and transportation science has tremendous potential to enhance the performance of ITS.

Travel time refers to a period spent traveling from origin to destination. Providing real-time travel information is indispensable for ITS. However, real-time travel time is unlikely to be observed because it has already been historical data rather than ‘real-time data’ since it was collected [149]. Using predictive methods to estimate future travel time is an effective way to provide real-time information. Furthermore, travel time prediction is a known and challenging research area because of the inherent uncertainty [150]. Existing studies on bus travel time prediction mainly focuses on improving the prediction accuracy of a single trip. It is inadequate for implementing efficient applications in an intelligent transportation system, where a bus journey has multiple bus trips [151]. Although the ConvLSTM has shown excellent performance in travel time prediction,

adding the attention mechanism to LSTM based models that have the potential to improve the predictive accuracy [152, 153]. It remains as an unsolved research task for integrating their strengths. The studies apply LSTM based deep learning methods with applications to journey travel time prediction that rely on high-quality, labeled data. However, data acquisition is a challenging task.

The contributions of this study are summarized as follows:

1) We design and develop an open-source data collection framework that can automatically collect and pre-process large amounts of high-quality data over a long period without involving personal privacy, for example, an entire season or even several years.
2) This paper proposes a hybrid model that applies the ConvLSTM network with an attention mechanism to explore a suitable model for the bus journey time prediction on open data.
3) We also discuss input features for journey travel time prediction and suggest directions for future research.

The remainder of the paper is organized as follows. Firstly, we demonstrate a brief overview of basic definitions. Secondly, an integrated system framework is introduced to target the problem of bus journey time prediction and provides a ConvLSTM based method with self-attention. Furthermore, the datasets baseline and evaluation metrics used are in this study. Finally, the findings and suggestions for further studies are summarized.

2 Related Works

The sustainable development of smart cities requires reliable and efficient transportation systems [25]. Internet of Things (IoT) can be applied with the existing infrastructure and service networks for the design of the transportation systems, such as

software-defined networks and communication technologies[154-156]. IoT based Intelligent transportation system (IoT-ITS) can be classified into four main fields: Advanced Traveler Information System (ATIS), Advanced Public Transportation System (APTS), Advanced Traffic Management System (ATMS), and Emergency Management System (EMS) [156]. Transportation systems are shifting from conventional technology-driven systems to more powerful multifunctional data-driven ITS [29], [157, 158]. Massive traffic sensor data gathered by various sensors are vital for informed, scientific decision-make processes in traffic operation, pavement design, and transportation planning [32]. Data analytics in ITS consider important factors that influence decision-making processes, such as travel time or traffic congestion of public transport services [33, 34]. The fusion of traffic data from multiple sources produces a better understanding of the observations for reaching a better inference in ITS [35] [159-161].

Accurate estimation of travel time is essential to the success of ATMS and ATIS [162]. The approaches to studying travel time prediction can be mainly divided into three categories: knowledge-driven, model-driven, and data-driven. Knowledge-driven approaches usually employ a database, a knowledge base in the form of rules, and an inference engine in the form of algorithms [135]. Lee et al. proposed a knowledge-based expert system that predicted travel time by combining general rules from location-based service applications and meta-rules from human domain experts [163]. Nonetheless, as the knowledge base becomes increasingly large, the time to obtain accurate predictions increases as well. Model-driven approaches can be divided into four levels: macroscopic (e.g. TOPL [164]), mesoscopic (e.g. DynaMIT [165], and Dynasmart [166]), cellular automaton (CA) (e.g. OLSIM [166]), and microscopic method (e.g. AIMSUM online [168]) [169]. In the past, most of the researches for travel time forecasting that have focused on model-based methods. Transport simulation software is intended for

simulating traffic state information on virtual networks. It is primarily focused on research in traffic control and management, such as the effects of ramp metering, variable speed limits, and traffic incidents. To perform research on model-based practices, we need to acquire and use travel demand data, which is known origin-destination (OD) matrix or population data [149]. Nevertheless, accurate OD data is difficult to obtain, time-consuming, and expensive. Presently, only a few institutions have accumulated essentially useful OD data to build integrated travel time forecasting systems.

Recently, data-driven approaches have been receiving increased attention and gained interest within the transportation research community due to the increased computing power being available and the vast amount of data collected in ITS. Deep learning leads to an advantage over conventional machine learning algorithms with big data analytics of urban traffic. Kumar et al. compared the performance of the data-driven artificial neural network (ANN) approach and the model-based Kalman filter (KF) approach, concerning bus travel time prediction in [170]. The experimental results show that the data-driven ANN can achieve better performance, but compared to KF, the model needs a rich set of data for neural network training. Hou and Edara proposed long short-term memory (LSTM) and convolutional neural network (CNN) to predict travel time in a road network; compared to CNN, random forests (RF), and gradient boosting machines (GBM), the computation time of LSTM is the shortest in the model training process and prediction process [171]. Petersen et al. utilised the convolutional LSTM to propose a multi-output, multi-time-step system for bus travel time prediction [152]. Yu et al. presented a random forest based on the near neighbor (RFNN) model to predict the travel times of buses between bus stops, which include running time and waiting time as two input variables separately. Correspondingly, the model also considers traffic conditions, which is an essential factor affecting bus travel time [172]. However, the study on the bus

journey time forecasting is rather limited. Our work focuses on forecasting the travel time of the bus journey for travelers. A *trip* is to use one transport mode to travel on a single line or route, and a *journey* has one or more trips where occurs transfers between bus services during a period of travel time [151]. Therefore, there is still a need for developing a well-designed system framework to discover the advantages of various methods that achieve a deterministic and the meaningful outcome, which is closer to the real world's needs.

However, none of the existing studies consider the travel time problem of a bus journey via the ConvLSTM with the self-attention mechanism. Thus, the objective of our study is to predict the travel time of bus journeys by leveraging a data fusion component, which offers appropriate inputs to deep learning models.

3 Methodology

3.1 Bus Travel Time

In this section, we define some terms in Table 1, which will be used throughout the rest of the paper.

Table 1. List of Important Notations

Symbol	Description
T	bus trip id T
n	number of bus stops in T
S	a bus stop in a trip T
t_d	bus departure time from the station S
t_a	bus arrival time at the station S
t_{total}	total time of a trip T
R	actual running time in T

D	actual waiting time in T
$\hat{R}$	predicted running time in T
$\hat{D}$	predicted waiting time in T
Y	actual value of evaluation metrics
$\hat{Y}$	predicted value of evaluation metrics

A bus usually runs along a fixed route based on a regular schedule. Travel time depicted in Figure 1 is the time cost to complete a trip, which departs at time t. It follows an itinerary characterized by an original station A, a destination station B, and some stops (e.g., station $S_1$and station S_2).

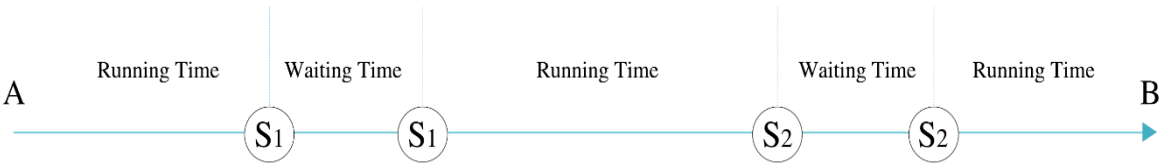


Figure 1. Running time and waiting time for a bus trip

In this paper, we predict the total travel time of a bus journey by using actual running time and waiting time from open data. For any stops in the trip, a bus is scheduled to arrive and depart from a stop S at different specified times, defined in the timetable, respectively, $t_d(T,S)$ and $t_a(T,S)$. In general, travel time forecasting is an estimate of the trip from a station of origin to a station of destination. Running time is the absolute difference between the arrival time of the current station and the departure time of the previous station, such as $R_2 = t_a(T,S_2) - t_d(T,S_1)$. Waiting time is the absolute difference between the departure time and the arrival time in a fixed stop station, such as, $D_1 = t_d(T,S_1) - t_a(T,S_1)$.

Our study defines segments based on information about the stops of a trip pattern. The segment-based method divides the stop points into running time and waiting time

segments. Our predictive models predict running and waiting times based on different t_a and t_d. According to Figure 1, it is evident that the numbers of input data for the prediction of running time and waiting time are different. Because for each trip of a specific bus, the running time will have one more record than the waiting time. The total travel time of a bus journey could be described with the equation (1).

$$t_{total} = \sum_i^n \hat{R} + \sum_i^{n-1} \hat{D}_i \quad (1)$$

3.2 Leveraging Machine Learning and Logical Reasoning

With the rapid development of ITS in recent years, data availability issues have always plagued researchers. Notably, the studies of multi-modal transport require a large amount of data from diverse data sources. Open data platforms release a variety of data that is freely available to everyone to reuse. Moreover, domain experts structure and classify data, such as General Transit Feed Specification (GTFS) and GTFS-Realtime [173]. Researchers can create structured data, namely the process of data curation, for the corresponding studies through data cleansing and data fusion. To predict a complex and uncertain event, we need to have multiple sources of data to provide more information for generating a predictive model.

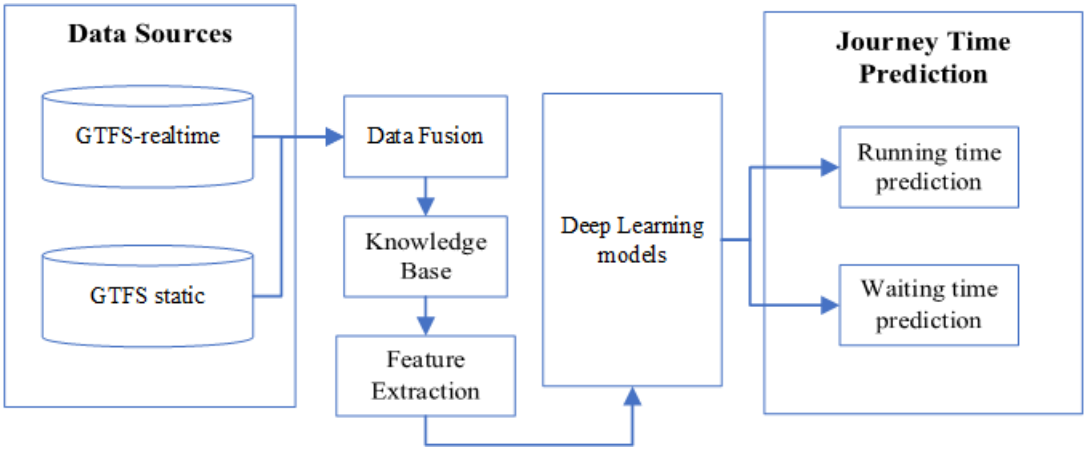


Figure 2. The framework of journey time prediction

Figure 2 illustrates the framework of an integrated system for journey time prediction, which consists of 6 components: GTFS-Realtime and GTFS static data stores,

data fusion, knowledge-base, feature extraction, deep learning models, and running time prediction and waiting time prediction. As Figure 2 shows, in the first step, we collected data from two types of GTFS and cleansed them, for example, by deleting duplicate data and sorting the data in chronological order. In order to build a knowledge base, the data fusion approach plays an essential role. Data from different data sources sometimes cannot be integrated and saved into a relational database or a two-dimensional data format, due to some data fail to match one-to-one or one-to-many mapping relationships, such as the running time from the station S_1 to S_2 and probe vehicle speed data. The use of the knowledge base enables deep learning models to exploit logical reasoning from data. Applying domain knowledge classify the raw data not only avoids the impact of irrelevant data but also reduces the computation time of the model. Furthermore, data fusion employs mathematical methods and programming languages to synthesize useful information or inferences. The theoretical framework can also be developed as an extended version to involve verification mechanisms [174].

3.3 Bus Journey Travel Time with Multi-Step Time Series Prediction

ConvLSTM model is a powerful kind of recurrent neural network (RNN), with a combination of convolutional and LSTM layers, which contains operation inside the LSTM cell [139]. On the other hand, the travel time prediction of a bus journey can be treated as a time series prediction problem. In recent years, LSTM is an elegant solution to the time series analysis by exploiting spatiotemporal data. Additionally, the ConvLSTM applies the convolution operators to capture the spatial and temporal dependencies in the dataset so that it generally performs better than fully connected LSTM (FC-LSTM) [139]. The calculation steps are as follows.

Firstly, calculate the input gate:

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \quad (2)$$

forget gate:

$$f_t = \sigma\left(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right) \quad (3)$$

cell state:

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

output gate:

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \circ c_t + b_o) \quad (5)$$

hidden state:

$$h_t = o_t \circ tanh(c_t) \quad (6)$$

where σ is a sigmoid function, $\circ$ is the Hadamard product, $*$ is the convolution operator. W_{xi}, W_{xf}, W_{xc}, W_{xo} are the weight matrices connecting the inputs $x_1,\ldots, x_t$ to three gates and the cell input, W_{hi}, W_{hf}, W_{hc}, W_{ho} are the weight matrices connecting the hidden states $h_1,\ldots, h_{t-1}$ to three gates and the cell input, W_{ci}, W_{cf}, W_{co} are the weight matrices connecting the $c_1,\ldots, c_t$ to three gates, b_i, b_f, b_c, b_o are the bias terms of three gates and the cell state.

Recently, the attention mechanism has succeeded in a wide range of sequence-to-sequence learning tasks [124] [175, 176]. Liang et al. presented a multi-level attention-based recurrent neural network for predicting geo-sensory time series [177]. The attention model focuses on the vital issue with LSTM based model for bus travel time prediction, which tends to select near-term data that is highly correlated to future travel time. In our experiments, the encoder is the underlying ConvLSTM model generating the hidden state representation h_t. We leverage a self-attention mechanism to the inputs after the operations of (1)-(6).

$$m_{t,t'} = tanh(W_m h_t + W_{m'} h_{t'} + b_m) \quad (7)$$

$$e_{t,t'} = \sigma(W_a m_{t,t'} + b_a) \quad (8)$$

$$a_t = softmax(e_t) \tag{9}$$

$$l_t = \sum_{t'=1}^{n} a_{t,t'} \cdot h_{t'} \tag{10}$$

where $a_{t,t'}$ is an attention matrix. b_m and b_a express bias terms. W_m, and $W_{m'}$ express weight matrices corresponding to the hidden states h_t, $h_{t'}$. Finally, l_t represents a weighted sum of $h_{t'}$ [178].

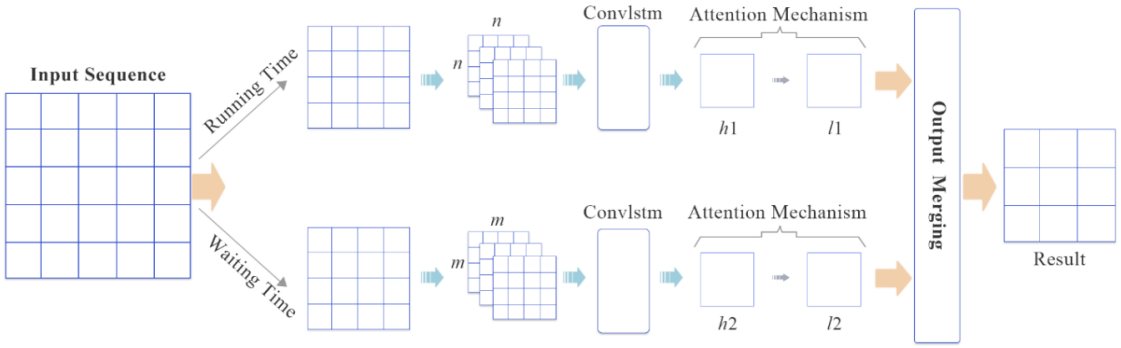


Figure 3. Self-attention based ConvLSTM network

Figure 3 demonstrates an overview of our proposed model, which consists of two main components: running time prediction and waiting time prediction, which are two independent components for estimating running and waiting times based on GTFS-Realtime. The first step is dividing historical observations from a sequence dataset into two smaller sequence datasets so that arranging the input data of the ConvLSTM model into a 3D-tensor for a single bus line. For example, in N days samples, time steps k, a sequence of running times R_i with a single bus line can be represented as (N, k, R_i). Secondly, l_1 and l_2 shows how much weight of historical observations affects predicted values. Finally, the outputs are merged to get the results by using equation (1).

The entire training process of an attention ConvLSTM is presented in Algorithm 1. We firstly construct multiple historical observation sequences as inputs. Then, the model is trained to predict the running time and waiting time separately.

Algorithm 1: Attention-based ConvLSTM Training Algorithm

Require:

Historical running time and waiting time observations:

$(R_1^T, R_2^T \ldots R_n^T)$ *and* $(D_1^T, D_2^T \ldots D_{n-1}^T)$;

Sequence length: *n*;

Lengths of running time, waiting time*:* l_R, l_D ;

running time: *R*;

waiting time: *D*.

Ensure: Attention-based ConvLSTM Model

for $epoch = 1$ to *max – epoch* do

Perform forward propagation recurrently using equation (2)-(10) to calculate

$$S_R = (R_1^T, R_2^T \ldots R_n^T)$$

$$S_D = (D_1^T, D_2^T \ldots D_n^T)$$

compute output error:

$$Y_R - \hat{Y}_R$$

$$Y_D - \hat{Y}_D$$

merging the predicted outputs to obtain the total travel time:

$$t_{total} = \hat{Y}_R + \hat{Y}_D$$

end for

4 Experiments and Discussion

4.1 Dataset Description and Preprocessing

We verify our model on real-world traffic datasets from TfNSW (Transport for NSW) Open Data, Bus Realtime Trip Update (BRTU) collected by a Python program that

read the TfNSW real-time feed Application Programming Interfaces (APIs) [13]. The dataset contains key attributes of bus journey information with corresponding timestamps, as detailed below.

BRTU was gathered from Sydney's bus system in real-time. For our experiment, the data is collected every 60 seconds, about 12 GB of data a day. (Note: the better frequency is 10 seconds, around 60 GB a day). The period used is from 6th May 2019 to 28th Jun 2019 except the weekends. We select the first three weeks of historical travel time records as a training set, and the rest serves as a test set, respectively. BRTU has information about departure time, arrival time, delay, route. GTFS-static contains station names, coordinates, and route names.

Table 2. Training details about self-attention based ConvLSTM

Variable	Value
learning rate	0.001
epochs	20
batch size	16
loss	Mean Squared Error
optimizer	Adam

The proposed model and other comparative models are implemented in Python via the TensorFlow Framework [143] and trained with the Adam algorithm [179]. The proposed network composed of several layers, a ConvLSTM2D [139], a flatten layer, a RepeatVector layer, a self-attention layer, and two TimeDistributed Layers. Training details about the network are presented in Table 2.

4.2 Evaluation Metrics and Results

In our experiments, we applied two standard metrics to evaluate the performance of running time prediction and waiting time prediction, including root mean square errors (RMSE) and mean absolute errors (MAE). They have been defined as presented in Equation (11) and Equation (12), where y_t represents the actual value for sample *t* and $\hat{y}_t$ represents the predicted value. As the multi-time-step model predicts bus travel time for all stops for the next *n* time-steps, both y_t and $\hat{y}_t$ have the dimensionality (N, k, R_i).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - \hat{y}_t)^2} \tag{11}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_t - \hat{y}_t| \tag{12}$$

We explored the patterns of bus running time and waiting time on weekdays. Respectively, Table 3 and Table 4 present the results of the trip id "27134" from Campbelltown station to Narellan Town Centre station. The trip "27134" has 37 records per day. As evidenced by the results, the performance of three types of LSTM does not have many differences. The output of our experiments is consistent with Greff et al.'s findings as well [141]. Standard LSTM and variant versions do not have significant performance differences.

Our design is to explore the pattern of each record (a stop). As can be seen from Table 3 and Table 4, we found that the attention ConvLSTM is a more stable model by observing each prediction result. It adjusts the predictions reasonably based on previous inputs. However, it cannot model very long-range temporal dependencies (e.g., period and trend), and training becomes more complicated when the depth increases [180].

Simply put, when the amount of input data increases, the time calculated by the model will increase dramatically. The attention mechanism can effectively overcome the drawbacks of modeling long-range temporal dependencies. Additionally, it could reduce computation time in every training by using less training data.

Table 3. Performance comparison of the bus running time prediction models for a stop

Models	RMSE (sec)		MAE (sec)	
	Mean	**SD**	**Mean**	**SD**
CNN	121.770	15.350	115.095	18.318
LSTM	49.849	5.046	47.146	4.583
ConvLSTM	43.720	15.468	37.533	13.821
Attention-ConvLSTM	41.449	5.623	36.328	4.539

Table 4. Performance comparison of the bus waiting time prediction models for a stop

Models	RMSE (sec)		MAE (sec)	
	Mean	**SD**	**Mean**	**SD**
CNN	7.891	6.415	6.912	1.747
LSTM	6.415	0.283	5.544	0.284
ConvLSTM	5.683	0.113	5.060	0.134
Attention-ConvLSTM	3.740	0.227	3.166	0.441

Our study defines segments based on information about the stops of a trip pattern. The segment-based method divides the stop points into running time and waiting time segments. Our predictive models predict running and waiting times based on different t_a and t_d. According to Figure 1, it is evident that the numbers of input data for the prediction of running time and waiting time are different. Because for each trip of a specific bus, the running time will have one more record than the waiting time. The total travel time of a bus journey could be described with the equation (1).

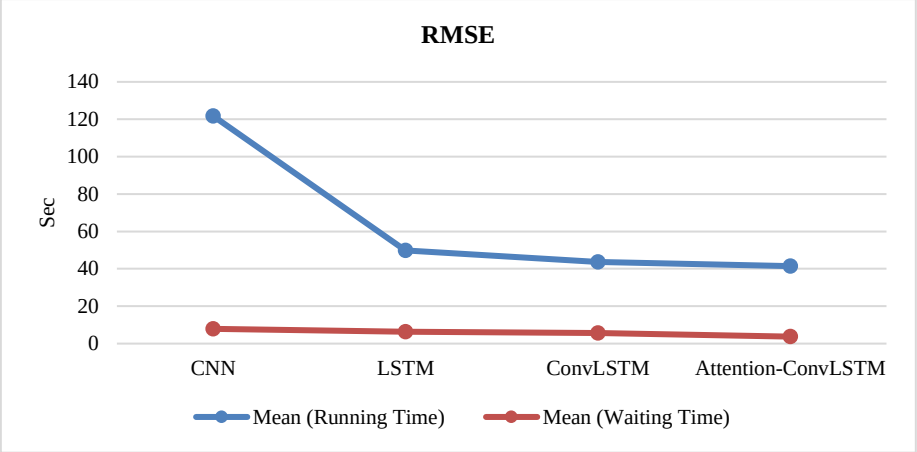
To further verify the performance, we use LSTM and attention-based ConvLSTM to predict the running time and waiting time of one of the stops, “Mt Annan Leisure

Centre, Welling Dr" (stop 18), respectively. In Table 3, a significant difference occurred. By observing each predicted value of the CNN model, we find that it has a significant difference between upper and lower bounds for the CNN model. In this case, the prediction of the model is very unreliable. Compared with the results of LSTM models, it can be seen that the forecast results are improved in Table 3 and Table 4. Attention-based ConvLSTM's mean errors and standard deviation (SD) are the lowest. In conclusion, attention-based ConvLSTM achieves the best overall performance compared to other models in Table 3,4. It is a more reliable model to predict travel time on data with large residuals than other models.

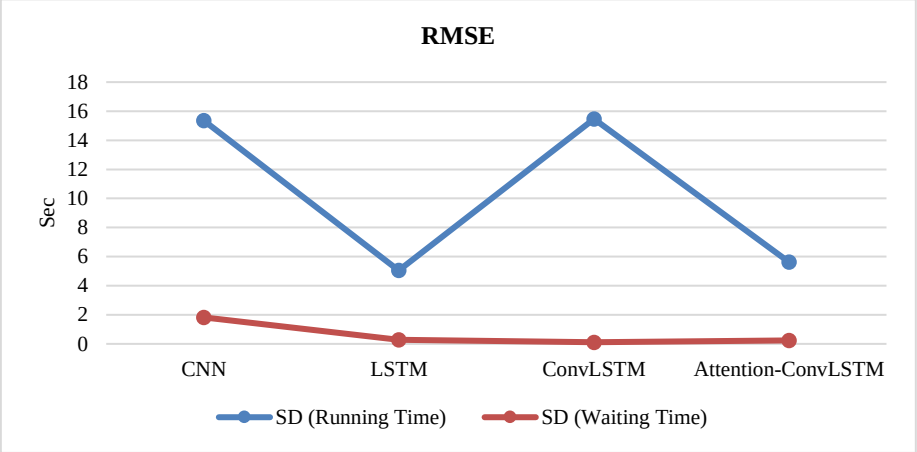
It is worth to mention that our aim is not to solely improve the accuracy of predictions, due to deep neural networks are less interpretable. Instead, we strive to find a practical data-driven model on open data by exploring the combination of deep learning methods and domain knowledge. Moreover, GTFS provides uncertainty values, which can be utilized to test the robustness of the generic model. The model based on GTFS will have a level of portability and reproducibility to the application in real scenarios.

Figure 4 reports the performance of CNN, LSTM, ConvLSTM, and Attention-ConvLSTM for predicting running time and waiting time. The *y*-axes of RMSE and MAE from (**a**), (**b**), (**c**), (**d**) represent the errors in seconds, respectively. All models have significant prediction errors (mean and standard deviation) in running time predictions. Especially, CNN reaches the most significant prediction errors in all cases. The waiting times indicate small variations, which are to a great extent explained by the input in the corresponding models. A weak dependence on the journey travel time prediction is established. However, the variability of running times cannot be fully explained by the selected input variables. Additionally, it shows that Attention-ConvLSTM effectively

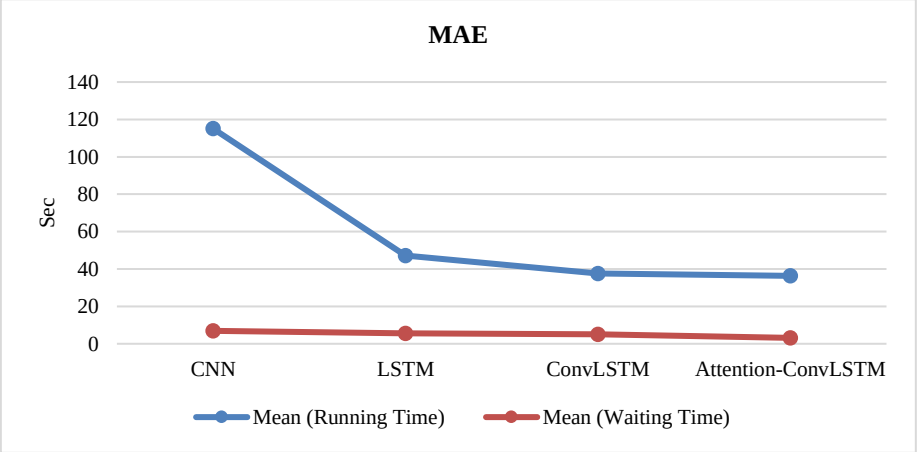
reduces errors. The proposed model needs to use more relevant factors to improve the predictions, such as vehicle speed or weather information.



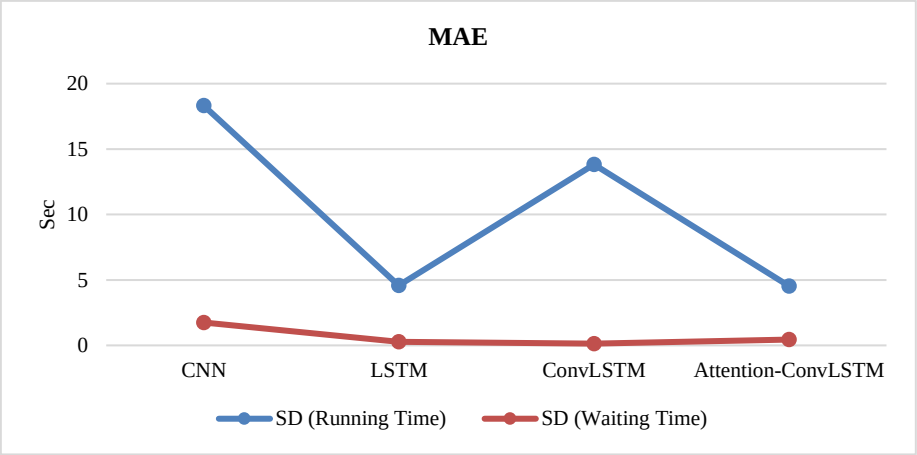
(a)



(b)



(c)



(d)

Figure 4. RMSE and MAE for the journey travel time prediction be listed as: (**a**) The mean of RMSE for running time and waiting time; (**b**) The standard deviation of RMSE for running time and waiting time; (**c**) The mean of MAE for running time and waiting time; (**d**) The standard deviation of MAE for running time and waiting time.

5 Conclusions and Future Work

In this paper, we investigated the problem of predicting bus journey travel time with public available GTFS data by taking into account the bus running time along the routes and waiting time at stop points. The basic idea is to use domain knowledge to classify raw data for obtaining a knowledge base, which can offer useful information for assisting in the deep learning models to explore the hidden patterns of the data. Thus, we proposed a comprehensive framework for using open data to bridge deep learning models and logical reasoning from a knowledge base. We used an attention-based ConvLSTM to predict the running time and waiting time separately. Ultimately, the total travel time prediction is obtained by merging the predicted outputs.

In the future, we will consider adding weather information, vehicle speed, and traffic condition data into our deep learning models. Furthermore, we will explore

evolutionary algorithms to find the best dataset size for accurate prediction of travel time, and to find the best model number of layers and number of units per layer. According to our experiments, using GTFS data exchange API will be easier to obtain high-quality input data for multi-modal traffic prediction studies. Our future work will also focus on employing more advanced data-driven models to shift from single-mode prediction to multi-modal prediction.

Bibliography

[1] United Nations. "68% of the World Population Projected to Live in Urban Areas by 2050, Says UN." https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html (accessed on: 25 May 2020).

[2] R. v. Nes, *Design of Multimodal Transport Networks : a Hierarchical Approach*. Delft, Netherlands: DUP Science, 2002.

[3] K. Sadler. "Swedish operator Stockholmstag develops model to predict and avoid rail delays." https://www.globalrailwayreview.com/news/24761/swedish-operator-stockholmstag-develops-model-to-predict-and-avoid-rail-delays(accessed on: 29 May 2021).

[4] J. Wu, Q. Wu, J. Shen, and C. Cai, "Towards Attention-Based Convolutional Long Short-Term Memory for Travel Time Prediction of Bus Journeys," *Sensors*, vol. 20, no. 12, p. 3354, 2020.

[5] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: a Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787-2805, 2010.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[7] F. Ghofrani, Q. He, R. M. Goverde, and X. Liu, "Recent Applications of Big Data Analytics in Railway Transportation Systems: a Survey," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 226-246, 2018.

[8] J. Wu, L. Zhou, C. Cai, J. Shen, S. K. Lau, and J. Yong, "Data Fusion for MaaS: Opportunities and Challenges," in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2018: IEEE, pp. 642-647.

[9] N. Papathanasiou, B. T. Adey, and M. Burkhalter, "Defining and Quantifying

Railway Service to Plan Infrastructure Interventions," *Infrastructure Asset Management*, vol. 7, no. 3, pp. 146-166, 2019.

[10] S. J. Barbeau, "Quality Control-Lessons Learned from the Deployment and Evaluation of GTFS-realtime Feeds," in *97th Annual Meeting of the Transportation Research Board, Washington, DC*, 2018. [Online]. Available: https://trid.trb.org/view/1496848

[11] C. Brakewood, G. S. Macfarlane, and K. Watkins, "The Impact of Real-Time Information on Bus Ridership in New York City," *Transportation Research Part C: Emerging Technologies*, vol. 53, pp. 59-75, 2015.

[12] L. Tang and P. V. Thakuriah, "Ridership Effects of Real-Time Bus Information System: A Case Study in the City of Chicago," *Transportation Research Part C: Emerging Technologies*, vol. 22, pp. 146-161, 2012.

[13] TfNSW. "General Transit Feed Specification (GTFS) and GTFS-Realtime (GTFS-R)." https://opendata.transport.nsw.gov.au/documentation (accessed on: 24 Jun 2020).

[14] C. Wen *et al.*, "Train Dispatching Management with Data-Driven Approaches: a Comprehensive Review and Appraisal," *IEEE Access*, vol. 7, pp. 114547-114571, 2019.

[15] L. Oneto *et al.*, "Train Delay Prediction Systems: A Big Data Analytics Perspective," *Big Data Research*, vol. 11, pp. 54-64, 2018.

[16] M. Yaghini, M. M. Khoshraftar, and M. Seyedabadi, "Railway Passenger Train Delay Prediction via Neural Network Model," *Journal of Advanced Transportation*, vol. 47, no. 3, pp. 355-368, 2013.

[17] S. Pongnumkul, T. Pechprasarn, N. Kunaseth, and K. Chaipah, "Improving Arrival Time Prediction of Thailand's Passenger Trains Using Historical Travel Times,"

in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2014: IEEE, pp. 307-312.

[18] L. Oneto *et al.*, "Dynamic Delay Predictions for Large-Scale Railway Networks: Deep and Shallow Extreme Learning Machines Tuned via Thresholdout," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2754-2767, 2017.

[19] Google. "Google Transit APIs." Google, Inc. https://developers.google.com/transit/ (accessed on: 20 July 2020).

[20] A. Owen and D. M. Levinson, "Developing a Comprehensive US Transit Accessibility Database," in *Seeing Cities Through Big Data*: Springer, 2017, pp. 279-290.

[21] T. H. Davenport, "The Human Side of Big Data and High-Performance Analytics," *International Institute for Analytics*, vol. 1, no. 1, pp. 1-13, 2012.

[22] R. Nair *et al.*, "An Ensemble Prediction Model for Train Delays," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 196-209, 2019.

[23] TfNSW. "Sydney Network -Trains Punctuality Performance." https://www.transport.nsw.gov.au/data-and-research/passenger-travel/historical-trains-punctuality-performance-sydney-network (accessed on: 10 Dec 2020).

[24] N. Bešinović, "Resilience in Railway Transport Systems: A Literature Review and Research Agenda," *Transport Reviews*, vol. 40, no. 4, pp. 457-478, 2020/07/03 2020, doi: 10.1080/01441647.2020.1728419.

[25] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer, "UAV-Enabled Intelligent Transportation Systems for the Smart City: Applications and Challenges," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 22-28, 2017.

[26] A. Maimaris and G. Papageorgiou, "A Review of Intelligent Transportation Systems from A Communications Technology Perspective," in *IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 54-59.

[27] J. Wu, Y. Huang, J. Kong, Q. Tang, and X. Huang, "A Study on the Dependability of Software Defined Networks," in *International Conference on Materials Engineering & Information Technology Applications*, 2015, pp. 314-318.

[28] P. Patel, Z. Narmawala, and A. Thakkar, "A Survey on Intelligent Transportation System Using Internet of Things," Singapore, 2019: Springer Singapore, in Emerging Research in Computing, Information, Communication and Applications, pp. 231-240.

[29] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624-1639, 2011, doi: 10.1109/TITS.2011.2158001.

[30] K. Qureshi and A. Abdullah, "A Survey on Intelligent Transportation Systems," *Middle-East Journal of Scientific Research*, vol. 15, pp. 629-642, 2013.

[31] Y. Yuan and F. Wang, "Towards Blockchain-Based Intelligent Transportation Systems," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2663-2668, doi: 10.1109/ITSC.2016.7795984.

[32] G. Zhang and Y. Wang, "Machine Learning and Computer Vision-Enabled Traffic Sensing Data Analysis and Quality Enhancement," in *Data-Driven Solutions to Transportation Problems*, Y. Wang and Z. Zeng Eds.: Elsevier, 2019, pp. 51-79.

[33] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big Data Analytics in Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent*

Transportation Systems, vol. 20, no. 1, pp. 383-398, 2019, doi: 10.1109/TITS.2018.2815678.

[34] S. Duleba and S. Moslem, "Examining Pareto Optimality in Analytic Hierarchy Process on Real Data: An Application in Public Transport Service Development," *Expert Systems with Applications*, vol. 116, pp. 21-30, 2019/02/01/ 2019, doi: https://doi.org/10.1016/j.eswa.2018.08.049.

[35] N.-E. E. Faouzi, H. Leung, and A. Kurian, "Data Fusion in Intelligent Transportation Systems: Progress and Challenges – A Survey," *Information Fusion*, vol. 12, no. 1, pp. 4-10, 2011, doi: https://doi.org/10.1016/j.inffus.2010.06.001.

[36] C. Bachmann, B. Abdulhai, M. J. Roorda, and B. Moshiri, "A Comparative Assessment of Multi-Sensor Data Fusion Techniques for Freeway Traffic Speed Estimation Using Microsimulation Modeling," *Transportation Research Part C Emerging Technologies*, vol. 26, no. JAN., pp. 33-48, 2013.

[37] M. Rostami Shahrbabaki, A. A. Safavi, M. Papageorgiou, and I. Papamichail, "A Data Fusion Approach for Real-Time Traffic State Estimation in Urban Signalized Links," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 525-548, 2018, doi: https://doi.org/10.1016/j.trc.2018.05.020.

[38] T. H. Chang, A. Y. Chen, C. W. Chang, and C. H. Chueh, "Traffic Speed Estimation through Data Fusion from Heterogeneous Sources for First Response Deployment," *Journal of Computing in Civil Engineering*, vol. 108, no. 6, pp. 30-6, 2014.

[39] A. S. Olafsson, T. S. Nielsen, and T. A. Carstensen, "Cycling in multimodal transport behaviours: Exploring modality styles in the Danish population," *Journal of transport geography*, vol. 52, pp. 123-130, 2016.

[40] C. Nobis, "Multimodality:Facets and Causes of Sustainable Mobility Behavior," *Transportation Research Record*, vol. 2010, no. 1, pp. 35-44, 2007, doi: 10.3141/2010-05.

[41] S. Heikkilä, "Mobility as a Service-a Proposal for Action for the Public Administration, Case Helsinki," https://aaltodoc.aalto.fi/handle/123456789/13133 (accessed on: 6 Nov 2020).

[42] D. A. Hensher, "Future Bus Transport Contracts under a Mobility as a Service (MaaS) Regime in the Digital Age: Are They Likely to Change?," *Transportation Research Part A: Policy and Practice*, vol. 98, pp. 86-96, 2017.

[43] E. Hannon, C. McKerracher, I. Orlandi, and S. Ramkumar, "An Integrated Perspective on the Future of Mobility," http://www.mckinsey.com/business-functions/sustainability-and-resource-productivity/our-insights/an-integrated-perspective-on-the-future-of-mobility (accessed on: 17 Nov 2020).

[44] C. Mulley *et al.*, "Mobility as a Service in Community Transport in Australia: Can It Provide a Sustainable Future?," *Transportation Research Part A: Policy and Practice*, vol. 131, pp. 107-122, 2020.

[45] M. Kamargianni and M. Matyas, "The Business Ecosystem of Mobility-as-a-Service," in *Transportation Research Board 96th Annual Meeting*, Washington DC, 2017.

[46] R. Merkert, J. Bushell, and M. J. Beck, "Collaboration as a Service (CaaS) to Fully Integrate Public Transportation–Lessons from Long Distance Travel to Reimagine Mobility as a Service," *Transportation Research Part A: Policy and Practice*, vol. 131, pp. 267-282, 2020.

[47] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban Computing: Concepts, Methodologies, and Applications," *ACM Transactions on Intelligent Systems and*

Technology (TIST), vol. 5, no. 3, pp. 1-55, 2014.

[48] M. K. E. Mahrsi, E. Côme, L. Oukhellou, and M. Verleysen, "Clustering Smart Card Data for Urban Mobility Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 712-728, 2017, doi: 10.1109/TITS.2016.2600515.

[49] M.-P. Pelletier, M. Trépanier, and C. Morency, "Smart Card Data Use in Public Transit: A Literature Review," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 557-568, 2011, doi: https://doi.org/10.1016/j.trc.2010.12.003.

[50] B. Du and P.-A. Dublanche, "Bus Bunching Identification using Smart Card Data," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018: IEEE, pp. 1087-1092.

[51] B. Du, "Estimating Travellers' Trip Purposes using Public Transport Data and Land Use Information," *Tenth Triennial Symposium on Transportation Analysis (TRISTAN X)*, 2019, pp. 1-4.

[52] Z. Gong, B. Du, Z. Liu, W. Zeng, P. Perez, and K. Wu, "SD-Seq2seq: A Deep Learning Model for Bus Bunching Prediction based on Smart Card Data," presented at the The 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, Hawaii, USA, 2020.

[53] P. Gaudette, R. Chapleau, and T. Spurr, "Bus Network Microsimulation with General Transit Feed Specification and Tap-in-Only Smart Card Data," *Transportation Research Record*, vol. 2544, no. 1, pp. 71-80, 2016, doi: 10.3141/2544-09.

[54] N. Nassir, A. Khani, S. G. Lee, H. Noh, and M. Hickman, "Transit Stop-Level Origin–Destination Estimation through Use of Transit Schedule and Automated

Data Collection System," *Transportation Research Record*, vol. 2263, no. 1, pp. 140-150, 2011.

[55] A. Alsger, B. Assemi, M. Mesbah, and L. Ferreira, "Validating and Improving Public Transport Origin–Destination Estimation Algorithm using Smart Card Fare Data," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 490-506, 2016.

[56] A. Giraud, F. Légaré, M. Trépanier, and C. Morency, "Data Fusion of APC, Smart Card and GTFS Data to Visualize Public Transit Use," [Online]. Available: https://trid.trb.org/view/1435839 (accessed on: 8 Jan 2021).

[57] P. Kumar, A. Khani, and Q. He, "A Robust Method for Estimating Transit Passenger Trajectories using Automated Data," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 731-747, 2018.

[58] A. Alsger, A. Tavassoli, M. Mesbah, L. Ferreira, and M. Hickman, "Public Transport Trip Purpose Inference using Smart Card Fare Data," *Transportation Research Part C: Emerging Technologies*, vol. 87, pp. 123-137, 2018.

[59] J. Wu, L. Zhou, C. Cai, F. Dong, J. Shen, and G. Sun, "Towards a General Prediction System for the Primary Delay in Urban Railways," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019: IEEE, pp. 3482-3487.

[60] W. Zeng, C.-W. Fu, S. M. Arisona, A. Erath, and H. Qu, "Visualizing Mobility of Public Transportation System," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1833-1842, 2014.

[61] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing Transportation Systems via Deep Learning: A Survey," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 144-163, 2019.

[62] T. Kusakabe and Y. Asakura, "Behavioural Data Mining of Transit Smart Card Data: A Data Fusion Approach," *Transportation Research Part C: Emerging Technologies,* vol. 46, pp. 179-191, 2014.

[63] Z. Huang *et al.*, "Modeling Real-Time Human Mobility Based on Mobile Phone and Transportation Data Fusion," *Transportation Research Part C: Emerging Technologies,* vol. 96, pp. 251-269, 2018.

[64] Z. Zhao, H. N. Koutsopoulos, and J. Zhao, "Discovering Latent Activity Patterns from Transit Smart Card Data: A Spatiotemporal Topic Model," *Transportation Research Part C: Emerging Technologies,* vol. 116, p. 102627, 2020.

[65] I. F. Ilyas and X. Chu, *Data Cleaning*. Association for Computing Machinery, 2019.

[66] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data: A Survey," *IEEE Transactions on Knowledge and data Engineering,* vol. 26, no. 9, pp. 2250-2267, 2013.

[67] D. J. Hill and B. S. Minsker, "Anomaly Detection in Streaming Environmental Sensor Data: A Data-Driven Modeling Approach," *Environmental Modelling & Software,* vol. 25, no. 9, pp. 1014-1022, 2010.

[68] N. D. Le, R. D. Martin, and A. E. Raftery, "Modeling Flat Stretches, Bursts Outliers in Time Series Using Mixture Transition Distribution Models," *Journal of the American Statistical Association,* vol. 91, no. 436, pp. 1504-1515, 1996.

[69] R. S. Tsay, D. Pena, and A. E. Pankratz, "Outliers in Multivariate Time Series," *Biometrika,* vol. 87, no. 4, pp. 789-804, 2000.

[70] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott, "Inferring Causal Impact using Bayesian Structural Time-Series Models," *The Annals of Applied Statistics,* vol. 9, no. 1, pp. 247-274, 2015.

[71] S. L. Scott and H. R. Varian, "Predicting the Present with Bayesian Structural Time Series," *International Journal of Mathematical Modelling and Numerical Optimisation,* vol. 5, no. 1-2, pp. 4-23, 2014.

[72] S. L. Scott and H. R. Varian, "Bayesian Variable Selection for Nowcasting Economic Time Series," *Economic Analysis of the Digital Economy,* p. 119, 2015.

[73] J. Peters, D. Janzing, and B. Schölkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

[74] S. R. Jammalamadaka, J. Qiu, and N. Ning, "Multivariate Bayesian Structural Time Series Model," *The Journal of Machine Learning Research,* vol. 19, no. 1, pp. 2744-2776, 2018.

[75] M. Johnson and A. Willsky, "Stochastic Variational Inference for Bayesian Time Series Models," in *International Conference on Machine Learning*, 2014, pp. 1854-1862.

[76] Q. E. McCallum, *Bad Data Handbook: Cleaning Up the Data So You Can Get Back to Work*. " O'Reilly Media, Inc.", 2012.

[77] H. Fan, F. Wang, and M. Zheng, "Research on Knowledge Fusion Connotation and Process Model," in *China Conference on Knowledge Graph and Semantic Computing*, 2016: Springer, pp. 184-195.

[78] L. Snidaro, J. Garcia-Herrera, J. Llinas, and E. Blasch, "Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge," Advances in Computer Vision and Pattern Recognition, 2016.

[79] C. Zins, "Conceptual Approaches for Defining Data, Information, and Knowledge," *Journal of the American Society for Information Science and Technology,* vol. 58, no. 4, pp. 479-493, 2007.

[80] X. L. Dong *et al.*, "From Data Fusion to Knowledge Fusion," *Proceedings of the*

VLDB Endowment, vol. 7, no. 10, pp. 881-892, 2014.

[81] Y. Zheng, "Methodologies for Cross-Domain Data Fusion: An Overview," *IEEE Transactions on Big Data,* vol. 1, no. 1, pp. 16-34, 2015.

[82] C. Boutsidis and E. Gallopoulos, "SVD Based Initialization: A Head Start for Nonnegative Matrix Factorization," *Pattern Recognition,* vol. 41, no. 4, pp. 1350-1362, 2008.

[83] M. Žitnik and B. Zupan, "Data Fusion by Matrix Factorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 1, pp. 41-53, 2014.

[84] F. Zhang, N. J. Yuan, D. Wilkie, Y. Zheng, and X. Xie, "Sensing the Pulse of Urban Refueling Behavior: A Perspective from Taxi Mobility," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 6, no. 3, pp. 1-23, 2015.

[85] C. Xu, D. Tao, and C. Xu, "A Survey on Multi-View Learning," *arXiv preprint arXiv:1304.5634,* 2013.

[86] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-View Learning Overview: Recent Progress and New Challenges," *Information Fusion,* vol. 38, pp. 43-54, 2017.

[87] J. Sun, Y. Wang, H. Si, J. Yuan, and X. Shan, "Aggregate Human Mobility Modeling using Principal Component Analysis," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications,* vol. 1, no. 2/3, pp. 83-95, 2010, doi: 10.22667/JOWUA.2010.09.31.083.

[88] D. Lahat, T. Adal, and C. Jutten, "Multimodal Data Fusion: An Overview of Methods, Challenges and Prospects," *Proceedings of the IEEE,* vol. 103, no. 9, pp. 1449-1477, 2015.

[89] D. Chu, L. Z. Liao, M. K. Ng, and X. Zhang, "Sparse Canonical Correlation

Analysis: New Formulation and Algorithm," *IEEE Transactions on Pattern Analysis & Machine Intelligence,* vol. 35, no. 12, pp. 3050-3065, 2013.

[90] D. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical Correlation Analysis: An Overview with Application to Learning Methods," *Neural computation,* vol. 16, no. 12, pp. p. 2639-2664, 2004.

[91] Y. Luo, D. Tao, K. Ramamohanarao, C. Xu, and Y. Wen, "Tensor Canonical Correlation Analysis for Multi-View Dimension Reduction," *IEEE Transactions on Knowledge & Data Engineering,* vol. 27, no. 11, pp. 3111-3124, 2015.

[92] M. Gönen, G. B. Gönen, and F. Gürgen, "Bayesian Multiview Dimensionality Reduction for Learning Predictive Subspaces," *Frontiers in Artificial Intelligence & Applications,* vol. 263, pp. 387-392, 2014.

[93] A. Farasat, A. Nikolaev, S. N. Srihari, and R. H. Blair, "Probabilistic Graphical Models in Modern Social Network Analysis," *Social Network Analysis and Mining,* vol. 5, no. 1, p. 62, 2015.

[94] L. PourMohammadBagher, M. M. Ebadzadeh, and R. Safabakhsh, "Graphical Model Based Continuous Estimation of Distribution Algorithm," *Applied Soft Computing,* vol. 58, pp. 388-400, 2017.

[95] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong, "Discovering Urban Functional Zones Using Latent Activity Trajectories," *IEEE Transactions on Knowledge and Data Engineering,* vol. 27, no. 3, pp. 712-725, 2014.

[96] B. Alharbi, A. Qahtan, and X. Zhang, "Minimizing User Involvement for Learning Human Mobility Patterns from Location Traces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.

[97] J. Y. Zhu *et al.*, "pg-Causality: Identifying Spatiotemporal Causal Pathways for Air Pollutants with Urban Big Data," *IEEE Transactions on Big Data,* vol. 4, no.

4, pp. 571-585, 2017.

[98] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A Survey of Heterogeneous Information Network Analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17-37, 2016.

[99] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of Predictability in Human Mobility," *Science*, vol. 327, no. 5968, pp. 1018-1021, 2010.

[100] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson, "Approaching the Limit of Predictability in Human Mobility," *Scientific reports*, vol. 3, p. 2923, 2013.

[101] K. Zhao, D. Khryashchev, and H. Vo, "Predicting Taxi and Uber Demand in Cities: Approaching the Limit of Predictability," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-1, 2019, doi: 10.1109/TKDE.2019.2955686.

[102] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[103] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[104] S. Wang, W. Zhang, and X. Qu, "Trial-and-Error Train Fare Design Scheme for Addressing Boarding/Alighting Congestion at CBD Stations," *Transportation Research Part B: Methodological*, vol. 118, pp. 318-335, 2018, doi: https://doi.org/10.1016/j.trb.2018.11.003.

[105] S. Wang and X. Qu, "Station Choice for Australian Commuter Rail Lines: Equilibrium and Optimal Fare Design," *European Journal of Operational Research*, vol. 258, no. 1, pp. 144-154, 2017/04/01/ 2017, doi: https://doi.org/10.1016/j.ejor.2016.08.040.

[106] M. F. Gorman, "Statistical Estimation of Railroad Congestion Delay," *Transportation Research Part E: Logistics and Transportation Review,* vol. 45, no. 3, pp. 446-456, 2009.

[107] P. Kecman and R. M. Goverde, "Predictive Modelling of Running and Dwell Times in Railway Traffic," *Public Transport,* vol. 7, no. 3, pp. 295-319, 2015.

[108] L. Oneto *et al.*, "Advanced Analytics for Train Delay Prediction Systems by Including Exogenous Weather Data," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 17-19 Oct. 2016 2016, pp. 458-467, doi: 10.1109/DSAA.2016.57.

[109] F. Corman and P. Kecman, "Stochastic Prediction of Train Delays in Real-Time using Bayesian Networks," *Transportation Research Part C: Emerging Technologies,* vol. 95, pp. 599-615, 2018.

[110] P. Huang *et al.*, "A Bayesian Network Model to Predict the Effects of Interruptions on Train Operations," *Transportation Research Part C: Emerging Technologies,* vol. 114, pp. 338-358, 2020.

[111] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*. New York: Springer Science & Business Media, 2009.

[112] P. Wang and Q.-p. Zhang, "Train Delay Analysis and Prediction Based on Big Data Fusion," *Transportation Safety and Environment,* vol. 1, no. 1, pp. 79-88, 2019.

[113] M. A. Nabian, N. Alemazkoor, and H. Meidani, "Predicting Near-Term Train Schedule Performance and Delay using Bi-Level Random Forests," *Transportation Research Record,* vol. 2673, no. 5, pp. 564-573, 2019.

[114] G. P. Zhang, "Time Series Forecasting using A Hybrid ARIMA and Neural Network Model," *Neurocomputing,* vol. 50, pp. 159-175, 2003.

[115] M. Khashei and M. Bijari, "A Novel Hybridization of Artificial Neural Networks and ARIMA Models for Time Series Forecasting," *Applied Soft Computing,* vol. 11, no. 2, pp. 2664-2675, 2011.

[116] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[117] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series," *The Handbook of Brain Theory and Neural Networks,* vol. 3361, no. 10, pp. 255-258, Cambridge, MA: MIT Press, 1995.

[118] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan, "Deep Spatial–Temporal 3D Convolutional Neural Networks for Traffic Data Forecasting," *IEEE Transactions on Intelligent Transportation Systems,* vol. 20, no. 10, pp. 3913-3926, 2019.

[119] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[120] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724-1734.

[121] T. Lin, T. Guo, and K. Aberer, "Hybrid Neural Networks for Learning the Trend in Time Series," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 2273-2279, doi: 10.24963/ijcai.2017/316.

[122] S. Du, T. Li, Y. Yang, and S. Horng, "Deep Air Quality Forecasting Using Hybrid Deep Learning Framework," *IEEE Transactions on Knowledge and Data Engineering,* pp. 1-1, 2019, doi: 10.1109/TKDE.2019.2954510.

[123] J. Wu *et al.*, "Multivariate Multi-Step Train Delay Forecasting: A Hybrid LSTM-CPS Solution," presented at the Transportation Research Board (TRB) 100th

Annual Meeting, 2021. [Online]. Available: https://trid.trb.org/view/1759773. (accessed on: 6 March 2021).

[124] A. Vaswani *et al.*, "Attention is All You Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998-6008.

[125] N. Wessel and M. J. Widener, "Discovering the Space–Time Dimensions of Schedule Padding and Delay from GTFS and Real-Time Transit Data," *Journal of Geographical Systems*, vol. 19, no. 1, pp. 93-107, 2017.

[126] J. Wong, "Leveraging the General Transit Feed Specification for Efficient Transit Analysis," *Transportation Research Record*, vol. 2338, no. 1, pp. 11-19, 2013.

[127] A. Owen and D. M. Levinson, "Modeling the Commute Mode Share of Transit Using Continuous Accessibility to Jobs," *Transportation Research Part A: Policy and Practice*, vol. 74, pp. 110-122, 2015.

[128] A. Guthrie, Y. Fan, and K. V. Das, "Accessibility Scenario Analysis of A Hypothetical Future Transit Network: Social Equity Implications of A General Transit Feed Specification–Based Sketch Planning Tool," *Transportation Research Record*, vol. 2671, no. 1, pp. 1-9, 2017.

[129] A. Harvey, *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge University Press, 1990.

[130] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, vol. abs/1312.6114, 2014.

[131] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *International Conference on Machine Learning*, 2014: PMLR, pp. 1278-1286.

[132] A. Yamamura, M. Koresawa, S. Adachi, and N. Tomii, "Identification of Causes of Delays in Urban Railways," *WIT transactions on the built environment*, vol.

127, pp. 403-414, May. 2012, doi:10.2495/CR120341.

[133] H. Huang, K. Li, and P. Schonfeld, "Real-Time Energy-Saving Metro Train Rescheduling with Primary Delay Identification," *PloS one*, vol. 13, no. 2, pp. 1-22, 2018.

[134] L. Oneto *et al.*, "Delay Prediction System for Large-Scale Railway Networks Based on Big Data Analytics," in *Inns Conference on Big Data*, 2016: Springer, pp. 139-150.

[135] G. Spring, "Knowledge-Based Systems in Transportation," *Artificial Intelligence in Transportation: Information for Application, Transportation Research Circular, No. E-C113, Transportation Research Board of the National Academies,* pp. 7-16, 2007.

[136] F. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to Time Series Predictable through Time-Window Approaches," in *Proceedings of the International Conference on Artificial Neural Networks*, 2001, pp. 669-676.

[137] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *Proceedings*, 2015, vol. 89: Presses universitaires de Louvain, pp. 89-94.

[138] A. Graves, "Generating Sequences with Recurrent Neural Networks," *arXiv preprint arXiv:1308.0850,* 2013.

[139] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems* 2015, vol. 1, pp. 802-810.

[140] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," in *International Conference on Machine*

Learning, 2017: PMLR, pp. 1243-1252.

[141] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232, 2017, doi: 10.1109/TNNLS.2016.2582924.

[142] I. Kontoyiannis, P. H. Algoet, Y. M. Suhov, and A. J. Wyner, "Nonparametric Entropy Estimation for Stationary Processes and Random Fields, with Applications to English Text," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 1319-1327, 1998.

[143] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," presented at *the Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, Savannah, GA, USA, 2016.

[144] F. Pedregosa *et al.*, "Scikit-Learn: Machine Learning in Python," *the Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[145] E. Stawiarska and P. Sobczak, "The Impact of Intelligent Transportation System Implementations on the Sustainable Growth of Passenger Transport in EU regions," *Sustainability*, vol. 10, no. 5, p. 1318, 2018.

[146] ESCAP, "Intelligent Transportation Systems for Sustainable Development in Asia and the Pacific," 2015. [Online]. Available: http://www.unescap.org/sites/default/files/ITS.pdf. (accessed on: 20 July 2020).

[147] J. A. Guerrero-ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration Challenges of Intelligent Transportation Systems with Connected Vehicle, Cloud Computing, and Internet of Things Technologies," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 122-128, 2015, doi: 10.1109/MWC.2015.7368833.

[148] JSCE. “Intelligent Transport Systems (ITS) Introduction Guide.” *Technical Committee Coordination Meeting*. 2016. [Online]. Available: https://www.jsce-int.org/system/files/ITS_Introduction_Guide_2.pdf (accessed on: 20 July 2020).

[149] D. Yanjie, L. Yisheng, and W. Fei-Yue, " Travel Time Prediction with LSTM Neural Network," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 1-4 Nov. 2016 2016, pp. 1053-1058, doi: 10.1109/ITSC.2016.7795686.

[150] A. O. Sullivan, F. C. Pereira, J. Zhao, and H. N. Koutsopoulos, "Uncertainty in Bus Arrival Time Predictions: Treating Heteroscedasticity With a Metamodel Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3286-3296, 2016, doi: 10.1109/TITS.2016.2547184.

[151] P. He, G. Jiang, S.-K. Lam, and D. Tang, "Travel-Time Prediction of Bus Journey With Multiple Bus Trips," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4192-4205, 2019, doi: 10.1109/TITS.2018.2883342.

[152] N. C. Petersen, F. Rodrigues, and F. C. Pereira, "Multi-Output Bus Travel Time Prediction with Convolutional LSTM Neural Network," *Expert Systems with Application*, vol. 120, pp. 426-435, 2019.

[153] X. Ran, Z. Shan, Y. Fang, and C. Lin, "An LSTM-Based Method with Attention Mechanism for Travel Time Prediction," *Sensors*, vol. 19, no. 4, p. 861, 2019.

[154] A. Maimaris and G. Papageorgiou, " A Review of Intelligent Transportation Systems from a Communications Technology Perspective," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 54-59, doi: 10.1109/ITSC.2016.7795531.

[155] J. Wu, Y. Huang, J. Kong, Q. Tang, and X. Huang, " A Study on the Dependability of Software Defined Networks," in *International Conference on Materials Engineering and Information Technology Applications (MEITA 2015)*, 2015: Atlantis Press, pp. 314-318.

[156] P. Patel, Z. Narmawala, and A. Thakkar, "A Survey on Intelligent Transportation System Using Internet of Things," in *Emerging Research in Computing, Information, Communication and Applications*: Springer, 2019, pp. 231-240.

[157] K. Qureshi and A. Abdullah, "A Survey on Intelligent Transportation Systems," *Middle-East Journal of Scientific Research*, vol. 15, no. 5, pp. 629-642, 2013.

[158] Y. Yuan and F. Wang, "Towards Blockchain-Based Intelligent Transportation Systems," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 1-4 Nov. 2016 2016, pp. 2663-2668, doi: 10.1109/ITSC.2016.7795984.

[159] C. Bachmann, B. Abdulhai, M. J. Roorda, and B. Moshiri, "A Comparative Assessment of Multi-Sensor Data Fusion Techniques for Freeway Traffic Speed Estimation Using Microsimulation Modeling," *Transportation Research Part C Emerging Technologies*, vol. 26, no. JAN., pp. 33-48, 2013.

[160] M. R. Shahrbabaki, A. A. Safavi, M. Papageorgiou, and I. Papamichail, "A Data Fusion Approach for Real-Time Traffic State Estimation in Urban Signalized Links," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 525-548, 2018, doi: https://doi.org/10.1016/j.trc.2018.05.020. (accessed on: 20 July 2020).

[161] T. H. Chang, A. Y. Chen, C. W. Chang, and C. H. Chueh, "Traffic Speed Estimation through Data Fusion from Heterogeneous Sources for First Response Deployment," *Journal of Computing in Civil Engineering*, vol. 108, no. 6, pp. 30-6, 2014.

[162] L. Shen and M. Hadi, "Practical Approach for Travel Time Estimation from Point Traffic Detector Data," *Journal of Advanced Transportation*, vol. 47, no. 5, pp. 526-535, 2013.

[163] W. H. Lee, S. S. Tseng, and S. H. Tsai, "A Knowledge Based Real-Time Travel Time Prediction System for Urban Network," *Expert Systems with Applications*, vol. 36, no. 3p1, pp. 4239-4247, 2009.

[164] A. Chow *et al.*, "TOPL: Tools for Operational Planning of Transportation Networks," in *ASME 2008 Dynamic Systems and Control Conference*, 2008, pp. 1035-1042, doi: 10.1115/dscc2008-2243.

[165] M. Ben-Akiva, M. Bierlaire, D. Burton, H. N. Koutsopoulos, and R. Mishalani, "Network State Estimation and Prediction for Real-Time Traffic Management," *Networks and Spatial Economics*, vol. 1, no. 3, pp. 293-318, 2001/09/01 2001, doi: 10.1023/A:1012883811652.

[166] M. T. Initiative, "DYNASMART-X Evaluation for Real-Time TMC Application: CHART Test Bed," 2005.

[167] R. Chrobok, S. F. Hafstein, and A. Pottmeier, "Olsim: A New Generation of Traffic Information Systems," *Forschung und wissenschaftliches Rechnen*, vol. 63, pp. 11-25, 2004.

[168] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic Simulation with Aimsun," in *Fundamentals of Traffic Simulation*: Springer, 2010, pp. 173-232.

[169] S. Oh, Y. J. Byon, K. Jang, and H. Yeo, "Short-Term Travel-Time Prediction on Highway: A Review on Model-Based Approach," *Ksce Journal of Civil Engineering*, vol. 22, no. 1, pp. 298-310, 2018, doi:10.1007/s12205-017-0535-8.

[170] V. Kumar, B. A. Kumar, L. Vanajakshi, and S. C. Subramanian, " Comparison of Model Based and Machine Learning Approaches for Bus Arrival Time Prediction," in *Proceedings of the 93rd Annual Meeting*, 2014. [Online]. Available: https://trid.trb.org/view/1288350 (accessed on: 20 July 2020).

[171] Y. Hou and P. Edara, "Network Scale Travel Time Prediction using Deep Learning," *Transportation Research Record*, vol. 2672, no. 45, pp. 115-123, 2018, doi: 10.1177/0361198118776139.

[172] B. Yu, H. Wang, W. Shan, and B. Yao, "Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 4, pp. 333-350, 2018, doi: https://doi.org/10.1111/mice.12315.

[173] Google. "Google Transit APIs." Google, Inc. https://developers.google.com/transit/ (accessed on: 20 July 2020).

[174] H. R. Schmidtke, "A Survey on Verification Strategies for Intelligent Transportation Systems," *Journal of Reliable Intelligent Environments*, vol. 4, no. 4, pp. 211-224, 2018.

[175] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, sep 2015: Association for Computational Linguistics, pp. 1412-1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166. (accessed on: 20 July 2020).

[176] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," presented at *the Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 1, Montreal, Canada, 2015, pp. 577-585.

[177] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction," in *IJCAI*, 2018, pp. 3428-3434.

[178] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "OpenTag: Open Attribute Value Extraction from Product Profiles," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*: Association for Computing Machinery, 2018, pp. 1049–1058.

[179] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," presented at *the 3rd International Conference on Learning Representations (ICLR)*, 2015. *arXiv preprint arXiv:1412.6980.*

[180] J. Zhang, Y. Zheng, and D. Qi, "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction," presented at *the Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, 2017, pp. 1655-1661.