Data-driven travel time and demand estimation of vehicles in smart cities

Lin, Xi

2020

Lin, X. (2020). Data-driven travel time and demand estimation of vehicles in smart cities.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/142890>

<https://doi.org/10.32657/10356/142890>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0
International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 16:42:52 SGT

Data-driven Travel Time and Demand Estimation of Vehicles in Smart Cities

Lin Xi

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

18 Jan. 2020

.....

Date

.....

Lin Xi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

18 Jan. 2020

.....

Date

.....

A/Prof. Sourav Saha Bhowmick

Authorship Attribution Statement

This thesis contains material from 2 papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as [Xi Lin, Xiaokui Xiao, and Zengxiang Li. A scalable approach to inferring travel time in singapore's metro network using smart card data. In 2018 IEEE International Smart Cities Conference \(ISC2\), pages 1–8. IEEE, 2018. 21](#)

The contributions of the co-authors are as follows:

- A/Prof. Xiaokui Xiao provided the initial project direction, as well as advice and guidance along the course of the project on the methodologies, paper writing, and experiments. He also reviewed and edited the manuscript drafts.
- I conducted the literature survey and the design of the solutions. I performed the testings to fine-tune and choose the right model for the problem from the beginning to the end. I also prepared the initial complete manuscript draft, and thereafter reviewed and edited it with A/Prof. Xiaokui Xiao.
- The experiments, including data processing required by the solutions, were conducted by me.
- Dr. Zengxiang Li liaised with the Land Transport Authority, collected the datasets necessary for this work, and processed it into a suitable format for our solution. He also reviewed the manuscript draft and provided suggestions, such as which experiments to add in for the work.
- All three co-authors contributed to the addressing of the concerns by reviewers from the conference.

Chapter 4 is published as [Xi Lin, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S Bhowmick. Path travel time estimation using attribute-related hybrid trajectories network. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pages 1973–1982. ACM, 2019. 45](#)

The contributions of the co-authors are as follows:

- I proposed the initial project problem, designed and built the solutions, as well as conducted the experiments. I also performed the literature survey and prepared the manuscript drafts.
- Dr. Yequan Wang provided various suggestions to improve this work. These are in the form of technical and writing/presentation-related advice. He also helped to review and edit the manuscript draft.
- A/Prof. Xiaokui Xiao provided supervision of the project and ensured the quality of the work, e.g., effectiveness of solution, is met. He also reviewed

and provided suggestions to improve the paper, be it before or after the conference review results were out.

- Dr. Zengxiang Li supervised, reviewed and provided suggestions on the project motivations, writings, experiments, as well as helped to address conference reviewers' concerns.
- A/Prof Sourav Saha Bhowmick supervised, reviewed and provided suggestions on improving the paper writing and experiments, both before and after the conference review results were out.

18 Jan. 2020

.....

Date

.....

Lin Xi

Acknowledgements

I wish to express my greatest gratitude to my supervisors A/Prof. Xiaokui Xiao, Dr. Zengxiang Li, and A/Prof. Sourav Saha Bhowmick for their guidance and all the support throughout my whole Ph.D. journey. It has been a very enriching time spent learning from each of them. Through the guidance of A/Prof. Xiao, I have learnt a lot about how to conduct good research, the importance of perseverance in research, and effective communication styles. And through the guidance of Dr. Li, I learnt to appreciate the motivations behind solving research problems related to real-world industry problems and the importance of patience in research. Having being supervised by A/Prof. Bhowmick two years ago, I have also learnt about effective writings and a great deal about spirit of research and education.

I would also like to thank my friends Dr. Yequan Wang and Dr. Andrew Ho, both of whom had provided huge timely help at certain points of this journey. Dr. Yequan Wang had introduced me to the field of deep learning and the plenty of opportunities there, and spared no effort in ensuring our collaboration work went well. Dr. Andrew Ho promoted opportunities that aimed to spur the pace of my Ph.D. process, and worked hard to ensure that our collaborations moved in a prompt fashion.

Very importantly, I would also like to thank my parents and my girlfriend Ling for their immensely patient support and understanding throughout this period, especially since I was away from home most of the time. Without that, my Ph.D. journey would have been much more challenging.

Finally, this journey would not be as interesting without the lab colleagues and friends I met along the way, whom had also helped me in one way or another. They are namely: Keke Huang, Renchi Yang, Tenindra Abeywickrama, Jun Zhao, Yancheng Shi, Chiam Zhong Lin, Grace Lee, Sibow Wang, Wenya Wang, Xin Zheng, Youze Tang, Jun Zhang, Rao Divya, and Ong Jenn Bing.

Contents

Acknowledgements	ix
List of Figures	xv
List of Tables	xvii
Abstract	xix
1 Introduction	1
1.1 Opportunities and Challenges of Smart Cities	1
1.2 Urban Mobility in Smart Cities	2
1.3 Travel Time and Demand Estimation	3
1.4 Dissertation Overview and Contributions	5
1.4.1 Datasets considered in this Dissertation	6
1.4.2 Estimation of Travel Time within a Metro Network	6
1.4.3 Estimation of Path Travel Time within a Road Network	7
1.4.4 City-wide Regional Demand Prediction	8
1.5 Road Map of Dissertation	9
2 Literature Review	11
2.1 Network Component Time Estimation	11
2.2 Path Travel Time Estimation	14
2.3 Regional Demand Prediction	16
2.4 Other Notable Urban Mobility Problems	18
3 Scalable Estimation of Component Travel Times	21
3.1 Preliminaries	23
3.1.1 Background	23
3.1.2 Modelling Metro Networks	24
3.1.3 Problem Definition	25
3.2 Methodology	25
3.2.1 Challenges and Solution Overview	25
3.2.2 Phase 1: Estimations for Non-Interchanges	28
3.2.2.1 Addressing Issue 1	29

3.2.2.2	Addressing Issue 2	29
3.2.2.3	Estimations of Time Variables	31
3.2.3	Phases 2 and 3: Estimations for Interchanges	33
3.3	Experiments	34
3.3.1	Datasets	34
3.3.1.1	Fare Card Data Subsets	34
3.3.1.2	Ground Truth Link Time Dataset based on Recorded Train Arrival Times	35
3.3.2	Baselines for Comparisons	36
3.3.3	Experiment Settings and Results of Method Comparisons	37
3.3.4	Validation with Ground Truth Link Times	39
3.3.5	Computational Efficiency and Scalability	40
3.4	Discussion	40
3.4.1	Generalizability of Solution	41
3.4.2	In-station Waiting Time Analysis	42
3.4.3	Limitations	43
4	Path Travel Time Estimation with Attribute-related Hybrid Trajectories Net	45
4.1	Preliminaries	47
4.2	Attribute-related Hybrid Trajectories Network	48
4.2.1	Intuition of AtHy-TNet	48
4.2.2	Overview of AtHy-TNet	50
4.2.3	Attribute Correlation Module	51
4.2.4	Sub-path Representation Module	52
4.2.5	Sub-path Correlation Module	55
4.2.6	Joint Prediction Module	56
4.3	Experiments	57
4.3.1	Experimental Set-up	57
4.3.2	Evaluation Results	61
4.3.3	Transfer Applications and Limitations	67
5	Demand Prediction with Attribute-guided Multi-graph Convolutional Network	69
5.1	Preliminaries	72
5.2	Attribute-guided Multi-graph Convolutional Neural Network	72
5.2.1	Overview of AM-Net	73
5.2.2	Multi-contextual-gated RNN	74
5.2.3	Mutual-correlated Multi-GCN	77
5.2.4	Attribute Guidance	78
5.2.5	Prediction Module	80
5.3	Experiments	81
5.3.1	Experimental Set-up	81
5.3.2	Evaluation Results	86

5.3.3	Relevance of High-level Attributes	87
5.3.4	Performance across Different Hyper-parameters	88
5.3.5	Performance across Different Time Periods	88
5.3.6	Training and Testing Time	90
5.3.7	Application to other Problems and Limitations	91
6	Conclusion	93
A	Publications	103
A.1	Publication for Chapter 3	103
A.2	Publication for Chapter 4	112
	List of Author's Awards, Patents, and Publications	123

List of Figures

3.1	A small metro network	25
3.2	A train line in a metro network	27
3.3	Solution overview flowchart	28
3.4	A lognormal fit of travel times between two stations	31
3.5	A single lognormal distribution of travel times between two stations	32
3.6	Average hourly departure and arrival rate of a station	36
3.7	Combined running time of the three phases	41
4.1	A small map with trajectory overlays from two vehicle types (best viewed in colour).	49
4.2	Architecture of AyHy-TNet Model	50
4.3	Attribute Correlation Module. a_c , a_d , a_r and a_t represent embedded vectors for vehicle ID, day-of-week, rainfall, and time-of-day respectively. d_c (resp. d_f) represent the fine (resp. coarse) grained embedded distance vectors.	52
4.4	Performance over different K values	64
4.5	Performance over different bus data size	64
4.6	Performance of AtHy-TNet over different ζ	64
4.7	Performance over different travel times	65
4.8	Performance over different days of the week	66
4.9	Performance over different times of the day (weekday)	67
4.10	Performance over different times of the day (weekend)	67
5.1	Architecture of AM-Net Model. G^n denotes the graph that represents the neighbourhood relationship across regions of the city, while G^r and G^s denote the graphs that represent the road connectivity and functional similarity relationships respectively.	73
5.2	Multi-contextual-gated RNN	75
5.3	Taxi and Bike Demand across 4-hours Periods of the Day (Weekdays). 1 denotes the 0-4 hours of the day, while 6 denotes the 20-24 hours of the day.	82
5.4	Taxi and Bike Demand across 4-hours Periods of the Day (Weekends). 1 denotes the 0-4 hours of the day, while 6 denotes the 20-24 hours of the day.	82
5.5	Average daily Taxi and Bike Demand across Different Days of the Week. 1 denotes Monday, while 7 denotes Sunday.	83

5.6	RMSE w.r.t. Size of Attention	89
5.7	RMES w.r.t. Size of GCN	89
5.8	RMSE across 4-hours Periods of the Day (Weekdays).	90
5.9	RMSE across 4-hours Periods of the Day (Weekends).	90
5.10	RMSE across different days of the week	90

List of Tables

3.1	Comparison of RMSE, 10%ile and 90%ile errors on morning-peak . .	39
3.2	Comparison of RMSE, 10%ile and 90%ile errors on evening-peak . .	39
3.3	Comparison of RMSE, 10%ile and 90%ile errors on off-peak	39
3.4	RMSE, 10%ile and 90%ile errors for link time estimation	40
3.5	Mean, 10%ile and 90%ile estimated tap-in times	43
4.1	Trajectory data characteristics	59
4.2	MAPE, MAE, and RMSE on AtHy-TNet and the baselines. Best results are in bold face and second best underlined.	62
5.1	Demand Dataset Characteristics	82
5.2	RMSE and MAE on AM-Net and the baselines. Best results are in bold face.	86

Abstract

In recent times, governments around the world have increasingly focused on transforming their cities into smart cities. A large number of these smart city projects requires the collection and analysis of datasets on a large scale. These are conducted by the various entities of the society, e.g., government organizations or businesses. To date, numerous data-driven computational techniques have been proposed to perform such analysis, and to draw previously untapped insights from the datasets. This dissertation focuses on such techniques in the domain of urban mobility of smart cities. Specifically, it introduces three problems related to the estimation of travel time and demand of vehicles, which play critical roles in urban mobility. From the perspectives of transport operators, e.g., taxi or ride-hailing companies, such information encourages a more effective allocation of existing resources or planning of routes, which in turn improves commuter experience. For public organizations, such information helps in the understanding of the traffic conditions within the transportation network, which in turn supports urban planning. Finally, such information also helps commuters to understand the travel time of routes they take and plan their trips more effectively.

In the first problem, we recognize that existing work for the estimation of the travel time of trip components within the metro network either requires datasets that are usually hard to obtain, or considers the problem on scales too small (e.g., consider only single-line scenario). In some cases, there are also limitations in terms of computational efficiency. Due to the inaccessibility to fine-grained datasets like cell-phone datasets, the smart card data is a natural choice for such estimations. However, what makes such estimations challenging is that smart card data only provides information about the origins and destinations, but not the actual routes taken between them. Since there may often be multiple route choices between an origin and destination, it is not straightforward to know what exact routes were taken, and thus challenging to estimate the trip component times. To tackle these limitations, we propose a novel multi-stage approach that accounts for the issue of

multiple route choices within the network, while leveraging solely the smartcard dataset to infer these component times. The key idea is that we first seek to identify the exact routes taken by passengers and the travel time associated with it for selected trips, using insights to the metro network and approaches like trip time distribution fittings. Then, we try to estimate the time variables of trip components by considering linear formulations of the components of these trips, as well as other issues like overfitting and robustness of estimation.

In the second problem, we note that existing methods for path travel time estimation generally require single-source large datasets for a satisfactory accuracy, and may not be suitable for vehicles with small datasets. Furthermore, considering only single-source datasets neglects the estimation-improving insights available in other datasets. A challenge is how to effectively utilize such other trajectory sources. This challenge is also further compounded by that the path travel time is generally dependent on various temporal and spatial factors. As such, we propose Attribute-related Hybrid Trajectories Network (AtHy-TNet), the first neural model that effectively considers the relative importance of different attributes under different scenarios, as well as the spatial and temporal relationships across hybrid trajectory data, to estimate the path travel times. Not only does this model consider the various dependencies affecting travel time, it also models the multi-nature correlations across trajectories that may come of different vehicle types.

For the third problem, we recognize that existing work for regional vehicle demand prediction largely depends on the correlations of historical demands, which may not be accurate in some situations. As such, guiding the estimation via high-level attributes like period of day may help to resolve such inaccuracies. However, the state of the art largely ignores these high-level attributes and their relative importance under different circumstances. Furthermore, while the state of the art considers pair-wise relationships across different regions, it does not consider the mutual correlations across these relationships. Similar to path travel time estimation, what makes this problem challenging is that there exists many multi-nature dependencies for demand of vehicles. As such, we propose Attribute-guided Multi-graph Convolutional Network (AM-Net), a novel neural model that models the spatiotemporal dependencies of regional demands via three different pair-wise relationships and their mutual correlations, and guides the estimation using high-level temporal and spatial attributes.

For each of the three problems, we evaluate the proposed solution with real-world datasets, and demonstrate its superior performance over the corresponding state of the art. We also conduct in-depth analysis on how the method performs under different conditions, e.g., period of day and day of week.

Chapter 1

Introduction

A smart city is one that aims to leverage information and communication technologies to optimize the efficiency of its operations, processes and functions, while driving economic growth and improving the quality of the life of its residents. It also integrates the infrastructure of the city and modern technologies to improve the residents' interactions with the urban environment.

Over the recent years, the governments around the world have increasingly focused on and kicked off programmes that transform their cities into smart cities. It is predicted that the market size of smart cities will be worth USD 237.6 billion by 2025, while expanding at a compound annual growth rate of 18.9% from 2019 to 2025 [1]. These smart city initiatives are further motivated by the ever growing influx of people into the cities. According to the United Nations, the percentage of people living in cities throughout the world will increase from around 55% to 68% from 2018 to 2050 [2]. This rising trend will impact the infrastructure and calls for further effort by the governments to cope with this influx via smart city agendas.

1.1 Opportunities and Challenges of Smart Cities

The advent of smart cities also brings about a plethora of opportunities and challenges. The opportunities that smart cities and their projects bring about are aplenty and diverse. They include a more efficient urban mobility, enhanced homes and neighbourhoods, more efficient energy usage optimization and conservation,

and a stronger administrative service. In the domain of urban mobility, an application is the smart traffic management system that monitors and analyzes the traffic conditions within the city's road network to identify bottlenecks for future infrastructure planning purposes. Another example is ride-hailing companies like Uber which provide transportation services to passengers based on their demands in real time. For homes and neighbourhoods, one example is the smart street lighting system, which controls lights in real time to conserve energy and provide greater security to the neighbourhood, e.g., light up only when human traffic flow is detected. Some other smart city projects include smart payment, smart grids, and smart utility metering for electricity and water. To achieve these goals, several technologies such as internet-of-things, cloud/edge computing, communication, software and artificial intelligence solutions, are often considered.

However, smart cities often come about with their fair share of challenges, which can be classified into the following categories: digital security, regulations and policies, business funding and support, the suitability of adaptation by existing infrastructures, as well as the technological challenges [3]. Technologically speaking, a large part of these smart city projects requires the collection and analysis of datasets on a large scale, conducted by the various entities of the society, e.g., government organizations, businesses and individuals. Indeed, together with the advancement in data collection and data governance, datasets of various genres have increasingly been available and accessible, and these datasets often contain hidden insights that are valuable to these entities. However, the ever growing volume of data and sensors within the city also makes it increasingly more challenging to manage and analyze these datasets. As such, along with the development of computational resources, e.g., graphical processing units widely used for deep learning algorithms, numerous novel data-driven computational techniques have been proposed to draw such previously-untapped information.

1.2 Urban Mobility in Smart Cities

This dissertation focuses on one particular area where such techniques have found their place - urban mobility in smart cities. Urban mobility is a core component within any cities and it plays an indispensable role in ensuring the smooth operations of the cities of the world. Furthermore, urban mobility applications like

intelligent transport systems which are the products of cities' push towards smart cities either have huge potentials, or have demonstrated immense benefits. For example, a case study conducted by the McKinsey Global Institute demonstrates that the cost of the solution that leverages transportation technologies for the M42 motorway of the United Kingdom is almost one-sixth that of the traditional approach of widening roads, with both achieving the same outcome. In specific numbers, this translates to a cost-savings of \$750 million [4]. Indeed, spurred by the increased focus of governments towards leveraging data-driven computational technologies to improve the urban mobility operations within cities, much research effort has also been poured into this field. The range of constituting topics and applications is broad and existing work has considered various types of urban mobility datasets, including trajectory datasets obtained from vehicles like taxis or other ride-hailing vehicles, as well as passenger trip records of public buses, taxis or ambulances.

1.3 Travel Time and Demand Estimation

In spite of the diverse range of topics related to urban mobility, we specifically focus on the problems of travel time and demand estimation of vehicles within smart cities. This is because these are problems very important for urban mobility, and the knowledge of travel time and demand of vehicles in a city is valuable to many. The problem of travel time estimation deals mainly with the estimation of how long it takes for a vehicle or commuter to travel a path or sub-paths of a trip in the transportation network. Existing work to date has considered the travel time for public transportation like trains [5–9], as well as taxis and other on-demand vehicles [10–15]. These travel time estimation problems can be classified into two groups:

- Estimating the travel time of trip components based on the total trip travel time. For example, given the total time taken by buses to complete their trips, estimate the travel time at road-segments or sub-paths used in these trips.
- Estimating the total path or trip travel time based on the time variables of the constituting path components. For example, given the historical travel

time of the sub-paths within a path, estimate the total time taken by a taxi or ambulance to travel that path.

It is very important to solve these travel time estimation problems effectively because having an accurate knowledge of the travel time is very valuable to many stakeholders within the transportation system. From the perspectives of transport operators, e.g., train or taxi companies, it encourages a more effective allocation of existing resources or planning of routes, which then lead to stronger customer service. For example, a ride-hailing operator may consider the travel times of various candidate paths before assigning the most suitable path to its driver partner to fulfill a ride-hailing order. Letting the commuters know how long their ride may take or how long it takes for the ride-hailing car to arrive may also improve their satisfaction of the ride-hailing service. Likewise, courier delivery operators require such travel time information to plan the most effective routes for their delivery vehicles for better cost-savings. For public organizations, such information also helps in the understanding of the traffic conditions within the transportation network, which in turn supports urban planning. For example, having the knowledge that certain roads of a city tend to see a slow traffic movement during the peak hours of the day due to congestions may prompt government organizations to propose measures that mitigate such congestions. Hospitals also require accurate travel time for ambulances, especially when they are dealing with patients in critical cases like severe road accidents. Finally, such information also helps commuters to understand the travel time of routes they take and plan their trips more effectively.

Apart from the travel time, information about the demand for transportation like buses, trains, taxis or other ride-hailing vehicles at different regions of a city is also something very important for transportation planning and vehicular resource allocations. Such information helps operators of public transportation like buses to more effectively plan the daily journey schedules to meet the demands of the residents within the city. For example, if the demand for trains at certain parts of the morning peak hours is too high, the metro system may not be able to comfortably handle it, resulting in train breakdowns or severe congestions. Therefore, having a knowledge of this may prompt train operators to increase the frequency of trains during that period, or propose other measures like discounts for people who choose to take the trains at a different time. Such information also helps ride-hailing companies like Uber, Didi, and Grab in effective vehicle allocations, while

simultaneously improving commuters' experience through the reduction of waiting time for a car. For example, by understanding in real time that the demand at a region of the city is going to experience a spike over a consecutive few time periods as a large scale event will be ending there soon, ride-hailing operators can direct their driver partners to that region in advance. This in turn avoids the scenario where people have to wait for up to an hour or two just to get a ride back home. For public organizations, it also provides insights for a better understanding of the traffic complexities within a city during different times of the day and days of the week. As such, it is very valuable to forecast the demand at different regions of the city across the day.

1.4 Dissertation Overview and Contributions

The purpose of this dissertation is to present several data-driven computational methods that effectively estimate the travel time of vehicles and the demand for them within the transportation system. Specifically, it introduces three problems about travel time and demand estimation and highlights the respective novel data-driven computational methods we propose to address each of these problems. Collectively, these methods either generate insights that are valuable for the stakeholders of urban mobility, or provide greater accuracy and computational efficiency in generating these insights. In the first problem, which is about inferring travel time of trip components in a metro network, we propose a novel scalable approach to infer these values. For the second problem, we propose a neural architecture that, for the first time, leverages hybrid trajectory datasets for estimating path travel times. For the third problem, we propose a novel neural architecture that leverages high-level attributes and correlations across multiple pair-wise relationships between regions in a city to guide the estimation/prediction of the future demands at these regions. For each of the three problems, we evaluate the proposed solution with real-world datasets, and demonstrate its superior performance over the corresponding state of the art. We also conduct in-depth analysis on how the method performs under different conditions, e.g., period of day and day of week.

1.4.1 Datasets considered in this Dissertation

To address these three problems, we consider mainly two types of datasets. The first corresponds to datasets which contain information about the origin and destination of trips. One example is smart card datasets obtained from automated fare collection systems of metro networks. Another example is trip datasets of taxis and bicycles within a city. Typically, such datasets contain information of both the origin and destination of every trip in the dataset. Such information may include the date and time at which the trip starts or ends, as well as geospatial coordinates or station/stop identification numbers of the origin and destination. The second type of datasets correspond to the trajectory ones, which contain not only the information about the origin or destination, but also that of the intermediate points of every trip. Typically, the information associated with each point is such as the timestamp at which it is recorded, as well as the geospatial coordinates of that point. In this dissertation, we consider trajectory datasets of taxis and buses.

1.4.2 Estimation of Travel Time within a Metro Network

As mentioned, travel time estimation problems can be classified into two categories. Chapter 3 of the dissertation focuses on the first category of problems, and specifically on the problem of estimating travel time of trip components within a metro network. The rationale behind such a domain choice is that metro networks remain a key area within public transportation where this problem is still very challenging and existing work is still limited in terms of accuracy or computational efficiency. This is especially so since the smart card dataset obtainable from the metro systems generally contains only the geospatial and temporal information of the origin and destination of each trip, and not that of any intermediate locations during the trip.

In particular, a trip in a metro network generally comprises of various time components, such as time taken by the train to travel across individual links, time from passengers' tapping in at the gantry to boarding the train, etc. Knowledge of these times is indispensable for generating meaningful insights. First, such information allows us to estimate passenger route choices at different periods of the day, which can then be used for the analysis of risks in the case of train service breakdowns.

Second, understanding these time components allows us to estimate the crowd flow at any part of the network and at any time of the day. Third, it allows us to infer travel times between any station pair, which helps in identifying travel inefficiencies in the network, to assist in future train line designs.

However, existing work either requires real time train timetables [5, 16, 17] that may not always be readily available especially if the city’s metro system is run by several train operators, or mobile phone datasets [18] that are also hard to gain access to. Furthermore, existing work that seeks to perform such estimation using smart card datasets is either limited in scale, e.g., works only for a single train line [19], or is too computationally costly for timely insights generation, e.g., taking days to complete running [6–9].

Motivated by these, we propose a solution that addresses the above issues. This solution leverages a three-phase approach and is not only able to infer the travel time within the metro network using solely the smart card datasets, but is also, for the first time, scalable to massive datasets of millions of trips with comfortable computational efficiency. We also show our experiments conducted on a real-world dataset and demonstrate its superior performance over the state of the art.

1.4.3 Estimation of Path Travel Time within a Road Network

While Chapter 3 focuses on the first category of travel time estimation problems, Chapter 4 focuses on the second - estimating the total travel time of a trip or a path. Specifically, we consider the estimation of path travel time of vehicles within road networks. Although existing work has demonstrated satisfactory accuracy for such estimation, estimating path travel time has generally been associated with various forms of challenges. Specifically, it has been challenging due to multi-nature considerations. First, travel time is affected by spatial factors, such as the road characteristics, e.g., roads in business districts vs. residential ones. Second, it is also temporally dependent. For example, travel times in the downtown area may be longer during peak hours due to traffic congestions. Other factors that affect such estimations are such as the vehicle type, e.g., sedan cars vs. buses, and weather conditions.

However, existing work generally requires single-source massive trajectory datasets, i.e., hundreds of thousands to millions of trajectories, to achieve satisfactory accuracies [11, 12, 20, 21]. The implication is that these methods are unsuitable for vehicles which do not have such massive datasets, e.g., logistics trucks, ambulances or buses. However, the travel time information of certain such vehicles is also highly necessary. For example, it may be mission-critical to understand what is the fastest path an ambulance could take to get to the hospital on a Tuesday night. While we may consider pooling trajectories of different vehicle types together into a large dataset, estimating across hybrid trajectories, e.g., a mixture of a bus dataset that consists of tens thousand trajectories and a taxi dataset which has millions of trajectories, is challenging. The reason is that the characteristics of trajectories from different vehicle types are inherently different. For example, buses generally move more slowly than taxis and have to stop at intermediate stops. As a result, attributes such as vehicle type tend to be highly critical for such estimations. However, existing work generally does not consider the relative importance across these attributes under different circumstances. Furthermore, there has yet to exist any work that leverages hybrid trajectory dataset, for travel time estimation.

Due to these motivations, we propose AtHy-TNet, the first end-to-end neural architecture for travel time estimation using hybrid trajectory data, to tackle the issues above and to solve the novel problem of path travel time across hybrid trajectories. To this end, AtHy-TNet not only considers the multi-nature characteristics of trajectories via spatiotemporal and attributional modules, but also the correlations between trajectories of different vehicle types. We also demonstrate with experiments the benefits of AtHy-TNet in leveraging hybrid trajectories, and its stronger performance over existing work.

1.4.4 City-wide Regional Demand Prediction

In Chapter 5, we focus on the problem of predicting the future demand for taxis at different regions of a city. The reason for this choice is that estimating taxi demands has generally been very challenging due to the various dependencies affecting the demand at a region. First, taxi demand is affected by spatial-related characteristics, such as the region at which the demand is estimated. Second, demand at a region is also highly dependent on temporal periodicities, e.g., day of week or period of

day. Finally, such demands are also highly dependent on sporadic events, e.g., pop concerts. As such, various computational methods have been proposed to date to comprehensively capture the dependencies for a more accurate prediction of regional demands.

Existing work for demand predictions tend to depend heavily on historical demands [22–25] and spatial relationships across regions [25]. However, the heavy dependence on historical demands comes with its limitations and may not always be accuracy-improving. For example, 10 am of a weekday usually corresponds to the off-peak hours in some cities. If we were to predict the demand from 10 to 10:30 am using solely the demands over the past time periods, the results may be largely inaccurate since those periods belong to the morning peak hours. To circumvent this issue, we may consider integrating high-level temporal or spatial attributes like period-of-day or region ID to guide the estimation. However, one other important factor largely ignored in existing work is that it does not consider the relative importance across these attributes under different circumstances. Moreover, while existing work that leverages multiple pair-wise relationships may uncover previously hidden insights that improve estimations [25, 26], it does not consider the correlations across these diverse relationships.

Motivated by these, we propose AM-Net, a neural architecture that leverages multi-nature dependencies to predict the next-period demands at every region of the city. This model also leverages insights from high-level attributes of data samples to guide the estimation. Furthermore, it not only learns the various pair-wise relationships between regions and the temporal correlations of historical demands, but also models the mutual correlations across these pair-wise relationships.

1.5 Road Map of Dissertation

For the remaining parts of the dissertation, Chapter 2 will focus on the literature review of the existing work for travel time and demand estimation within smart cities. Chapter 3 focuses on the approach we propose for the estimation of travel time components within the metro network. Chapter 4 highlights AtHy-TNet, an approach we propose to estimate the path travel times across hybrid trajectory datasets. Chapter 5 sheds light on AM-Net, the method we propose to estimate the

demands of regions across the city. Finally, we provide our conclusion in Chapter 6 of the dissertation. In each of Chapter 3, 4, and 5, we highlight the detailed motivation behind each proposed solution, the methodologies, and the experiment results that demonstrate the effectiveness of the solutions proposed.

Chapter 2

Literature Review

In this chapter, we review the existing work for the estimation of travel time and demand for vehicles in smart cities. As mentioned, methods for travel time estimation can be classified into two groups. The first group focuses on the estimation of travel time of components of a trip, while the second group focuses on the estimation of the total travel time of the trip based on the travel time of the trip components. To date, the motivations for the methods of the first group have been the strongest in the domain of metro networks, and smart card data has been widely used to deal with such problems.

2.1 Network Component Time Estimation

As highlighted earlier, there is existing work that uses smart card data to provide insights to the time characteristics of the metro network. First, there is work that makes use of a combination of smart card data and train operation logs [5, 16, 17] to do this job. For example, Zhao et al. [5] propose a probabilistic model to estimate the passenger flows distributions on a metro network, down to the granularity of trains of each line. While methods that make use of train operation logs may shed light on the timings of the trip spent inside the trains, they do not reveal much information about the time spent outside the trains, such as the time from tapping in to boarding the trains, and the time from alighting at a station to tapping out. Moreover, train operation logs may not be always available or easily obtained. Another option is to make use of cell-phone data [18]. While this approach allows

a finer-grained determination of the route choices of passengers, it poses problems of considerable privacy concerns and such cell-phone data may not always be easily accessible.

Existing methods that make use solely of fare card data have several limitations. Sun et al. [19] make use of a regression model to estimate a few travel time component variables (train speed, average dwell time, average platform-gantry walking time). Nevertheless, only a single-line scenario was considered. In order to scale it to a larger network, more factors like trip transfers have to be considered in the estimations. Moreover, it assumes constant speed in the trains along the network, which is not always true in reality. To achieve analysis on the scale of a whole metro network, some methods employ Bayesian inference using computational approaches like the Markov Chain Monte Carlo (MCMC) methods to estimate network timing attributes and characterize the commuters route choice model [8, 9, 27]. To this end, these methods make use of the following trip information from the smart card data: trip start and end date/time, as well as origin and destination stations. In the proposed framework [8], the authors conduct the commuter route choice modeling simultaneously with the estimation of the following network time attributes: link timings, the time from tapping-in at station gantry to boarding a train, and time from alighting a train to tapping-out. In [27], Rahbar et al. extend this framework to the multi-modal setting, by considering additional smart-card-based transportation modes like bus and ferry. They calibrate and validate it using smart card data from Brisbane, Australia, and compared the inference results with the scheduled times from the General Transit Feed Specification.

In these cases, travel time variability caused by possible interruption during metro operation has been considered in this framework by assuming normal distributions for component times. However, it may not always be valid to have such assumptions since these component times may not certainly be normally-distributed. This is especially so since such timings are unlikely to be too short (close to 0) or too long (close to infinity) in reality. Zhang et.al [9] attempt to deal with this issue by considering normal distributions truncated at both ends. They also more accurately describe these timings by considering only origin-destination (OD) pairs with only one effective route instead of all OD pairs. Another drawback is that such methods tend to be computationally costly and may take up to days to complete all iterations

of the MCMC method [8, 9]. As a result, some data (trip timings and origin-destination pairs) which would otherwise contribute to the inference were omitted. This issue also poses a problem to the scalability of such methods to even larger networks. In general, there is a lack of work that conducts network component timings inference on the scale of large metro networks, yet at the same time not costly in computation.

From the broader perspective, there are some problems that are very closely related to network component time estimation. First, commuter flow assignment in a transportation network using smart card data has also been a very popular topic. Traditionally, this task was achieved by interviewing people at various locations and times of the day. However, such methods generally require too much manpower and effort for viable results. A wider collection of smart card data and improvement in data-driven analytical techniques suggest possible alternatives to address this limitation. However, estimating route choice on smart card datasets has the same challenges as network time component inference due to the same reason - limited card information that only reveals the origin and destination information but not the intermediate points. Apart from [8, 9, 27] that attempt to simultaneously find the route choices and these network component times, Fu et.al [28] formulate normal/lognormal mixture models for estimating posterior probabilities that represent the likelihood of a particular route taken by a commuter in a metro network. In the proposed models, every observed peak in the mixture plot corresponds to an actual commuter route choice. In a similar manner, Monterola et al. [29] claim that the travel time between each station pair follows an extreme value Gumbel distribution, and make use of Gumbel mixture model to represent the actual commuter route choices. It is observed that much of existing work requires input of additional data like train timetables. Apart from the latest work of this genre [5], there is also other existing work which makes use of scheduled timetables in conjunction with smart card data to model passenger flow assignments in metro networks [16, 17, 30].

In conclusion, existing work that deals with estimation of travel time of trip components largely tends to require datasets like realized train time tables or mobile data which are not always easily available. Work that does this estimation using smart card datasets either considers scenarios on scales too small, e.g., single-line

scenario, or is too computationally costly, i.e., taking up to days to complete running.

2.2 Path Travel Time Estimation

For the second group of travel time estimation methods, existing work largely leverages trajectory datasets and is generally classified into two categories: 1) Link-based, and 2) Sub-path-based methods. For the link-based methods, travel time estimation is carried out for the individual road segments [10–13, 31, 32], before these times are combined to form the travel time of the whole path. Yang et al. utilize spatiotemporal Markov models to model traffic behaviours and forecast the travel time at all the road segments [32]. In [10], Wang et al. make use of regression tree and probabilistic graphical methods based on some correlations observed from the traffic data. In [13], the authors propose a deep learning framework that predicts traffic speed of road segments using spatiotemporal information of contiguous road segments. Rahmania et al. propose a fixed point formulation that simultaneously infers route choice and travel time [12]. Jenelius et al. [11] use a multivariate probabilistic principal component analysis model that is able to model correlation patterns between days. The methods that belong to this category are mostly focused on estimating the travel time of single road segments. As such, accuracy is compromised when these methods are used for the estimation of path travel times. The reason why accuracy is compromised using such methods is that these methods do not consider the interactions between road segments, e.g., certain travel complexities on the sub-path level, such as typical time spent waiting at the traffic lights, as well as turn costs. Furthermore, they depend largely on the high quality of the travel speed dataset of each segment, which may not be always available.

As such, methods that make use of historical sub-paths to estimate travel times have been devised to resolve such issues more effectively [14, 15, 33, 34]. The key idea is that path travel times can be estimated by considering historical path/sub-path travel times, which implicitly consider costs that are not modeled in the link-based methods. In [33], Yuan et al. propose a time-dependent landmark graph to model the time-varying traffic patterns, and techniques to compute the fastest path based on this graph and the experience of drivers. In [34], the authors

utilize a tensor decomposition approach to resolve the issue of data sparsity, and a dynamic programming method to find the optimal concatenation of trajectories for path travel time estimation. Dai et al. propose the hybrid graph paradigm and techniques to address the issue of non-standard travel time distribution, data sparseness and the dependencies among roads travelled during a path [14]. Li et al. further consider scenarios where only a small number of probe vehicles is used to collect the data and travel times of the same path may have large variances [15]. More recently, deep learning has shown further success for this task. Wang et al. propose DeepTTE, which models the path travel time of taxis in an end-to-end fashion, while considering the spatiotemporal and attribute characteristics [21]. This is achieved through the usage of LSTM and convolutional layers, as well as multi-task learning. Zhang et al. partition the road network into grids, and consider various short and long-term features from the historical trajectories for the prediction [20]. Unlike [21], this method leverages the aggregation of historical trajectories to derive information like the average historical travel speed at a grid, before using such information to further infer the path travel time.

On the broader scale, there also exists work that estimates travel time given the input of an OD pair [35, 36]. Li et al. introduce a representation learning method based on multi-task learning that considers the road network structure and spatiotemporal information [35]. Hu et al. propose a method based on matrix factorization and graph convolutional recurrent neural network to forecast the travel time between an OD pair, using historical OD matrices [36]. The difference between such methods and those from the previous two categories is that it estimates the total trip travel time while avoiding the complexities brought about by different route choices.

In general, the existing work largely leverages datasets of a substantial size, i.e., hundreds of thousands to millions of trajectories, for a fairly accurate accuracy. This makes it unsuitable for the estimation for vehicles like public/private buses which only have a small trajectory dataset. Furthermore, while we may consider the mixture of trajectories to form a large dataset, the characteristics of the trajectories of different vehicles are inherently different. There has been no work to date that focuses on modeling the characteristics of hybrid datasets for such estimations.

2.3 Regional Demand Prediction

Traditional techniques for demand prediction or predictions related to other traffic time series mainly revolve around ARIMA and its variations [22, 23], based on the idea that incorporating autoregressive and historical moving average components may improve estimation of future demands. In order to further utilize external information like weather condition or type of venues, some studies focus on incorporating such information to fine-tune estimations [31, 37, 38]. There also exists work that provides preliminary consideration of some spatial factors that may affect traffic. In [39], the correlations among regions of a road network that are connected by roads are modeled using matrix factorizations. By assuming that traffic at a region should be similar to that of nearby regions, there also exists work that focuses on smoothing the prediction at nearby regions through the usage of regularization for close space and temporal dependencies. In the latest work of this genre, Tong et al. leverage a spatiotemporal regularization to perform this smoothing, and propose a unified linear regression model with features that have more than 200 million dimensions [40].

However, the above-mentioned methods are mainly based on historical demand information and do not consider the complex nonlinear space and temporal relationships. To capitalize on the superior modeling capabilities of deep learning, additional focus has been dedicated in the recent years to modeling the multi-nature complex dependencies for demand prediction using these techniques [24, 24, 25, 41–43]. Specifically, Ke et al. leverage the stacking and fusion of multiple convolutional long short-term memory layers, standard LSTM layers, and convolutional layers to capture the spatiotemporal characteristics and correlations of the dependencies affecting the demand [42]. In [24], the authors also propose a deep learning method which encodes the region-wise relationship as a graph using graph embedding methods, and leverages CNNs, LSTMs and to model the multi-nature dependencies for taxi demand prediction.

Indeed, recent methods for demand prediction have focused heavily on using deep learning techniques to model the spatiotemporal characteristics and correlations between regions or points in a city. In fact, the state of the art has largely leveraged graph convolutional networks due to their strong capabilities for learning the correlations across regions of the city. In these methods, the spatial structure of

the demand dataset is mostly formulated as a matrix, where each entry represents a rectangular region in the city. As such, these regions and the pair-wise relationships between them naturally formulate an Euclidean structure. However, non-Euclidean structured data is also prevalent in urban mobility [26, 44, 45]. For example, any two regions within a city may also be related to each other via functional similarity or road connectivity. As such, Geng et al. [25] propose a solution that not only models Euclidean, but also non-Euclidean pair-wise relationships that exist between these regions. To this end, they leverage multiple GCNs to model the various different relationships, as well as contextual-gated RNNs to model the temporal correlations affecting the demand for ride-hailing vehicles. Apart from work that focuses on prediction at the next immediate time period, Bai et al. [41] also propose a method that leverages graph convolutional networks for multi-step predictions. This method consists of a long-term encoder to model the historical demands, as well as a short-term encoder for generating the multiple next-step predictions. For the similar problem of multi-step prediction [43], Zhou et al. use an encoder-decoder framework based on LSTM and convolutional units to identify features that capture the spatiotemporal influences and pickup-dropoff interactions affecting passenger demands for ride-hailing vehicles.

On the broader scale, another topic that is very closely related to demand prediction is traffic flow prediction, due to the similarity in the nature of both problems, i.e., time series prediction. Indeed, it is an area where deep learning has also found its place [26, 45, 46]. In [45], Li et al. model the traffic flow as a diffusion process on a directed graph and capture the spatial dependencies with bidirectional random walks on this graph, as well as the temporal dependencies using an encoder-decoder architecture with scheduled sampling. In [46], Liu et al. propose a model that leverages LSTM units to model traffic under both normal conditions and extreme ones, e.g., accidents. Yu et al. also propose a spatiotemporal graph convolutional network to model the spatial and temporal dependencies [26].

In general, the state of the art for demand prediction leverages heavily on the historical demand information. However, over-dependence on such historical demand may not always be accurate, e.g., when trying to predict the demand at the time period right after peak hours. As such, integrating high-level attributes like period of day and day of week may be useful to resolve such issues. However, existing

work generally does not consider the relative importance across these high-level attributes. Furthermore, while pair-wise relationships of different natures have been considered, there has yet to exist a work that models the correlations across these relationships.

2.4 Other Notable Urban Mobility Problems

In general, apart from the estimations of travel time and demand within smart cities, there are several notable urban mobility problems that have been tackled using data-driven computational methods, made possible through the increasing availability of urban mobility datasets like smart card and trajectory datasets. For existing work that is concerned with the smart card dataset, other notable problems include OD matrices estimations and travel behaviour analysis. Specifically, an OD matrix is a matrix where each element contains a number that represents the number of trips made between a particular OD pair within the road or metro network. The main motivation behind work that attempts to estimate such matrices comes from the fact that in various parts of the world, the AFC system only records information related to tap-ins, i.e., only the trip start time and station/stop information is available. As such, if the arrival information of every trip is unavailable, the downside is that it is not straightforward to infer the destination of every trip record, as well as trips counts of each OD pair. In order to provide a solution that avoids such issues, existing work seeks to infer arrival information based on available smart card datasets [47–52]. For analysis of travel behaviours, existing work includes [53], where the authors propose a data mining approach to identify travel patterns for individual commuters, based on the spatial and temporal characteristics of the commuter’s trips. Lee et al. also leverage smart card data to infer information about the purpose of passenger trips in the urban context, via the usage of a decision tree [54].

Existing work that leverages smart card datasets is mostly concerned with buses or trains within a city. For other types of vehicles like taxis and ride-hailing cars, a wider range of datasets is generally available and used in existing work. Apart from travel time, demand and traffic flow estimations, trajectory data mining has also been considered for a wide range of problems. Some other notable ones of the recent years include destination prediction [55] for applications like recommending

sightseeing places, influential billboard placement for advertising applications [56], minimal on-road time route scheduling with parking facilities [57], as well as analysis of the public bus transportation systems [58]. In general, there is also a growing trend towards the designing of deep learning techniques to solve urban mobility problems, such as crowd-flow estimations [59], transportation mode prediction [60] and trajectory predictions for road agents [61–65].

Chapter 3

Scalable Estimation of Component Travel Times

This chapter highlights the approach we propose for inferring with scalability the travel time of trip components within a metro network ¹. Specifically, there has been existing work that makes use of both smart card data and train operation logs [5, 16, 17]. In the literature [5], a probabilistic model is used to estimate the passenger flow distributions on a metro network, down to the granularity of passenger train choices. However, while methods that leverage train operation logs may shed light on the times of the trip spent inside the trains, they do not reveal much information about the time spent outside, such as the time from tapping in to boarding the trains, and the time from alighting at a station to tapping out. Moreover, pre-planned train timetables may often be coarse-grained to the level of minutes, or provide only information of train frequency. Actual realized train operation logs may also not always be available or easily obtained, since the train service is often managed by multiple train operators. Another option is to make use of cell-phone data [18], but that poses problems of considerable privacy concerns, and the data is generally hard to obtain. Hence, for the inference of network time components, smart card data tends to be a viable choice to work with due to their relatively ease of accessibility.

However, existing methods that use smart card data face limitations that make them unsuitable for our system. A regression-based model was used in [19] to

¹The work in this chapter has been published in IEEE International Smart Cities Conference 2018 [66]

estimate a few travel time component variables before using them to analyze the individual trajectory during a metro trip. Nevertheless, only a single-train-line scenario was considered. In order to scale to larger network topologies, more factors like trip transfers have to be considered. Moreover, it assumed a constant train speed, which is not always true. To achieve analysis on the scale of a whole metro network, there exists work that employed Bayesian inference using computational approaches like the Markov Chain Monte Carlo (MCMC) methods, to estimate network time attributes [6–9]. The main idea of such work is to build a Bayesian framework that simultaneously model the route choices of commuters in the metro network and estimate the time distribution of every component in the network. The advantage is that by considering the commuter route choice model, more accurate estimations of network component time distributions can be obtained, which can in turn be used to refine the modeling of route choices, so on and so forth. Furthermore, such works also have an additional advantage in revealing network time variabilities. However, the issue about these works is that they also tend to be computationally costly and may take days to complete running. As a result, some data which would otherwise contribute to the inference were omitted, highlighting a problem of scalability to larger datasets. As we seek to provide the earlier-described insights in a timely manner, such methods tend to be unsuitable for our system. In general, there is lack of work that infers network component times on the scale of a large metro network like Singapore’s, yet at the same time computationally efficient.

Hence, in this chapter, we propose a solution that addresses the above issues. Specifically, the contributions include:

- We present a multi-stage method that is based solely on smart card data, and it infers time components in a metro network in a progressive and computational-efficient manner. This method is based on the fact that trains traveling within the metro network may exhibit speed changes, and that each time component in the network may take on a different value. We also take into account all the different possible transfers at each interchange station and their respective timing.
- We share and elaborate on several insights of the metro network and how we use them to address key technical challenges in the various stages of the solution,

and propose a list of criteria that helps us to select data to improve the accuracy of the inference.

- We conduct a large-scale experiment on a set of smart card data collected from Singapore’s metro network of a 1-month period, consisting of about 56M trips. Our experiments show that our proposed method significantly outperforms alternative solutions in terms of accuracy, and demonstrates scalability to large datasets.
- We further validate our method over a large portion of the metro network using the train arrival times manually collected over a 1-month period. We also provide an analysis of the waiting times estimated from our method and how they vary over the different periods of a day.

The rest of the chapter is organized as follows: Section 3.1 presents the preliminaries of our work and describes the data used in this study. The proposed multi-stage methodology for travel time estimation is described in details in Section 3.2, followed by the experiments and discussion about the results in Section 3.3. Section 3.4 provides the discussion about the generalizability of the proposed solution, its limitations, as well as analysis of the waiting time variables estimated.

3.1 Preliminaries

3.1.1 Background

A metro network system consists of railway lines, trains which traverse along the railway lines, and train stations where commuters get aboard and off the trains. In the context of Singapore, among the train stations, there are *terminals* that locate on either ends of a line, as well as *interchanges* that locate at the intersection points of more than one line. Figure 3.1 presents an example of one such small metro network that consists of two lines, Lines 1 and 2, such that vertices colored in blue (resp. red) represent stations on Line 1 (resp. Line 2). v_1 and v_5 are terminal stations of Line 1, while v_2 and v_4 are terminal stations of Line 2. In addition, v_2 and v_4 are also interchanges between the two lines.

When a passenger enters (resp. exits) a train station, he/she needs to pass through an AFC gantry and tap his/her smart card on the electronic card reader on the gantry; we refer to such an event as a *tap-in* (resp. *tap-out*). These events are recorded in a *fare card dataset* D , where each tuple is a quadruple $\langle o, d, t_o, t_d \rangle$, which indicates that a passenger begins his/her trip by tapping in at station o at time t_o , and ends his/her trip by tapping out at station d at time t_d ². During a trip that involves transfers at an interchange, a passenger can usually move from one platform to another within the interchange, and does not have to exit from the station and re-enter from another side.

3.1.2 Modelling Metro Networks

We model Singapore's metro network as a directed graph $G(V, E)$ with a vertex set V and an edge set E . Each edge $e = \langle u, v \rangle$ is associated a value $t_l(e)$ that equals the typical time from a train's departure at u to its arrival to v (including the time required by the train to stop and pick up passengers at u). We refer to $t_l(e)$ as the *link time* of e .

Meanwhile, for each vertex v on a train line, we associate v with four values $t_{fin}(v)$, $t_{fout}(v)$, $t_{rin}(v)$, and $t_{rout}(v)$. Among them, $t_{fin}(v)$ is the typical time from a passenger's tap-in at v to his/her on-boarding a train towards the *forward* direction, while $t_{fout}(v)$ denotes the typical time from a passenger's alighting a forward-direction train to his/her tap-out at v . $t_{rin}(v)$ and $t_{rout}(v)$ are defined similarly for the *reverse* direction of the train line. Notice that $t_{rin}(v)$ and $t_{rout}(v)$ include the time required by a passenger to wait at the platform to board a train; as a consequence, $t_{fin}(v)$ and $t_{fout}(v)$ can be drastically different when the numbers of passengers queueing for different train directions are different. We refer to $t_{fin}(v)$ (resp. $t_{rin}(v)$) as the forward (resp. reverse) *tap-in time* of v , and $t_{fout}(v)$ (resp. $t_{rout}(v)$) as the forward (resp. reverse) *tap-out time* of v .

In addition, if v is an interchange station of two train lines a and b , then we associate v with four additional values $t_{fa-fb}(v)$, $t_{fa-rb}(v)$, $t_{ra-fb}(v)$, and $t_{ra-rb}(v)$, where $t_{fa-fb}(v)$ denotes the typical time required by a passenger to transfer from a forward-direction train on Line a to a forward-direction train on Line b , and the

²We note that a fare card dataset may also contain other attributes such as the identification number of a trip. We omit such attributes as they are irrelevant to our study.

FIGURE 3.1: A small metro network

other three values are defined in a similar manner. We refer to $t_{fa-fb}(v)$, $t_{fa-rb}(v)$, $t_{ra-fb}(v)$, and $t_{ra-rb}(v)$ as the *transfer time* at v .

3.1.3 Problem Definition

Suppose that we are given a fare card dataset D from the metro network $G(V, E)$, and that we only know the topology of G but not the time values associated with any vertex or edge in G . Our objective is to use D to estimate the link time of each edge in G , the tap-in and tap-out time of each vertex, as well as the transfer time of each interchange.

In our application's context, we consider Singapore's metro network, which consists of 5 train lines and 92 stations, among which 15 are interchanges. Our fare card dataset D contains information about 56M trips over a period of 1 month.

3.2 Methodology

In this section, we first highlight the key challenges of estimating travel time in the metro network using the smart card data in the context of Singapore's metro network. Then, we provide an overview of the proposed multi-phase solution, before describing these phases in details, as well as how they resolve the challenges identified.

3.2.1 Challenges and Solution Overview

Observe that the duration of each trip in the fare card dataset D can be expressed as the sum of several time variables in the metro network G . For instance, in

Figure 3.1, suppose that the forward direction of Line 1 goes from v_1 to v_5 , and that a passenger's trip starts from v_1 and ends at v_3 . In that case, the trip duration can be modelled as

$$t_{fin}(v_1) + t_l(\langle v_1, v_2 \rangle) + t_l(\langle v_2, v_3 \rangle) + t_{fout}(v_3).$$

In other words, each tuple in D provides us a linear equation involving several time variables in G .

Based on the above observation, if we are to use D to infer the time variables in G , a straightforward solution is to construct a linear formulae from each tuple in D , and then perform a linear regression to obtain an estimation of each time variable. This solution, however, requires exact knowledge of the route taken by each passenger, which is absent from D and could be difficult to obtain. For example, considering the network in Figure 3.1, we observe that if a passenger wants to travel from v_9 to v_2 , he/she may choose to take a train leftwards on Line 2 for four stations. Alternatively, he/she may also take a train downwards for one station, do a transfer at v_4 , before taking another train leftwards for two stations on Line 1. As such, there are two possible routes from v_9 to v_2 , with one being four-station long and another being three-station long with a transfer. Given only the origin and destination of the passenger trip, it is difficult to decide which route is taken by the passenger, since the time variables in the metro network G are unknown.

To avoid the complications incurred by multiple possible routes for passenger trips, one may attempt to omit all passenger trips that involve transfers, and construct a linear system using only trips that start and end at the same train line. Unfortunately, such a solution is unable to provide any estimation of transfer time at interchange stations, since the omission of transfer trips would eliminate all transfer time variables from the linear system. Furthermore, we observe that focusing on single-train-line trips does not completely eliminate the problem caused by the existence of multiple route choices by the passengers. For example, considering the metro network line depicted in Figure 3.2, if a passenger wants to travel from v_2 to v_{28} , the most direct route is to take a train in a rightward direction. However, if the trains traveling in that direction are very crowded, then the passenger may choose to first traverse from v_2 to v_1 (which is a terminal station), secure a seat on a train from v_1 , and traverse to v_{28} . (We note that such choices of *non-simple paths*

FIGURE 3.2: A train line in a metro network

are rather common in our fare card dataset, especially during the peak hours.) Therefore, it is non-trivial to construct linear systems even from single-train-line trips.

To address the above challenges, we propose a three-phase approach to inferring time variables, and Figure 3.3 shows the overview of our method.

- Phase 1: This phase considers passenger trips that (i) traverse only one train line and (ii) do not start from or end at an interchange station. In this phase, a filtering process is first used to obtain such trips. This is followed by the key of our solution, which is a lognormal mixture fitting process that eliminates the effects of passenger trips following non-direct routes. The fitting process generates a summary for every origin-destination pair of the filtered trips, which we then utilize to infer the tap-in and tap-out time for non-interchange stations, as well as the link time for the edges in G that are not adjacent to interchanges.
- Phase 2: The processes in this phase are functionally similar to that of Phase 1, except that we focus on passenger trips that (i) traverse only one train line and (ii) start from, end at, or pass by an interchange station. We use such trip data, as well as the estimated time from the previous phase, to estimate the tap-in, tap-out, and link time pertinent to interchange stations. We separate the estimation of these time variables from those considered in Phase 1, so as to reduce the number of variables in the linear system in either phase to avoid over-fitting.
- Phase 3: In this phase, we consider passenger trips that involve transfers at interchange stations, and use them to infer the transfer time at each interchange. The basic idea is to identify transfer trips whose routes are relatively easy to pinpoint, and then compare the durations of these trips against the time estimations from Phases 1 and 2 to derive the transfer time values. The processes are functionally similar to the previous two phases, except that instead of solely using the identified transfer trips, this phase also utilize the estimated time from the previous two phases.

FIGURE 3.3: Solution overview flowchart

We will elaborate each phase of our solution in Sections 3.2.2 and 3.2.3, respectively.

3.2.2 Phase 1: Estimations for Non-Interchanges

As mentioned in Section 3.2.1, Phase 1 of our solution aims to estimate the tap-in and tap-out time at non-interchange stations, as well as the link time between any two non-interchanges. For this purpose, we would like to utilize the passenger trips that do not involve any transfers. However, there are two issues that we need to tackle:

- Issue 1: How do we identify passenger trips without transfers?
- Issue 2: Given a set of non-transfer passenger trips, how do we mitigate the effects of non-simple paths taken by the passengers?

3.2.2.1 Addressing Issue 1

A naive solution to tackling Issue 1 is to consider only those passenger trips where the origin and destination are on the same train line. However, we observe that this solution is often inaccurate. To explain, consider the metro network in Figure 3.1. Suppose that a passenger trip starts from v_6 and ends at v_4 . Although both v_6 and v_4 are on Line 2, it is not necessarily the case that the trip follows a non-transfer path $P_1 = \langle v_6, v_7, v_8, v_9, v_4 \rangle$. In particular, there exists an alternative path $P_2 = \langle v_6, v_2, v_3, v_4 \rangle$ that involves a transfer to Line 1. If P_2 is a faster path than P_1 , then passengers may be more inclined to choose P_2 over P_1 . (This phenomenon occurs in several places of the metro network that we study.) Therefore, we cannot identify non-transfer trips solely based on whether the origin and destination are on the same line.

To tackle the above problem, we adopt the following heuristics to filter out passenger trips that may involve transfers. First, we omit all trips from a station u to a station v , if (i) u and v are not on the same train line or (ii) u or v is an interchange. The reason is that it is often difficult to infer the exact routes corresponding to such trips. Second, if the shortest path from u to v (in terms of the number of hops) is a path involving transfers, then we do not consider the trip, because the path involving transfers may be chosen by the passengers. On our fare card dataset, we observe that these heuristics are highly effective in eliminating transfer trips, albeit somewhat conservative. For any pair $\langle u, v \rangle$ that passes through all of the above filtering procedure, we refer to it as a *qualifying origin-destination (OD) pair for Phase 1*.

3.2.2.2 Addressing Issue 2

Suppose that we have a qualifying OD pair $\langle u, v \rangle$ for Phase 1, and we have retrieved the set of passenger trips from u to v . Our next step is to address Issue 2, i.e., to alleviate the effects of non-simple paths. Towards this end, we exploit an observation made in previous work [28, 29]: when there exist multiple route choices from u to v , the travel time of passengers' trips from u to v often follows a mixture of distributions, such that each distribution corresponds to one route choice. For example, Figure 3.4 shows the distribution of passengers' trip durations during the morning peak hours (i.e., 7:30am to 9:30am) on weekdays, for an OD pair in our

fare card dataset. The OD pair concerns two stations u and v on the same train line, and they are twelve stations apart from each other. From Figure 3.4, it is evident that the passengers can be divided into two groups with very different travel time. It turns out that the group of users with the smaller travel time are those who traverse directly from u to v , whereas the other group of users choose to first traverse from u to the terminal station of the train line (which is right next to u in the reverse direction), and then secure a seat in the train that travels from the terminal station to v . This behaviour by passengers is called back-tracking and it is often prevalent even though the train frequencies are higher during the morning peak hours of the day than during the off-peak hours.

From our fare card dataset, we observe that, for each OD pair, the distribution of passengers' trip durations can be accurately modeled as a lognormal mixture. Specifically, the decision to apply lognormal mixture was arrived in two stages. In the first stage, we try to fit the trip travel time distribution with different models, e.g., normal and lognormal distributions. We observe that the trip distribution largely resembles a lognormal one, and decides to consider that for the fitting of our trip distributions. In the second stage, we compare the fitting accuracy across the different modes of the lognormal distributions. Specifically, we look at the Bayesian information criterion (BIC) across these modes and choose the one with the lowest BIC. Since the chosen mode is very often larger than 1 (single lognormal distribution), we conclude that a lognormal mixture is able to accurately model the trip distribution. For example, the red line in Figure 3.4 shows the result of fitting a lognormal mixture to the trip duration distribution.

Motivated by this, we fit a lognormal mixture on the passenger trips for each OD pair $\langle u, v \rangle$. If the mixture contains only one lognormal distribution, we use the mode of the distribution, i.e., the travel time that is associated with its peak, as an estimation of the travel time of the most direct route from u to v . Figure 3.5 shows an example of such a mixture. On the other hand, if the mixture contains more than one lognormal distributions, we will take the distribution with the smallest mode, and use this mode as an estimation of the travel time of the most direct route. A technical issue here is that the algorithms for fitting lognormal mixtures require as input the total number of lognormal distributions in the mixture, which is unknown in advance. To address this issue, we vary the number of lognormal distributions, and measure the accuracy of each fit using the BIC; then, we choose

FIGURE 3.4: A lognormal fit of travel times between two stations

the model with the smallest BIC. In addition, if the number of passenger trips for an OD pair is small (i.e., < 1500), then we assume that all those trips follow a lognormal distribution instead of a lognormal mixture, so as to avoid inaccurate model fitting.

In summary, for each qualifying OD pair $\langle u, v \rangle$, we transform the set of passenger trips from u to v into a quadruple $\langle u, v, \mu, cnt \rangle$, where $\mu_{u,v}$ is the mode of the lognormal distribution that corresponds to the “fastest” route taken by the passengers, and cnt is the number of passenger trips from u to v . We refer to $\langle u, v, \mu, cnt \rangle$ as the *OD summary* for $\langle u, v \rangle$.

3.2.2.3 Estimations of Time Variables

Suppose that we obtain a set S of OD summaries as a result of the previous steps. For each OD summary $\langle u, v, \mu, cnt \rangle$, we retrieve the shortest path p from u to v (in terms of the number of hops), and express the travel time from u to v as a sum μ'

FIGURE 3.5: A single lognormal distribution of travel times between two stations

of the relevant time variables:

$$\mu' = \begin{cases} t_{fin}(u) + t_{fout}(v) + \sum_{e \in p} t_l(e), & \text{if } p \text{ follows the forward direction of the} \\ & \text{train line} \\ t_{rin}(u) + t_{rout}(v) + \sum_{e \in p} t_l(e), & \text{otherwise} \end{cases}$$

After that, we apply a ridge regression to obtain time variables that minimize

$$\sum_{\langle u, v, \mu, cnt \rangle \in S} \left(cnt \cdot (\mu - \mu')^2 \right),$$

with a regularization term to avoid overfitting. From the regression results, we obtain the estimations of the tap-in and tap-out time for non-interchange stations, as well as the estimation of the link time of each edge between non-interchanges. We refer to these estimations as the *estimated time for non-interchanges*. The results may also contain the estimated link time for some edges pertinent to interchange stations, but we discard such estimations since they will be estimated more systematically in Phase 2.

3.2.3 Phases 2 and 3: Estimations for Interchanges

Phase 2. After we derive the time variables for non-interchange stations, we proceed to Phase 2, which aims to estimate the tap-in, tap-out, and link time pertinent to interchange stations. In particular, we first identify the set of qualifying OD pairs for Phase 2, in a manner similar to the case of Phase 1; after that, we derive an OD summary for each qualifying OD pair based on lognormal fits, based on which we construct a linear system for estimation of time variables.

In particular, we say that an OD pair $\langle u, v \rangle$ is a qualifying OD pair for Phase 2, if it satisfies all of the following conditions: (i) u or v is an interchange, and they are on the same train line; (ii) the shortest path (in terms of the number of hops) from u to v does not involve transfers. The rationale of these conditions is to ensure that the shortest path from u to v along the train line that both u and v are on, is highly likely to be the route taken by passengers.

Once we have the qualifying OD pairs for Phase 2, we convert each pair $\langle u, v \rangle$ into an OD summary $\langle u, v, \mu, cnt \rangle$, using the same method applied in Phase 1. That is, we first retrieve all passenger trips from u to v , and fit a lognormal mixture model on the trip durations. After that, we identify the lognormal distribution in the mixture with the smallest mode, and set μ to the mode. In addition, we set cnt to the number of passenger trips from u to v .

Finally, we construct, from each OD summary, a linear formula of time variables, and we apply a ridge regression to derive the variables pertinent to interchange stations, in the same way as in Phase 1 but with one modification: if any linear formula involves a time variable for a non-interchange station, then we replace the variable with its estimated value that we derived from Phase 1. In other words, we treat the estimated time for non-interchange stations from Phase 1 as constants (instead of variables) in the ridge regression, and we focus only on the time variables for interchanges. This reduces the number of variables in the regression task, which helps improve the robustness of the results.

Phase 3. Our solution for Phase 3 is similar in spirit to those of Phases 1 and 2, except that it focuses only on the transfer time of interchange stations. In particular, we define the qualifying OD pairs for Phase 3 as the pairs $\langle u, v \rangle$ that satisfy all of the following requirements: (i) u and v are on two different train lines;

(ii) the shortest path (in terms of the number of hops) from u to v contains at most three stations other than u and v . These requirements are to ensure that the shortest path from u to v is likely to be traversed by passengers.

Given the qualifying OD pairs, we transform each pair $\langle u, v \rangle$ into an OD summary $\langle u, v, \mu, cnt \rangle$, using the lognormal fitting procedure introduced in Section 3.2.2.2. After that, for each OD summary, we construct a linear formula of time variables, and we substitute all variables for tap-in, tap-out, and link time with estimated values obtained from Phases 1 and 2. This leads to a set of formula with only variables concerning transfer time. Finally, we perform a ridge regression to obtain estimations of transfer time for all interchange stations.

3.3 Experiments

This section includes an evaluation of our solution using three different ways. We first evaluate it in terms of its accuracy for route planning purposes, and compare its accuracy with several baselines. Sections 3.3.1 and 3.3.2 elaborate on the datasets and baselines, and Section 3.3.3 elaborate on the comparison results. Then, we evaluate our solution in terms of how accurate the estimated link times are relative to ground truth values. The findings are then presented in Section 3.3.4. Finally, we evaluate our solution in terms its scalability to increasing dataset sizes, and present the results in Section 3.3.5.

3.3.1 Datasets

3.3.1.1 Fare Card Data Subsets

As mentioned, we use the fare card dataset consisting of around 56M trips collected on one month, October 2015, from Singapore’s metro network with 5 train lines and 92 stations. We generate three subsets of the dataset as follows:

- **morning-peak**, which consists of all trips made from 7:30am to 9:30am on week days;
- **evening-peak**, which contains all trips made from 5:30pm to 7:30pm on week days;

- off-peak, which comprises of all trips made on week days, other than those in morning-peak and evening-peak.

The key purpose of considering the three subsets separately is not only to account for the differences in travel characteristics across different times of the week. It is also because the train frequencies are different across these different time periods. Since the difference in train frequency may be up to the level of several minutes, training the model of our solution agnostic to the time periods may result in large errors in waiting time. To demonstrate how trip characteristics may differ over the three periods of the day, we make use of an example of a selected station present in our data set. Figure 3.6 shows the average hourly count of passengers departing from and arriving at this station, during the weekday morning peak, evening peak, and off-peak hours. The observations are as follows.

- During the morning peak hours, passenger departure rate at the station far exceeds the passengers arrival rate.
- During evening peak hours, the arrival rate is far greater than the departure rate.
- The departure and arrival rates are roughly the same during the off-peak hours.

It turns out that this station corresponds to a train station situated within a predominantly housing area, where numerous nearby residents leave by the train for work in the morning and return by the train from work in the evenings. As observed, the passenger travel patterns are not necessarily constant throughout the day. While this is only an example of a selected station, differences in trip characteristics over the periods of a day are often true for other stations in the network as well, as we observe from our data set. Hence, we consider the subsets individually. Finally, we also omit the weekend trips due to data sparsity.

3.3.1.2 Ground Truth Link Time Dataset based on Recorded Train Arrival Times

In the second part of our evaluation, we make use of the actual recorded train arrival times obtained from Singapore’s metro transit operator, manually recorded over a period of one month. This record covers a total of two train lines of the Singapore’s metro network, and consists a total of 54 stations. Based on this log,

FIGURE 3.6: Average hourly departure and arrival rate of a station

we derive a dataset that contains the link time measurements of the two lines. We call this dataset the ground truth link times and break it down into three subsets according to the time periods similar to those of the fare card data subsets.

3.3.2 Baselines for Comparisons

We consider two baseline methods in our experiments, dubbed Baseline 1 and Baseline 2, respectively. These baselines are chosen for their computational efficiency and scalability to large datasets, which are essential to provide results in a prompt manner. We note that there exists another class of methods based on Bayesian inference and MCMC-based sampling [6–9]. However, we do not consider them in our comparisons as they do not possess the computational efficiency required by our system. In particular, these methods have running time that extends into order of days, even with substantial data omission.

Baseline 1. This method is based on prior work [19]. The method assumes that the train speed on each line is a constant, and it infers this constant using a linear regression based on the observed travel time between the stations on the

train line. In addition, its regression procedure also outputs the tap-in and tap-out time of each station. However, the method does not consider the case of interchanges between multiple train lines. To address this issue, we use our solution in Section 3.2.3 to infer transfer time for Baseline 1. In particular, we first apply the method in [19] to estimate the train speed and other time variables in each line, and then convert them into a set of tap-in, tap-out, and link time, which is then fed into our solution in Section 3.2.3 to obtain estimations of transfer time at interchange stations.

Baseline 2. This method is an improvement over Baseline 1, in that it does not assume constant train speed in each train line. Instead, it is identical to our solution, except that it omits the lognormal fitting procedure that is used in the Phase 1 and 2 of our solution. That is, given the qualifying OD pairs in each phase, it directly performs a ridge regression using the observed trip durations as the dependent variables (instead of using the mode of a lognormal distribution in a lognormal mixture). Furthermore, similar to Baseline 1, it does not draw a distinction between interchange and non-interchange stations. That is, instead of inferring the network times of non-interchange stations and interchange stations in two separate phases, it does so in one single phase. Effectively, Baseline 2 serves as a middle ground between Baseline 1 and our proposed solution.

3.3.3 Experiment Settings and Results of Method Comparisons

We evaluate each method in the context of route planning. Specifically, given a fare card dataset, we use each method to infer the time variables in the metro network G . Let G' denote the metro network with the estimated tap-in, tap-out, link, and transfer time. Then, we consider a number of route planning queries, each of which asks for the smallest travel time from a station u to another station v . For each query $\langle u, v \rangle$, we compute the shortest path (in terms of travel time) from u to v in G' , and then obtain the estimated travel time t' from u to v (including tap-in and tap-out time). After that, we would measure the accuracy of t' . However, this brings us a challenge: as we do not know the ground truth travel time from u to v , it is difficult to assess the accuracy of t' . To address this issue, we utilize the lognormal fitting technique in Section 3.2.2.2. In particular, we obtain all passenger trips from

u to v , and fit a lognormal mixture model on the trip durations. After that, we identify the lognormal distribution in the mixture with the smallest mode, and use this mode as the ground truth travel time from u to v . To ensure the accuracy of this approach, we require that there must exist at least 1500 passenger trips from u to v . In addition, we would omit all trips from u to v during the inference of time variables in the metro network.

Based on the above discussions, we evaluate each method M as follows.

- Given a fare card dataset D , we identify the OD pairs with at least 1500 passenger trips. Then, we randomly select 20 such OD pairs, put them into a hold-out set H , and we remove from D all passenger trips corresponding to those OD pairs. Let D' denote the resulting dataset.
- We use M to infer the time variables in the metro network, and obtain a version G' of the metro network with estimated time values.
- For each OD pair $\langle u, v \rangle$ in H , we use G' to derive the estimated travel time from u to v , and then compare it against the ground truth computed from a lognormal fit on the passenger trips from u to v .

We repeat the above procedure 50 times, which results in 1000 travel time queries, as well as 1000 answers from each method. We divide the queries into 4 groups as follows. First, we sort the ground truth answers for the 1000 queries in ascending order. Then, we divide the sorted sequence into 4 equal-size subsequence. After that, queries corresponding to i -th subsequence is put into the i -th group. For each method, we compute the *root-mean-square error (RMSE)*, the *10-th percentile error (10%ile)*, and the *90-th percentile error (90%ile)* of its query answer in each query group.

Tables 3.1, 3.2, and 3.3 show the RMSE, 10%ile, and 90%ile errors of each method on **morning-peak**, **evening-peak**, and **off-peak**, respectively. Observe that the RMSE, 10%ile and 90%ile errors of our solution are significantly smaller than Baseline 2, which demonstrates the effectiveness of our lognormal fitting procedure in mitigating the effects of passengers' heterogenous route choices. In turn, Baseline 2 is more accurate than Baseline 1, since the "constant train speed" assumption [19] adopted by Baseline 1 incurs considerable estimation errors. The errors of each method tend to increase from query group 1 to 4, since queries with longer travel

TABLE 3.1: Comparison of RMSE, 10%ile and 90%ile errors on **morning-peak**

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	1070			1744			2171			2872		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	132	27	164	269	34	377	394	54	573	543	158	862
Baseline 2	195	63	250	188	69	306	214	35	375	246	43	394
Our solution	57	6	74	118	3	201	143	16	255	156	4	279

TABLE 3.2: Comparison of RMSE, 10%ile and 90%ile errors on **evening-peak**

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	705			1504			2141			3055		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	132	7	218	271	32	364	399	33	515	525	45	796
Baseline 2	188	65	213	225	33	391	283	33	451	267	58	435
Our solution	40	2	73	103	4	172	175	8	287	176	7	302

TABLE 3.3: Comparison of RMSE, 10%ile and 90%ile errors on **off-peak**

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	679			1610			2258			3367		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	158	34	315	360	61	488	500	113	715	504	113	692
Baseline 2	191	100	239	270	38	458	297	16	407	381	103	564
Our solution	57	9	100	136	3	249	176	27	242	286	16	501

time tend to involve a larger number of time variables, which makes them more difficult to answer accurately. Overall, the RMSE of our method is around 5% of the ground truth answer, which shows that our method provides satisfactory accuracy for travel time estimation. Moreover, the 90%ile error of our method is not larger than 16% of the ground truth answer, which implies that our method still performs fairly well even in the poor cases.

3.3.4 Validation with Ground Truth Link Times

We also evaluate, for each fare card data subset, the inference accuracy based on how close the estimated link times are to their ground truths. To this end, we first run our proposed inference method on each fare card data subset: morning-peak, evening-peak and off-peak. Similar to the previous stage of validation, we compute the RMSE, 10%ile, and 90%ile errors for the estimated link times, based on its corresponding ground truth in the link time subset. Table 3.4 shows the results for morning-peak, evening-peak and off-peak. The average ground truth link time remains relatively unchanged over the three time periods. Observe that the

TABLE 3.4: RMSE, 10%ile and 90%ile errors for link time estimation

	morning-peak	evening-peak	off-peak
Average ground truth link time (s)	152	154	151
RMSE	27	26	26
10%ile error (s)	3	4	4
90%ile error (s)	45	46	44

RMSE is around 17% of the average ground truth, while the 10%ile (resp. 90%ile) error is around 2% (resp. 29%). These results demonstrate our method’s general consistency on average in producing fairly accurate estimation across the links.

3.3.5 Computational Efficiency and Scalability

We implemented our solution using Python and conducted our evaluation on a Linux machine with an Intel Xeon 2.1GHz CPU and 32GB RAM. To evaluate its scalability and computational efficiency, we compare the total running time of all three phases of our method, across sample sets of the smart card dataset with sizes varying from 2M to 16M tuples. Figure 3.7 illustrates the average running time for each sample set. Observe that the running time exhibits a linear growth with the increase in number of tuples, and all running times fall within the range of hundreds of seconds. These results indicate that our solution is scalable to datasets with millions of tuples with a comfortable computational efficiency. This makes our solution a viable choice for timely insights provision.

3.4 Discussion

As demonstrated in our experiments, we have proposed a solution that estimates the travel time in a metro network with a comfortable computational efficiency. It addresses the issue of multiple route choices, which may be brought about by network congestions, or by the presence of multiple viable routes between stations. We also consider the fact that the train frequency is higher during the peak hours, which may mitigate the effects of the congestion to some extent, and conduct our experiments separately across the three time periods highlighted in Section 3.3.1.

FIGURE 3.7: Combined running time of the three phases

Furthermore, considering the estimation separately also allows us to resolve the issue of travel characteristic differences across these periods.

Our solution uses ridge regression which has a L2 regularization term that consists of a scaling factor and the sum-of-squares of the estimated coefficients. This scaling parameter has to be tuned to an appropriate value for an accurate estimation. For our experiments, we choose the scaling parameter that gives the best results in Section 3.3.4, which is 0.1. In the rest of this section, we highlight the generalizability of the proposed solution, an analysis of an important time variable in the metro network, as well as some limitations of our work.

3.4.1 Generalizability of Solution

Our solution addresses the issues identified in metro networks where train lines have terminals on both ends, and may include interchanges, as described in Section 3.1 in the context of Singapore. However, to the best of our knowledge, our method can be applied to metro networks elsewhere in the world, which commonly comprise of train lines as such. There are occasional cases where a train line forms a loop and does not have any terminals. As a result, there may exist two ways to travel from any station of the line to another on the same line, which brings about the issue of multiple route choices. However, this can be addressed using the same approach highlighted in Section 3.2.2.2, since travel times of OD pairs with multiple route

choices can be modeled by a lognormal mixture. We do not conduct experiments of our solution on such train lines in this work as they do not exist in Singapore’s metro network, from which we obtain the smart card data.

Furthermore, in this work, while the dataset considered during validation is from a network that is static, we also note that certain metro networks may not be so. For example, metro networks may consider A/B scheduling, where trains along a line may follow either one of the two (A or B) schedules. However, our proposed solution is also able to account for cases like the A/B scheduling, as long as we have information about such schedules. For example, since the stations in a schedule A may differ from those of a schedule B, we can consider only trips which start or end at the stations in A if we want to estimate travel time of trip components in the A-schedule. While in theory this is possible, we skip the validation for such networks, since we do not have their ground-truth datasets.

3.4.2 In-station Waiting Time Analysis

Knowledge of the time passengers have to wait between entering the station and boarding a train at the station platform, and how this duration varies across the different periods of the day, is valuable for both train operators and passengers. To this end, we note that the tap-in time estimated from our solution includes both the time the passenger takes to get from the station gantry to the train platform and the waiting time for the train on the platform. We analyze the train waiting time based on these estimated tap-in times by computing the mean, 10%ile, and 90%ile values, as shown in Table 3.5.

By referring to the 90%ile values across the three data sets, we observe that the majority of tap-in times across these periods fall within the range of less than 2.5 minutes. This shows that under the usual operating scenarios, passengers generally do not have to wait particularly long to get into the trains, even during the peak hours of the day. Comparing across the different time periods, we observe that the tap-in times during the off-peak hours tend to be longer than that of the morning and evening peak hours. The reason is that while there are larger crowds during peak hours, the frequencies of trains during these periods are also higher. For example, based on the Singapore’s train operation information of one of its lines, the train frequencies during peak hours are around 1.5 to 2.5 minutes per train

TABLE 3.5: Mean, 10%ile and 90%ile estimated tap-in times

	morning-peak	evening-peak	off-peak
Mean (s)	91	95	100
10%ile (s)	46	55	52
90%ile (s)	135	144	144

higher than the off-peak hours [67]. As a result, passengers may have to wait longer during the off-peak hours than the peak-hours.

3.4.3 Limitations

In our solution, we use ridge regression to avoid overfitting. The implication of using ridge regression is that while it penalizes overly large estimates and keep them within a reasonable range, it introduces bias into the estimation. However, we also note that the penalty in ridge regression is more pronounced only when the estimate is extremely large. Since all the time variables in our case of metro network fall within the same order of magnitude in our case and have a small variance, the bias may not significantly affect the accuracy of the estimation.

Due to the lack of ground truth for the tap-in and tap-out times, we have skipped the empirical validation for these times estimated from our solution. Also, the scalability of the solution comes with a cost in that it generates only a single value to represent each time variable, and it may often be valuable if more information than a single value is provided. For example, instead of knowing only the typical waiting time at a particular station, it may also be helpful for the commuters to be aware of the maximum and minimum waiting times they can expect to face at that station. On the other hand, existing methods that are capable of expressing such time variabilities also tend to be computational costly when they are conducted on a large scale. Hence, this points us towards a potential future direction, that is to design a travel time estimation method that reveals time variabilities in a computational efficient manner.

Chapter 4

Path Travel Time Estimation with Attribute-related Hybrid Trajectories Net

While the previous chapter focuses mainly on estimating individual time components based on total trip time of trains in a metro network, the reverse, i.e., estimating total trip time using information from the individual spatio-temporal components of the trip, may sometimes be valuable to many. Specifically, in this chapter, we focus on the problem of estimating the travel time of any path/trip within a road network based on the multi-nature properties of all sub-paths that exist within the path ¹.

Existing work for path travel time estimation traditionally falls under two approaches. The first is link-based methods, where the intuition is to first estimate the travel time of every road-segment, before combining them to get the path travel time [10–13, 32]. These methods have a limitation in that they do not consider traffic complexities outside the road-segments, such as traffic lights. The second class of methods resolves such issues to some extent; the main idea is that travel time can be estimated by extracting the historical travel time of its sub-paths, which implicitly contains certain travel costs unconsidered in the link-based ones [14, 15, 33, 34]. More recently, deep learning has demonstrated further success for

¹The work in this chapter has been published in ACM International Conference on Information and Knowledge Management 2019 [68]

this task, based on its ability for powerful representation and implicitly modeling multi-natured traffic complexities [20, 21].

However, existing work mostly leverages single-source trajectory data that are usually also large in size. While having large-scale data benefits estimation since it usually also means larger road coverage, data of such size is not always attainable. For example, public bus trajectory data sizes in many cities of the world are small due to the much fewer number of daily trips and the less extensive coverage of the road network, as compared to other types of vehicles like taxis. A newcomer to ride-sharing in a city also does not have access to large data due to the low number of trips that it has serviced initially. For the same reasons, the lack of large-scale data exists for vehicles where accurate path travel time estimation is valuable for highly-critical timely arrivals, e.g., ambulances. It is also prevalent for vehicles that require such estimation for global optimizations to meet business goals, e.g., logistics trucks.

Moreover, considering trajectory data from single sources neglects the potential estimation-improving insights present in other sources. To the best of our knowledge, there has yet to exist any work that leverages hybrid trajectory dataset for travel time prediction based on deep learning, e.g., using a mixture of bus trajectory data and a massive taxi trajectory data within the same city for bus travel time estimation. Estimating across hybrid trajectories, however, is also a challenging task mainly due to differences in trajectory characteristics. For example, buses tend to move slower than taxis due to their larger sizes and passenger loads, and have to stop at scheduled bus stops for pick-ups and drop-offs. As a result, attributes that come along with these trajectories tend to be highly crucial for such estimations, with some relatively more important than others, e.g., vehicle ID that reveals the vehicle type. However, existing work generally does not consider the relationships between attributes, nor attend the important ones.

As such, we propose AtHy-TNet, the first end-to-end neural architecture for travel time estimation using hybrid trajectory data, to tackle the issues above. Its value lies in its applicability for use-cases when only small training data is available to the transport operator, e.g., public buses, shuttle buses, and urban logistics trucks. It also enables small/medium transport operators to enhance their services by leveraging massive trajectory data which are owned by giant operators or government

agencies. To this end, we have considered various spatio-temporal and attributional treatments. The main contributions of this chapter are summarized as follows:

- For the first time, we propose a neural method that effectively tackles the novel problem of estimating path travel time using hybrid data sources. More importantly, AtHy-TNet implicitly models the multi-nature factors affecting path travel time, and their correlations across trajectories of heterogeneous vehicle sources.
- We propose: i) an Attribute Correlation Module that models path attributes correlations and attend to the more important attributes, ii) a Sub-path Representation Module that utilizes a Tri-CNN Encoder and on-road information to create a representation for each sub-path, iii) a Sub-path Correlation Module that models correlations across sub-paths, and utilizes a Correlation component to learn the path-attribute relationships, and iv) a Joint-Learning Prediction Module that learns the path travel time estimation task with an auxiliary task of sub-path travel time estimation.
- We conduct extensive experiments on trajectory data obtained in Singapore. We demonstrate the benefits of considering hybrid data for travel time estimation, and show that AtHy-TNet significantly outperforms state-of-the-art methods. We also show the robustness of its performance across different conditions, e.g., different days of the week.

4.1 Preliminaries

A trajectory of a vehicle within a road network is defined as a sequence of GPS records, $T_x = (x_1, x_2, x_3, \dots, x_n)$, where x_i denotes a record within the trajectory, and n denotes the total number of records in this trajectory. Each record $x_i = (x_i.lat, x_i.lng, x_i.t, x_i.state, x_i.id)$, where $x_i.lat$, $x_i.lng$, $x_i.t$, $x_i.state$ and $x_i.id$ denote the latitude, the longitude, the timestamp, the vehicle state (e.g., busy, stop etc.), and the map-matched road ID respectively. For each trajectory, it is associated with T_a , a set of attributes which can be categorized under four classes: vehicular, spatial, temporal, and weather. Vehicular attributes provide information about the vehicle that travels the trajectory, e.g., vehicle ID. The weather attributes

include information like rainfall level in the city. The temporal attributes include periodic information like time of day (at which T_x starts) and day of week. Finally, the spatial attributes are geospatial information like the total distance (traveled by T_x). Note that a trajectory is obtainable from different vehicle types, e.g., taxi and bus, and thus may exhibit different spatiotemporal characteristics depending on its vehicular source. As such, the vehicle ID of a trajectory acts as a unique identifier that also reveals the vehicle type. More formally, we define a trajectory data instance, which includes the trajectory and all its associated attributes, as $T = (T_x, T_a)$. In this work, we use a hybrid collection, i.e., a mixture, of trajectories obtained from more than one vehicle type.

Our objective: Given the input of a hybrid trajectory dataset built from a small trajectory dataset of one type of vehicles and a massive trajectory dataset of another type, we want to train a neural model that can estimate the travel time required by the small-dataset vehicles to complete any path within the road network, at any time of the day and on any day of the week.

4.2 Attribute-related Hybrid Trajectories Network

In this section, we present our proposed method. First, we highlight the intuition of how our method is able to leverage hybrid trajectories to improve travel time estimation. Second, we provide an overview of our architecture along with the motivations behind the constituting modules. Finally, we describe in details the functions of each of these modules, the rationale behind their designs and how these designs contribute to effective travel time estimation.

4.2.1 Intuition of AtHy-TNet

A small trajectory dataset usually implies that the coverage of its vehicles is less extensive than vehicles of a massive dataset. However, if both datasets are sampled from the same geographical space, the much larger coverage of the latter may overlap with that of the former. AtHy-TNet leverages these overlaps within the hybrid dataset and the fact that vehicles within a dataset belong to the same

FIGURE 4.1: A small map with trajectory overlays from two vehicle types (best viewed in colour).

type. The intuition is to learn the characteristics of trajectories of both the small-dataset and big-dataset vehicles considered in the hybrid dataset, and model the correlations at parts where they overlap. The correlation learnt is then used to infer the travel time of small-dataset trajectories untraveled before, yet whose paths overlap with the big-dataset ones used during training.

For instance, Figure 4.1 shows a roadmap of a small district that is overlaid with trajectories from two datasets, each obtained from a different type of vehicles. Traces in green represent trajectories generated from a small dataset A , while traces in red represent trajectories obtained from a larger dataset B . Traces in blue highlight the roads where the trajectories of the two datasets overlap. As observed, trajectories from A also have a less-extensive coverage of the road network than its larger counterpart. As such, if an operator who possesses only A wishes to estimate travel time for paths in regions outside its coverage, e.g., red regions, there may be large errors since A does not consider the complexities there yet. In this case, AtHy-TNet leverages the correlations between trajectories of the two vehicle types at the blue regions and the larger coverage of B at the red regions. Specifically, it first models the multi-natured characteristics of the trajectories in both the green and red regions, while learning the correlations of these characteristics at the blue

FIGURE 4.2: Architecture of AyHy-TNet Model

regions. This correlation can then be used to estimate the travel time at the out-of-coverage red regions.

4.2.2 Overview of AtHy-TNet

The architecture of the proposed AtHy-TNet is shown in Figure 4.2. It comprises of four modules: Attribute Correlation, Sub-path Representation, Sub-path Correlation, and Joint Prediction. The input to our model is a trajectory and its associated attributes. Path attributes are very important when dealing with hybrid data due to the different trajectory natures. However, existing work generally does not consider the correlations across these attributes nor their relative importance. As such, we design an Attribute Correlation Module to model such relationships among trajectory attributes, e.g., vehicle ID that differentiates a type of vehicles from another. We support our main path travel time estimation task with an auxiliary task of sub-path travel time estimation. The rationale is that by accurately estimating the sub-path times in a path, the path travel time can in turn be more accurately estimated. Furthermore, by modeling sub-path characteristics, it enables the model to learn the trajectory differences across different vehicle types to

a finer detail. To this end, we first extract the abstract spatial features of sub-paths by designing a Sub-path Representation Module. For the first time, we consider a design that effectively utilizes on-road information, e.g., map-matched road GPS coordinates. To model the temporal relationships across the sub-paths, and the correlation between each sub-path and the path attributes, we design a Sub-path Correlation Module. Finally, we leverage the Joint Prediction Module to combine the main and auxiliary task described earlier.

4.2.3 Attribute Correlation Module

Path travel time is largely affected by various external factors. We are the first to utilize the relative importance of each of these attributes through our proposed Attribute Correlation Module, as depicted in Figure 4.3. Specifically, the input here is T_a , a set of vehicular, weather, temporal and spatial attributes, i.e., vehicle ID, rainfall, day of week, time of day, and total distance. The Attribute Embedder embeds all of these attributes into higher-dimension vectors via a linear layer. For all attributes other than distance, we denote them as a_c , a_r , a_d and a_t respectively.

Global Distance Representation. Path distance is an important attribute in determining path travel time. Existing deep-learning based work focuses mainly on coarse-grained great-circle distances [21]. However, fine-grained distance data is able to reveal additional information about the path traveled. Hence, we integrate not only coarse-grained, but also fine-grained distance in our model. Specifically, we consider two attributes: a coarse-grained global trip distance, and a fine-grained global trip distance derived from on-road distance information. We convert each distance into a $\mathbb{R}^{16}$ vector, denoted as d_c and d_f , via the Attribute Embedder. Then we concatenate them and apply a linear transformation:

$$a_g = W_d[d_c, d_f] + b_d, \quad (4.1)$$

where a_g is the global distance representation and W_d and b_d are the parameters of this operation. The square brackets denote the concatenate operator.

Attribute Attention. Certain attributes are relatively more important than others on some occasions in revealing the travel time. Vehicle ID is generally more important when dealing with hybrid data because the trajectory of a type of

FIGURE 4.3: Attribute Correlation Module. a_c , a_d , a_r and a_t represent embedded vectors for vehicle ID, day-of-week, rainfall, and time-of-day respectively. d_c (resp. d_f) represent the fine (resp. coarse) grained embedded distance vectors.

vehicles may differ greatly from the trajectory of another, e.g., public buses move much slower than taxis. A vehicle traveling in the downtown during peak hours will be slow, due to traffic congestion. In this case, the day and time of the day are more important attributes. Since attention mechanism is a powerful way to capture such correlations, we leverage it here, as follows:

$$z_i = a_i c_i \quad (4.2)$$

$$\alpha_i = \frac{\exp(z_i)}{\sum_{i \in A} \exp(z_i)} \quad (4.3)$$

$$v_{\text{attr}} = \sum_{i \in A} \alpha_i a_i, \quad (4.4)$$

where α_i is the weight for the attribute vector a_i , $A = \{c, d, t, r, g\}$. Parameter c_i is used to calculate an intermediate score z_i . We characterize our attention mechanism with $\{a_i\}$ and $\{c_i\}$, and v_{attr} denotes the output of our attention mechanism and this module.

4.2.4 Sub-path Representation Module

The Sub-path Representation Module is the encoder for the input trajectory. First, the Trajectory Point Embedder is used to embed the trajectory point information to a higher-dimension representation. Then, the Tri-CNN Encoder, which is an encoder with three-channel output, is utilized to extract high-level abstract sub-path features based on roads used in the trajectory. To attend to the important

channel of sub-path features, we design a Sub-path Attention mechanism, inspired by [69, 70]. Finally, we use a Local Distance Representation encoder to create a more accurate representation for the sub-path distance.

Trajectory Point Embedder. The input of this module is a sequence of trajectory points T_x . For each point x_i in T_x , we map-match it to a road using [71] and embed its road ID to a vector. Due to the fact that there are often thousands of roads in large cities, using traditional one-hot encoding results in a vector too large. As such, we use the idea of condensed word vector from natural language processing and convert it to $\mathbb{R}^{32}$ for an accurate representation while saving space. Since the GPS coordinates of each point reveal fine-grained information of the vehicle, we seek to ensure that their impact are not buried by the large embedded road ID vector. This is achieved by converting a point’s latitude to $\mathbb{R}^{16}$, and doing the same for its longitude. We embed the vehicle status to $\mathbb{R}^2$. We then concatenate the embedded vectors and convert it to $\mathbb{R}^{16}$ using a non-linear layer, as follows:

$$u_a = \tanh(W_c x_i + b_c), \quad (4.5)$$

where W_c and b_c are the parameters of the embedder.

Tri-CNN Encoder. This encoder is used to extract high-level abstract sub-path features, while mitigating the noise incurred during the recording process. The rationale is that trajectory recording is a noisy process that often results in imprecise GPS coordinates recorded. Map-matching, on the other hand, is capable of estimating precisely the roads used during the trajectory. As such, by integrating the GPS coordinates of its map-matched road, it helps in extracting high-level abstract sub-path features in a more accurate fashion. To this end, we create u_s and u_e using the same procedure as Equation 4.5. The difference is that instead of using the $x_i.lat$ and $x_i.lng$ as inputs during the creation of u_s (resp. u_e), the latitude and longitude of the start (resp. end) point of the road x_i belongs to are used.

Inspired by DeepTTE [21], we then utilize a 1D convolutional neural network (CNN) to create a spatial representation for the sub-paths in u_a . CNNs have been widely used in various image recognition tasks [72, 73] for their power in capturing spatial and temporal dependencies. For tasks where the input is a 1D-sequence, e.g., sentence classification [74], the 1D-CNN is very effective in extracting local

subsequences of data instances and derive patterns within each subsequence. As such, it is applicable to our context of path travel time estimation, where the 1D input sequence (resp. subsequence) is a trajectory (resp. sub-path) and each GPS point is a data instance. In particular, 1D-CNN is effective in capturing finer-grained information like turns at road junctions [21]. Typically, a 1D-CNN contains a filter of a fixed width that slides across the data instances to capture patterns within each sub-sequence. In our model, the formulation is as follows:

$$v_a = \text{ELU}(W_a * u_a), \quad (4.6)$$

where $W_a \in \mathbb{R}^{k \times d \times 16}$ is the parameter, $*$ is the convolution operation, k is the filter width and d is the output dimension. ELU is the activation function [75], where $\text{ELU}(x) = \exp(x) - 1$ for $x \leq 0$ and x for $x > 0$. We then use the same 1D-CNN to create another sub-path spatial representation v_s (resp. v_e) from u_s (resp. u_e) respectively.

Sub-path Attention. Different CNN outputs may have different importance. If T is very noisy, a greater importance should be assigned to v_s or v_e , instead of v_a . As such, we pass the output of the Tri-CNN Encoder through our Sub-path Attention mechanism. This attention mechanism is similar in spirit to the Attribute Attention proposed in Section 4.2.3, with one exception. The formulation is as follows:

$$y_i = \sum_{j=1}^{n-k} v_{i,j} f_i \quad (4.7)$$

$$\beta_i = \frac{\exp(y_i)}{\sum_{i \in S} \exp(y_i)} \quad (4.8)$$

$$v_m = \sum_{i \in S} \beta_i v_i, \quad (4.9)$$

where β_i is the weight for v_i , $S = \{s, a, e\}$, and $\sum_{i \in S} \beta_i = 1$. To learn β_i , we use a parameter vector f_i . Note that unlike Attribute Attention, the intermediate attention score y_i is calculated by the sum of the inner product between $v_{i,j}$ and f_i across all j , where $v_{i,j}$ denotes the j -th sub-path vector of v_i .

Local Distance Representation. The sub-path distance is an important information. As mentioned in Section 4.2.3, supporting it by the integration of fine-grained distance may create a more accurate representation of the distance

traveled. To this end, we consider two inputs, the coarse-grained local distance of the sub-paths and its fine-grained version. We denote these by $l_c, l_f \in \mathbb{R}^{n-k}$ respectively. Similar to Section 4.2.3, we convert each of l_c and l_f into $\mathbb{R}^{(n-k) \times 16}$ using a linear layer. Then, we perform a concatenation on l_c and l_f , and convert it to $\mathbb{R}^{(n-k) \times 16}$ using another linear layer. Finally, we concatenate this vector with v_m of the Spatial Representation Attention. We denote the final output of this module as m .

4.2.5 Sub-path Correlation Module

Correlation Component. Travel time is highly dependent on the path attributes, as mentioned earlier. To capture the inter-dependencies between the attributes and the sub-paths, we design a Correlation component where v_{attr} is first concatenated with the representation of each sub-path in m , before passing through a linear layer, as follows:

$$s_i = W_s[m_i, v_{\text{attr}}] + b_s, \quad (4.10)$$

where m_i is the representation of the i -th sub-path, $1 \leq i \leq n - k$, and W_s and b_s are the parameters of this operation.

LSTM Layer. To further capture the temporal dependencies among sub-paths, we feed the results into long short-term memory (LSTM) network, a type of recurrent neural network (RNN). Generally, RNNs enable the processing at every step of a sequential input to leverage information from the previous steps. In particular, LSTM is capable of learning longer-term dependencies by using a memory cell, and resolving the issue of vanishing and exploding gradient in standard RNNs. Briefly speaking, the hidden state h_t and memory cell c_t are function of h_{t-1} and c_{t-1} from the previous step, and input vector s_t , or formally:

$$c_t, h_t = \text{LSTM}(c_{t-1}, h_{t-1}, s_t), \quad (4.11)$$

LSTM uses an input and a forget gate to control the information flow within the recursive operation. We also use two stacked LSTM to allow greater model complexity. For more details about LSTM, we refer to [76]. We denote the output of this layer as h_{sub} .

4.2.6 Joint Prediction Module

This module jointly learns the main task of path travel time estimation, with an auxiliary task of sub-path travel time estimation, inspired by [21, 77]. The intuition is that the travel time of a path is essentially made up of the sum of its sub-path travel times. By accurately estimating the sub-path times, the path travel time can be more accurately predicted, vice versa.

Sub-path Aggregation. Due to length variability of h_{sub} , we convert it to a fixed-length vector via pooling. While mean-pooling treats all sub-paths equally, attention-pooling gives more attention to critical sub-paths, e.g., many traffic lights or road segments [21]. As such, we utilize an attention mechanism that is similar in structure as Attribute Attention of Section 4.2.3. However, instead of an arbitrary parameter vector, we convert v_{attr} via a linear layer to a vector that is similar in size as a vector in h_{sub} , and use it as the parameter. The output of the aggregation is denoted by r_{path} .

Main and Auxiliary Task Learning. To generate the estimated path travel time, we use deep residual learning. Deep residual learning is a technique for training of very deep neural networks and has shown superior performance across multiple recognition [78] and certain geo-spatial tasks [59]. The intuition is that each residual unit focuses on fine-tuning the output from the previous unit by only learning an added "residual". This is achieved through shortcut connections across the units. Such designs mitigate the degradation issue of deep neural networks where adding more layers results in lower accuracies, and makes it very effective to train deep networks for maximal representation power. Specifically, we pass the concatenated vector $[r_{\text{path}}, v_{\text{attr}}]$ through a linear layer, followed by a series of residual units. The formulation of a residual unit is:

$$r_l = r_{l-1} + \text{ReLU}(W_r r_{l-1}), \quad (4.12)$$

where r_{l-1} denotes the output from the $(l-1)$ -th residual unit. Also, $1 \leq l \leq L$, where L is the total number of residual units. ReLU is the activation [79]. W_l is the learnable parameters of this unit. Finally, we pass r_L through a linear layer to get the estimated path travel time $t_{\text{path}} \in \mathbb{R}^1$. The training objective for this task

is:

$$C_{\text{path}} = \left| \frac{t_{\text{path}} - g_{\text{path}}}{g_{\text{path}}} \right|, \quad (4.13)$$

where g_{path} is the ground-truth path travel time.

To get the estimated travel time $t \in \mathbb{R}^{(n-k) \times 1}$ for the sub-paths, we simply pass h_{sub} through several fully-connected linear layers. The objective for this task is:

$$C_{\text{sub}} = \frac{1}{n-k} \sum_{i=1}^{n-k} \left| \frac{t_i - g_i}{g_i} \right|, \quad (4.14)$$

where t_i (resp. g_i) denotes the estimated (resp. ground-truth) value for the i -th sub-path.

For optimization during training, we use the following loss function:

$$C_{\text{join}} = \zeta C_{\text{path}} + (1 - \zeta) C_{\text{sub}}, \quad (4.15)$$

where ζ is a weighing factor that considers the relative importance of the error of each task.

4.3 Experiments

To ascertain the effectiveness of our proposed method, we conduct experiments on real-world datasets. Specifically, we focus on the application of estimating public bus travel time using a hybrid dataset of public bus and taxi trajectories.

4.3.1 Experimental Set-up

This section describes the datasets used, the baselines we consider for comparisons, and the evaluation metrics.

Datasets. For a fair validation process and to construct a hybrid trajectory dataset, we require datasets obtained from the same geographical space. As such, we make use of the following datasets:

- **Singapore Smart Card Dataset:** This dataset contains passenger trip records obtained from CEPAS card records from Singapore’s public bus network. After pre-processing, there are 204,000 trajectories in all. We explain the preprocessing steps below.
- **Singapore Taxi Dataset:** We use a dataset that has 11.2M taxi trajectories. Each point within the trajectory contains the GPS coordinates, the time instance at which it was sampled, and the status of the taxi (i.e., busy, idle, stop, on-call, or hired) at that time instance. Each trajectory also includes the following information: a vehicle ID, time of day, and day of week. The longest trajectory has 128 points.

We further collect city-scale rainfall records; a total of 5 distinct levels, from Singapore. All the above datasets are obtained over an overlapping time period in March 2016.

Pre-processing: Bus Trajectory Derivation from Smart Card Dataset. A passenger trip record in the smart card dataset consists of the following features: origin stop, destination stop, bus number, tap-in time, and tap-out time. Specifically, tap-in (resp. tap-out) time refers to the time at which the passenger boards (resp. alights) the bus at origin (resp. destination) stop by tapping in (resp. out) at the on-bus fare gantry. A bus journey starts from a bus terminal, passes a series of scheduled bus stops, and ends at a bus terminal. Unlike taxis, public buses generally follow scheduled routes. Through basic sorting and processing, we derive a trajectory for each bus journey in the dataset, where each point within the trajectory contains the GPS coordinates of the bus stop, as well as the timestamp of the first tap-in at that stop during the journey. We obtain the exact on-road distance between any two consecutive stops traveled by the bus from the publicly accessible bus schedules. As such, we can further derive the on-road distance between any two consecutive GPS points of the trajectory, as well as the total on-road journey distance. Since the date and time of the journey are recorded, we can get the following attributes required by our model: day of week, and time of day. Finally, instead of using the vehicle states of taxis, we assign a constant bus state to each GPS point of the bus trajectory to further differentiate bus trajectories from taxi ones.

TABLE 4.1: Trajectory data characteristics

Statistical Index	Bus	Taxi
Travel Time Mean (s)	3,245	692
25%ile Travel Time (s)	1,841	283
75%ile Travel Time (s)	4,192	933
Travel Time Std (s)	1,756	573
Time Gap Mean (s)	204	34
Sampling Interval (s)	-	30

Data Characteristics. Table 5.1 shows some statistics of the trajectory datasets. Observe that the typical bus data is vastly different from the taxi data. The mean, 25%-ile and 75%-ile travel times show that the total trip duration of a bus trajectory is typically several times that of a taxi’s. Bus travel time also has a larger variance, and the time-gaps between consecutive trajectory points are larger. Moreover, the points in a taxi trajectory are sampled at a regular frequency, but the points in a bus trajectory are sampled when the bus is at the bus stops.

Training, Validation and Testing Sets. We create a bus training set Bus-Training by sampling 70% of the bus lines in Singapore Bus Dataset and collecting all the trajectories of these lines. Similarly, we sample 10% (resp. 20%) of the remaining bus lines for the validation (resp. testing) data, denoted as Bus-Validation (resp. Bus-Testing). We also create a taxi training set Taxi-Training, by randomly sampling from the Singapore Taxi Dataset a number of trajectories that is 5 times that of the number of trajectories in Bus-Training. Finally, we create a hybrid dataset Hybrid-Training by combining Bus-Training and Taxi-Training. This is done by simply putting the bus and taxi trajectories in a common dataset. For example, if Bus-Training = $\{T_1, T_2\}$ and Taxi-Training = $\{T_3, T_4, T_5\}$, where each $T_i, 1 \leq i \leq 5$ denotes a trajectory instance, Hybrid-Training = $\{T_1, T_2, T_3, T_4, T_5\}$. We repeat the above steps for 5 times to get 5 different sets of data.

Evaluation Metrics. For our experiments, we adopt the following evaluation metrics: mean absolute percentage error (MAPE), mean average error (MAE), and root mean square error (RMSE).

Implementation Details and Parameter Setting. We implement our model on Pytorch 0.4.1. Our model training/evaluation is done on Nvidia Tesla M40 GPU, and Linux machine with an Intel Xeon 2.6GHz CPU and 64GB RAM. We use Adam optimization algorithm to fine-tune the parameters. The learning rate we

use is 0.001 and we train our model in batches of 400 over 100 epochs. For hyperparameters settings, we set $k = 3$ and $c = 32$ in the Sub-path Representation Module. The embedding size of the attributes or inputs are chosen based on empirical testings. In the LSTM layer of our Sub-path Correlation Module, we use the hidden vector size of 128. In the Correlation component, we use an output size that is the same as the input. In the Joint Learning Module, we set the number of residual units used as 3; the size for each unit is 128. We also set the number of fully-connected layers used in the auxiliary task prediction as 1. Finally, we set ζ as 0.7, for a stronger focus on the main task loss function. To provide sufficient clout for the relatively smaller bus trajectory sets, we also associate a constant vehicle ID for all the bus trajectories.

Baselines. We consider several baselines in our experiment. Each baseline has a superscript $*$, $\dagger$ or $\ddagger$, which denote that the model uses Bus-Training, Taxi-Training, Hybrid-Training for training respectively. In all cases, Bus-Validation and Bus-Testing are used for validation and testing. The baselines are as follows:

- **DeepTTE^{*}** As there is no prior work on hybrid-trajectory-based estimation, we compare with the state of the art single-source method. DeepTTE [21] is chosen as it is the best-performing one to date, yet is also modifiable to support hybrid trajectories. Specifically, instead of considering the driver ID feature in this baseline, we use vehicle ID to differentiate bus trajectories from taxi ones. Instead of using the weather feature that is required in DeepTTE, we replace that feature with the rainfall level present in our datasets. In this baseline, only Bus-Training is used for model training to demonstrate the effectiveness when only a single-source small trajectory dataset is available.
- **DeepTTE[†]** This baseline is similar to DeepTTE^{*} except that Taxi-Training is used for training. It is chosen to explore the effectiveness of solely utilizing a larger trajectory dataset, but of different characteristics.
- **DeepTTE[‡]** This baseline is similar to the previous two, except that Hybrid-Training is used for training. This baseline serves to show how estimation changes when the training dataset uses not only a small single-source dataset, but also a larger one obtained from the same geographical space.
- **LightGBM[‡]** LightGBM is one of the most efficient and high-performing gradient boost decision tree method [80]. The trajectory input for this baseline

is the same as AtHy-TNet. Apart from the total path distance, the attributes of the trajectory are specified as categorical features in the program.

- **HyMLP[‡]** This baseline is a simple 5-layer fully-connected network with ReLU activation. The input is the same as AtHy-TNet and each categorical attribute in the input is embedded to a low-dimensional vector.
- **HyLSTM[‡]** This baseline uses a 2-stacked LSTM. The input is the same as AtHy-TNet, except that neither sub-path representation nor attribute correlation is considered. Similar to HyMLP[‡], each categorical attribute is embedded to a low-dimensional vector, before feeding into the LSTM. The estimation is obtained by mean-pooling the LSTM output and passing through residual fully-connected layers.
- **AtHy-TNet^{*}** This baseline uses AtHy-TNet on Bus-Training instead of Hybrid-Training. The purpose is to demonstrate the raw improvements brought about by our model when a single-source small dataset is used.
- **AtHy-TNet-a[‡]** This baseline is a simplified model of AtHy-TNet. Instead of using the Sub-path Representation Module, AtHy-TNet-a[‡] uses the Geo-Conv Layer proposed in [21].

4.3.2 Evaluation Results

We conduct experiments on the five sets of {Bus-Training, Taxi-Training, Hybrid-Training, Bus-Validation, Bus-Testing} described earlier, and record the average testing results on Bus-Testing in Table 5.2. As observed, methods that utilize hybrid trajectories perform better than their single-source versions. For example, the performance of DeepTTE[‡] is at least 2% stronger than DeepTTE^{*} and DeepTTE[†]. Despite the fact that Taxi-Training is much larger than Bus-Training and supposedly not as affected by data sparsity, performance of DeepTTE[†] is significantly worse than DeepTTE^{*}. This shows that simply replacing a dataset with another results in large errors, due to the vast differences across different sources. AtHy-TNet-a[‡] shows a stronger performance than all other hybrid-data methods by at least 2%. This demonstrates effectiveness of our model in estimating across hybrid trajectories, and by considering correlations of the relevant attributes. The performance of AtHy-TNet[‡] is a further improvement from AtHy-TNet-a[‡] by around

TABLE 4.2: MAPE, MAE, and RMSE on AtHy-TNet and the baselines. Best results are in bold face and second best underlined.

Method	MAPE (%)	MAE (s)	RMSE
DeepTTE*	15.16	367.16	374.51
DeepTTE [†]	41.93	1244.69	1433.67
DeepTTE [‡]	13.15	350.60	358.26
LightGBM [‡]	14.60	393.83	544.25
HyMLP [‡]	17.40	497.70	608.60
HyLSTM [‡]	14.81	388.70	406.70
AtHy-TNet*	13.66	329.45	342.0
AtHy-TNet [†]	38.72	927.78	1132.86
AtHy-TNet-a [‡]	<u>10.91</u>	<u>288.32</u>	<u>295.73</u>
AtHy-TNet [‡]	10.24	264.21	273.12

0.7%. This demonstrates the benefits of extracting high level sub-paths spatial features through fine-grained road-based information. As such, our experiment results show that the effectiveness of our model in learning multi-nature correlations of hybrid trajectory data. Note that AtHy-TNet* demonstrates a performance stronger than DeepTTE* by 1.5%, which shows that our method is effective even when single-source training data is used.

Comparison with Google Maps. We also compare AtHy-TNet with Google Maps API [81]. Adopting the treatment in [82], we query Google Maps Directions API at the same time and weekday as the starting time of each trip, by assuming a strong weekly periodicity. Since the API does not support the querying of past trips, all the queries correspond to the current or future time. As arbitrary public bus routes outside of the existing bus lines are unsupported in the query, the travel mode is set as Driving. Due to credit limits, 30k trips within a set of Bus-Testing are randomly chosen for comparison. The MAPE, MAE, and RMSE are 20.9%, 491, and 647 respectively. In contrast, the errors are 11.24%, 237, and 241.3 respectively when the same trips are estimated using AtHy-TNet. A reason why Google Maps does not perform as well may be that the data sources for its estimations in the Driving mode are largely based on non-public-bus vehicles. Since the travel time of arbitrary public bus routes are unsupported in online map services, it further highlights the necessity of a solution for vehicles with small trajectory datasets.

Effect of Varying Taxi and Bus Data Size in Hybrid-Training. Next, we keep the bus trajectories in Hybrid-Training constant and vary the concentration

of taxi trajectories. Specifically, we consider taxi data that is K times the size of the bus data. We evaluate the performance of AtHy-TNet[‡] over different values of K , using Bus-Validation and Bus-Testing. The results are shown in Figure 4.4. As observed, the performance is significantly worse when no taxi trajectories are used in the hybrid training data, but improves rapidly when they are introduced. As early as $K = 1$, the decrease in error is more than 2%. The performance strengthens with increasing K and settles at its peak from $K = 4$ onward. The maximum achievable improvement is nearly 4%. This demonstrates benefits of hybrid data, and capacity of AtHy-TNet[‡] to effectively utilize hybrid data, even when the external data used is small. Error tends to be larger when K is small, since a smaller external taxi data often implies a smaller extended area coverage. As such, estimation does not improve as much as when larger taxi data/coverage is considered in the hybrid data.

We also vary the concentration of bus trajectories while keeping the taxi trajectories in Hybrid-Training constant. The results are shown in Figure 4.5. As observed, the accuracy improves as the bus trajectory data size increases. The reason is that when the bus data is small, the area coverage of buses that can be used for correlation-learning with taxis tends to be small too. As such, the error is larger as compared to when more bus data is available for correlation-learning. Nonetheless, AtHy-TNet[‡] demonstrates a satisfactory accuracy using bus data sizes even as small as 20k and 40k. This shows that our method is capable of leveraging external data for estimation to address the inaccuracy brought about by a single-source small dataset. Moreover, in both cases, AtHy-TNet[‡] shows a stronger performance and a steeper descent over increasing data size than DeepTTE[‡]. This shows the suitability of our model over prior art in effectively modeling across hybrid trajectories.

Performance across Different ζ in the Joint Learning Module. Figure 4.6 shows how the performance of AtHy-TNet varies across different values of ζ , from 0.01 to 1.0. As observed, the performance is satisfactory in general across all values. Although the performance remains around a constant level in the middle of the range, the errors are slightly larger at the ends, i.e., 0 and 1.0. This demonstrates the benefits of learning the main task of path travel time estimation with an auxiliary one and considering the errors of both during training.

FIGURE 4.4: Performance over different K values

FIGURE 4.5: Performance over different bus data size

FIGURE 4.6: Performance of AtHy-TNet over different ζ

Performance across Different Trip Travel Times. Figure 4.7 compares the model performance across trajectories of different trip durations. Generally, AtHy-TNet consistently shows a stronger performance than DeepTTE and LightGBM across all trip durations. It is interesting to note that the error tends to decrease as trip duration increases for all methods. The reason may be that unlike a short trip, a long trip usually has more constituting sub-paths. Since each sub-path provides valuable information that is useful for estimation, having more sub-paths means the longer trips have more information usable by the methods. This results in the travel time of these trips to be estimated more effectively than the shorter ones. Furthermore, the rate of error-decrease for AtHy-TNet is the fastest as compared to other methods. Specifically, error decreases from 16% to around 11% when the trip duration increases from 0-1000s to 1000-2000s, stays constant at 2000-3000s, before further decreasing to around 8% for durations within 3000-5000s. In contrast, the errors of DeepTTE stay nearly constant during the shorter trips and only starts to

FIGURE 4.7: Performance over different travel times

decrease towards the longer ones (3000-5000s).

Performance across Different Days of the Week. Figure 4.8 shows the performance of AtHy-TNet and baselines across the different days of the week where 1 represents Monday and 7 represents Sunday. As observed, AtHy-TNet out-performs DeepTTE and LightGBM across all days. It is interesting to know that the trend of errors' variation across the week is similar across all three methods. Specifically, error increases almost linearly from Monday to Sunday, with dips on Wednesday and Friday. Estimation errors are higher on weekends than weekdays since the travel patterns of passengers are less regular. It may also be because the number of trajectories collected on weekends is generally smaller as there are fewer weekend days in a week. The rate of increase for AtHy-TNet, as shown by the gradient of the slope, is slow as compared to the other two methods. Furthermore, its errors fall within the satisfactory performance range of 9-11%. This demonstrates the performance robustness of AtHy-TNet across different days of the week.

Performance across Different Times of the Day. We also evaluate the estimation performance across different times of the day. We conduct the test separately for weekdays and weekends, since the travel characteristics of weekdays and weekends are largely different. The results are shown in Figure 4.9 and 4.10 respectively, based on the 24-hour clock. As observed, AtHy-TNet out-performs the baselines across all times of the day for both cases. The errors of AtHy-TNet also show a smaller variance in both cases. Similar to Figure 4.8, the MAPEs during the

FIGURE 4.8: Performance over different days of the week

weekends are mostly higher than during the weekdays. For both weekdays and weekends, errors remain relatively constant for all methods during the later parts of the day, i.e., 9 hrs onwards. Note that the error of the earliest period, i.e., 0-3 hrs, of weekdays is higher than that of the later periods. In contrast, the error over the same time period during weekends is lower than the later ones. The reason for such discrepancy may be because there are fewer bus services that run at that period. As such, there are much fewer trajectories available for training and testing, and this results in a large variance in errors between weekdays and weekends. Furthermore, specific to AtHy-TNet, the error during the 6-9 hrs of weekdays is the highest as compared to the rest of the day. This is reasonable since the 6-9 hrs period is the morning peak hours when traffic congestions are rampant, due to people going to a common set of places. As a result, the difficulty of estimation is higher. Compared to the baselines, this observation also shows that our model manages to reveal such temporal-dependencies in the datasets.

Attribute/Feature Importance Evaluation. We study the effect of each of the attributes used in AtHy-TNet, by evaluating the performance of AtHy-TNet when only that attribute is excluded from the model. When the day-of-week attribute is excluded, the MAPE worsens by 0.46%. When time-of-day attribute is excluded, the MAPE worsens by 0.99%. This is aligned to our intuition that traffic complexities, e.g., traffic congestions, are different in different times of the day/week. As such, temporal attributes are important to model such differences. When the rainfall attribute is excluded, MAPE increases by 0.43%. This is coherent with the well-studied fact that vehicles tend to move slower on rainy days. To study the effect of differentiating bus trajectories from taxi ones, we exclude vehicle ID. Recall

FIGURE 4.9: Performance over different times of the day (week-day)

FIGURE 4.10: Performance over different times of the day (weekend)

that the bus states used are generally different from the taxi ones. As such, we also exclude the vehicle state of each GPS point. As a result, the MAPE worsens by 0.96%. This highlights the importance of differentiation of vehicle types when dealing with hybrid trajectories, since vehicles of different types may have distinct travel behaviours.

Predicting Time. The predicting time of AtHy-TNet is fast, in spite of its slower training time. The time taken to estimate a batch of 400 trajectories is around 0.023s.

4.3.3 Transfer Applications and Limitations

AtHy-TNet is the first work that demonstrates the benefit of using hybrid dataset for a task. A possible transfer application of the method is to use it for the trajectory classification problem of a vehicle type that has a small dataset, e.g., classify bus trajectories based on the degree of in-bus crowdedness. Due to privacy and legal concerns, it is often hard for one operator, e.g., taxi company, to share its data with another, e.g., bus company. As such, another potential extension is the use of federated learning for the training process. For example, instead of training on a hybrid dataset, the model may first train on the taxi dataset, before fine-tuning by training another time on the bus dataset. In this case, only the model needs to be shared among the companies while the datasets are kept private.

A limitation of AtHy-TNet is that it is a preliminary attempt at demonstrating the effectiveness of using hybrid-trajectory data. As such, the model can be improved in several ways, like including finer-grained road information such as regulated speed limits, and if the road has electronic road tolls installed. Furthermore, AtHy-TNet deals with the estimation of a single type of vehicles when the single-source dataset is small. An extended exploration on leveraging hybrid datasets for multi-modal path travel time estimation may be useful, to public organizations for traffic studies, and to commuters for multi-modal trip planning.

Chapter 5

Demand Prediction with Attribute-guided Multi-graph Convolutional Network

For the problem of prediction of regional demands, various computational methods have been proposed to comprehensively capture the dependencies for a more accurate prediction. Existing work for regional demand prediction generally falls under two classes: traditional statistical methods, and machine learning methods. The former comprises mainly of Auto Regressive Moving Average model (ARIMA) and its variations; these methods utilize both autoregressive and historical moving average components for prediction [22, 23]. However, such methods focus primarily on the temporal dependencies and not factors of other natures. To resolve this, recent methods have been designed to take into account more factors that affect such demands. Specifically, the state of the art leverages deep learning techniques, which has demonstrated considerable success for demand prediction and many other spatiotemporal tasks, due to the superior modelling capabilities. In [24], Yao et al. leverage CNNs, LSTMs and graph embedding methods to model the spatial, temporal and structural factors affecting the demand. This model allows a more comprehensive consideration of factors as compared to solely using the temporal information of the regional demands. Then, based on the idea that integrating non-Euclidean pair-wise relationships between regions of a city may reveal further insights about the demand at a region, Geng et al. [25] propose a spatiotemporal multi-GCN (ST-MGCN) to tackle this issue. ST-MGCN models the different

pair-wise relationships and demonstrates the advantages in considering these relationships over existing work that leverages solely Euclidean-based relationships, such as [26].

Future demand prediction has generally been very challenging due to the various dependencies affecting the demand at a region. First, taxi demand is affected by spatial-related characteristics, such as the region at which the demand is estimated. For example, a region that falls within the downtown of a city is likely to experience a higher demand than a region from a less populous part of the city. This is supported by the fact that a city is often demarcated by the local government into various regions, with each serving a particular function, e.g., financial district, or residential district. Second, demand at a region is also highly dependent on temporal periodicities. For example, the demand for taxis is likely to be high at a residential region during the morning peak hours on a weekday, since that is typically the time when most people leave home for work. In contrast, the demand at an international airport is likely to be very high all day long. Finally, such demands are also highly dependent on sporadic events. For example, a region where a large-scale pop-music concert had just ended may see a high demand over a few consecutive time periods.

However, the state of the art for demand prediction tends to mostly consider historical demands and pair-wise relationships between regions for prediction. However, it may not always be accurate to consider solely historical demands for the prediction of future demands. For example, 10 am of a weekday usually corresponds to the off-peak hours in some cities. If we were to predict the demand from 10 to 10:30 am using solely the demands over the past time periods, the results may be largely inaccurate. This is because those periods fall within the morning peak hours, which generally see much higher demands for taxis due to people traveling to work. To circumvent this issue, we may consider integrating high-level temporal or spatial attributes like period-of-day or region ID to guide the estimation and reduce the overdependence on the historical demands. However, one important factor largely ignored in existing work is that certain attributes are sometimes relatively more important than others in revealing the demand at a region. For example, the period of the day is generally important for revealing demand information. However, this may not be as true during the weekends when the periodicity is not as strong

as that of the weekdays. In the latter case, the day of week may be the more important attribute instead.

Moreover, while leveraging multiple pair-wise relationships may uncover previously hidden insights that improve estimations, existing work does not consider the correlations across these diverse relationships. For example, the demands at regions that fall directly within the neighbourhood of a region may often reveal the information about that region. However, this does not hold true in all cases, especially when these regions are functionally dissimilar. As such, different relationships may be relatively more important than others under different conditions.

Motivated by these, we propose AM-Net, a neural architecture that leverages multi-nature dependencies to predict the next-period demands at every region of the city. This model also leverages insights from high-level attributes of data samples to guide the estimation. Furthermore, it not only learns the various pair-wise relationships between regions and the temporal correlations of historical demands, but also models the mutual correlations across these pair-wise relationships.

Specifically, our novel contributions include:

- We propose a Mutual-correlated Mutli-GCN module that models the spatial correlations through different pair-wise relationships across regions, as well as the correlations across these different relationships.
- We also propose for the first time an Attribute Attention Guidance module that guides the mutual-correlated multi-GCN module towards a more effective prediction of demand. The module achieves this by learning the spatial and temporal correlations based on high-level attributes that may provide insights about the demand, as well as the relative importance across these attributes.
- We conduct extensive experiments on demand records obtained from New York, and demonstrate that AM-Net significantly outperforms state-of-the-art methods. We show the benefits of the proposed modules and evaluate the performance of AM-Net across different conditions, e.g., different days of the week and periods of the day.

5.1 Preliminaries

In this work, we formally define the city-wide regional demand record at a time period t as $D^t = (X^t, C^t)$. $X^t = (X_1^t, X_2^t, X_3^t, \dots, X_N^t)$, where X_i^t denotes the demand during the t -th period at the i -th region of the city, and N denotes the total number of regions in the city. X^t is associated with C^t , a set of attributes that reveals high-level information about the demand record of all regions. Specifically, each X_i^t has an associated C_i^t that comprises of spatial and temporal attributes, where spatial attributes may include features like the ID of the region from which X_i^t is sampled, while temporal attributes refer to features like the period of the day or day of the week on which X_i^t is sampled.

A region within a city may be connected to another via different pair-wise relationships, e.g. neighbourhood, road connectivity, or functional similarity. For each of these relationships, we represent it using a graph $G_j = (V, A_j)$ with a vertex set $V \in \mathbb{R}^N$ and an adjacency matrix $A_j \in \mathbb{R}^{N \times N}$. Each vertex corresponds to a region within the city, while each entry in A_j represents the strength of connection that exists between the two corresponding regions. j denotes the type of relationship, and if there are M relationships between the regions, $j \leq M$.

Our objective is that given the input of a sequence of historical demands $(X^t, X^{t-2}, \dots)$, a set of high-level attributes C^{t+1} , and a set of pair-wise relationships, $G_1, G_2, \dots, G_M$, we want to predict X^{t+1} , the next-period city-wide regional demand.

5.2 Attribute-guided Multi-graph Convolutional Neural Network

In this section, we propose a method that comprehensively models the multi-nature dependencies for an effective prediction of next-period taxi demands. We first present an overview of our solution and explain the purpose of each module within the architecture. Then, we highlight in details each of these modules and explain the motivations and intuitions behind their designs.

FIGURE 5.1: Architecture of AM-Net Model. G^n denotes the graph that represents the neighbourhood relationship across regions of the city, while G^r and G^s denote the graphs that represent the road connectivity and functional similarity relationships respectively.

5.2.1 Overview of AM-Net

Figure 5.1 shows an overview of AM-Net. The model starts off by using Multi-contextual-gated RNN module to model the temporal correlations across the historical demands of each region. This is done with the help of integrating both the global historical demand information and that of the regions which it is relationally close to. Since there are different types of pair-wise relationships present between regions of a city, three different graphs are used to represent the different types of relationships. A Contextual-gated RNN (CGRNN) is applied to each of these graphs to get a high-order representative vector of the historical demands at each region. Then, these representative vectors are passed through a Mutual-correlated Multi-GCN module, which models the pair-wise relationships across regions, as well as the mutual correlations across these relationships. An external Attribute Guidance module is designed to model the relative importance of the high level attributes affecting the demands at each region under different circumstances. This module also directs the prediction at the Attribute-guided Prediction module. Finally, the output of the prediction module is our desired outcome, i.e., the predicted demand over the next time period.

5.2.2 Multi-contextual-gated RNN

The demand at a region during a certain period is often correlated with the historical demands at that region. For example, the demand during the morning peak hour of 8:30-9:00 am on a Tuesday tends to be very similar to that of the preceding time period, i.e., 8:00-8:30 am. This demand is also likely to be very close to the demand at this region over the same period one day ago, as well as the demand over the same period exactly one week ago, i.e., 8:30-9:00 am on Monday and 8:30-9:00 am last Tuesday. As such, it is important to model such temporal correlations and Multi-contextual-gated RNN module is designed for this purpose.

Contextual-gated RNNs and Leveraging Multiple Pair-wise Relationships across Regions. The key idea of the CGRNN lies in that in addition to modelling the temporal correlations across the historical demands at a region, the demand information on the global/city scale can also be integrated to provide a more insightful representation of the historical demands at that region. Similarly, such information derived from regions that are relationally close to each region can also be considered. As mentioned, a region can be spatially correlated to other regions within the city via different ways. For example, the neighbourhood of a region comprises of regions which it is geographically directly adjacent to. A region can also be related to another based on whether these two regions are connected by roads, i.e., road connectivity. Finally, a region may also be related to another based on how similar they are in terms of the functions of the facilities or buildings present within these regions, i.e., functional similarity. Both CGRNN and the consideration of multiple network relationships for spatial and temporal correlation modelling are originally proposed in [25].

As depicted in Figure 5.2, we adopt the approach with the above-mentioned pair-wise relationships: 1) neighbourhood, 2) road connectivity, and 3) functional similarity, represented by $G^n = (V, A^n)$, $G^r = (V, A^r)$ and $G^s = (V, A^s)$ respectively. Each entry in A^n takes on the value of 1 (resp. 0) if its two corresponding regions are (resp. are not) directly adjacent to each other. Each entry in A^r takes on the value of 1 (resp. 0) if the two corresponding regions are (resp. are not) connected via roads. Finally, each entry in A^s represents the functional similarity between two corresponding regions, we use the following formulation.

$$A_{u,v}^s = \text{sim}(F_{V_u}, F_{V_v}) \in [0, 1] \quad (5.1)$$

FIGURE 5.2: Multi-contextual-gated RNN

where sim is the similarity function of our choice. F_{V_u} and F_{V_v} are point-of-interest (POI) vectors of the u -th and v -th regions respectively, where $u, v \leq N$. The size of each POI vector is equal to the total number of POI categories available in a city, and each element within a POI vector is the total number of POIs within that region that belongs to that category. We first consider the neighbourhood relationship. We pass the input $X \in \mathbb{R}^{N \times P}$ and A into a global contextualization component. P denotes the total number of historical demands considered for each region at the time period $t + 1$. For the historical demand at each time step, $X^t \in \mathbb{R}^N$, a GCN operation is applied to capture the demand information of the regions surrounding each region using the Chebyshev polynomial basis filters [83] with order $K - 1$, as follows.

$$\tilde{X}^p = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) X^p, \text{ for } p = 1, 2, \dots, P \quad (5.2)$$

where $T_k(\tilde{L})$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{L} = 2L/\lambda_{\max} - I_N$. $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$ is the normalized graph Laplacian. $D \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix, where $D_{c,c} = \sum_d A_{c,d}$. $I_N \in \mathbb{R}^{N \times N}$ is an identity matrix, and $\lambda_{\max}$ corresponds to the largest eigenvalue of L . We then concatenate each $\tilde{X}^t$ with X^t , combine them and derive $\tilde{X} \in \mathbb{R}^{N \times P \times 2}$,

before passing it through a mean pooling layer as follows.

$$Z^p = \frac{1}{N} \sum_{i=0}^N \tilde{X}_i^p, \text{ for } p = 1, 2, \dots, P \quad (5.3)$$

where $\tilde{X}_i^p$ corresponds to the element in $\tilde{X}$ that corresponds to the i -th region at time period p . This pooling is done to get a global summary observation across all regions for each time step. We denote the collection of Z^p across all time steps as Z .

Within a sequence of historical demands at a region, different elements may have different importance in revealing information about the demand of the next time period. For example, the demand during the previous time period may be more important than that of three periods ago if there are spurious events like car accidents. However, the demand at a region over the same period one week ago may sometimes reveal more insights than the demand one period ago, especially during times of the day when demands may fluctuate a lot, e.g., early morning peak-hours. As such, multi-channel attention is applied across the all the time steps to get a weighing based on the global and local demand information, as follows.

$$S_1 = \text{ReLU}(W_1(Z) + b_1) \quad (5.4)$$

$$S = \sigma(W_2(S_1) + b_2) \quad (5.5)$$

where $S \in \mathbb{R}^P$, and W_1 , W_2 , b_1 and b_2 are the parameters of the operation. The activations used are $\text{ReLU}(x) = \max(0, x)$ and the Sigmoid function, $\sigma = \frac{1}{1+e^{-x}}$ respectively. Then, this attention weighting is applied to the input X , as follows.

$$Y^p = X^p S^p, \text{ for } p = 1, 2, \dots, P \quad (5.6)$$

Finally, a recurrent layer is applied to model the temporal correlations of the historical demands across the different time steps at all regions, as follows.

$$H_i = \text{RNN}(Y_i^1, Y_i^2, \dots, Y_i^P), \text{ for } i = 1, 2, \dots, N \quad (5.7)$$

where each region is denoted by i , and the output H_i is the representation vector for the historical demands at that region. We denote the output of the RNN layer for the temporal correlations based on the neighbourhood relationship as

$H^n \in \mathbb{R}^{N \times m_1}$, where m_1 is the dimension of the RNN. In a similar fashion, we perform these operations to obtain the temporal correlations in the context of the two other relationships, i.e., road connectivity and functional similarity, to obtain H^r and H^s respectively.

5.2.3 Mutual-correlated Multi-GCN

Apart from the temporal correlations of historical demands, the demand at a region also has spatial correlations with other regions within the city, depending on the type of pair-wise relationship with those regions. For example, a region may experience similar demand to that of another region because both regions are adjacent to each other geographically and likely to have similar travel patterns, e.g., people traveling out to work in the morning. A region may also exhibit similar demand characteristics as another region if both of them are functionally similar, e.g., industrial hubs across the city, even when both are geographically far apart. To this end, existing work has considered using multi-GCNs to model different pair-wise spatial correlations, by leveraging network relationships of different types [25]. In this work, we propose a multi-GCN model that not only models the spatial correlations across the different regions of a city, but also the mutual correlations between the different pair-wise relationships.

Multi-graph Correlation Layer. To this end, we first pass the output from the Multi-Contextual Gated RNN through a Multi-graph Correlation Layer. The key motivation behind this layer is that the demand characteristics from a type of relationship may be correlated to that of another type, and learning such mutual correlations may result in a stronger accuracy. For example, the demand characteristics at a region may sometimes be very much correlated with that of the neighbouring regions. Since these neighbouring regions are also likely to be functionally similar, the representation vector that corresponds to this region in H^n may experience some correlations with that in H^s . We model such correlations by applying a linear layer across each element of H^n , H^r , and H^s , as follows.

$$U_{i,g} = W_{3,g}[H_{i,g}^n, H_{i,g}^r, H_{i,g}^s] + b_g, \text{ for } g = 1, 2, \dots, m_1 \quad (5.8)$$

where $H_{i,g}^n$, $H_{i,g}^r$ and $H_{i,g}^s$ correspond to the g -th element of the i -th region within H^n , H^r and H^s respectively. $W_{3,g} \in \mathbb{R}^{3 \times 3}$ and b_g are the associated parameters

of this operation. We combine all derived $U_{i,g}$ and denote the collective output of this operation as $U \in \mathbb{R}^{3 \times N \times m_1}$. Then, we split U into three parts along the first axis to get U^n , U^r , and U^s .

Spatial Correlations Modeling. To model the spatial correlations across the regions of the city, we apply a sequence of GCN layers to each of U^n , U^r , U^s . The idea is similar in spirit to that of Section 5.2.2, except for the differences in the dimensions of the input and output. Specifically, in the first GCN layer, the size of input (resp. output) is m_1 (resp. $m_2 \in \mathbb{R}$). The formulation, in the case of U^n , based on the extension of the equation in Section 5.2 to multi-dimensional tensors is as follows.

$$J_h^n = \sum_{l=1}^{m_1} \sum_{k'=0}^{K'-1} \theta_{k',l,h} T_{k'}(\tilde{L}) U_l^n, \text{ for } h = 1, 2, \dots, m_2 \quad (5.9)$$

with the $m_1 \times m_2 \times K'$ vectors of the Chebyshev coefficients $\theta_{k',l,h}$. U_l^n is the l -th element of the vector U^n . $J^n \in \mathbb{R}^{N \times m_2}$ is the feature vector output of this operation that considers spatial correlations between a region and the regions away from it. A key characteristic of GCN operations is that for each region, the demand characteristics of regions which it has a closer relationships with tend to be weighed higher than that of other regions. The intuition is that by focusing more on these relationally-close regions, the derived feature vector may more accurately represent the demand characteristics of this region. However, the strong correlations with such regions may not always be true in reality. For example, a neighbouring region may be functionally different and thus exhibit different demand characteristics from the region at concern. It may then not be accurate to weigh this region higher. As such, we pass J^n through a linear layer for a re-weighing of importance of the information derived from the related regions, before passing it through another GCN layer that is structurally similar to the one before. Similarly, we perform the above operations for the road-connectivity, and functional similarity relationships, and obtain the outputs J^r , J^s respectively.

5.2.4 Attribute Guidance

The state of the art tends to leverage mostly historical demands and the network relationships across the regions for prediction. However, it may not always be accurate to consider only historical demands for the prediction of future demands.

For example, 10 am of a weekday usually corresponds to the off-peak hours in some cities. If we were to predict the demand from 10 to 10:30 am based solely on the demands over the past time periods which correspond to the morning peak hours, the results may be largely inaccurate. To resolve this issue, we propose for the first time an Attribute Guidance module that leverages high-level attributes to guide the prediction process.

Specifically, these attributes can be classified into two classes: temporal and spatial ones. For temporal attributes, we consider the period of day depending on the time frame we are looking at, e.g., 0-48 in the case of 30-minutes time frames. We also consider the day of the week, i.e., 1-7. The rationale behind including such temporal information is to account for the periodicities of the demands. For example, the demand during the morning peak hours at a residential region may generally be higher than that during the non-peak hours. However, the demand may not be the same during the same time on a weekend. Including such information may also help to resolve the issue highlighted above, since the day of week and period of day are pieces of information that are unaffected by historical demands. For spatial attributes, we consider a numerical ID for each region of the city. The rationale behind including this feature is that the demands at a region may be very different from those of another region. For example, the demands within a shopping district throughout the day are likely to be very different from the demands at a residential one.

To model the insights from these attributes, we design our module as shown in Figure 5.1. Specifically, we first embed each of these attributes into a higher-dimension vector using the condensed word vector approach in natural language processing. Certain attributes are relatively more important than others on some occasions in revealing the demand at a region. For example, period of day is generally important in determining the demand across the day. However, this may not be as true during the weekends when the periodicity is not as strong as that of weekdays. To account for such differences in relative importance across attributes, we propose an Attribute Attention layer, where we first convert the embedded vectors derived earlier into vectors of the same size via a linear layer, and denote these vectors as r_c , r_p , and r_d respectively. Then, we pass these vectors through an

attention mechanism designed as follows.

$$z_q = r_q s_q \quad (5.10)$$

$$\alpha_q = \frac{\exp(z_q)}{\sum_{q \in Q} \exp(z_q)} \quad (5.11)$$

$$J_{\text{attr}} = \sum_{q \in Q} \alpha_q r_q, \quad (5.12)$$

where α_q is the weight for the attribute vector r_q , $Q = \{c, p, d\}$. Parameter s_q is used to calculate the intermediate score z_q . We characterize our attention mechanism with $\{r_q\}$ and $\{s_q\}$, and J_{attr} denotes the output of our attention mechanism and this module.

5.2.5 Prediction Module

Finally, we design a module to combine the information of the previous stages. To integrate the relationship-specific information learnt in J^n , J^r , and J^s , we apply mean-pooling across the three representation vectors to get an overall representation of the information derived from the three relationships, $\tilde{J}$. Then, we concatenate the outputs from the Attribute-guidance module and Mutual-correlated Multi-GCN module, before passing the results through a non-linear layer, as follows.

$$X_i^{t+1} = \text{ReLU}(W_f[J_{\text{attr}}, \tilde{J}_i] + b_f), \text{ for } i = 1, 2, \dots, N \quad (5.13)$$

where ReLU is the activation function, and W_f and b_f are the associated parameters of this operation. $\tilde{J}_i$ is the element in $\tilde{J}$ that corresponds to the i -th region. Since the high-level attributes across all regions are the same for each input X , we concatenate each $\tilde{J}_i$ with J_{attr} .

Training Objective. To train the model such that it effectively predicts not only typical demand values, but also extremely large and small ones, we apply the root-mean-squared error loss. Specifically, the training objective is as follows.

$$L = \sqrt{\sum_{i=1}^N X_i^{t+1} - R_i^{t+1}}, \quad (5.14)$$

where X_i^{t+1} corresponds to the estimated next-period demand for the i -th region, and R_i^{t+1} is its associated ground-truth value.

5.3 Experiments

To evaluate the effectiveness of our proposed method, we conduct experiments on a real-world dataset. Specifically, we focus on the application of estimating taxi and bike demands at each region over the next time period.

5.3.1 Experimental Set-up

This section describes the dataset used, the baselines we consider for comparisons, and the evaluation metrics.

Datasets. For the validation process, we leverage origin-destination records of taxis within the New York City (NYC), US. Since the other open source dataset available is the NYC Bike Dataset, we also consider it for experiments. Some properties of these datasets are as follows.

- **NYC Taxi Dataset:** This dataset contains taxi passenger trip records from January and February 2015, obtained from NYC Open Data. Each trip record consists of the pick-up and drop-off locations and date/time, as well as the GPS coordinates of these pick-up and drop-off locations. Altogether, there are about 25M trip records in this dataset.
- **NYC Bike Dataset:** This dataset contains bicycle trip records from June to October 2015, also obtained from NYC Open Data. Each trip record consists of the docking stations and date/time at which the bike rider begins and ends his/her trip, as well as the GPS coordinates of these pick-up and drop-off locations. Altogether, there are about 2.1B trip records.

Since the NYC Taxi Dataset consists of only the information about individual trips within the city, we conduct some processing to get the aggregated demand of every region during every period of the two months. To this end, we define the regions considered for this city, by breaking the New York City map into 10×20 grids

TABLE 5.1: Demand Dataset Characteristics

Statistical Index	NYC Taxi	NYC Bike
Mean Periodic Demand	41.8	14.1
10%ile Periodic Demand	0	0
90%ile Periodic Demand	72.0	33.0
Standard Deviation	157.8	50.7

FIGURE 5.3: Taxi and Bike Demand across 4-hours Periods of the Day (Weekdays). 1 denotes the 0-4 hours of the day, while 6 denotes the 20-24 hours of the day.

FIGURE 5.4: Taxi and Bike Demand across 4-hours Periods of the Day (Weekends). 1 denotes the 0-4 hours of the day, while 6 denotes the 20-24 hours of the day.

and assign each passenger trip to its respective region. We also consider the time period of 30 minutes intervals, i.e., a total of 48 time periods in a day. Then, we sort the trips of each region and aggregate the total number of trips during each time period. We consider the demand data from the first five weeks of this processed dataset as the training dataset, that of the third last week as validation dataset, and the remaining as testing dataset. We denote these datasets as Taxi-Training, Taxi-Validate, and Taxi-Testing respectively. In a similar fashion, we conduct the similar processing for NYC Bike Dataset and derive Bike-Training, Bike-Validate, and Bike-Testing, while considering the time period of 60 minutes intervals, i.e., a total of 24 time periods in a day.

Demand Dataset Characteristics. Table 5.1 shows some statistics of the demand dataset. As observed, the average demands for NYC Taxi and Bike datasets are 41.8 and 14.1 respectively. The 10%ile for both datasets are 0, showing that a significant part of periods within the regions considered in the datasets does not

FIGURE 5.5: Average daily Taxi and Bike Demand across Different Days of the Week. 1 denotes Monday, while 7 denotes Sunday.

see any demands for taxis and bikes. This is because there is a large part of the day that falls under the non-waking hours, when people generally demand much fewer transportation of any sorts. Furthermore, the higher values of 72 and at the 90%ile and the standard deviation of 157.8 show that the demands fluctuate greatly across different periods and regions of the city. Figure 5.3 also shows the normalized demand for taxis and bikes in the two months over 4-hours time periods of weekdays, while Figure 5.4 shows this demand over the different periods of the day on weekends. In general, the demand for taxis is larger than that of bikes across different periods of the day and days of the week. Both the demand for taxis and that for bikes in the early hours of weekdays tend to be lower than that of the later parts of the day. Furthermore, there is an increasing trend in the demand throughout the day. The high demands at the 8-16 hrs period are likely due to people doing work-related travels, while those of the later time periods are likely due to people traveling back home after work or traveling for post-work leisure activities. In contrast, the only time period when a low demand for taxis is observed on weekends is during the late hours of the night, i.e., 4-8 hrs. The demand for taxis at 0-4 hrs is much higher than that during the weekdays because people are generally more willing to travel for late-night events then, since they are less tied up by work schedules on weekends. However, the demand during the similar period for bikes is low, since people may not ride bikes at those hours due to safety concerns. Figure 5.5 shows how the average daily demand varies across different days of the week, where 1 denotes Monday and 7 denotes Sunday. As observed, the demand does not vary too much across the different days and the demand for

taxi on Saturdays on average is the highest.

Evaluation Metrics. For our experiments, we adopt the following evaluation metrics: root mean square error (RMSE) and mean absolute error (MAE).

Implementation Details and Parameter Setting. We implement our model on Keras. Our model training/evaluation is done on Nvidia Tesla M40 GPU, and Linux machine with an Intel Xeon 2.6GHz CPU and 64GB RAM. We use Adam optimization algorithm to fine-tune the parameters. The learning rate we use is 0.001. For the historical observations, we consider $P = 5$ historical observations. Similar to [59] and [25], these observations include 3 latest closeness components, 1 period component and 1 latest trend component. In our work, these respectively correspond to the demand at the past 3 time periods, the demand at the same time period one day ago, and the demand at the same period one week ago. For hyper-parameters settings, K and K' are set as 3. m_1 , the size of RNNs considered is set as 64. We also use two stacked RNNs to allow greater model complexity and performance. Similarly, m_2 , the size of GCNs within the Mutual-correlated Multi-GCN module is also set as 64. For the similarity function used in G_s , we apply the Euclidean distance similarity. Within the Attribute-directing module, the time period, day of week, and region ID attributes are embedded into $\mathbb{R}^5$, $\mathbb{R}^3$ and $\mathbb{R}^8$ vectors respectively. The size of the attention mechanism considered is 64.

Baselines. We consider several baselines in our experiment. In these cases, the demand records within Taxi-Training and Bike-Training are used for training, while those within Taxi-Validate and Bike-Validate, as well as Taxi-Testing and Bike-Testing, are used for validation (wherever applicable) and testing respectively. The baselines are as follows:

- **LASSO, Ridge Regression:** These methods take the historical demands over a few time periods and feed them into a linear regression model, with regularization terms. The historical demands considered at each region and time period are the same as those of AM-Net. Furthermore, the high level attributes are encoded via one-hot encoding.
- **At-MLP:** This method leverages a simple three-layers Multi-layer Perception and models the historical demand data, as well as the high level attributes. The input for this method is the same as LASSO, Ridge Regression.

- **LightGBM:** Gradient-boosting machines have always been highly effective for classification and regression problems. We compare with LightGBM [80], which is one of the most effective one within this class of methods. Specifically, we run LightGBM with the number of iterations as 1000, and number of leaves as 31.
- **ST-GCN:** This method is a state of the art for traffic prediction [26]. It leverages GCNs with temporal gates to model the spatial correlations across regions, while considering information from the temporal correlations of historical demands.
- **ST-MGCN:** ST-MGCN is the state of the art for the demand prediction problem, and is the best performing to date for single-step prediction [25]. Instead of considering GCNs on solely Euclidean-based pair-wise relationships across regions as in ST-GCN, this method also leverages pair-wise relationships of other types, i.e., road connectivity and functional similarity relationships. It also considers contextual information in the temporal modeling of the historical demands across the regions.
- **AM-Net-a:** This method is a variant of AM-Net. In this method, we remove the Attribute-directing module, to see how the model fares when high-level attributes are totally disregarded in the estimation.
- **AM-Net-b:** In this method, instead of using the attention layer to learn the relative importance across the attributes, we simply concatenate the embedded attributes r_c , r_p , and r_d and consider the resulting vector as the output of the Attribute-directing module. The purpose is to see how our method fare when there is a consideration of estimation directing via high level attributes, albeit in a naive manner.
- **AM-Net-c:** Here, we remove components that learn the mutual correlations across the pair-wise relationships, to see how the AM-Net fares when no mutual correlations are considered. Specifically, instead of using the Mutual-correlated Multi-graph Convolutional module, we replace for each pair-wise relationship a sequence of three GCNs.

TABLE 5.2: RMSE and MAE on AM-Net and the baselines. Best results are in bold face.

	NYC Taxi		NYC Bike	
Method	RMSE	MAE	RMSE	MAE
Lasso Regression	19.36	5.53	14.65	3.58
Ridge Regression	19.34	7.01	14.57	4.45
LightGBM	18.42	5.07	12.20	2.89
At-MLP	18.25	6.28	11.48	3.44
ST-GCN	23.27	6.18	13.70	3.937
ST-MGCN	17.36	4.87	12.18	2.93
AM-Net-a	16.95	<u>4.82</u>	11.43	<u>2.84</u>
AM-Net-b	16.66	4.93	12.04	3.01
AM-Net-c	<u>16.51</u>	4.92	10.84	2.74
AM-Net	16.02	4.79	<u>11.15</u>	2.85

5.3.2 Evaluation Results

To ensure the fairness of the experiments, we conduct the experiments of our proposed approach and the baselines across three pre-defined seeds, except when it is not applicable, e.g., Ridge Regression. As observed, the accuracy for AM-Net is the highest among all compared methods. In the case of NYC Taxi, it is at least 5% stronger than the next best-performing method, ST-MGCN. In the case of NYC Bike, the improvement over ST-MGCN is about 8%. This demonstrates the effectiveness of considering high level attributes for directing the estimation of vehicle demands, while modeling the temporal correlations associated with historical demands. It also demonstrates the effectiveness of integrating the correlations between the different pair-wise relationships across regions. When the correlations across the high-level attributes are considered simply by a concatenation instead of an attention mechanism in AM-Net-b, the accuracy worsens by about 0.64 in terms of RMSE in the case of NYC Taxi dataset, and 0.89 in the case of NYC Bike dataset. This demonstrates the effectiveness in integrating an attention mechanism to consider the various relative importance across these attributes under different circumstances. Furthermore, AM-Net-c performs worse than AM-Net by about 0.49 in terms of RMSE for NYC Taxi. This demonstrates the importance of considering the mutual relationships across the graphs within the Mutual-correlated Multi-GCNs. However, it is interesting to note that this improvement is not observed for NYC Bike. A possible reason may be that AM-Net is designed with focus

on vehicles like taxis, via considerations like road connectivity, which may not be as applicable for bikes. For example, bikes may travel on smaller roads/pavements that are not on road maps to get from one place to another. Furthermore, the characteristics of a bike trip is also vastly different from that of taxi ones, e.g., a bike trip tends to be shorter in distance. Finally, when the Attribute-directing module is removed in AM-Net-a, the RMSE also worsens significantly for NYC Taxi data and to a fair degree for NYC Bike data. This highlights the importance of using high-level attributes to direct the prediction.

5.3.3 Relevance of High-level Attributes

To test the effectiveness of each of the high-level attributes, we evaluate the accuracy of the AM-Net when only that attribute is removed. As mentioned, since our model is more focused on vehicles like taxis, we evaluate on NYC Taxi. We observe that when the region ID is removed, the accuracy worsens to about 17.14 and 5.27 in terms of RMSE and MAE respectively, i.e., about 7% degradation with respect to RMSE. When the period of day is removed, the accuracy worsens to 18.02 and 5.33 respectively, which is about 12% degradation with respect to RMSE. We note that this accuracy is worse than when the Attribute-directing module and all the high-level attributes are removed. This demonstrates that if the set of high-level attributes is not chosen suitably, it does more harm than good to the modeling of the multi-nature dependencies. It further highlights the necessity for the careful choice of the right attributes to be included in the model. Finally, when the day of week is removed, the accuracy worsens to 16.19 in terms of RMSE. The degradation is the smallest as compared to the other two attributes. A likely reason may be that while the purpose of the day of week attribute is to account for the weekly temporal periodicity, there already exists plenty of other features that provide similar information within the model. One example is the historical demand at a region one week ago, which is provided as an input to the model. As such, the inclusion of the day of week attribute may provide some but lesser information as compared to the other two attributes.

5.3.4 Performance across Different Hyper-parameters

We also evaluate how AM-Net performs on the NYC Taxi dataset across attention mechanisms of different size, as well as how it performs when the size of the GCNs in the Mutual-correlated Multi-GCN module is changed. As shown in Figure 5.6, the performances vary to some extent across attention mechanism of different sizes. The error is the lowest when the size is 16 to 32. However, it reaches a peak of almost 17.4 when the size is 48, before dropping to relatively lower values when the sizes are 64 and 80. This shows that even when high-level attributes are used to direct the estimation, fine-tuning is very important to effectively capture the correlations of these additional features to improve the estimation. Likewise, the performance of AM-Net varies with respect to the size of GCNs considered in the Mutual-correlated Multi-GCNs. As shown in Figure 5.7, when the size considered is either too small, i.e., 32, or too large, i.e., 160, the accuracy is severely dampened. For all the other sizes in between that are considered, satisfactory accuracies are achieved. This also highlights the importance of tuning parameters in order for GCNs to achieve their potential in effectively modeling spatial correlations across regions to improve estimation.

5.3.5 Performance across Different Time Periods

We also evaluate how our method performs on NYC Taxi dataset over different time periods of the day, for both weekdays and weekends. We conduct the test separately for weekdays and weekends since the travel characteristics for weekdays are different from those of weekends. We consider 4 hour time periods and the results are shown in Figure 5.8 and Figure 5.9 respectively. As observed, the performance of AM-Net mostly outperforms the performance of the two next-best methods across all periods of the day on both weekdays and weekends. In general, there are some rough similar trends in terms of the errors for the three different methods. For example, in the case of weekdays, there is an rising trend in terms of errors from the 0 to 12 hours of the day, followed by a small dip at the 12-16 hours time period, before another rising trend for the rest of the day. On weekdays, a possible reason why the errors at the 8-12 and 20-24 periods are mostly the highest may be because these are the time periods where travel has the most irregularities. For example, these periods correspond to the morning peak hours and lunchtime,

FIGURE 5.6: RMSE w.r.t. Size of Attention

FIGURE 5.7: RMSE w.r.t. Size of GCN

and post-work hours of the day, respectively. Both of these are periods when people travel the most and may exhibit much irregularities in travel. For example, people may not follow the regular travel schedules as much during lunchtime as compared to other periods like the morning peak hours, i.e., before 8, which see most people going to the same location for work.

One interesting thing to note is that the overall trends of errors throughout the day for both weekdays and weekends show some similarities to those of Figure 5.3 and Figure 5.4. For example, both Figure 5.8 and Figure 5.3 exhibit mostly increasing trends throughout the periods of the day, while both Figure 5.9 and Figure 5.4 show a dip at period 2, i.e., 4-8 hours of the day, when the demand is generally the lowest. While there are some exceptions, e.g., lunchtime of weekdays due to the reasons mentioned above, it demonstrates to some extent that it tends to be more difficult to perform accurate estimations when there are more people demanding taxi-rides than when there are less. Note that in the case of weekends, the errors of AM-Net are higher than those of LightGBM for two particular time periods. A possible reason may be that deep learning usually performs better when there is a larger training dataset. However, the training dataset from weekends is smaller in our case as there are fewer weekend days in a week than weekdays. As such, the performance of AM-Net on weekends may not always be relatively better than the baselines.

We also conduct this comparison across the different days of the week. As observed, AM-Net largely outperforms ST-MGCN and LightGBM throughout the days of the week. In terms of the trends in the errors, the three methods also demonstrate large

FIGURE 5.8: RMSE across 4-hours Periods of the Day (Weekdays).

FIGURE 5.9: RMSE across 4-hours Periods of the Day (Weekends).

FIGURE 5.10: RMSE across different days of the week

degrees of similarities, e.g., falling trend from Monday to Wednesday, and rising trend from Wednesday to Thursday. We observe that the RMSE of AM-Net on Saturdays tends to fare slightly worse than LightGBM. This may possibly be due to the same reason highlighted earlier, i.e., a smaller weekend dataset that prevent deep learning techniques from performing as well.

5.3.6 Training and Testing Time

Despite the longer training time of a few hours for AM-Net, the prediction time is fast, taking about an average of 0.0876s to complete a single round of prediction for all the regions within the NYC dataset.

5.3.7 Application to other Problems and Limitations

We have demonstrated with experiments the superior performance of AM-Net over state of the art for the demand prediction problem. However, in general, this model and the insights obtained through this work could also be applied to various other spatiotemporal problems. This is especially so given that several other time series problems like vehicle supply prediction are also heavily dependent on the similar factors that affect the demand of vehicles. For example, the supplies of empty taxis of a region are very much associated with temporal dependencies like day of week and period of day, e.g., downtown of a city is likely to see a larger supply of empty taxis during the morning peak hours due to these taxis dropping off their passengers who are going to work. A limitation that this work faces is that while real-time conditions are considered in the estimation, such data comes with a lag factor. For example, suppose that a large-scale event like a music concert has just ended at a region. When that happens, methods like AM-Net and existing work in general will not be able to pick up this information immediately, since the historical demand data that is used as input does not include the current demand until at least a time period has passed. This may be a severe issue if the time periods considered are large, e.g., intervals of 1 hr and above. As such, it would be more accurate if we could integrate real-time information in a more timely fashion. Some potential ideas include incorporating real-time social network information, e.g., tweets, or leveraging pre-planned concert schedules.

Chapter 6

Conclusion

In this dissertation, we propose several approaches for estimation of travel time and demand, which play very important roles in smart cities. Information about the travel time and demand is very valuable to many. From the perspectives of transport operators, e.g., taxi or ride-hailing companies, such information encourages a more effective allocation of existing resources or planning of routes, which then improves the travel experience of commuters. Through the lens of public organizations, such information helps in the understanding of the traffic conditions within the transportation network, which in turn supports urban planning. Finally, such information also helps commuters to understand the travel time of routes they take and plan their trips more effectively.

Chapter 3 presents a new solution to inferring travel time in Singapore’s metro network using only smart card data. The key idea of the solution is to (i) divide the time variables in the network into different categories and estimate them separately, and (ii) utilize heuristics and lognormal mixture models to mitigate the complexities caused by heterogeneous route choices by the passengers. We evaluate the solution in the context of route planning, and show that it significantly outperforms the baseline approaches in terms of accuracy.

Chapter 4 proposes an Attribute-related Hybrid Trajectories Network capable of estimating path travel times by effectively utilizing hybrid trajectory data. This method implicitly models the correlations across hybrid trajectories, something largely ignored in existing work. It also extracts the correlations of attributes that affect path travel time and leverage on-road information to extract sub-path

features. Through experiments, we demonstrate the superior performance of our method over the state of the art methods.

Chapter 5 of the thesis proposes an Attribute-guided Multi-graph Convolution Network capable of estimating future demand for ride-hailing vehicles or taxis at regions of a city by effectively utilizing historical demands, pair-wise relationships across regions of the city, and high-level spatial and temporal attributes. This method implicitly models the correlations across the different pair-wise relationships across regions of the city. It also guides the estimation by considering high-level attributes that address the limitations brought about by considering solely historical demands for estimation. Through experiments, we demonstrate the superior performance of our solution over the state of the art methods.

In general, travel time and vehicle demand are pieces of information that are very important for urban mobility in smart cities. As future work, we may consider leveraging the inferred travel time information to analyze passengers' route choices within the metro and road networks, as well as investigate how passenger route choices would change when there are new train or bus lines added into the transportation network. Furthermore, we may also consider for future work an extension to trajectory classification problems, e.g., whether a trajectory corresponds to a vehicle carrying a heavy load or not.

Apart from travel time and vehicle demand, there is certain other information that is valuable to many within smart cities, yet challenging to estimate. For example, predicting the future trajectories of vehicles is also an important task that helps transport planners or public organizations to more strongly understand where the supply of vehicles will be in the future. CrowdfLOW prediction also helps public organizations to understand where the traffic bottlenecks within a city may be, and assists marketing businesses in knowing where to plant their advertisements. As such, some potential future work also includes leveraging data-driven computational methods to solve these problems, while leveraging the insights obtained through the course of this dissertation. For example, instead of considering solely single source trajectories, we can leverage hybrid trajectories for a more comprehensive understanding of crowdfLOWS of different natures, e.g., passengers from different vehicle types.

Bibliography

- [1] Grant View Research. Smart cities market size worth \$237.6 billion by 2025. <https://www.grandviewresearch.com/press-release/global-smart-cities-market/>, 2019. [Online; accessed 12-January-2020]. [1](#)
- [2] 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/>, 2018. [Online; accessed 12-January-2020]. [1](#)
- [3] Norman Akhtar and Kevin Hasley. Smart cities face challenges and opportunities. <https://www.computerweekly.com/opinion/Smart-cities-face-challenges-and-opportunities/>, 2018. [Online; accessed 12-January-2020]. [2](#)
- [4] Dia Hussein. The smart future of urban mobility. <https://www.smartcitiesdive.com/ex/sustainablecitiescollective/smart-future-urban-mobility/1049321/>. [Online; accessed 12-January-2020]. [3](#)
- [5] Juanjuan Zhao, Fan Zhang, Lai Tu, Chengzhong Xu, Dayong Shen, Chen Tian, Xiang-Yang Li, and Zhengxi Li. Estimation of passenger route choice pattern using smart card data for complex metro systems. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):790–801, 2017. [3](#), [7](#), [11](#), [13](#), [21](#)
- [6] Xinyue Xu, Liping Xie, Haiying Li, and Lingqiao Qin. Learning the route choice behavior of subway passengers from afc data. *Expert Systems with Applications*, 95:324–332, 2018. [7](#), [22](#), [36](#)
- [7] Mohadeseh Rahbar, Mark Hickman, Mahmoud Mesbah, and Ahmad Tavassoli. Calibrating a bayesian transit assignment model using smart card data. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–10, 2018.
- [8] Lijun Sun, Yang Lu, Jian Gang Jin, Der-Horng Lee, and Kay W Axhausen. An integrated bayesian approach for passenger flow assignment in metro networks. *Transportation Research Part C: Emerging Technologies*, 52:116–131, 2015. [12](#), [13](#)

- [9] Yong-Sheng Zhang and En-Jian Yao. Splitting travel time based on afc data: estimating walking, waiting, transfer, and in-vehicle travel times in metro system. *Discrete Dynamics in Nature and Society*, 2015, 2015. [3](#), [7](#), [12](#), [13](#), [22](#), [36](#)
- [10] Dong Wang, Wei Cao, Mengwen Xu, and Jian Li. Etcps: An effective and scalable traffic condition prediction system. In *International Conference on Database Systems for Advanced Applications*, pages 419–436. Springer, 2016. [3](#), [14](#), [45](#)
- [11] Erik Jenelius and Haris N Koutsopoulos. Urban network travel time prediction based on a probabilistic principal component analysis model of probe data. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):436–445, 2018. [8](#), [14](#)
- [12] Mahmood Rahmani, Haris N Koutsopoulos, and Erik Jenelius. Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach. *Transportation Research Part C: Emerging Technologies*, 85: 628–643, 2017. [8](#), [14](#)
- [13] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. Traffic speed prediction and congestion source exploration: A deep learning method. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 499–508. IEEE, 2016. [14](#), [45](#)
- [14] Jian Dai, Bin Yang, Chenjuan Guo, Christian S Jensen, and Jilin Hu. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment*, 10(3):85–96, 2016. [14](#), [15](#), [45](#)
- [15] Yang Li, Dimitrios Gunopulos, Cewu Lu, and Leonidas Guibas. Urban travel time prediction using a small number of gps floating cars. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 3. ACM, 2017. [3](#), [14](#), [15](#), [45](#)
- [16] Yanshuo Sun and Ruihua Xu. Rail transit travel time reliability and estimation of passenger route choice behavior: Analysis using automatic fare collection data. *Transportation Research Record: Journal of the Transportation Research Board*, (2275):58–67, 2012. [7](#), [11](#), [13](#), [21](#)
- [17] Feng Zhou and Rui-hua Xu. Model of passenger flow assignment for urban rail transit based on entry and exit time constraints. *Transportation Research Record: Journal of the Transportation Research Board*, (2284):57–61, 2012. [7](#), [11](#), [13](#), [21](#)
- [18] Hasan Poonawala, Vinay Kolar, Sebastien Blandin, Laura Wynter, and Sambit Sahu. Singapore in motion: Insights on public transport service level through farecard and mobile data analytics. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 589–598. ACM, 2016. [7](#), [11](#), [21](#)

- [19] Lijun Sun, Der-Horng Lee, Alex Erath, and Xianfeng Huang. Using smart card data to extract passenger's spatio-temporal density and train's trajectory of mrt system. In *Proceedings of the ACM SIGKDD international workshop on urban computing*, pages 142–148. ACM, 2012. [7](#), [12](#), [21](#), [36](#), [37](#), [38](#)
- [20] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. *arXiv preprint arXiv:1802.02147*, 2018. [8](#), [15](#), [46](#)
- [21] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. AAAI, 2018. [8](#), [15](#), [46](#), [51](#), [53](#), [54](#), [56](#), [60](#), [61](#)
- [22] Xiaolong Li, Gang Pan, Zhaohui Wu, Guande Qi, Shijian Li, Daqing Zhang, Wangsheng Zhang, and Zonghui Wang. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science*, 6(1):111–121, 2012. [9](#), [16](#), [69](#)
- [23] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013. [16](#), [69](#)
- [24] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [16](#), [69](#)
- [25] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *2019 AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019. [9](#), [16](#), [17](#), [69](#), [74](#), [77](#), [84](#), [85](#)
- [26] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017. [9](#), [17](#), [70](#), [85](#)
- [27] Mohadeseh Rahbar, Mahmoud Mesbah, Mark Hickman, and Ahmad Tavasoli. Determining route choice behaviour of public transport passengers using bayesian statistical inference. In *Australasian Transport Research Forum (ATRF), 38th, 2016, Melbourne, Victoria, Australia*, 2016. [12](#), [13](#)
- [28] Qian Fu, Ronghui Liu, and Stephane Hess. A bayesian modelling framework for individual passenger's probabilistic route choices: a case study on the london underground. Leeds, 2014. [13](#), [29](#)
- [29] Christopher Monterola, Erika Fille Legara, Di Pan, Kee Khoon Lee, and Gih Guang Hung. Non-invasive procedure to probe the route choices of commuters in rail transit systems. *Procedia Computer Science*, 80:2387–2391, 2016. [13](#), [29](#)

- [30] Sung-Pil Hong, Yun-Hong Min, Myoung-Ju Park, Kyung Min Kim, and Suk Mun Oh. Precise estimation of connections of metro passengers from smart card data. *Transportation*, 43(5):749–769, 2016. [13](#)
- [31] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604. IEEE, 2012. [14](#), [16](#)
- [32] Bin Yang, Chenjuan Guo, and Christian S Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proceedings of the VLDB Endowment*, 6(9):769–780, 2013. [14](#), [45](#)
- [33] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. T-drive: Enhancing driving directions with taxi drivers’ intelligence. *IEEE Trans. Knowl. Data Eng.*, 25(1):220–232, 2013. [14](#), [45](#)
- [34] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34. ACM, 2014. [14](#), [45](#)
- [35] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *International Conference on Knowledge Discovery and Data Mining, (KDD)*, 2018. [15](#)
- [36] Jilin Hu, Chenjuan Guo, Bin Yang, Christian S Jensen, and Lu Chen. Recurrent multi-graph neural networks for travel cost prediction. *arXiv preprint arXiv:1811.05157*, 2018. [15](#)
- [37] Dong Wang, Wei Cao, Jian Li, and Jieping Ye. DeepSD: supply-demand prediction for online car-hailing services using deep neural networks. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 243–254. IEEE, 2017. [16](#)
- [38] Fei Wu, Hongjian Wang, and Zhenhui Li. Interpreting traffic dynamics using ubiquitous urban data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 69. ACM, 2016. [16](#)
- [39] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1525–1534. ACM, 2016. [16](#)
- [40] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1653–1662. ACM, 2017. [16](#)

- [41] Lei Bai, Lina Yao, Salil Kanhere, Xianzhi Wang, Quan Sheng, et al. Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. *arXiv preprint arXiv:1905.10069*, 2019. [16](#), [17](#)
- [42] Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Michael Chen. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies*, 85:591–608, 2017. [16](#)
- [43] Xian Zhou, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. Predicting multi-step citywide passenger demands using attention-based neural networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 736–744. ACM, 2018. [16](#), [17](#)
- [44] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. Modeling spatial-temporal dynamics for traffic prediction. *arXiv preprint arXiv:1803.01254*, 2018. [17](#)
- [45] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017. [17](#)
- [46] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 777–785. SIAM, 2017. [17](#)
- [47] Daming Li, Yongjie Lin, Xinliang Zhao, Hongjun Song, and Nan Zou. Estimating a transit passenger trip origin-destination matrix using automatic fare collection system. *Database systems for adanced applications*, pages 502–513, 2011. [18](#)
- [48] Marcela A Munizaga and Carolina Palma. Estimation of a disaggregate multi-modal public transport origin–destination matrix from passive smartcard data from santiago, chile. *Transportation Research Part C: Emerging Technologies*, 24:9–18, 2012.
- [49] James Barry, Robert Freimer, and Howard Slavin. Use of entry-only automatic fare collection data to estimate linked transit trips in new york city. *Transportation Research Record: Journal of the Transportation Research Board*, (2112):53–61, 2009.
- [50] Marcela Munizaga, Flavio Devillaine, Claudio Navarrete, and Diego Silva. Validating travel behavior estimated from smartcard data. *Transportation Research Part C: Emerging Technologies*, 44:70–79, 2014.
- [51] Azalden A Alsger, Mahmoud Mesbah, Luis Ferreira, and Hamid Safi. Use of smart card fare data to estimate public transport origin–destination matrix. *Transportation Research Record: Journal of the Transportation Research Board*, (2535):88–96, 2015.

- [52] António A Nunes, Teresa Galvão Dias, and João Falcão e Cunha. Passenger journey destination estimation from automated fare collection system data using spatial validation. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):133–142, 2016. [18](#)
- [53] Xiaolei Ma, Yao-Jan Wu, Yinhai Wang, Feng Chen, and Jianfeng Liu. Mining smart card data for transit riders’ travel patterns. *Transportation Research Part C: Emerging Technologies*, 36:1–12, 2013. [18](#)
- [54] Sang Gu Lee and Mark Hickman. Trip purpose inference using automated fare collection data. *Public Transport*, 6(1-2):1–20, 2014. [18](#)
- [55] Andy Yuan Xue, Jianzhong Qi, Xing Xie, Rui Zhang, Jin Huang, and Yuan Li. Solving the data sparsity problem in destination prediction. *The VLDB Journal—The International Journal on Very Large Data Bases*, 24(2):219–243, 2015. [18](#)
- [56] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. Trajectory-driven influential billboard placement. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2748–2757. ACM, 2018. [19](#)
- [57] Lei Li, Kai Zheng, Sibao Wang, Wen Hua, and Xiaofang Zhou. Go slow to go fast: minimal on-road time route scheduling with parking facilities using historical trajectory. *The VLDB Journal—The International Journal on Very Large Data Bases*, 27(3):321–345, 2018. [19](#)
- [58] Kien Nguyen, Jingyun Yang, Yijun Lin, Jianfa Lin, Yao-Yi Chiang, and Cyrus Shahabi. Los angeles metro bus data analysis using gps trajectory and schedule data (demo paper). In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 560–563. ACM, 2018. [19](#)
- [59] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017. [19](#), [56](#), [84](#)
- [60] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *IJCAI*, volume 16, pages 2618–2624, 2016. [19](#)
- [61] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. [19](#)
- [62] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

- [63] Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, and Feng Zhu. On prediction of user destination by sub-trajectory understanding: A deep learning based approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1413–1422. ACM, 2018.
- [64] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. *arXiv preprint arXiv:1812.04767*, 2018.
- [65] Yuexin Ma, Xinge Zhu, Sibozhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. *arXiv preprint arXiv:1811.02146*, 2018. 19
- [66] Xi Lin, Xiaokui Xiao, and Zengxiang Li. A scalable approach to inferring travel time in singapore’s metro network using smart card data. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8. IEEE, 2018. 21
- [67] Transport services. available online: https://www.sbstransit.com.sg/transport/trpt_nel_1st.aspx (accessed on 21 november 2018). 43
- [68] Xi Lin, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S Bhowmick. Path travel time estimation using attribute-related hybrid trajectories network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1973–1982. ACM, 2019. 45
- [69] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 53
- [70] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016. 53
- [71] Youze Tang, Andy Diwen Zhu, and Xiaokui Xiao. An efficient algorithm for mapping vehicle trajectories onto road networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 601–604. ACM, 2012. 53
- [72] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 53
- [73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 53
- [74] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 53

- [75] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 54
- [76] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 55
- [77] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 56
- [78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 56
- [79] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 56
- [80] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017. 60, 85
- [81] Google. Google maps platform, 2019. URL <https://developers.google.com/maps/documentation/>. 62
- [82] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):19, 2019. 62
- [83] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016. 75

Appendix A

Publications

A.1 Publication for Chapter 3

A Scalable Approach to Inferring Travel Time in Singapore's Metro Network using Smart Card Data

Xi Lin

Nanyang Technological University
xlin012@ntu.edu.sg

Xiaokui Xiao

National University of Singapore
xkxiao@nus.edu.sg

Zengxiang Li

Institute of High Performance
Computing, Singapore
liz@ihpc.a-star.edu.sg

Abstract—Knowledge of the travel time in each component of the Singapore's metro network is an important building block in our ongoing construction of an analytics system that aims to generate insights to support transportation planning and business decision making. It is useful for understanding individual passengers' route choices, estimating crowd flow across the network, and identifying network deficiencies to facilitate future train line designs for improving passenger travels. Existing work on inferring travel time in a metro network, however, tends to consider the problem on scales too small, requires data sources that are not easily available, or exhibit limitations in terms of computational efficiency. To address this problem, we propose an automated multi-stage method for inferring the time variable in various components of a metro network. This method requires only the input of a smart card dataset, and it can easily scale to data with millions of tuples. We evaluate the proposed method for a route planning application, using smart card data from Singapore, and compared our estimated results with ground truth values. Our experimental results show that the proposed method demonstrates a satisfactory inference accuracy, offers much higher accuracy than baseline methods based on prior work, and is scalable to large datasets.

Index Terms—metro network, smart card data, inference

I. INTRODUCTION

The onset of metro systems over the past decades has brought about vast transformations to many areas of the world, creating lasting impacts from the city-level scale all the way down to the levels of individuals that reside in the cities. They have become an inseparable part of our lives as we increasingly depend on it for our daily commutes. Along with the growing prevalence of automatic fare collection (AFC) systems around the world in recent years, the smart card data, which provides information about trips made using public transportation like the metro system, has also become increasingly accessible.

Leveraging transportation data like smart card data, we are building an analytics system that generates insights useful to public organizations for transportation planning and business applications, in Singapore's metro network. To this end, we first note that a trip in a metro network generally comprises of various time components, such as time taken by the train to travel across individual links, time from passengers' tapping in at the gantry to boarding the train, etc. Knowledge of these times is indispensable for generating meaningful insights. First, such information allows us to estimate passenger

route choices at different periods of the day. Based on these estimations, we can identify areas of the network where passenger travels are most severely affected, should there be a train breakdown there. Second, understanding these time components allows us to estimate the position of any passenger in the train network at any time instance during the trip. This allows us to predict the crowd flow at any part of the network and at any time. Third, it allows us to infer travel times between any station pair, which helps in identifying travel inefficiencies in the network, for future train line designs.

However, while accessible data like smart card data has spurred much research advances, estimating these time components using solely smart card data remains challenging, because records of this data only tell us the starting time and location of the whole trip, as well as the ending time and location, but they do not reveal any information on the time spent on each of the individual components of the trip. Furthermore, it does not tell us what routes are used in the trip, which are some information crucial for the inference of time components.

On the other hand, there is existing work that makes use of both smart card data and train operation logs [1]–[3]. In the literature [3], a probabilistic model is used to estimate the passenger flow distributions on a metro network, down to the granularity of passenger train choices. However, while methods that leverage train operation logs may shed light on the times of the trip spent inside the trains, they do not reveal much information about the time spent outside, such as the time from tapping in to boarding the trains, and the time from alighting at a station to tapping out. Moreover, train operation logs may not always be available or easily obtained. Another option is to make use of cell-phone data, but that poses problems of considerable privacy concerns, and the data is generally hard to obtain. Hence, for the inference of network time components, smart card data tends to be a viable choice to work with due to their relatively ease of accessibility.

However, existing methods that use smart card data face limitations that make them unsuitable for our system. A regression-based model was used in [4] to estimate a few travel time component variables before using them to analyze the individual trajectory during a metro trip. Nevertheless, only a single-train-line scenario was considered. In order to scale to larger networks, more factors like trip transfers have to be considered. Moreover, it assumed a constant train speed,

which is not always true. To achieve analysis on the scale of a whole metro network, some works employed Bayesian inference using computational approaches like the Markov Chain Monte Carlo (MCMC) methods, to estimate network time attributes and characterize the commuters route choice model [5]–[7]. While such methods have an advantage in revealing network time variabilities, they tend to be computationally costly and may take days to complete running. As a result, some data which would otherwise contribute to the inference were omitted, highlighting a problem of scalability to larger datasets. As we seek to provide the earlier-described insights in a timely manner, such methods tend to be unsuitable for our system. In general, there is lack of work that infers network component times on the scale of a large metro network like Singapore’s, yet at the same time computationally efficient.

Hence, in this paper, we propose a solution that addresses the above issues. Specifically, the contributions include:

- We present a multi-stage method that is based solely on smart card data, and it infers time components in a metro network in a progressive and computational-efficient manner. Within the method, we assume that trains traveling within the metro network may exhibit speed changes depending on the link they are on, and that each time component in the network may take on a different time measurement. We also take into account all the different possible transfers at each interchange station and their respective timing.
- We share some insights on the metro network and how we use them to address key technical challenges in the various stages of the solution, and propose a list of criteria that helps us to select data to improve the accuracy of the inference.
- We conduct a large-scale experiment on a set of smart card data collected from Singapore’s metro network of a 1-month period, consisting of about 56M trips. Our experiments show that our proposed method significantly outperforms alternative solutions in terms of accuracy, and demonstrates scalability to large datasets.

II. PRELIMINARIES

A. Background

A metro network system consists of railway lines, trains which traverse along the railway lines, and train stations where commuters get aboard and off the trains. In Singapore’s context, among the train stations, there are *terminals* that locate on either ends of a line, as well as *interchanges* that locate at the intersection points of more than one line. Figure 1 presents an example of one such small metro network that consists of two lines, Lines 1 and 2, such that vertices colored in blue (resp. red) represent stations on Line 1 (resp. Line 2). v_1 and v_5 are terminal stations of Line 1, while v_2 and v_4 are terminal stations of Line 2. In addition, v_2 and v_4 are also interchanges between the two lines.

When a passenger enters (resp. exits) a train station, he/she needs to pass through an AFC gantry and tap his/her smart card on the electronic card reader on the gantry; we refer to

Fig. 1. A small metro network

such an event as a *tap-in* (resp. *tap-out*). We assume that these events are recorded in a *fare card dataset* D , where each tuple is a quadruple $\langle o, d, t_o, t_d \rangle$, which indicates that a passenger begins his/her trip by tapping in at station o at time t_o , and ends his/her trip by tapping out at station d at time t_d ¹.

B. Modelling Metro Networks

We model Singapore’s metro network as a directed graph $G(V, E)$ with a vertex set V and an edge set E . Each edge $e = \langle u, v \rangle$ is associated a value $t_l(e)$ that equals the typical time from a train’s departure at u to its arrival to v (including the time required by the train to pick up passengers at u). We refer to $t_l(e)$ as the *link time* of e .

Meanwhile, for each vertex v on a train line, we associate v with four values $t_{fin}(v)$, $t_{fout}(v)$, $t_{rin}(v)$, and $t_{rout}(v)$. Among them, $t_{fin}(v)$ is the typical time from a passenger’s tap-in at v to his/her on-boarding a train towards the *forward* direction, while $t_{fout}(v)$ denotes the typical time from a passenger’s alighting a forward-direction train to his/her tap-out at v . $t_{rin}(v)$ and $t_{rout}(v)$ are defined similarly for the *reverse* direction of the train line. Notice that $t_{rin}(v)$ and $t_{rout}(v)$ include the time required by a passenger to wait at the platform to board a train; as a consequence, $t_{fin}(v)$ and $t_{fout}(v)$ can be drastically different when the numbers of passengers queueing for different train directions are different. We refer to $t_{fin}(v)$ (resp. $t_{rin}(v)$) as the forward (resp. reverse) *tap-in time* of v , and $t_{fout}(v)$ (resp. $t_{rout}(v)$) as the forward (resp. reverse) *tap-out time* of v .

In addition, if v is an interchange station of two train lines a and b , then we associate v with four additional values $t_{fa-fb}(v)$, $t_{fa-rb}(v)$, $t_{ra-fb}(v)$, and $t_{ra-rb}(v)$, where $t_{fa-fb}(v)$ denotes the typical time required by a passenger to transfer from a forward-direction train on Line a to a forward-direction train on Line b , and the other three values are defined in a similar manner. We refer to $t_{fa-fb}(v)$, $t_{fa-rb}(v)$, $t_{ra-fb}(v)$, and $t_{ra-rb}(v)$ as the *transfer time* at v .

C. Problem Definition

Suppose that we are given a fare card dataset D from the metro network $G(V, E)$, and that we only know the topology of G but not the time values associated with any vertex or edge in G . Our objective is to use D to estimate the link time of each edge in G , the tap-in and tap-out time of each vertex, as well as the transfer time of each interchange.

In our application’s context, we consider Singapore’s metro network, which consists of 5 train lines and 92 stations, among

¹We note that a fare card dataset may also contain other attributes such as the identification number of a trip. We omit such attributes as they are irrelevant to our study.

Fig. 2. A train line in a metro network

which 15 are interchanges. Our fare card dataset D contains information about 56M trips over a period of 1 month.

III. CHALLENGES AND SOLUTION OVERVIEW

Observe that the duration of each trip in the fare card dataset D can be expressed as the sum of several time variables in the metro network G . For instance, in Figure 1, suppose that the forward direction of Line 1 goes from v_1 to v_5 , and that a passenger's trip starts from v_1 and ends at v_3 . In that case, the trip duration can be modelled as

$$t_{fin}(v_1) + t_l(\langle v_1, v_2 \rangle) + t_l(\langle v_2, v_3 \rangle) + t_{fout}(v_3).$$

In other words, each tuple in D provides us a linear equation involving several time variables in G .

Based on the above observation, if we are to use D to infer the time variables in G , a straightforward solution is to construct a linear formulae from each tuple in D , and then perform a linear regression to obtain an estimation of each time variable. This solution, however, requires exact knowledge of the route taken by each passenger, which is absent from D and could be difficult to obtain. For example, considering the network in Figure 1, we observe that if a passenger wants to travel from v_9 to v_2 , he/she may choose to take a train leftwards on Line 2 for four stations. Alternatively, he/she may also take a train downwards for one station, do a transfer at v_4 , before taking another train leftwards for two stations on Line 1. As such, there are two possible routes from v_9 to v_2 , with one being four-station long and another being three-station long with a transfer. Given only the origin and destination of the passenger trip, it is difficult to decide which route is taken by the passenger, since the time variables in the metro network G are unknown.

To avoid the complications incurred by multiple possible routes for passenger trips, one may attempt to omit all passenger trips that involve transfers, and construct a linear system using only trips that start and end at the same train line. Unfortunately, such a solution is unable to provide any estimation of transfer time at interchange stations, since the omission of transfer trips would eliminate all transfer time variables from the linear system. Furthermore, we observe that focusing on single-train-line trips does not completely eliminate the problem caused by the existence of multiple route choices by the passengers. For example, considering the metro network line depicted in Figure 2, if a passenger wants to travel from v_2 to v_{28} , the most direct route is to take a train in a rightward direction. However, if the trains traveling in that direction are very crowded, then the passenger may choose to first traverse from v_2 to v_1 (which is a terminal station), secure a seat on a train from v_1 , and traverse to v_{28} . (We note that such choices of *non-simple paths* are rather common in our fare card dataset, especially during the peak hours.) Therefore,

it is non-trivial to construct linear systems even from single-train-line trips.

To address the above challenges, we propose a three-phase approach to inferring time variables as follows.

- Phase 1: This phase considers passenger trips that (i) traverse only one train line and (ii) do not start from or end at an interchange station. We utilize such trip data to infer the tap-in and tap-out time for non-interchange stations, as well as the link time for the edges in G that are not adjacent to interchanges. The key of our solution is a lognormal mixture model that is used to eliminate the effects of passenger trips following non-direct routes.
- Phase 2: In this phase, we focus on passenger trips that (i) traverse only one train line and (ii) start from, end at, or pass by an interchange station. We use such trip data to estimate the tap-in, tap-out, and link time pertinent to interchange stations. We separate the estimation of these time variables from those considered in Phase 1, so as to reduce the number of variables in the linear system in either phase to avoid over-fitting.
- Phase 3: In this phase, we consider passenger trips that involve transfers at interchange stations, and use them to infer the transfer time at each interchange. The basic idea is to identify transfer trips whose routes are relatively easy to pinpoint, and then compare the durations of these trips against the time estimations from Phases 1 and 2 to derive the transfer time values.

In the following section, we will elaborate each phase of our solution in Sections IV and V, respectively.

IV. PHASE 1: ESTIMATIONS FOR NON-INTERCHANGES

As mentioned in Section III, Phase 1 of our solution aims to estimate the tap-in and tap-out time at non-interchange stations, as well as the link time between any two non-interchanges. For this purpose, we would like to utilize the passenger trips that do not involve any transfers. However, there are two issues that we need to tackle:

- Issue 1: How do we identify passenger trips without transfers?
- Issue 2: Given a set of non-transfer passenger trips, how do we mitigate the effects of non-simple paths taken by the passengers?

Addressing Issue 1. A naive solution to tackling Issue 1 is to consider only those passenger trips where the origin and destination are on the same train line. However, we observe that this solution is often inaccurate. To explain, consider the metro network in Figure 1. Suppose that a passenger trip starts from v_6 and ends at v_4 . Although both v_6 and v_4 are on Line 2, it is not necessarily the case that the trip follows a non-transfer path $P_1 = \langle v_6, v_7, v_8, v_9, v_4 \rangle$. In particular, there exists an alternative path $P_2 = \langle v_6, v_2, v_3, v_4 \rangle$ that involves a transfer to Line 1. If P_2 is a faster path than P_1 , then passengers may be more inclined to choose P_2 over P_1 . (This phenomenon

occurs in several places of the metro network that we study.) Therefore, we cannot identify non-transfer trips solely based on whether the origin and destination are on the same line.

To tackle the above problem, we adopt the following heuristics to filter out passenger trips that may involve transfers. First, we omit all trips from a station u to a station v , if (i) u and v are not on the same train line or (ii) u or v is an interchange. The reason is that it is often difficult to infer the exact routes corresponding to such trips. Second, if the shortest path from u to v (in terms of the number of hops) is a path involving transfers, then we do not consider the trip, because the path involving transfers may be chosen by the passengers. On our fare card dataset, we observe that these heuristics are highly effective in eliminating transfer trips, albeit somewhat conservative. For any pair $\langle u, v \rangle$ that passes through all of the above filtering procedure, we refer to it as a *qualifying origin-destination (OD) pair for Phase 1*.

Addressing Issue 2. Suppose that we have a qualifying OD pair $\langle u, v \rangle$ for Phase 1, and we have retrieved the set of passenger trips from u to v . Our next step is to address Issue 2, i.e., to alleviate the effects of non-simple paths. Towards this end, we exploit an observation made in previous work [8], [9]: when there exist multiple route choices from u to v , the travel time of passengers' trips from u to v often follows a mixture of distributions, such that each distribution corresponds to one route choice. For example, Figure 3 shows the distribution of passengers' trip durations during the morning peak hours (i.e., 7:30am to 9:30am) on weekdays, for an OD pair in our fare card dataset. The OD pair concerns two stations u and v on the same train line, and they are twelve stations apart from each other. From Figure 3, it is evident that the passengers can be divided into two groups with very different travel time. It turns out that the group of users with the smaller travel time are those who traverse directly from u to v , whereas the other group of users choose to first traverse from u to the terminal station of the train line (which is right next to u in the reverse direction), and then secure a seat in the train that travels from the terminal station to v .

From our fare card dataset, we observe that, for each OD pair, the distribution of passengers' trip durations can be accurately modeled as a lognormal mixture. For example, the red line in Figure 3 shows the result of fitting a lognormal mixture to the trip duration distribution. Motivated by this, we fit a lognormal mixture on the passenger trips for each OD pair $\langle u, v \rangle$, and if the mixture contains more than one lognormal distributions, we will take the distribution with the smallest mode, and use this mode as an estimation of the travel time of the most direct route from u to v . A technical issue here is that the algorithms for fitting lognormal mixtures require as input the total number of lognormal distributions in the mixture, which is unknown in advance. To address this issue, we vary the number of lognormal distributions, and measure the accuracy of each fit using the Bayesian information criterion (BIC); then, we choose the model with the smallest BIC. In addition, if the number of passenger trips for an OD pair is

Fig. 3. A lognormal fit of travel times between two stations

small (i.e., < 1500), then we assume that all those trips follow a lognormal distribution instead of a lognormal mixture, so as to avoid inaccurate model fitting.

In summary, for each qualifying OD pair $\langle u, v \rangle$, we transform the set of passenger trips from u to v into a quadruple $\langle u, v, \mu, cnt \rangle$, where $\mu_{u,v}$ is the mode of the lognormal distribution that corresponds to the “fastest” route taken by the passengers, and cnt is the number of passenger trips from u to v . We refer to $\langle u, v, \mu, cnt \rangle$ as the *OD summary* for $\langle u, v \rangle$.

Estimations of Time Variables. Suppose that we obtain a set S of OD summaries as a result of the previous steps. For each OD summary $\langle u, v, \mu, cnt \rangle$, we retrieve the shortest path p from u to v (in terms of the number of hops), and express the travel time from u to v as a sum μ' of the relevant time variables:

$$\mu' = \begin{cases} t_{fin}(u) + t_{fout}(v) + \sum_{e \in p} t_l(e), & \text{if } p \text{ follows the forward direction of the train line} \\ t_{rin}(u) + t_{rout}(v) + \sum_{e \in p} t_l(e), & \text{otherwise} \end{cases}$$

After that, we apply a ridge regression to obtain time variables that minimize

$$\sum_{\langle u, v, \mu, cnt \rangle \in S} \left(cnt \cdot (\mu - \mu')^2 \right),$$

with a regularization term to avoid overfitting. From the regression results, we obtain the estimations of the tap-in and tap-out time for non-interchange stations, as well as the estimation of the link time of each edge between non-interchanges. We refer to these estimations as the *estimated time for non-interchanges*. The results may also contain the estimated link time for some edges pertinent to interchange stations, but we discard such estimations since they will be estimated more systematically in Phase 2.

V. PHASES 2 AND 3: ESTIMATIONS FOR INTERCHANGES

Phase 2. After we derive the time variables for non-interchange stations, we proceed to Phase 2, which aims

to estimate the tap-in, tap-out, and link time pertinent to interchange stations. In particular, we first identify the set of qualifying OD pairs for Phase 2, in a manner similar to the case of Phase 1; after that, we derive an OD summary for each qualifying OD pair based on lognormal fits, based on which we construct a linear system for estimation of time variables.

In particular, we say that an OD pair $\langle u, v \rangle$ is a qualifying OD pair for Phase 2, if it satisfies all of the following conditions: (i) u or v is an interchange, and they are on the same train line; (ii) the shortest path (in terms of the number of hops) from u to v does not involve transfers. The rationale of these conditions is to ensure that the shortest path from u to v along the train line that both u and v are on, is highly likely to be the route taken by passengers.

Once we have the qualifying OD pairs for Phase 2, we convert each pair $\langle u, v \rangle$ into an OD summary $\langle u, v, \mu, cnt \rangle$, using the same method applied in Phase 1. That is, we first retrieve all passenger trips from u to v , and fit a lognormal mixture model on the trip durations. After that, we identify the lognormal distribution in the mixture with the smallest mode, and set μ to the mode. In addition, we set cnt to the number of passenger trips from u to v .

Finally, we construct, from each OD summary, a linear formula of time variables, and we apply a ridge regression to derive the variables pertinent to interchange stations, in the same way as in Phase 1 but with one modification: if any linear formula involves a time variable for a non-interchange station, then we replace the variable with its estimated value that we derived from Phase 1. In other words, we treat the estimated time for non-interchange stations from Phase 1 as constants (instead of variables) in the ridge regression, and we focus only on the time variables for interchanges. This reduces the number of variables in the regression task, which helps improve the robustness of the results.

Phase 3. Our solution for Phase 3 is similar in spirit to those of Phases 1 and 2, except that it focuses only on the transfer time of interchange stations. In particular, we define the qualifying OD pairs for Phase 3 as the pairs $\langle u, v \rangle$ that satisfy all of the following requirements: (i) u and v are on two different train lines; (ii) the shortest path (in terms of the number of hops) from u to v contains at most three stations other than u and v . These requirements are to ensure that the shortest path from u to v is likely to be traversed by passengers.

Given the qualifying OD pairs, we transform each pair $\langle u, v \rangle$ into an OD summary $\langle u, v, \mu, cnt \rangle$, using the lognormal fitting procedure introduced in Section IV. After that, for each OD summary, we construct a linear formula of time variables, and we substitute all variables for tap-in, tap-out, and link time with estimated values obtained from Phases 1 and 2. This leads to a set of formula with only variables concerning transfer time. Finally, we perform a ridge regression to obtain estimations of transfer time for all interchange stations.

VI. EXPERIMENTS

This section evaluates our solution using three different ways. We first evaluate it in terms of its accuracy for route

planning purposes, and compare its accuracy with several baselines. Sections VI-A and VI-B elaborate on the datasets and baselines, and Section VI-C elaborate on the comparison results. Then, we evaluate our solution in terms of how accurate the estimated link times are relative to ground truth values. The findings are then presented in Section VI-D. Finally, we evaluate our solution in terms its scalability to increasing dataset sizes, and present the results in VI-E.

A. Datasets

1) *Fare Card Data Subsets:* As mentioned, we use the fare card dataset consisting of around 56M trips collected on one month, October 2015, from Singapore’s metro network with 5 train lines and 92 stations. We generate three subsets of the dataset as follows:

- **morning-peak**, which consists of all trips made from 7:30am to 9:30am on week days;
- **evening-peak**, which contains all trips made from 5:30pm to 7:30pm on week days;
- **off-peak**, which comprises of all trips made on week days, other than those in **morning-peak** and **evening-peak**.

We consider these three subsets separately, since (i) the characteristics of passenger trips differ considerably in the three subsets, and (ii) the metro network has increased train frequencies during the morning and evening peak hours. In addition, we omit the weekend trips due to data sparsity.

2) *Ground Truth Link Time Dataset:* In the second part of our evaluation, we also use a dataset that contains link time measurements manually collected over a week that falls within the one-month period of the fare card dataset, covering a total of 25 stations of Singapore’s metro network. We call this dataset the ground truth link times and break it down into three subsets using time periods similar to those of the fare card data subsets.

B. Baselines for Comparisons

We consider two baseline methods in our experiments, dubbed Baseline 1 and Baseline 2, respectively. These baselines are chosen for their computational efficiency and scalability to large datasets, which are essential to our analytic system to provide results in a prompt manner. We note that there exists another class of methods based on Bayesian inference and MCMC-based sampling [5]–[7]. However, we do not consider them in our comparisons as they do not possess the computational efficiency required by our system. In particular, these methods have running time that extends into order of days, even with substantial data omission.

Baseline 1. This method is based on prior work [4]. The method assumes that the train speed on each line is a constant, and it infers this constant using a linear regression based on the observed travel time between the stations on the train line. In addition, its regression procedure also outputs the tap-in and tap-out time of each station. However, the method does not consider the case of interchanges between multiple train

TABLE I
COMPARISON OF RMSE, 10%ILE AND 90%ILE ON morning-peak

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	1070			1744			2171			2872		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	132	27	164	269	34	377	394	54	573	543	158	862
Baseline 2	195	63	250	188	69	306	214	35	375	246	43	394
Our solution	57	6	74	118	3	201	143	16	255	156	4	279

TABLE II
COMPARISON OF RMSE, 10%ILE AND 90%ILE ON evening-peak

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	705			1504			2141			3055		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	132	7	218	271	32	364	399	33	515	525	45	796
Baseline 2	188	65	213	225	33	391	283	33	451	267	58	435
Our solution	40	2	73	103	4	172	175	8	287	176	7	302

TABLE III
COMPARISON OF RMSE, 10%ILE AND 90%ILE ON off-peak

	Query group 1			Query group 2			Query group 3			Query group 4		
Average ground truth travel time (s)	679			1610			2258			3367		
Error Type	RMSE	10%ile (s)	90%ile (s)	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile	RMSE	10%ile	90%ile
Baseline 1	158	34	315	360	61	488	500	113	715	504	113	692
Baseline 2	191	100	239	270	38	458	297	16	407	381	103	564
Our solution	57	9	100	136	3	249	176	27	242	286	16	501

TABLE IV
RMSE, 10%ILE AND 90%ILE FOR LINK TIME ESTIMATION

	morning-peak	evening-peak	off-peak
Average ground truth link time (s)	186	184	183
RMSE	16	21	16
10%ile (s)	3	2	1
90%ile (s)	26	28	28

lines. To address this issue, we use our solution in Section V to infer transfer time for Baseline 1. In particular, we first apply the method in [4] to estimate the train speed and other time variables in each line, and then convert them into a set of tap-in, tap-out, and link time, which is then fed into our solution in Section V to obtain estimations of transfer time at interchange stations.

Baseline 2. This method is an improvement over Baseline 1, in that it does not assume constant train speed in each train line. Instead, it is identical to our solution, except that it omits the lognormal fitting procedure that is used in the Phase 1 and 2 of our solution. That is, given the qualifying OD pairs in each phase, it directly performs a ridge regression using the observed trip durations as the dependent variables (instead of using the mode of a lognormal distribution in a lognormal mixture). Furthermore, similar to Baseline 1, it does not draw a distinction between interchange and non-interchange stations. That is, instead of inferring the network times of non-interchange stations and interchange stations in two separate phases, it does so in one single phase. Effectively, Baseline 2 serves as a middle ground between Baseline 1 and

our proposed solution.

C. Experiment Settings and Results of Method Comparisons

We evaluate each method in the context of route planning. Specifically, given a fare card dataset, we use each method to infer the time variables in the metro network G . Let G' denote the metro network with the estimated tap-in, tap-out, link, and transfer time. Then, we consider a number of route planning queries, each of which asks for the smallest travel time from a station u to another station v . For each query $\langle u, v \rangle$, we compute the shortest path (in terms of travel time) from u to v in G' , and then obtain the estimated travel time t' from u to v (including tap-in and tap-out time). After that, we would measure the accuracy of t' . However, this brings us a challenge: as we do not know the ground truth travel time from u to v , it is difficult to assess the accuracy of t' . To address this issue, we utilize the lognormal fitting technique in Section IV. In particular, we obtain all passenger trips from u to v , and fit a lognormal mixture model on the trip durations. After that, we identify the lognormal distribution in the mixture with the smallest mode, and use this mode as the ground truth travel time from u to v . To ensure the accuracy of this approach,

we require that there must exist at least 1500 passenger trips from u to v . In addition, we would omit all trips from u to v during the inference of time variables in the metro network.

Based on the above discussions, we evaluate each method M as follows.

- Given a fare card dataset D , we identify the OD pairs with at least 1500 passenger trips. Then, we randomly select 20 such OD pairs, put them into a hold-out set H , and we remove from D all passenger trips corresponding to those OD pairs. Let D' denote the resulting dataset.
- We use M to infer the time variables in the metro network, and obtain a version G' of the metro network with estimated time values.
- For each OD pair $\langle u, v \rangle$ in H , we use G' to derive the estimated travel time from u to v , and then compare it against the ground truth computed from a lognormal fit on the passenger trips from u to v .

We repeat the above procedure 50 times, which results in 1000 travel time queries, as well as 1000 answers from each method. We divide the queries into 4 groups as follows. First, we sort the ground truth answers for the 1000 queries in ascending order. Then, we divide the sorted sequence into 4 equal-size subsequence. After that, queries corresponding to i -th subsequence is put into the i -th group. For each method, we compute the *root-mean-square error (RMSE)*, the *10-th percentile error (10%ile)*, and the *90-th percentile error (90%ile)* of its query answer in each query group.

Tables I, II, and III show the RMSE, 10%ile, and 90%ile of each method on morning-peak, evening-peak, and off-peak, respectively. Observe that the RMSE, 10%ile and 90%ile errors of our solution are significantly smaller than Baseline 2, which demonstrates the effectiveness of our lognormal fitting procedure in mitigating the effects of passengers' heterogeneous route choices. In turn, Baseline 2 is more accurate than Baseline 1, since the "constant train speed" assumption [4] adopted by Baseline 1 incurs considerable estimation errors. The errors of each method tend to increase from query group 1 to 4, since queries with longer travel time tend to involve a larger number of time variables, which makes them more difficult to answer accurately. Overall, the RMSE of our method is around 5% of the ground truth answer, which shows that our method provides satisfactory accuracy for travel time estimation. Moreover, the 90%ile error of our method is not larger than 16% of the ground truth answer, which implies that our method still performs fairly well even in the poor cases.

D. Validation with Ground Truth Link Times

We also evaluate, for each fare card data subset, the inference accuracy based on how close the estimated link times are to their ground truths. To this end, we first run our proposed inference method on each fare card data subset: morning-peak, evening-peak and off-peak. Similar to the previous stage of validation, we compute the RMSE, 10%ile, and 90%ile errors for the estimated link times, based on its corresponding ground truth in the link time subset. Table IV shows the results

Fig. 4. Running time

for morning-peak, evening-peak and off-peak. The average ground truth link time remains relatively unchanged over the three time periods. Observe that the RMSE is around 10% and the 90%ile error is around 15%, of the average ground truth. These results demonstrate our method's general consistency in producing fairly accurate estimation across the links.

E. Computational Efficiency and Scalability

We implemented our solution using Python. To evaluate its scalability and computational efficiency, we compare its running time over sample sets of the smart card dataset with sizes varying from 2M to 16M tuples. Figure 4 illustrates the average running time for each sample set. Observe that the running time exhibits a linear growth with the increase in number of tuples, and all running times fall within the range of hundreds of seconds. These results indicate that our solution is scalable to datasets with millions of tuples with a comfortable computational efficiency. This makes our solution a viable choice in our analytics system for timely insights provision.

VII. RELATED WORK

A plethora of research about the analytics of smart card data have sprung up over the years. As shared earlier, there is existing work that uses smart card data to provide insights to the time characteristics of the network, such as in [4] where a few explanatory variables of a line in the metro network are estimated. Another popular approach is one based on Bayesian statistical inference [5], [7], where methods simultaneously infer passenger route choice behaviour. In the area of route choice estimations, numerous works require the use of scheduled timetables in tandem with smart card data to model passenger flow assignments in metro networks [1]–[3], [10], [11]. In the latest work of this genre by Zhao et al. [3], the proposed solution considers the "travelers left behind" situation, a phenomenon caused by crowded trains. Leveraging on the work in [11], Kyung et al. [12] investigate the degree and manner in which crowding affect commuter route choices.

Many AFC systems record only information of trip origins, i.e. trip start time and stop/station, but neglect the trip

destinations, as well as the trip counts of each OD pair. To address this, much work has been done for estimations of trip destinations and travel demand matrix, which is used to represent the trip counts between any two stops or stations [13]–[18]. While the estimated matrices reveal travel demands between any two locations, they do not provide information about the trip component times between them. Furthermore, while it may be easy to infer the route taken if it is a bus trip since public buses generally follow standard routes, inference gets tricky in the context of metro trips due to the existence of multiple route choices.

Other aspects of commuter travel behaviour and applications can also be derived from smart card data. Ma et al. [19] propose a data mining approach to identify travel patterns for individual commuters, based on the spatial and temporal characteristics of the trips. Kusakabe et al. [20] leverage the strengths of both smart card and travel survey data and develop a data fusion methodology for estimating commuter trip purposes over long time periods in Osaka, Japan. Long et al. [21] seek to understand extreme public transit riders using both traditional household surveys and data sources like smart card data. In [22], Pelletier et al. present a survey that highlights the research areas concerning smart card data, as well as the prominent work in each.

In general, there has been limited focus on metro network time inference using smart card data. Related works involving smart card data are either inapplicable to this problem, or in the case of existing time inference methods, face scalability and computational efficiency issues.

VIII. CONCLUSION

This paper presents a new solution to inferring travel time in Singapore's metro network using only smart card data. The key idea of the solution is to (i) divide the time variables in the network into different categories and estimate them separately, and (ii) utilize heuristics and lognormal mixture models to mitigate the complexities caused by heterogeneous route choices by the passengers. We evaluate the solution in the context of route planning, and show that it significantly outperforms the baseline approaches in terms of accuracy. For future work, we plan to utilize the time variables estimated from our solution in our analytic system, and in particular, we seek to analyze passengers' route choices and study the passenger waiting time profiles at each station of the metro network. In addition, we plan to investigate how passenger route choices would change when there are new train lines added into the metro network.

IX. ACKNOWLEDGEMENTS

This paper is supported in part by grant MOE2015-T2-2-069 from MOE Singapore.

REFERENCES

- [1] Y. Sun and R. Xu, "Rail transit travel time reliability and estimation of passenger route choice behavior: Analysis using automatic fare collection data," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2275, pp. 58–67, 2012.
- [2] F. Zhou and R.-h. Xu, "Model of passenger flow assignment for urban rail transit based on entry and exit time constraints," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2284, pp. 57–61, 2012.
- [3] J. Zhao, F. Zhang, L. Tu, C. Xu, D. Shen, C. Tian, X.-Y. Li, and Z. Li, "Estimation of passenger route choice pattern using smart card data for complex metro systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 790–801, 2017.
- [4] L. Sun, D.-H. Lee, A. Erath, and X. Huang, "Using smart card data to extract passenger's spatio-temporal density and train's trajectory of mrt system," in *Proceedings of the ACM SIGKDD international workshop on urban computing*. ACM, 2012, pp. 142–148.
- [5] L. Sun, Y. Lu, J. G. Jin, D.-H. Lee, and K. W. Axhausen, "An integrated bayesian approach for passenger flow assignment in metro networks," *Transportation Research Part C: Emerging Technologies*, vol. 52, pp. 116–131, 2015.
- [6] Y.-S. Zhang and E.-J. Yao, "Splitting travel time based on afc data: estimating walking, waiting, transfer, and in-vehicle travel times in metro system," *Discrete Dynamics in Nature and Society*, vol. 2015, 2015.
- [7] M. Rahbar, M. Mesbah, M. Hickman, and A. Tavassoli, "Determining route choice behaviour of public transport passengers using bayesian statistical inference," in *Australasian Transport Research Forum (ATRF)*, 38th, 2016, Melbourne, Victoria, Australia, 2016.
- [8] C. Monterola, E. F. Legara, D. Pan, K. K. Lee, and G. G. Hung, "Non-invasive procedure to probe the route choices of commuters in rail transit systems," *Procedia Computer Science*, vol. 80, pp. 2387–2391, 2016.
- [9] Q. Fu, R. Liu, and S. Hess, "A bayesian modelling framework for individual passenger's probabilistic route choices: a case study on the london underground," Leeds, 2014.
- [10] T. Kusakabe, T. Iryo, and Y. Asakura, "Estimation method for railway passengers train choice behavior with smart card transaction data," *Transportation*, vol. 37, no. 5, pp. 731–749, 2010.
- [11] S.-P. Hong, Y.-H. Min, M.-J. Park, K. M. Kim, and S. M. Oh, "Precise estimation of connections of metro passengers from smart card data," *Transportation*, vol. 43, no. 5, pp. 749–769, 2016.
- [12] K. M. Kim, S.-P. Hong, S.-J. Ko, and D. Kim, "Does crowding affect the path choice of metro passengers?" *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 292–304, 2015.
- [13] D. Li, Y. Lin, X. Zhao, H. Song, and N. Zou, "Estimating a transit passenger trip origin-destination matrix using automatic fare collection system," *Database systems for advanced applications*, pp. 502–513, 2011.
- [14] M. A. Munizaga and C. Palma, "Estimation of a disaggregate multi-modal public transport origin-destination matrix from passive smartcard data from santiago, chile," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 9–18, 2012.
- [15] J. Barry, R. Freimer, and H. Slavin, "Use of entry-only automatic fare collection data to estimate linked transit trips in new york city," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2112, pp. 53–61, 2009.
- [16] M. Munizaga, F. Devillaine, C. Navarrete, and D. Silva, "Validating travel behavior estimated from smartcard data," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 70–79, 2014.
- [17] A. A. Alsger, M. Mesbah, L. Ferreira, and H. Safi, "Use of smart card fare data to estimate public transport origin-destination matrix," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2535, pp. 88–96, 2015.
- [18] A. A. Nunes, T. G. Dias, and J. F. e Cunha, "Passenger journey destination estimation from automated fare collection system data using spatial validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 133–142, 2016.
- [19] X. Ma, Y.-J. Wu, Y. Wang, F. Chen, and J. Liu, "Mining smart card data for transit riders travel patterns," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 1–12, 2013.
- [20] 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.
- [21] Y. Long, X. Liu, J. Zhou, and Y. Chai, "Early birds, night owls, and tireless/recurring itinerants: An exploratory analysis of extreme transit behaviors in beijing, china," *Habitat International*, vol. 57, pp. 223–232, 2016.
- [22] 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.

A.2 Publication for Chapter 4

Path Travel Time Estimation using Attribute-related Hybrid Trajectories Network

Xi Lin

xlin012@e.ntu.edu.sg

Nanyang Technological University

Yequan Wang*

tshwangyequan@gmail.com

Advertising and Marketing Services,
Tencent Inc.

Xiaokui Xiao

xkxiao@nus.edu.sg

National University of Singapore

Zengxiang Li

liz@ihpc.a-star.edu.sg

Institute of High Performance
Computing, Singapore

Sourav S. Bhowmick

assourav@ntu.edu.sg

Nanyang Technological University

ABSTRACT

Estimation of path travel time provides great value to applications like bus line designs and route plannings. Existing approaches are mainly based on single-source trajectory datasets that are usually large in size to ensure a satisfactory performance. This leads to two limitations: 1) Large-scale data may not always be attainable, e.g. city-scale public bus data is usually small compared to taxi data due to relative fewer bus trips in a day. 2) Considering only single-source trajectory data neglects the potential estimation-improving insights of external data, e.g. trajectory dataset of other vehicle sources obtained from the same geographical region. A challenge is how to effectively utilize such other trajectory sources. Moreover, existing work does not attend the important attributes of a trajectory including vehicle ID, day of week, rainfall level etc., which are important for estimating the path travel time. Motivated by these and the recent successes of neural network models, we propose Attribute-related Hybrid Trajectories Network (AtHy-TNet), a neural model that effectively utilizes the attribute correlations, as well as the spatial and temporal relationships across hybrid trajectory data. We apply this to a novel problem of estimating path travel time of a type of vehicles using a hybrid trajectory dataset that includes trajectories from other vehicle types. We demonstrate in our experiments the benefits of considering hybrid data for travel time estimation, and show that AtHy-TNet significantly outperforms state-of-the-art methods on real-world trajectory datasets.

KEYWORDS

deep learning; path travel time; hybrid trajectory

ACM Reference Format:

Xi Lin, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S. Bhowmick. 2019. Path Travel Time Estimation using Attribute-related Hybrid Trajectories Network. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3357384.3357927>

1 INTRODUCTION

The knowledge of the travel time of any path within a road network is valuable to many. Through the lens of public transportation, this information is useful for bus line designs and road traffic monitoring. From the perspectives of ride-hailing (e.g. Didi, Grab and Uber) and courier delivery businesses, it improves user experience, and enables cost (e.g. time and fuel) savings, respectively. Such information also helps commuters in their trip-plannings.

With the growth in data collection technology, massive data is increasingly available for use, and to date, there has been much work which focuses on path travel time estimation using trajectory data. Estimating path travel times has been challenging due to multi-nature considerations. First, travel time is affected by spatial factors, such as the road characteristics. For example, a vehicle may travel faster along an expressway than within the residential district. Second, it is also temporally dependent. For example, travel times in the downtown area may be longer during peak hours due to traffic congestions. Other factors that affect such estimations are such as the vehicle type and weather conditions. For example, taxi sedans tend to move faster than large public buses.

Existing work traditionally falls under two approaches. The first is link-based methods, where the intuition is to first estimate the travel time of every road-segment, before combining them to get the path travel time [12, 19, 25, 28, 31]. These methods have a limitation in that they do not consider traffic complexities outside the road-segments, such as traffic lights. The second class of methods resolves such issues to some extent; the main idea is that travel time can be estimated by extracting the historical travel time of its sub-paths, which implicitly contains certain travel costs unconsidered in the link-based ones [5, 16, 30, 32]. More recently, deep learning has demonstrated further success for this task, based on its ability for powerful representation and implicitly modeling multi-natured traffic complexities [26, 33].

However, existing work mostly leverages single-source trajectory data that are usually also large in size. While having large-scale

*This work was done when Yequan was a Ph.D student at Tsinghua University and a visiting Ph.D student at Nanyang Technological University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '19, November 3–7, 2019, Beijing, China

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6976-3/19/11...\$15.00

<https://doi.org/10.1145/3357384.3357927>

data benefits estimation since it usually also means larger road coverage, data of such size is not always attainable. For example, public bus trajectory data sizes in many cities of the world are small due to the much fewer number of daily trips and the less extensive coverage of the road network, as compared to other types of vehicles like taxis. A newcomer to ride-sharing in a city also does not have access to large data due to the low number of trips that it has serviced initially. For the same reasons, the lack of large-scale data exists for vehicles where accurate path travel time estimation is valuable for highly-critical timely arrivals, e.g. ambulances. It is also prevalent for vehicles that require such estimation for global optimizations to meet business goals, e.g. logistics trucks.

Moreover, considering trajectory data from single sources neglects the potential estimation-improving insights present in other sources. To the best of our knowledge, there has yet to exist any work that leverages hybrid trajectory dataset for travel time prediction based on deep learning, e.g. using a mixture of bus trajectory data and a massive taxi trajectory data within the same city for bus travel time estimation. Estimating across hybrid trajectories, however, is also a challenging task mainly due to differences in trajectory characteristics. For example, buses tend to move slower than taxis due to their larger sizes and passenger loads, and have to stop at scheduled bus stops for pick-ups and drop-offs. As a result, attributes that come along with these trajectories tend to be highly crucial for such estimations, with some relatively more important than others, e.g. vehicle ID that reveals the vehicle type. However, existing work generally does not consider the relationships between attributes, nor attend the important ones.

As such, we propose Attribute-related Hybrid Trajectories Network (AtHy-TNet), the first end-to-end neural architecture for travel time estimation using hybrid trajectory data, to tackle the issues above. Its value lies in its applicability for use-cases when only small training data is available to the transport operator, e.g. public buses, shuttle buses, and urban logistics trucks. It also enables small/medium transport operators to enhance their services by leveraging massive trajectory data which are owned by giant operators or government agencies. To this end, we have considered various spatio-temporal and attributional treatments. The main contributions of this paper are summarized as follows:

- For the first time, we propose a neural method that effectively tackles the novel problem of estimating path travel time using hybrid data sources. More importantly, AtHy-TNet implicitly models the multi-nature factors affecting path travel time, and their correlations across trajectories of heterogeneous vehicle sources.
- We propose: i) an Attribute Correlation Module that models path attributes correlations and attend to the more important attributes, ii) a Sub-path Representation Module that utilizes a Tri-CNN Encoder and on-road information to create a representation for each sub-path, iii) a Sub-path Correlation Module that models correlations across sub-paths, and utilizes a Correlation component to learn the path-attribute relationships, and iv) a Joint-Learning Prediction Module that learns the path travel time estimation task with an auxiliary task of sub-path travel time estimation.

- We conduct extensive experiments on trajectory data obtained in Singapore. We demonstrate the benefits of considering hybrid data for travel time estimation, and show that AtHy-TNet significantly outperforms state-of-the-art methods. We also show the robustness of its performance across different conditions, e.g. different days of the week.

2 PRELIMINARIES

A trajectory of a vehicle within a road network is defined as a sequence of GPS records, $T_x = (x_1, x_2, x_3, \dots, x_n)$, where x_i denotes a record within the trajectory, and n denotes the total number of records in this trajectory. Each record $x_i = (x_i.lat, x_i.lng, x_i.t, x_i.state, x_i.id)$, where $x_i.lat$, $x_i.lng$, $x_i.t$, $x_i.state$ and $x_i.id$ denote the latitude, the longitude, the timestamp, the vehicle state (e.g. busy, stop etc.), and the map-matched road ID respectively. For each trajectory, it is associated with T_a , a set of attributes which can be categorized under four classes: vehicular, spatial, temporal, and weather. Vehicular attributes provide information about the vehicle that travels the trajectory, e.g. vehicle ID. The weather attributes include information like rainfall level in the city. The temporal attributes include periodic information like time of day (at which T_x starts) and day of week. Finally, the spatial attributes are geospatial information like the total distance (traveled by T_x). Note that a trajectory is obtainable from different vehicle types, e.g. taxi and bus, and thus may exhibit different spatiotemporal characteristics depending on its vehicular source. As such, the vehicle ID of a trajectory acts as a unique identifier that also reveals the vehicle type. More formally, we define a trajectory data instance, which includes the trajectory and all its associated attributes, as $T = (T_x, T_a)$. In this work, we use a hybrid collection, i.e. a mixture, of trajectories obtained from more than one vehicle type.

Our objective: Given the input of a hybrid trajectory dataset built from a small trajectory dataset of one type of vehicles and a massive trajectory dataset of another type, we want to train a neural model that can estimate the travel time required by the small-dataset vehicles to complete any path within the road network, at any time of the day and on any day of the week.

3 ATTRIBUTE-RELATED HYBRID TRAJECTORIES NETWORK

In this section, we present our proposed method. First, we highlight the intuition of how our method is able to leverage hybrid trajectories to improve travel time estimation. Second, we provide an overview of our architecture along with the motivations behind the constituting modules. Finally, we describe in details the functions of each of these modules, the rationale behind their designs and how these designs contribute to effective travel time estimation.

3.1 Intuition of AtHy-TNet

A small trajectory dataset usually implies that the coverage of its vehicles is less extensive than vehicles of a massive dataset. However, if both datasets are sampled from the same geographical space, the much larger coverage of the latter may overlap with that of the former. AtHy-TNet leverages these overlaps within the hybrid dataset and the fact that vehicles within a dataset belong to the same type. The intuition is to learn the characteristics of trajectories

Figure 1: A small map with trajectory overlays from two vehicle types (best viewed in colour).

of both the small-dataset and big-dataset vehicles considered in the hybrid dataset, and model the correlations at parts where they overlap. The correlation learnt is then used to infer the travel time of small-dataset trajectories untraveled before, yet whose paths overlap with the big-dataset ones used during training.

For instance, Figure 1 shows a roadmap of a small district that is overlaid with trajectories from two datasets, each obtained from a different type of vehicles. Traces in green represent trajectories generated from a small dataset A , while traces in red represent trajectories obtained from a larger dataset B . Traces in blue highlight the roads where the trajectories of the two datasets overlap. As observed, trajectories from A also have a less-extensive coverage of the road network than its larger counterpart. As such, if an operator who possesses only A wishes to estimate travel time for paths in regions outside its coverage, e.g. red regions, there may be large errors since A does not consider the complexities there yet. In this case, AtHy-TNet leverages the correlations between trajectories of the two vehicle types at the blue regions and the larger coverage of B at the red regions. Specifically, it first models the multi-natured characteristics of the trajectories in both the green and red regions, while learning the correlations of these characteristics at the blue regions. This correlation can then be used to estimate the travel time at the out-of-coverage red regions.

3.2 Overview of AtHy-TNet

The architecture of the proposed AtHy-TNet is shown in Figure 2. It comprises of four modules: Attribute Correlation, Sub-path Representation, Sub-path Correlation, and Joint Prediction. The input to our model is a trajectory and its associated attributes. Path attributes are very important when dealing with hybrid data due to the different trajectory natures. However, existing work generally does not consider the correlations across these attributes nor their relative importance. As such, we design an Attribute Correlation Module to model such relationships among trajectory attributes, e.g. vehicle ID that differentiates a type of vehicles from another. We support our main path travel time estimation task with an auxiliary task of sub-path travel time estimation. The rationale is that by accurately estimating the sub-path times in a path, the path travel time can in turn be more accurately estimated. Furthermore, by modeling sub-path characteristics, it enables the model to learn

Figure 2: Architecture of AtHy-TNet Model

the trajectory differences across different vehicle types to a finer detail. To this end, we first extract the abstract spatial features of sub-paths by designing a Sub-path Representation Module. For the first time, we consider a design that effectively utilizes on-road information, e.g. map-matched road GPS coordinates. To model the temporal relationships across the sub-paths, and the correlation between each sub-path and the path attributes, we design a Sub-path Correlation Module. Finally, we leverage the Joint Prediction Module to combine the main and auxiliary task described earlier.

3.3 Attribute Correlation Module

Path travel time is largely affected by various external factors. We are the first to utilize the relative importance of each of these attributes through our proposed Attribute Correlation Module, as depicted in Figure 3. Specifically, the input here is T_a , a set of vehicular, weather, temporal and spatial attributes, i.e. vehicle ID, rainfall, day of week, time of day, and total distance. The Attribute Embedder embeds all of these attributes into higher-dimension vectors via a linear layer. For all attributes other than distance, we denote them as a_c , a_r , a_d and a_t respectively.

Global Distance Representation. Path distance is an important attribute in determining path travel time. Existing deep-learning based work focuses mainly on coarse-grained great-circle distances [26]. However, fine-grained distance data is able to reveal additional information about the path traveled. Hence, we integrate not only coarse-grained, but also fine-grained distance in our model. Specifically, we consider two attributes: a coarse-grained global trip distance, and a fine-grained global trip distance derived from on-road distance information. We convert each distance into a $\mathbb{R}^{16}$ vector, denoted as d_c and d_f , via the Attribute Embedder. Then we concatenate them and apply a linear transformation:

$$a_g = W_d[d_c, d_f] + b_d, \quad (1)$$

where a_g is the global distance representation and W_d and b_d are the parameters of this operation. The square brackets denote the concatenate operator.

Attribute Attention. Certain attributes are relatively more important than others on some occasions in revealing the travel time.

Figure 3: Attribute Correlation Module. a_c, a_d, a_r and a_t represent embedded vectors for vehicle ID, day-of-week, rainfall, and time-of-day respectively. d_c (resp. d_f) represent the fine (resp. coarse) grained embedded distance vectors.

Vehicle ID is generally more important when dealing with hybrid data because the trajectory of a type of vehicles may differ greatly from the trajectory of another, e.g. public buses move much slower than taxis. A vehicle traveling in the downtown during peak hours will be slow, due to traffic congestion. In this case, the day and time of the day are more important attributes. Since attention mechanism is a powerful way to capture such correlations, we leverage it here, as follows:

$$z_i = a_i c_i \quad (2)$$

$$\alpha_i = \frac{\exp(z_i)}{\sum_{i \in A} \exp(z_i)} \quad (3)$$

$$v_{attr} = \sum_{i \in A} \alpha_i a_i, \quad (4)$$

where α_i is the weight for the attribute vector a_i , $A = \{c, d, t, r, g\}$. Parameter c_i is used to calculate an intermediate score z_i . We characterize our attention mechanism with $\{a_i\}$ and $\{c_i\}$, and v_{attr} denotes the output of our attention mechanism and this module.

3.4 Sub-path Representation Module

The Sub-path Representation Module is the encoder for the input trajectory. First, the Trajectory Point Embedder is used to embed the trajectory point information to a higher-dimension representation. Then, the Tri-CNN Encoder, which is an encoder with three-channel output, is utilized to extract high-level abstract sub-path features based on roads used in the trajectory. To attend to the important channel of sub-path features, we design a Sub-path Attention mechanism, inspired by [1, 29]. Finally, we use a Local Distance Representation encoder to create a more accurate representation for the sub-path distance.

Trajectory Point Embedder. The input of this module is a sequence of trajectory points T_x . For each point x_i in T_x , we map-match it to a road using [23] and embed its road ID to a vector. Due to the fact that there are often thousands of roads in large cities, using traditional one-hot encoding results in a vector too large. As such, we use the idea of condensed word vector from natural language processing and convert it to $\mathbb{R}^{32}$ for an accurate representation while saving space. Since the GPS coordinates of each point reveal fine-grained information of the vehicle, we seek to ensure that their impact are not buried by the large embedded road ID

vector. This is achieved by converting a point's latitude to $\mathbb{R}^{16}$, and doing the same for its longitude. We embed the vehicle status to $\mathbb{R}^2$. We then concatenate the embedded vectors and convert it to $\mathbb{R}^{16}$ using a non-linear layer, as follows:

$$u_a = \tanh(W_c x_i + b_c), \quad (5)$$

where W_c and b_c are the parameters of the embedder.

Tri-CNN Encoder. This encoder is used to extract high-level abstract sub-path features, while mitigating the noise incurred during the recording process. The rationale is that trajectory recording is a noisy process that often results in imprecise GPS coordinates recorded. Map-matching, on the other hand, is capable of estimating precisely the roads used during the trajectory. As such, by integrating the GPS coordinates of its map-matched road, it helps in extracting high-level abstract sub-path features in a more accurate fashion. To this end, we create u_s and u_e using the same procedure as Equation 5. The difference is that instead of using the $x_i.lat$ and $x_i.lng$ as inputs during the creation of u_s (resp. u_e), the latitude and longitude of the start (resp. end) point of the road x_i belongs to are used.

Inspired by DeepTTE [26], we then utilize a 1D convolutional neural network (CNN) to create a spatial representation for the sub-paths in u_a . CNNs have been widely used in various image recognition tasks [20, 22] for their power in capturing spatial and temporal dependencies. For tasks where the input is a 1D-sequence, e.g. sentence classification [14], the 1D-CNN is very effective in extracting local subsequences of data instances and derive patterns within each subsequence. As such, it is applicable to our context of path travel time estimation, where the 1D input sequence (resp. subsequence) is a trajectory (resp. sub-path) and each GPS point is a data instance. In particular, 1D-CNN is effective in capturing finer-grained information like turns at road junctions [26]. Typically, a 1D-CNN contains a filter of a fixed width that slides across the data instances to capture patterns within each sub-sequence. In our model, the formulation is as follows:

$$v_a = \text{ELU}(W_a * u_a), \quad (6)$$

where $W_a \in \mathbb{R}^{k \times d \times 16}$ is the parameter, $*$ is the convolution operation, k is the filter width and d is the output dimension. ELU is the activation function [4], where $\text{ELU}(x) = \exp(x) - 1$ for $x \leq 0$ and x for $x > 0$. We then use the same 1D-CNN to create another sub-path spatial representation v_s (resp. v_e) from u_s (resp. u_e) respectively. **Sub-path Attention.** Different CNN outputs may have different importance. If T is very noisy, a greater importance should be assigned to v_s or v_e , instead of v_a . As such, we pass the output of the Tri-CNN Encoder through our Sub-path Attention mechanism. This attention mechanism is similar in spirit to the Attribute Attention proposed in Section 3.3, with one exception. The formulation is as follows:

$$y_i = \sum_{j=1}^{n-k} v_{i,j} f_i \quad (7)$$

$$\beta_i = \frac{\exp(y_i)}{\sum_{i \in S} \exp(y_i)} \quad (8)$$

$$v_m = \sum_{i \in S} \beta_i v_i, \quad (9)$$

where β_i is the weight for v_i , $S = \{s, a, e\}$, and $\sum_{i \in S} \beta_i = 1$. To learn β_i , we use a parameter vector f_i . Note that unlike Attribute Attention, the intermediate attention score y_i is calculated by the sum of the inner product between $v_{i,j}$ and f_i across all j , where $v_{i,j}$ denotes the j -th sub-path vector of v_i .

Local Distance Representation. The sub-path distance is an important information. As mentioned in Section 3.3, supporting it by the integration of fine-grained distance may create a more accurate representation of the distance traveled. To this end, we consider two inputs, the coarse-grained local distance of the sub-paths and its fine-grained version. We denote these by $l_c, l_f \in \mathbb{R}^{n-k}$ respectively. Similar to Section 3.3, we convert each of l_c and l_f into $\mathbb{R}^{(n-k) \times 16}$ using a linear layer. Then, we perform a concatenation on l_c and l_f , and convert it to $\mathbb{R}^{(n-k) \times 16}$ using another linear layer. Finally, we concatenate this vector with v_m of the Spatial Representation Attention. We denote the final output of this module as m .

3.5 Sub-path Correlation Module

Correlation Component. Travel time is highly dependent on the path attributes, as mentioned earlier. To capture the inter-dependencies between the attributes and the sub-paths, we design a Correlation component where v_{attr} is first concatenated with the representation of each sub-path in m , before passing through a linear layer, as follows:

$$s_i = W_s[m_i, v_{\text{attr}}] + b_s, \quad (10)$$

where m_i is the representation of the i -th sub-path, $1 \leq i \leq n - k$, and W_s and b_s are the parameters of this operation.

LSTM Layer. To further capture the temporal dependencies among sub-paths, we feed the results into long short-term memory (LSTM) network, a type of recurrent neural network (RNN). Generally, RNNs enable the processing at every step of a sequential input to leverage information from the previous steps. In particular, LSTM is capable of learning longer-term dependencies by using a memory cell, and resolving the issue of vanishing and exploding gradient in standard RNNs. Briefly speaking, the hidden state h_t and memory cell c_t are function of h_{t-1} and c_{t-1} from the previous step, and input vector s_t , or formally:

$$c_t, h_t = \text{LSTM}(c_{t-1}, h_{t-1}, s_t), \quad (11)$$

LSTM uses an input and a forget gate to control the information flow within the recursive operation. We also use two stacked LSTM to allow greater model complexity. For more details about LSTM, we refer to [10]. We denote the output of this layer as h_{sub} .

3.6 Joint Prediction Module

This module jointly learns the main task of path travel time estimation, with an auxiliary task of sub-path travel time estimation, inspired by [2, 26]. The intuition is that the travel time of a path is essentially made up of the sum of its sub-path travel times. By accurately estimating the sub-path times, the path travel time can be more accurately predicted, vice versa.

Sub-path Aggregation. Due to length variability of h_{sub} , we convert it to a fixed-length vector via pooling. While mean-pooling treats all sub-paths equally, attention-pooling gives more attention to critical sub-paths, e.g. many traffic lights or road segments [26].

As such, we utilize an attention mechanism that is similar in structure as Attribute Attention of Section 3.3. However, instead of an arbitrary parameter vector, we convert v_{attr} via a linear layer to a vector that is similar in size as a vector in h_{sub} , and use it as the parameter. The output of the aggregation is denoted by r_{path} .

Main and Auxiliary Task Learning. To generate the estimated path travel time, we use deep residual learning. Deep residual learning is a technique for training of very deep neural networks and has shown superior performance across multiple recognition [9] and certain geo-spatial tasks [34]. The intuition is that each residual unit focuses on fine-tuning the output from the previous unit by only learning an added "residual". This is achieved through shortcut connections across the units. Such designs mitigate the degradation issue of deep neural networks where adding more layers results in lower accuracies, and makes it very effective to train deep networks for maximal representation power. Specifically, we pass the concatenated vector $[r_{\text{path}}, v_{\text{attr}}]$ through a linear layer, followed by a series of residual units. The formulation of a residual unit is:

$$r_l = r_{l-1} + \text{ReLU}(W_l r_{l-1}), \quad (12)$$

where r_{l-1} denotes the output from the $(l-1)$ -th residual unit. Also, $1 \leq l \leq L$, where L is the total number of residual units. ReLU is the activation [6]. W_l is the learnable parameters of this unit. Finally, we pass r_L through a linear layer to get the estimated path travel time $t_{\text{path}} \in \mathbb{R}^1$. The training objective for this task is:

$$C_{\text{path}} = \left| \frac{t_{\text{path}} - g_{\text{path}}}{g_{\text{path}}} \right|, \quad (13)$$

where g_{path} is the ground-truth path travel time.

To get the estimated travel time $t \in \mathbb{R}^{(n-k) \times 1}$ for the sub-paths, we simply pass h_{sub} through several fully-connected linear layers. The objective for this task is:

$$C_{\text{sub}} = \frac{1}{n-k} \sum_{i=1}^{n-k} \left| \frac{t_i - g_i}{g_i} \right|, \quad (14)$$

where t_i (resp. g_i) denotes the estimated (resp. ground-truth) value for the i -th sub-path.

For optimization during training, we use the following loss function:

$$C_{\text{join}} = \zeta C_{\text{path}} + (1 - \zeta) C_{\text{sub}}, \quad (15)$$

where ζ is a weighing factor that considers the relative importance of the error of each task.

4 EXPERIMENTS

To ascertain the effectiveness of our proposed method, we conduct experiments on real-world datasets. Specifically, we focus on the application of estimating public bus travel time using a hybrid dataset of public bus and taxi trajectories.

4.1 Experimental Set-up

This section describes the datasets used, the baselines we consider for comparisons, and the evaluation metrics.

Datasets. For a fair validation process and to construct a hybrid trajectory dataset, we require datasets obtained from the same geographical space. As such, we make use of the following datasets:

- **Singapore Smart Card Dataset:** This dataset contains passenger trip records obtained from CEPAS card records from Singapore’s public bus network. After pre-processing, there are 204,000 trajectories in all. We explain the preprocessing steps below.
- **Singapore Taxi Dataset:** We use a dataset that has 11.2M taxi trajectories. Each point within the trajectory contains the GPS coordinates, the time instance at which it was sampled, and the status of the taxi (i.e. busy, idle, stop, on-call, or hired) at that time instance. Each trajectory also includes the following information: a vehicle ID, time of day, and day of week. The longest trajectory has 128 points.

We further collect city-scale rainfall records; a total of 5 distinct levels, from Singapore. All the above datasets are obtained over an overlapping time period in March 2016.

Pre-processing: Bus Trajectory Derivation from Smart Card Dataset. A passenger trip record in the smart card dataset consists of the following features: origin stop, destination stop, bus number, tap-in time, and tap-out time. Specifically, tap-in (resp. tap-out) time refers to the time at which the passenger boards (resp. alights) the bus at origin (resp. destination) stop by tapping in (resp. out) at the on-bus fare gantry. A bus journey starts from a bus terminal, passes a series of scheduled bus stops, and ends at a bus terminal. Unlike taxis, public buses generally follow scheduled routes. Through basic sorting and processing, we derive a trajectory for each bus journey in the dataset, where each point within the trajectory contains the GPS coordinates of the bus stop, as well as the timestamp of the first tap-in at that stop during the journey. We obtain the exact on-road distance between any two consecutive stops traveled by the bus from the publicly accessible bus schedules. As such, we can further derive the on-road distance between any two consecutive GPS points of the trajectory, as well as the total on-road journey distance. Since the date and time of the journey are recorded, we can get the following attributes required by our model: day of week, and time of day. Finally, instead of using the vehicle states of taxis, we assign a constant bus state to each GPS point of the bus trajectory to further differentiate bus trajectories from taxi ones.

Data Characteristics. Table 1 shows some statistics of the trajectory datasets. Observe that the typical bus data is vastly different from the taxi data. The mean, 25%-ile and 75%-ile travel times show that the total trip duration of a bus trajectory is typically several times that of a taxi’s. Bus travel time also has a larger variance, and the time-gaps between consecutive trajectory points are larger. Moreover, the points in a taxi trajectory are sampled at a regular frequency, but the points in a bus trajectory are sampled when the bus is at the bus stops.

Training, Validation and Testing Sets. We create a bus training set Bus-Training by sampling 70% of the bus lines in Singapore Bus Dataset and collecting all the trajectories of these lines. Similarly, we sample 10% (resp. 20%) of the remaining bus lines for the validation (resp. testing) data, denoted as Bus-Validation (resp. Bus-Testing). We also create a taxi training set Taxi-Training, by randomly sampling from the Singapore Taxi Dataset a number of trajectories that is 5 times that of the number of trajectories in Bus-Training. Finally, we create a hybrid dataset Hybrid-Training

Table 1: Trajectory Data Characteristics

Statistical Index	Bus	Taxi
Travel Time Mean (s)	3,245	692
25%ile Travel Time (s)	1,841	283
75%ile Travel Time (s)	4,192	933
Travel Time Std (s)	1,756	573
Time Gap Mean (s)	204	34
Sampling Interval (s)	-	30

by combining Bus-Training and Taxi-Training. This is done by simply putting the bus and taxi trajectories in a common dataset. For example, if Bus-Training = $\{T_1, T_2\}$ and Taxi-Training = $\{T_3, T_4, T_5\}$, where each T_i , $1 \leq i \leq 5$ denotes a trajectory instance, Hybrid-Training = $\{T_1, T_2, T_3, T_4, T_5\}$. We repeat the above steps for 5 times to get 5 different sets of data.

Evaluation Metrics. For our experiments, we adopt the following evaluation metrics: mean absolute percentage error (MAPE), mean average error (MAE), and root mean square error (RMSE).

Implementation Details and Parameter Setting. We implement our model on Pytorch 0.4.1. Our model training/evaluation is done on Nvidia Tesla M40 GPU, and Linux machine with an Intel Xeon 2.6GHz CPU and 64GB RAM. We use Adam optimization algorithm to fine-tune the parameters. The learning rate we use is 0.001 and we train our model in batches of 400 over 100 epochs. For hyper-parameters settings, we set $k = 3$ and $c = 32$ in the Sub-path Representation Module. The embedding size of the attributes or inputs are chosen based on empirical testings. In the LSTM layer of our Sub-path Correlation Module, we use the hidden vector size of 128. In the Correlation component, we use an output size that is the same as the input. In the Joint Learning Module, we set the number of residual units used as 3; the size for each unit is 128. We also set the number of fully-connected layers used in the auxiliary task prediction as 1. Finally, we set ζ as 0.7, for a stronger focus on the main task loss function. To provide sufficient clout for the relatively smaller bus trajectory sets, we also associate a constant vehicle ID for all the bus trajectories.

Baselines. We consider several baselines in our experiment. Each baseline has a superscript *, † or ‡, which denote that the model uses Bus-Training, Taxi-Training, Hybrid-Training for training respectively. In all cases, Bus-Validation and Bus-Testing are used for validation and testing. The baselines are as follows:

- **DeepTTE*** As there is no prior work on hybrid-trajectory-based estimation, we compare with the state of the art single-source method. DeepTTE [26] is chosen as it is the best-performing one to date, yet is also modifiable to support hybrid trajectories. Specifically, instead of considering the driver ID feature in this baseline, we use vehicle ID to differentiate bus trajectories from taxi ones. Instead of using the weather feature that is required in DeepTTE, we replace that feature with the rainfall level present in our datasets. In this baseline, only Bus-Training is used for model training to demonstrate the effectiveness when only a single-source small trajectory dataset is available.

Table 2: MAPE, MAE, and RMSE on AtHy-TNet and the baselines. Best results are in bold face and second best underlined.

Method	MAPE (%)	MAE (s)	RMSE
DeepTTE*	15.16	367.16	374.51
DeepTTE [†]	41.93	1244.69	1433.67
DeepTTE [‡]	13.15	350.60	358.26
LightGBM [‡]	14.60	393.83	544.25
HyMLP [‡]	17.40	497.70	608.60
HyLSTM [‡]	14.81	388.70	406.70
AtHy-TNet*	13.66	329.45	342.0
AtHy-TNet-a [‡]	<u>10.91</u>	<u>288.32</u>	<u>295.73</u>
AtHy-TNet [‡]	10.24	264.21	273.12

- **DeepTTE[†]** This baseline is similar to DeepTTE* except that Taxi-Training is used for training. It is chosen to explore the effectiveness of solely utilizing a larger trajectory dataset, but of different characteristics.
- **DeepTTE[‡]** This baseline is similar to the previous two, except that Hybrid-Training is used for training. This baseline serves to show how estimation changes when the training dataset uses not only a small single-source dataset, but also a larger one obtained from the same geographical space.
- **LightGBM[‡]** LightGBM is one of the most efficient and high-performing gradient boost decision tree method [13]. The trajectory input for this baseline is the same as AtHy-TNet. Apart from the total path distance, the attributes of the trajectory are specified as categorical features in the program.
- **HyMLP[‡]** This baseline is a simple 5-layer fully-connected network with ReLU activation. The input is the same as AtHy-TNet and each categorical attribute in the input is embedded to a low-dimensional vector.
- **HyLSTM[‡]** This baseline uses a 2-stacked LSTM. The input is the same as AtHy-TNet, except that neither sub-path representation nor attribute correlation is considered. Similar to HyMLP[‡], each categorical attribute is embedded to a low-dimensional vector, before feeding into the LSTM. The estimation is obtained by mean-pooling the LSTM output and passing through residual fully-connected layers.
- **AtHy-TNet*** This baseline uses AtHy-TNet on Bus-Training instead of Hybrid-Training. The purpose is to demonstrate the raw improvements brought about by our model when a single-source small dataset is used.
- **AtHy-TNet-a[‡]** This baseline is a simplified model of AtHy-TNet. Instead of using the Sub-path Representation Module, AtHy-TNet-a[‡] uses the Geo-Conv Layer proposed in [26].

4.2 Evaluation Results

We conduct experiments on the five sets of {Bus-Training, Taxi-Training, Hybrid-Training, Bus-Validation, Bus-Testing} described earlier, and record the average testing results on Bus-Testing in Table 2. As observed, methods that utilize hybrid trajectories perform better than their single-source versions. For example, the

performance of DeepTTE[‡] is at least 2% stronger than DeepTTE* and DeepTTE[†]. Despite the fact that Taxi-Training is much larger than Bus-Training and supposedly not as affected by data sparsity, performance of DeepTTE[†] is significantly worse than DeepTTE*. This shows that simply replacing a dataset with another results in large errors, due to the vast differences across different sources. AtHy-TNet-a[‡] shows a stronger performance than all other hybrid-data methods by at least 2%. This demonstrates effectiveness of our model in estimating across hybrid trajectories, and by considering correlations of the relevant attributes. The performance of AtHy-TNet[‡] is a further improvement from AtHy-TNet-a[‡] by around 0.7%. This demonstrates the benefits of extracting high level sub-paths spatial features through fine-grained road-based information. As such, our experiment results show that the effectiveness of our model in learning multi-nature correlations of hybrid trajectory data. Note that AtHy-TNet* demonstrates a performance stronger than DeepTTE* by 1.5%, which shows that our method is effective even when single-source training data is used.

Comparison with Google Maps. We also compare AtHy-TNet with Google Maps API [7]. Adopting the treatment in [27], we query Google Maps Directions API at the same time and weekday as the starting time of each trip, by assuming a strong weekly periodicity. Since the API does not support the querying of past trips, all the queries correspond to the current or future time. As arbitrary public bus routes outside of the existing bus lines are unsupported in the query, the travel mode is set as Driving. Due to credit limits, 30k trips within a set of Bus-Testing are randomly chosen for comparison. The MAPE, MAE, and RMSE are 20.9%, 491, and 647 respectively. In contrast, the errors are 11.24%, 237, and 241.3 respectively when the same trips are estimated using AtHy-TNet. A reason why Google Maps does not perform as well may be that the data sources for its estimations in the Driving mode are largely based on non-public-bus vehicles. Since the travel time of arbitrary public bus routes are unsupported in online map services, it further highlights the necessity of a solution for vehicles with small trajectory datasets.

Effect of Varying Taxi and Bus Data Size in Hybrid-Training. Next, we keep the bus trajectories in Hybrid-Training constant and vary the concentration of taxi trajectories. Specifically, we consider taxi data that is K times the size of the bus data. We evaluate the performance of AtHy-TNet[‡] over different values of K , using Bus-Validation and Bus-Testing. The results are shown in Figure 4. As observed, the performance is significantly worse when no taxi trajectories are used in the hybrid training data, but improves rapidly when they are introduced. As early as $K = 1$, the decrease in error is more than 2%. The performance strengthens with increasing K and settles at its peak from $K = 4$ onward. The maximum achievable improvement is nearly 4%. This demonstrates benefits of hybrid data, and capacity of AtHy-TNet[‡] to effectively utilize hybrid data, even when the external data used is small. Error tends to be larger when K is small, since a smaller external taxi data often implies a smaller extended area coverage. As such, estimation does not improve as much as when larger taxi data/coverage is considered in the hybrid data.

We also vary the concentration of bus trajectories while keeping the taxi trajectories in Hybrid-Training constant. The results are

Figure 4: Performance over different K values

Figure 5: Performance over different bus data size

Figure 6: Performance of AtHy-TNet over different ζ

shown in Figure 5. As observed, the accuracy improves as the bus trajectory data size increases. The reason is that when the bus data is small, the area coverage of buses that can be used for correlation-learning with taxis tends to be small too. As such, the error is larger as compared to when more bus data is available for correlation-learning. Nonetheless, AtHy-TNet[‡] demonstrates a satisfactory accuracy using bus data sizes even as small as 20k and 40k. This shows that our method is capable of leveraging external data for estimation to address the inaccuracy brought about by a single-source small dataset. Moreover, in both cases, AtHy-TNet[‡] shows a stronger performance and a steeper descent over increasing data size than DeepTTE[‡]. This shows the suitability of our model over prior art in effectively modeling across hybrid trajectories.

Performance across Different ζ in the Joint Learning Module. Figure 6 shows how the performance of AtHy-TNet varies across different values of ζ , from 0.01 to 1.0. As observed, the performance is satisfactory in general across all values. Although the performance remains around a constant level in the middle of the range, the errors are slightly larger at the ends, i.e. 0 and 1.0. This demonstrates the benefits of learning the main task of path travel time estimation with an auxiliary one and considering the errors of both during training.

Performance across Different Trip Travel Times. Figure 7 compares the model performance across trajectories of different trip durations. Generally, AtHy-TNet consistently shows a stronger performance than DeepTTE and LightGBM across all trip durations. It is interesting to note that the error tends to decrease as trip duration increases for all methods. The reason may be that unlike a short trip, a long trip usually has more constituting sub-paths. Since each sub-path provides valuable information that is useful for estimation, having more sub-paths means the longer trips have more information usable by the methods. This results in the travel time of these trips to be estimated more effectively than the shorter ones. Furthermore, the rate of error-decrease for AtHy-TNet is the fastest

Figure 7: Performance over different travel times

as compared to other methods. Specifically, error decreases from 16% to around 11% when the trip duration increases from 0-1000s to 1000-2000s, stays constant at 2000-3000s, before further decreasing to around 8% for durations within 3000-5000s. In contrast, the errors of DeepTTE stay nearly constant during the shorter trips and only starts to decrease towards the longer ones (3000-5000s).

Performance across Different Days of the Week. Figure 8 shows the performance of AtHy-TNet and baselines across the different days of the week where 1 represents Monday and 7 represents Sunday. As observed, AtHy-TNet out-performs DeepTTE and LightGBM across all days. It is interesting to know that the trend of errors' variation across the week is similar across all three methods. Specifically, error increases almost linearly from Monday to Sunday, with dips on Wednesday and Friday. Estimation errors are higher on weekends than weekdays since the travel patterns of passengers are less regular. It may also be because the number of trajectories collected on weekends is generally smaller as there are fewer weekend days in a week. The rate of increase for AtHy-TNet, as shown by the gradient of the slope, is slow as compared to the other two methods. Furthermore, its errors fall within the satisfactory performance range of 9-11%. This demonstrates the performance robustness of AtHy-TNet across different days of the week.

Performance across Different Times of the Day. We also evaluate the estimation performance across different times of the day. We conduct the test separately for weekdays and weekends, since the travel characteristics of weekdays and weekends are largely different. The results are shown in Figure 9 and 10 respectively, based on the 24-hour clock. As observed, AtHy-TNet out-performs the baselines across all times of the day for both cases. The errors of AtHy-TNet also show a smaller variance in both cases. Similar to Figure 8, the MAPEs during the weekends are mostly higher than during the weekdays. For both weekdays and weekends, errors remain relatively constant for all methods during the later parts of the day, i.e. 9 hrs onwards. Note that the error of the earliest period, i.e. 0-3 hrs, of weekdays is higher than that of the later periods. In contrast, the error over the same time period during weekends is lower than the later ones. The reason for such discrepancy may be because there are fewer bus services that run at that period. As such, there are much fewer trajectories available for training and testing, and this results in a large variance in errors between weekdays and weekends. Furthermore, specific to AtHy-TNet, the error during the 6-9 hrs of weekdays is the highest as compared to the rest of the day. This is reasonable since the 6-9 hrs period is the morning peak hours when traffic congestions are rampant, due to people going

Figure 8: Performance over different days of the week

Figure 9: Performance over different times of the day (weekday)

Figure 10: Performance over different times of the day (weekend)

to a common set of places. As a result, the difficulty of estimation is higher. Compared to the baselines, this observation also shows that our model manages to reveal such temporal-dependencies in the datasets.

Attribute/Feature Importance Evaluation. We study the effect of each of the attributes used in AtHy-TNet, by evaluating the performance of AtHy-TNet when only that attribute is excluded from the model. When the day-of-week attribute is excluded, the MAPE worsens by 0.46%. When time-of-day attribute is excluded, the MAPE worsens by 0.99%. This is aligned to our intuition that traffic complexities, e.g. traffic congestions, are different in different times of the day/week. As such, temporal attributes are important to model such differences. When the rainfall attribute is excluded, MAPE increases by 0.43%. This is coherent with the well-studied fact that vehicles tend to move slower on rainy days. To study the effect of differentiating bus trajectories from taxi ones, we exclude vehicle ID. Recall that the bus states used are generally different from the taxi ones. As such, we also exclude the vehicle state of each GPS point. As a result, the MAPE worsens by 0.96%. This highlights the importance of differentiation of vehicle types when dealing with hybrid trajectories, since vehicles of different types may have distinct travel behaviours.

Predicting Time. The predicting time of AtHy-TNet is fast, in spite of its slower training time. The time taken to estimate a batch of 400 trajectories is around 0.023s.

4.3 Transfer Applications and Limitations.

AtHy-TNet is the first work that demonstrates the benefit of using hybrid dataset for a task. A possible transfer application of the method is to use it for the trajectory classification problem of a vehicle type that has a small dataset, e.g. classify bus trajectories based on the degree of in-bus crowdedness. Due to privacy and legal concerns, it is often hard for one operator, e.g. taxi company,

to share its data with another, e.g. bus company. As such, another potential extension is the use of federated learning for the training process. For example, instead of training on a hybrid dataset, the model may first train on the taxi dataset, before fine-tuning by training another time on the bus dataset. In this case, only the model needs to be shared among the companies while the datasets are kept private.

A limitation of AtHy-TNet is that it is a preliminary attempt at demonstrating the effectiveness of using hybrid-trajectory data. As such, the model can be improved in several ways, like include finer-grained road information such as regulated speed limits, and if the road has electronic road tolls installed. Furthermore, AtHy-TNet deals with the estimation of a single type of vehicles when the single-source dataset is small. An extended exploration on leveraging hybrid datasets for multi-modal path travel time estimation may be useful, to public organizations for traffic studies, and to commuters for multi-modal trip planning.

5 RELATED WORK

Travel time estimation using trajectory data are generally classified into two categories: 1) Link-based, and 2) Sub-path-based methods.

For the link-based methods, travel time estimation is carried out for the individual road segments [12, 18, 19, 25, 28, 31], before these times are combined to form the travel time of the whole path. Yang et al. utilize spatio-temporal Markov models to model traffic behaviours and forecast the travel time at all the road segments [31]. In [25], Wang et al. make use of regression tree and probabilistic graphical methods based on some correlations observed from the traffic data. In [28], the authors propose a deep learning framework that predicts traffic speed of road segments using spatio-temporal information of contiguous road segments. Rahmania et al. propose a fixed point formulation that simultaneously infers route choice and travel time [19]. Jenelius et al. [12] use a multivariate probabilistic principal component analysis model that is able to model correlation patterns between days. One limitation about such methods is that they do not consider certain travel complexities on the path level, such as traffic lights, and turn costs.

Sub-path-based methods mitigate this limitation to some extent [5, 16, 30, 32]. The key idea is that path travel times can be estimated by considering historical path/sub-path travel times, which implicitly consider costs that are not modeled in the link-based methods. In [32], Yuan et al. propose a time-dependent landmark graph to model the time-varying traffic patterns, and techniques to compute the fastest path based on this graph and the experience of drivers. In [30], the authors utilize a tensor decomposition approach to resolve the issue of data sparsity, and a dynamic programming method to find the optimal concatenation of trajectories for path travel time estimation. Dai et al. propose the hybrid graph paradigm and techniques to address the issue of non-standard travel time distribution, data sparseness and the dependencies among roads travelled during a path [5]. Li et al. further consider scenarios where only a small number of probe vehicles is used to collect the data and travel times of the same path may have large variances [16]. More recently, deep learning has shown further success for this task. Wang et al. propose DeepTTE, which models the path travel time of taxis in an end-to-end fashion, while considering

the spatio-temporal and attribute characteristics [26]. Zhang et al. partition the road network into grids, and consider various short and long-term features [33].

On the broader scale, there also exists work that estimates travel time given the input of an origin-destination (OD) pair [11, 15]. However, while these methods are advantageous in faster computations, they have a limited consideration in terms of the vehicle path choice variabilities between the origin and destination, which the travel time of a vehicle is highly dependent on. Furthermore, deep learning has been used in various spatio-temporal problems, such as crowd-flow prediction [34], transportation mode prediction [21] and trajectory predictions for road agents [3, 8, 17, 24, 35]. In general, utilizing deep learning for path travel time estimation is a new field, and there has not been any work which seeks to perform this task across hybrid trajectories.

6 CONCLUSION

In this paper, we propose an Attribute-related Hybrid Trajectories Network capable of estimating path travel times by effectively utilizing hybrid trajectory data. This method implicitly models the correlations across hybrid trajectories, something largely ignored in existing work. It also extracts the correlations of attributes that affect path travel time and leverage on-road information to extract sub-path features. Through experiments, we demonstrate the superior performance of our method over the state of the art methods. For future work, we will explore how path travel time estimation can be used in identifying real-time traffic conditions.

7 ACKNOWLEDGEMENTS

This work is supported in part by National University of Singapore under the SUG Grant. It is also supported in part by MOE, Singapore under grant MOE2015-T2-2-069. We gratefully acknowledge the support of NVIDIA AI Tech Center (NVAITC) for our research.

REFERENCES

- [1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
- [2] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–57.
- [3] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. 2018. TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. *arXiv preprint arXiv:1812.04767* (2018).
- [4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).
- [5] Jian Dai, Bin Yang, Chenjuan Guo, Christian S Jensen, and Jilin Hu. 2016. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment* 10, 3 (2016), 85–96.
- [6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.
- [7] Google. 2019. Google Maps Platform. <https://developers.google.com/maps/documentation/>
- [8] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2255–2264.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [11] Jilin Hu, Chenjuan Guo, Bin Yang, Christian S Jensen, and Lu Chen. 2018. Recurrent Multi-Graph Neural Networks for Travel Cost Prediction. *arXiv preprint arXiv:1811.05157* (2018).
- [12] Erik Jenelius and Haris N Koutsopoulos. 2018. Urban network travel time prediction based on a probabilistic principal component analysis model of probe data. *IEEE Transactions on Intelligent Transportation Systems* 19, 2 (2018), 436–445.
- [13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.
- [14] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
- [15] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.
- [16] Yang Li, Dimitrios Gunopulos, Cewu Lu, and Leonidas Guibas. 2017. Urban Travel Time Prediction using a Small Number of GPS Floating Cars. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 3.
- [17] Yuxin Ma, Xinge Zhu, Sibao Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. 2018. TrafficPredict: Trajectory prediction for heterogeneous traffic-agents. *arXiv preprint arXiv:1811.02146* (2018).
- [18] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. 2012. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 595–604.
- [19] Mahmood Rahmani, Haris N Koutsopoulos, and Erik Jenelius. 2017. Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach. *Transportation Research Part C: Emerging Technologies* 85 (2017), 628–643.
- [20] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [21] Xuan Song, Hiroshi Kanazaki, and Ryosuke Shibasaki. 2016. DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level. In *IJCAI*, Vol. 16. 2618–2624.
- [22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
- [23] Youze Tang, Andy Diwen Zhu, and Xiaokui Xiao. 2012. An efficient algorithm for mapping vehicle trajectories onto road networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 601–604.
- [24] Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–7.
- [25] Dong Wang, Wei Cao, Mengwen Xu, and Jian Li. 2016. Etcps: An effective and scalable traffic condition prediction system. In *International Conference on Database Systems for Advanced Applications*. Springer, 419–436.
- [26] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. AAAI.
- [27] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 19.
- [28] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 499–508.
- [29] Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*. 606–615.
- [30] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 25–34.
- [31] Bin Yang, Chenjuan Guo, and Christian S Jensen. 2013. Travel cost inference from sparse, spatio temporally correlated time series using Markov models. *Proceedings of the VLDB Endowment* 6, 9 (2013), 769–780.
- [32] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.* 25, 1 (2013), 220–232.
- [33] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DeepTravel: a Neural Network Based Travel Time Estimation Model with Auxiliary Supervision. *arXiv preprint arXiv:1802.02147* (2018).
- [34] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In AAAI. 1655–1661.
- [35] Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, and Feng Zhu. 2018. On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning based Approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1413–1422.

List of Author's Awards, Patents, and Publications

Awards

- **Best Paper Nomination**, “A Scalable Approach to Inferring Travel Time in Singapore’s Metro Network using Smart Card Data” *IEEE International Conference for Smart Cities 2018*.
- **CIKM Travel Grant**, “Attribute-related Hybrid Trajectories Network for Path Travel Time Estimation” *ACM International Conference for Information and Knowledge Management 2019*.

Conference Proceedings

- **Xi Lin**, Xiaokui Xiao, and Zengxiang Li, “A Scalable Approach to Inferring Travel Time in Singapore’s Metro Network using Smart Card Data” in *IEEE International Conference for Smart Cities 2018*.
- **Xi Lin**, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S. Bhowmick, “Attribute-related Hybrid Trajectories Network for Path Travel Time Estimation” in *ACM International Conference for Information and Knowledge Management 2019*.

Papers in Submission or under Review

- **Xi Lin**, Yequan Wang, Shen Ren, Xiaokui Xiao, Zengxiang Li, and Sourav S. Bhowmick, “Attribute-guided Multi-graph Convolutional Network for Taxi Demand Prediction” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2020*.
- **Xi Lin**, Xiaokui Xiao, and Zengxiang Li, “Estimating Travel Times in Singapore’s Metro Network with Scalability using Smart Card Data” in *IET Intelligent Transport Systems 2020*.