LOW-DIMENSIONAL MODELS FOR COMPRESSION, ESTIMATION AND PREDICTION OF LARGE-SCALE TRAFFIC DATA

Nikola Mitrovic

School of Electrical & Electronic Engineering

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

2016
Statement of Originality

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

................................. .................................
Date                           Nikola Mitrovic
Acknowledgements

To my Mother who passed away before this dissertation was completed and who has always supported, encouraged and believed in me.

I would like to express my sincere gratitude to my advisor, Professor Justin Dauwels for his support and excellent guidance from the very moment when I joined Nanyang Technological University (NTU). I was truly impressed by Prof. Dauwels’s consistent dedication to research, teaching and mentoring. It would not be an overstatement if I say that Prof Dauwels is my Guru who helped me discover my research potentials. I am very grateful for that.

I would also like to thank our collaborator Prof. Patrick Jaillet for his constructive suggestions and insightful discussions.

Special thanks go to Muhammad Tayyab Asif for his continuous support, encouragement and fruitful discussions. Also, many thanks to all of my lab mates and friends for providing me with an excellent environment for doing research.

Last but not least, I would like to thank my father, Bratislav, my sister, Milena, and my girlfriend, Lea, who stood by me through the good times and bad. I would also like to thank all of them for their constant and never ending love that has helped me complete this 4-year journey successfully!
Abstract

Low-dimensional models for compression, estimation and prediction of large-scale traffic data

by

Nikola Mitrovic

Supervisor: Asst. Prof. Justin Dauwels

1 Nanyang Technological University, Singapore

Intelligent Transportation Systems (ITS) often operate on large road networks and collect traffic data with high temporal resolution. The volume of the collected data severely limits the scalability of real-time traffic operations. We propose data-driven models that can help intelligent transportation systems in dealing with large amounts of traffic information in the context of online operations. We also propose non-centralized architectures for the implementation of the presented method on smartphones. Finally, we explore whether incorporating additional information (e.g., rainfall intensity) would lead to a better prediction and overall performance of the proposed system.

Our main objective is to develop generic low-dimensional models that can enhance the scalability of traffic operations in practical city-scale networks, which are typically composed of different types of roads. We propose a column based (CX) matrix decomposition that leads to low-dimensional models where the components correspond to individual road segments (links) in the network. The resulting models can be easily interpreted and used for compression, compressed sensing and prediction. Hence, traffic states of large networks can be efficiently estimated by observing a
small subset of road segments. To achieve this, we carefully select a small number of links from the original network. Then, we use historical information to learn the relationship functions between the selected subset of the links and the rest of the network. These functions allow us to estimate the traffic parameters (e.g., traffic speed) at any link in the network using information from a small subset of the links. Similarly, we perform traffic prediction for the whole network, by developing prediction models for only the representative subset of road segments.

We model large networks by observing traffic conditions on a few roads and then extrapolating this information (through straightforward vector-matrix multiplication) across the network. This approach seems to be suitable for non-centralized architectures in which the workload is shared between central servers and user devices. To this end, we propose decentralized and hybrid architectures for system implementation, based on smartphones (as user devices) since they have remarkable performance nowadays. We study the computational performance of decentralized and hybrid architectures for applications of traffic speed estimation and prediction, and travel time prediction. We also analyze the performance of various development Android platforms and smartphone devices for different sizes of the test network.

Weather conditions tend to have a significant impact on driving behavior and traffic performance. Assessing this impact might prove useful for the proposed low-dimensional models since they heavily rely on existing and future conditions at certain links in the network, which might be affected by inclement weather. However, quantifying this impact is not trivial. While light showers and a moderate amount of rainfall may not have any significant effect on road traffic, heavy showers, on the other hand, typically have a strong impact on driving conditions on roads. Furthermore, the resultant changes in traffic due to rain may vary during a day. We investigate the impact of rainfall on traffic conditions at different times of the day and for various rainfall intensities and road categories. In addition, we also investigate whether the incorporation of rainfall data would increase the performance of data-driven models for short-term traffic prediction.
In summary, the proposed column-based (CX) method demonstrates a significant reduction in prediction time without significant degradation in prediction performance. Regarding the system implementation, the proposed decentralized and hybrid architectures provide flexibility for app developers and offer an opportunity to offload significant load from busy central servers. More importantly, the proposed decentralized architecture significantly reduces the overhead of the communication network and paves the way for new cooperative traffic applications and operations. Finally, the numerical results illustrate that, for certain links, incorporation of rainfall information would lead to better prediction performance of the traffic stream parameters.
Author’s Publications

Journal Papers:


Nanyang Technological University

Singapore

**Conference Papers:**


Patent:

Contents

Acknowledgements i

Abstract iii

Author’s Publications vii

List of Contents xi

List of Figures xv

List of Tables xvii

1 Introduction 1

1.1 Motivation ........................................ 1

1.2 Related Work ....................................... 4

1.2.1 Low-Dimensional Models ...................... 4

1.2.2 Traffic App. ..................................... 6

1.2.3 Impact of Rainfall on Traffic Conditions and Prediction Performance .................. 7

1.3 Contributions and Outline of the Thesis .......... 9
## 2 Low-dimensional Models of Large Scale Traffic Data

2.1 Introduction ........................................... 14  

2.2 Column-based (CX) Decomposition .......................... 15  

2.2.1 Column Selection ................................... 16  

2.2.2 Relationship Matrix ................................. 18  

2.3 CX-based Method for Traffic Applications ................. 19  

2.3.1 Temporal Patterns in Urban Traffic Networks ............ 19  

2.3.2 Compression .......................................... 20  

2.3.3 Compressed Sensing .................................. 21  

2.3.4 Compressed Prediction ............................... 26  

2.4 Experimental Setup ..................................... 28  

2.4.1 Data Set ............................................. 28  

2.4.2 Support Vector Regression ............................ 29  

2.4.3 Performance Measures ............................... 31  

2.5 Results ................................................ 32  

2.5.1 Temporal Patterns in Urban Traffic Networks ............ 33  

2.5.2 Compression .......................................... 37  

2.5.3 Compressed Sensing .................................. 41  

2.5.4 Compressed Prediction ............................... 43  

2.6 Discussion ............................................... 48
CONTENTS

3 Implementation of CX Models on Smartphones

3.1 Introduction .............................................. 50

3.2 Column-based Method for Traffic Applications on Smartphone .... 53

3.2.1 Column-based (CX) Matrix Approximation .................. 54

3.2.2 CX Method for Traffic Speed Estimation ............... 54

3.2.3 CX Method for Speed Prediction ......................... 58

3.2.4 CX Method for Travel Time Prediction .................... 59

3.3 System Infrastructure ....................................... 60

3.3.1 Mode of Operations .................................. 60

3.3.2 Android Platforms .................................... 62

3.4 Experimental Setup .......................................... 63

3.4.1 Performance of Low-dimensional Models ................. 65

3.4.2 Computation Time ..................................... 67

3.5 Results .......................................................... 68

3.5.1 Performance of the CX Models for Traffic Applications .... 69

3.5.2 Computation Time of the App ............................ 73

3.6 Discussion ...................................................... 77

4 Impact of Rainfall on Traffic .................................. 79

4.1 Introduction .................................................... 80

4.2 Data Collection and Processing ................................ 83

4.2.1 Traffic data ................................................ 84

4.2.2 Rainfall Data .............................................. 84

NANYANG TECHNOLOGICAL UNIVERSITY  
SINGAPORE
4.3 Data Analysis ......................................................... 87
  4.3.1 Impact of the Rainfall on Level of Congestion ............. 87
  4.3.2 Impact of the Rainfall on Prediction Performance ........... 88
4.4 Results .................................................................... 92
  4.4.1 Impact of the Rainfall on the Level of Congestion .......... 94
  4.4.2 Impact of the Rainfall on Prediction Performance .......... 96
4.5 Discussion .............................................................. 98

5 Conclusion and future directions .................................... 103
  5.1 Conclusion ............................................................ 103
  5.2 Future Directions .................................................... 105

Bibliography ................................................................. 109
List of Figures

2.1 Compressed sensing. ........................................... 23
2.2 The idea of the compressed prediction. ....................... 25
2.3 Test network: City-scale network of Singapore. .......... 30
2.4 Assigned weights (importance) by Energy sampling method. 34
2.5 Assigned weights (importance) by SVD sampling method. 35
2.6 Performance of CX method for compression of traffic data. 36
2.7 Road importance in the network. .............................. 40
2.8 Road categories in study network. ............................ 41
2.9 Accuracy of compressed sensing. ............................. 42
2.10 Accuracy of the proposed and traditional methods. ....... 44
2.11 Performance of the proposed and traditional methods during a workday. 45
2.12 Prediction performance for different road categories. ....... 47
3.1 Decentralized non adaptive (D-NA) scenario. ............... 55
3.2 Decentralized adaptive (D-AD) scenario. .................... 57
3.3 Operation Modes ............................................. 61
3.4 Major components of the traffic app. ......................... 64
3.5 Test network for evaluation the performance of traffic app. ......... 65
3.6 Comparison of the computation times of the emulator and smartphone. 68
3.7 The performance of compressed and traditional prediction methods. . 71

4.1 Average rainfall in Singapore and other Pacific Rim cities. .......... 82
4.2 Test network with the category of road segments. ................. 85
4.3 Rain map of Singapore. ............................................. 86
4.4 Integration of the rainfall and weather data. ...................... 87
4.5 Average speed band in the network during rainy and non-rainy time instants for weekday (top) and weekend (bottom). ................. 93
4.6 SVM grid search method for parameter selection. ................. 100
4.7 Histogram of the links where rainfall data is useful (blue) and not useful (green) for Adaboost prediction method. ................. 101
4.8 Histogram of the links where rainfall data is useful (blue) and not useful (green) for bagging prediction method. ................. 102

5.1 Concept diagram showing the relationships between weather, accidents and traffic. .................................................. 106
5.2 Potential of smartphones for traffic applications. ................. 106
List of Tables

2.1 Consistency of road selection. ........................................ 38
2.2 Mean Squared Error (MSE) of compressed sensing. .............. 43
2.3 Mean Squared Error (MSE) of compressed prediction. ............ 46
2.4 Computation time. ......................................................... 48

3.1 Technical characteristics of the tested smartphones and the server. ... 69
3.2 Computation times of smartphone for application of estimation. ... 70
3.3 Computation time for compressed and traditional prediction methods. 72
3.4 Performance of the proposed and benchmark methods for travel time estimation. ................................................. 73
3.5 Computation time of different programming models and smartphones in the case of D-NA scenario and highway traffic network. .............. 74
3.6 Computation time of different programming models and smartphones in the case of D-NA scenario and large traffic network. .............. 74
3.7 The execution time of the app for the D-AD scenario and different networks ......................................................... 76
3.8 Required computation time [ms] that traffic app takes to assess the future speed for the list of links, followed by computation of travel time. ................................................. 77
4.1 Impact of the rainfall intensity on average congestion level in the network for different types of the roads, periods of the week and rainfall thresholds. ................................. 95

4.2 Impact of the rainfall intensity on average congestion level in the network for different periods throughout the day and type of the roads. 96

4.3 Prediction performance of SVM and boosting methods. ............... 98
Chapter 1

Introduction

1.1 Motivation

Enhancements in sensor technologies such as GPS probes have led to rapid developments in the field of Intelligent Transportation Systems (ITS). These systems gather traffic-related information from multiple sources and with high temporal resolution. Once this data is received within traffic management centers (TMC), decision processes are executed, and this information then drives real-time applications/systems such as travel time messages and variable speed limits. The generated traffic information is frequently disseminated via a wireless network and delivered to the end-user through his/her smartphone or in-vehicle device. However, the sheer volume of the collected data, combined with the large network size and high number of the users (e.g., drivers) can limit the scalability of real-time traffic operations. Firstly, the TMC often monitors and processes high-resolution traffic information from each road segment in the network although there might be a significant correlation between (neighboring) segments. These correlation functions might help traffic operators to obtain reliable estimation of the network conditions by explicitly observing only portion of that network. In this way TMC can reduce the operational costs and increase the scalability of operations since less data needs to be explicitly processed. Secondly, TMC disseminates the processed traffic information for each
road segment to all users in the network. This fully centralized approach (where all operations are performed within TMCs) imposes significant bandwidth requirements to already congested wireless networks. To deal with these issues, we propose a novel method that requires only data from a few locations in the network in order to assess the traffic conditions in the entire network. The proposed method, that relies on column-based (CX) low-dimensional methods for matrix approximations can be implemented on the user smartphone; and assign them certain operations that are traditionally performed on traffic servers. In this way, the proposed method might improve the scalability and restrain the communication bandwidth of TMC real-time operations; even in the context of practical city-scale networks, which are typically composed of different types of roads.

Let us start by explaining how the proposed mechanism is used to improve the scalability of real-time ITS operations. Typically, intelligent transportation systems process data with high spatial resolution (i.e., from every single road segment) although there is a significant level of correlation among different segments in the network. By contrast, the proposed CX mechanism is built on the observation that the values of collected traffic data exhibit similar patterns over time, or that traffic conditions on different roads within the network tend to demonstrate a varying level of correlations. At the core of our approximation lies the notion of compressed traffic state, a set of explicit values from the collected measurements at a few road segments in the network that can be used to linearly approximate traffic conditions in the rest of the network. We use historical information to learn these relationship functions between the compressed traffic state and the rest of the network. It is noteworthy that the relationship functions might be updated with the help of upcoming traffic information. These functions allow us to estimate the traffic parameters (e.g., traffic speed) at any segment in the network using information from a small subset of the links. We refer to this as compressed sensing. Similarly, we perform traffic prediction for the whole network, by developing prediction models for only the representative subset of road segments. We refer to this as the compressed prediction. It is crucial to note that “extrapolation” of the collected / predicted traffic data...
for the representative subset of road segments to the entire network is solely based
on straightforward matrix multiplication, which in general can be easily done in a
fraction of second. Hence, the proposed CX mechanism improves the computation
times of ITS applications by the factor of $k$, where the $k$ is the ratio of the total
number of links in the network and a number of links in the compressed traffic state.

In the following section, we will discuss how the estimated and predicted traffic data
are transferred from the Traffic Management Centers (TMCs) to end-users (e.g.,
motorists). Typically, dissemination of traffic data is deployed in a fully-centralized
manner where all computations are performed within TMCs while end-user devices
such as smartphones and vehicle navigation devices are used only to display final
outputs. This centralized architecture puts severe constraints to the communication
workload since it requires all data to be sent from the TMC to each vehicle in the
network. Furthermore, the great potential of smartphones, which have significant
computation power nowadays, is not efficiently utilized in this centralized framework.
We propose decentralized and hybrid architectures, based on a CX mechanism where
the network extrapolation is performed on a smartphone, to further restrain the
busy TMC servers and reduce the communication bandwidth between the TMC
and end-users.

So far we have proposed the novel low-dimensional methods to improve the scalabil-
ity and reduce the communication cost of TMC operations. Intuitively, this method
might not work well during non-recurrent traffic conditions. This abnormal behavior
of traffic network is frequently caused by external factors such as heavy rainfall and
traffic incidents, which are, in general, hard to predict. To tackle these issues we
carefully studied the information about the rainfall and corresponding traffic data
for a considerable period of time. In particular, we (i) investigate the impact of the
rainfall on the various levels of congestions; and (ii) determine whether including
information about rainfall can improve the predictive accuracy of the state-of-the-
art prediction algorithms such as Support Vector Machine (SVM) and boosting
methods.
1.2 Related Work

In this section we briefly review the literature related to low-dimensional models, architectures for smartphone traffic applications and impact of the inclement weather on traffic performance. The existing studies often deal with the scenarios where traffic data for each segment in the network is explicitly collected and processed. The application of low-dimensional models is frequently restricted to the compression and pre-processing of collected data. Moreover, most of the existing traffic applications rely on a traditional centralized architecture where all operations are performed on the server. Only final outputs (e.g., suggestions for the drivers) are sent to the users’ smartphones where this information is visualized and delivered to the customers. In this way, the significant potential of the smartphone computation resources has not been either used or even systematically evaluated for traffic applications. Finally, the scope of the studies that are dealing with the weather and traffic data is often restricted to special cases such as single intersections or short corridors. Furthermore, these studies often deal with limited databases where traffic and weather data are collected for a short period of time (e.g., a few weeks) and aggregated on hourly or even daily increments. These are the major shortcomings of the previous studies.

1.2.1 Low-Dimensional Models

In this study we propose a novel low-dimensional network model to improve the scalability of estimation and prediction operations in ITS. Proposed models can also be used to compress and store traffic data in an intuitive yet accurate way. Low-dimensional representation of large traffic data sets might be obtained using various techniques. In the recent study, M. T. Asif et al. analyzed the reconstruction efficiency of singular value decomposition (SVD), discrete cosine transform (DCT), wavelet transform and non-negative matrix factorization for low-dimensional representation [1]. In addition, the authors proposed a near-lossless compression algorithm based on Huffman coding on low-dimensional network space. The results of
this study show that DCT and wavelet based methods provide the best compression performance in terms of relative error. Principal Component Analysis (PCA), is commonly applied to obtain low-dimensional representations of traffic data [2, 3, 4]. PCA provides an effective low-dimensional representation in terms of latent variables and corresponding basis vectors. However, these latent variables are hard to interpret since they represent a linear combination of (up to all) original dimensions [5, 3]. Moreover, even if we obtain the basis vectors from historical data, we still need to collect data from all sensors during online operation. Therefore, PCA and the method studied in [1] are often applied for offline operations such as compression, future selection and missing data imputation [3, 6, 7, 8, 9].

Simulation models and data-driven approaches are traditionally used to perform traffic estimation and prediction [10, 11]. Simulation models can be used for traffic management operations at various levels of network granularity [10, 12, 13]. For large areas, macroscopic and mesoscopic simulation tools (e.g., DYNAMIT) have been adopted to build custom models, relying on historical speed-density link relationships for that specific network [13]. In recent years, large volumes of collected data have served for extensive model calibration of traffic dynamics [14, 15]. Extensive calibration enhances the credibility of built simulation models. However, such models are not generic and cannot be translated from one network to another in a straightforward manner. An alternative is to consider data driven methods. These methods offer greater flexibility due to their generic structure. Consequently, these methods are used to develop highly accurate traffic estimation and prediction models [16, 11, 17]. There is a vast literature in the area of traffic forecasting [18, 19, 20, 21, 22, 23, 24, 25, 16, 26, 27, 28, 29, 17]. Recently, Lippi et al. have performed a comprehensive analysis of the forecasting performance of time series and supervised learning techniques [11]. The authors observed that seasonal autoregressive integrated moving average (SARIMA) in combination with the Kalman filter ends up being the best model in terms of accuracy while simultaneously requires the largest training time [11]. They also conclude that Support Vector Regression (SVR) represents an interesting tradeoff solution between computational costs and
prediction precision. In this and other relevant studies, data-driven techniques explicitly predict traffic variables at each link in the observed network while seeking for a best performance. For large traffic networks prediction of every single segment may not be possible, especially in the context of online operations. By contrast, we wish to consider a more practical option where only a subset of links is explicitly monitored, processed and predicted.

In summary, the sheer volume of data combined with the large network size, however, can limit the scalability of both data-driven and model-driven techniques [30]. Nonetheless, traffic conditions on different roads within the network tend to exhibit varying level of correlations. This trend can be seen even for large and diverse networks. We propose to utilize these relationships to obtain low-dimensional representations for traffic networks. Specifically, we use these low-dimensional representations to perform traffic compression, estimation and prediction. For traffic prediction we apply SVR as as this method is commonly used for traffic applications [11]. However, the proposed compressed prediction framework is quite flexible and can incorporate any other prediction algorithm in a straightforward manner.

1.2.2 Traffic App.

In this study, we explore the computational capabilities of smartphones for different modes of operations and traffic applications. Being equipped with smart sensors and high visualization capabilities, smartphones have found many applications in the domain of intelligent transportation systems (ITS) [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]. Numerous smart sensors such as GPS, manometers, barometers etc., have been used to detect the speed, location and activity of the travelers, mode of transportation and user (driver) behavior [31, 32, 33, 34, 35, 36, 37]. Conversely, smartphones are frequently seen as a convenient way to deliver real-time traffic information to the travelers. This information encompasses current and future traffic conditions, relevant transit information, as well as step-by-step guidance from origin to the destination [38, 39, 40, 41]. In these studies, traffic apps often deal with the
computationally light operations in a centralized framework and typically do not use all available CPU resources. Although smartphones possess significant computational power nowadays, these resources, to our best knowledge, have not yet been evaluated for computationally intensive traffic applications. The only exceptions are a few safety related applications [42] where all the computations are internally performed with the help of native programming languages.

Development of traffic (and other Android) applications is typically done in the Java programming language with the help of the Android Software Development Kit (SDK), third party libraries, and other useful tools [43]. The SDK platform is “easy-programmable”, portable, and supported by most Android features such as services and content providers [44]. Unlike the SDK, the Native Development Kit (NDK) platform uses C/C++ libraries for computation components of the intensive smartphone applications such as video games, image processing applications, etc. [44, 45]. Recent studies on the Android platform show that applications written in C/C++ achieve better performance than those in Java [44, 45, 46]. These benefits of using native languages vary across the applications and underlying architecture of the devices [44, 45, 46]. If only the front-end of application is implemented on the mobile device (such as in [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]), then the improvement in performance may not be that significant [44, 45, 46]. For the intensive smartphone applications, this improvement might be significant [44, 45, 46]. Although the NDK development platform may yield a significant improvement in performance, it is still not clear whether it is sufficient for the ITS applications at hand. To answer this question, one needs to implement and test the NDK development platform for various traffic applications.

1.2.3 Impact of Rainfall on Traffic Conditions and Prediction Performance

Weather conditions tend to have a significant impact on the driving behavior and the key parameters of traffic stream. Various studies have shown that different
weather factors such as temperature [47, 48], precipitation [49, 50] and visibility [51, 52], can degrade the traffic parameters such as capacity and free-flow speed while simultaneously increase a frequency of accidents on the roads. Rainfall intensity has been frequently identified as an external factor that has the highest impact on driving conditions and network performance [49, 53]. Even light rain tends to enhance the drivers’ carefulness and a number of interactions in traffic flow, which, in turn, leads to the reduction in roadway capacity. This decrease in throughput then changes the relationships between the fundamental traffic parameters (e.g., volume, speed, and density) [49]. Moreover, the slippery road conditions and reduced visibility can cause a dramatic increase in a total number of crashes while simultaneously reducing the number of casualties and severity of these accidents as the consequence of enhanced drivers’ consciousness [50].

Different rainfall intensities can have different impacts on the key traffic stream parameters and their relationships [53, 54, 55, 56, 57]. This quantifiable effect is frequently utilized to develop mathematical and regression models that can predict traffic conditions based on the given rainfall intensities [58, 53, 54, 59]. These prediction methods, that frequently rely on neural networks and various vector autoregressive moving average algorithms, use the information about the rainfall as an exogenous variable. The results indicate that including exogenous variables only marginally improves or sometimes even degrades the prediction performance [60, 61, 58, 62].

The scope of the relevant studies remains limited to special cases. For instance, the applied prediction algorithms are restricted to vector autoregressive moving average methods and neural networks. In addition, the studies [60, 61, 62, 63, 53] consider a short segment or single intersection only. Moreover, most of the studies deal with a limited database where the data is frequently collected for a short periods such as a few weeks or even days and often aggregated in hourly or even daily increments [60, 61, 62, 64, 63, 53, 58]. Such an approach might be inappropriate due to the intrinsic nature of the rainfall accidents, which are in general quite rare events whose severity often varies temporally and spatially.
In this study, we investigate the impact of the rainfall on the traffic speed and explore whether the additional information about the present rainfall rate might improve the performance of the state-of-the-art prediction algorithms such as Support Vector Machine (SVM) and boosting methods. Unlike the other relevant studies, we rely on an impressive data set comprising rainfall and traffic information for more than 4,000 road segments. This information is collected for a period of seven months and with a sampling interval of 5 minutes.

1.3 Contributions and Outline of the Thesis

The thesis is structured as follows:

- In chapter 2, we discuss the problem of scalability in the context of traffic estimation and prediction in large networks. We also present a novel compressed prediction algorithm that, as opposed to the methods from the related studies, does not require data from all links in the network. In this way the proposed method significantly reduces the total computation time at the cost of a minor increase in the error. We decompose the overall error of the presented method into several components and suggest strategies to further improve the performance of these components.

Contributions:

Chapter 1 Introduction


- J. Dauwels, M. T. Asif, N. Mitrovic, and P. Jaillet, Compression, Compressed Sensing and Compressed Prediction of Urban Traffic Data, Singapore patent application no. 10201406502P.

In chapter 3, we investigate the problem of implementation of the method that we proposed in chapter 2. The deployment of the similar traffic applications is frequently done in a centralized framework where all communications are performed on the server. We study various architectures and evaluate their performance for different modeling platforms, smartphone devices and traffic networks. We also propose decentralized and hybrid frameworks that significantly reduce the volume of communication bandwidth and pave the way for new cooperative traffic applications and operations.

Contributions:


In chapter 4, we discuss the problem of estimating the impact of the rainfall intensity on traffic performance in large traffic networks. We also explore the problem of integration rainfall data in traffic prediction. Impact of the rainfall on traffic and its prediction performance is a commonly studied problem. How-
ever, a comprehensive analysis is often missing due to specific requirements in the analysis of rainfall accidents which are in general quite rare events and might have different durations and intensity across the network. Furthermore, only a few study have explicitly investigated an integration of the weather information in the state-of-the-art prediction algorithms, especially in the context of high-resolution rainfall data. We assess the impact of the rainfall on link-level performance and evaluate the accuracy of state-of-the-art prediction algorithms in the context of large traffic networks and rich weather data sets. We conduct our analysis for different rainfall thresholds, road categories and periods of the day.

Contributions:


• We conclude the thesis by discussing potential extensions related to our current work in chapter 5.

• In addition to the above mentioned works, the author also contributed to the works published in [1, 65, 66, 67, 68, 69, 70] and [71].
Chapter 2

Low-dimensional Models of Large Scale Traffic Data

Summary - Advanced sensing and surveillance technologies often collect traffic information from various sources with high temporal and spatial resolution. Recorded data is essential for many real-time applications related to traffic management systems. However, the volume of the collected data severely limits the scalability of real-time traffic operations. Low-dimensional models can help ease these constraints by providing compressed representations for the networks. In this study, we propose a low-dimensional network representation, explicitly built of a small subset of the road segments present in the network. This formulation allows us to efficiently store collected data in an intuitive way. Furthermore, we utilize the compressed representation to assess the current condition in the network by collecting traffic information from a small subset of segments. Similarly, we perform traffic prediction for the whole network, by developing prediction models for only the representative subset of road segments. For the analysis, we consider a large network comprising 17,967 road segments. Numerical results show that our method can achieve competitive compression performance compared to PCA. More importantly, results demonstrate a significant reduction in prediction time without significant degradation in prediction performance.

Nanyang Technological University Singapore
2.1 Introduction

Intelligent Transportation Systems (ITS) collect real-time traffic information from various sources such as probe vehicles, smartphone devices and infrastructure based traffic sensors. With advancements in sensor technology, traffic data (e.g., volume and speed) can be recorded on a large scale and with high temporal resolution [72]. Recorded data is frequently used for historical analysis and traffic management operations such as network monitoring, transportation planning and congestion avoidance applications [73]. These applications heavily rely on fast and accurate assessment of current (estimation) and future (prediction) network states.

To model the road network, existing studies explicitly address each road segment in that network. For large traffic networks and real-time applications such an approach may not be feasible. To overcome this problem we focus on low-dimensional network models where only a subset of road segments needs to be explicitly monitored.

In this study, we use column-based (CX) matrix decomposition to express the original network in terms of a small subnetwork. We refer to the small subnetwork as the compressed state of the original network. We learn the relationship functions between compressed and original (uncompressed) network by analyzing the recorded data in offline manner. In this way, we can represent the traffic network as a product of two low-rank matrices: (i) the subnetwork data and (ii) the corresponding relationship matrix. We refer to this as CX compression scheme. The CX compression scheme is a stepping stone to compressed sensing and compressed prediction applications.

In the case of compressed sensing we aim to infer the present state of the entire network from the current traffic state of the subnetwork [74]. We use the training data (collected offline) to infer the relationship matrix. To assess the network state we multiply the data from the subnetwork, obtained from the testing set, with the relationship matrix, inferred from the training data set. Our underlying assumption is that traffic variables often vary smoothly across the traffic network [4].
In the matter of compressed prediction we apply the CX-based method to infer the future state of the network. First, we explicitly predict the traffic state for the subnetwork using traditional prediction algorithms. Then we multiply the predicted data of the subnetwork with the relationship matrix, inferred from the training data set. Similarly to compressed sensing we rely on the observation that traffic conditions tend to follow distinct patterns and traffic parameters often vary smoothly [75, 4].

For our analysis, we consider the city-scale traffic network in Singapore, comprising 17,967 road segments. The numerical results illustrate that the proposed methods can accurately compress the traffic information. More importantly, the proposed methods can infer the current and future states of the network, while substantially improving the processing speed of the underlying modeling algorithm. The reduction in computational time is proportional to the compression ratio, i.e., the ratio of the number of links in the subnetwork and the total number of links.

2.2 Column-based (CX) Decomposition

The column-based (CX) method has recently found applications in many fields such as text processing, finance and biology [76, 77, 78]; it uses only a subset of the columns to reconstruct the entire data matrix. In our previous study, we applied the column and row (CUR) based method to impute a matrix of traffic data from a few columns (links) and few rows (time instants) of that matrix [79]. The resulting CUR models can be easily interpreted, and can also be used for application of compression and extraction of the common traffic patterns. However, the CUR method occasionally requires traffic data for the entire network, and, hence it cannot be applied for compressed prediction. An alternative would be to consider a Non-Negative Matrix Factorization (NMF) which approximates a certain matrix as product of two low-rank matrices with the property that all three matrices have no negative elements [80]. However, the NMF, frequently seen as suboptimal solution to SVD in fitting the data, requires that the weights (or relationship functions) are
non-negative [80]. The CX-based method instead does not have this requirement. In the following, we will briefly review the CX-based method.

**Definition 1**: Let \( \mathbf{A} \in \mathbb{R}^{m \times n} \) be a given matrix. Let \( \mathbf{C} \in \mathbb{R}^{m \times c} \) be a matrix consisting of \( c \) columns of the matrix \( \mathbf{A} \). The column-based (CX) matrix approximation \( \hat{\mathbf{A}} \) of \( \mathbf{A} \) is defined as \( \hat{\mathbf{A}} = \mathbf{C}\mathbf{X} \), where \( \mathbf{X} \in \mathbb{R}^{c \times n} \) is a matrix that expresses every column of \( \mathbf{A} \) in terms of the basis provided by the columns of \( \mathbf{C} \) [81].

### 2.2.1 Column Selection

In order to select the best subset of columns, for a given size \( c \), one could test all possible combinations. However, the computational complexity of this brute-force approach is \( O(n^c) \) [76]. Due to this complexity, testing all possible choices of \( c \) columns is typically not practical. Therefore, the selection of a subset of columns is still an interesting open problem [76, 81, 82, 83]. To alleviate this problem, several randomized algorithms have been proposed [81, 82]. In the following we briefly explain three sampling strategies that we consider in our study.

#### 2.2.1.1 SVD Sampling Strategy

In this strategy we assign a weight to each column (road segment) in proportion to the Euclidean (or \( L^2 \)) norm of the top \( k \) right singular vectors \( (v) \) of (traffic) matrix \( \mathbf{A} \) [76, 81, 83]. The Euclidean (or \( L^2 \)) norm, assigns to each of \( k \) right singular vectors the length (or magnitude) of its arrows. To approximate \( k \) we use the number of columns (links) that needs to be selected:

\[
P_s(a_i) = \frac{1}{c} \sum_{j=1}^{c} v_{ij}^2 \quad \forall \ i \in \{1, \ldots, n\}. \tag{2.1}
\]

We refer to this scheme as SVD sampling method where \( v_{ij} \) is the \( i \)-th coordinate of \( j \)-th right singular vector. To compute the right singular vectors, we perform a
singular value decomposition (SVD) of the original data matrix $A$:

$$A = U \Sigma V^T.$$  \hspace{1cm} (2.2)

Since the matrix $V$ is unitary, the quantities $P_s(a_i)$ sum to one. The computational complexity of the SVD sampling method is $O(n^3)$ [81], which can be too expensive for extremely large matrices [82].

The resulting reconstruction error can be upper bounded as follows [81]:

**Theorem 1:** Given a matrix $A \in \mathbb{R}^{m \times n}$ and an integer $k \ll \min\{m,n\}$, there exists a randomized algorithm that selects exactly $c = O(k^2 \log(1/\delta)/\varepsilon^2)$ columns of $A$ to construct $C$, such that with probability at least $1 - \delta$,

$$\min \| A - CX \|_F = \| A - CC^+A \|_F \leq (1 + \varepsilon) \| A - A_k \|_F,$$  \hspace{1cm} (2.3)

where $\| A \|_F = \left( \sum_i \sum_j a_{i,j}^2 \right)^{\frac{1}{2}}$ is the Frobenius norm of the matrix $A$. $C$ contains the sampled columns of $A$, $CC^+A$ is the projection of $A$ on the subspace spanned by the chosen columns and $\varepsilon$ is the acceptable deviation. $A_k$ is the best rank-$k$ approximation to $A$, obtained by singular value decomposition (SVD). $\delta$ is the failure probability in the case of randomized algorithms. The algorithm performs column sampling $O(\log(\frac{1}{\delta}))$ times and returns the subset of columns such that $\| A - CC^+A \|_F$ is smallest over all $O(\log(\frac{1}{\delta}))$ trials [81].

### 2.2.1.2 Energy Sampling Strategy

In this strategy we assign a score to each column of matrix $A$ based on the Euclidean norm of that column. The scores are then normalized to obtain the probability of selection for each column. Hence, the assigned probability of the column is proportional to the energy (L2 norm) of that column $2.4$. We refer to it as energy...
The computational complexity of the energy sampling method is $O(n^3)$ as it requires the computation of the Frobenius norm of the matrix.

### 2.2.1.3 Random Sampling Strategy

In this strategy we assign an identical probability to each column of the matrix ($P_r(a_i) = n^{-1}$). We refer to it as random sampling. The computational complexity of the random sampling method is $O(1)$. We evaluate the random strategy since it has practical applications in the context of probe (or mobile) vehicles whose locations in the network cannot be predefined (detailed information is given in Section 3).

### 2.2.2 Relationship Matrix

For the sampled column matrix $C \in \mathbb{R}^{m \times c}$, we compute the relationship matrix $X \in \mathbb{R}^{c \times n}$, which will allow us to represent the columns of matrix $A \in \mathbb{R}^{m \times n}$ in terms of columns of the matrix $C$ [81]. The matrix $X$ can be regarded as an extrapolation matrix that maps the subnetwork associated with $C$ to the entire network represented by $A$.

For given matrices $C$ and $A$, we compute the matrix $X$ as:

$$X = C^+ A,$$

where $C^+$ is Moore-Penrose pseudo-inverse of matrix $C$ [84].
2.3 CX-based Method for Traffic Applications

In this section, we discuss how the sampling strategies of CX based method can be utilized to infer meaningful temporal patterns in the network. More importantly, we also discuss how CX method can be used to perform compression, compressed sensing, and compressed prediction of traffic data. For this purpose, we consider the traffic data in the form of a matrix \( A \in \mathbb{R}^{m \times n} \) where the columns of the matrix \( \{a_i\}_{i=1}^n \) contain traffic data from different roads \( \{s_i\}_{i=1}^n \). Rows represent time instants \( \{t_i\}_{i=1}^m \) at which the traffic data is recorded. Each matrix cell \((a_{ij})\) shows the numerical value of an observed traffic variable (e.g., speed, volume) at location \(s_j\) during the interval of time \((t_i - T, t_i)\) where \(T\) is the sampling period (e.g., 5 or 15 minutes). Therefore, the \(i\)-th row vector \(\alpha_i = [z(s_1, t_i) ... z(s_n, t_i)]\) of \(A\) contains the traffic state for the entire network at a particular time \(t_i\). Similarly, the \(j\)-th column vector \(a_j = [z(s_j, t_1) ... z(s_j, t_m)]^T\) of \(A\) contains the observed condition at location \(s_j\) during the entire recording period. Hence, we can write traffic data matrix as \(A = [a_1 ... a_n]\). For the sake of simplicity, we use subscripts \(h, p\) and \(f\) in the rest of the paper to denote historical, present and future values, respectively.

2.3.1 Temporal Patterns in Urban Traffic Networks

We start by explaining how the assigned weights, obtained by applying different sampling strategies, can be used to infer network-wise temporal variations within a day and for different days. Let us consider the historical data matrix \(A_h \in \mathbb{R}^{m \times n}\) as explained in Section 2.3. We assess the network-wise traffic conditions at any time \(a \{t_i\}_{i=1}^m\), by assigning certain weights to this time instant (i.e., the \(i\)-th column of \(A_h^T \in \mathbb{R}^{n \times m}\)) using the Energy sampling method. Intuitively, the higher assigned weights, in the case of traffic speed, refer to the smooth or free-flow traffic conditions (see (2.4)). However, for the time instants with congested traffic, such as these during peak periods, we assign lower weights. Similarly, we apply the SVD sampling strategy on the transposed historical data matrix \((A_h^T)\) to find correlations.
between different time instants. For example, if a large number of time instants follow similar traffic patterns then we can easily represent them by a few components. However, if a certain time instant does not conform to normal traffic behavior, then we assign a higher weight to that time instant (see (2.1)).

Column-based sampling schemes lead to practical benefits to traffic management. The understanding of the network-wise temporal patterns on different time scales, might prove helpful for management centers. We will analyze these temporal patterns on various time scales in Section 2.5.1.

### 2.3.2 Compression

In this section we explain how the column-based decomposition method can be used for compression of recorded traffic data. Suppose that the matrix $C_h$ contains the observed traffic parameter at $c$ specific locations in the network, such that \{c_1, ..., c_c\} $\subseteq$ \{a_1, ..., a_n\}. Then, we can approximate the data matrix $A_h$ as

$$\hat{A}_h = C_h X_h,$$  \hspace{1cm} (2.6)

where the matrix $X_h$ contains the relationships between the traffic parameter at different locations in the network. Hence, instead of storing the large matrix $A_h$, we store the two smaller matrices $C_h$ and $X_h$. The compression ratio (CR) of such low-dimensional approximation is given by:

$$CR_h = \frac{mn}{mc + cn},$$  \hspace{1cm} (2.7)

where the numerator equals the total number of elements in the original (uncompressed) matrix $A_h$. The denominator in (2.7) represents the total number of elements in the low-dimensional (compressed) approximation. It is noteworthy that the matrix $X$ has to be stored for the compression of traffic data. Consequently, the compression ratio differs from the ratio of the total number of links ($n$) and the number of links in compressed network state ($c$).
The column-based (CX) compression scheme leads to simple network representation. Unlike Principal Component Analysis (PCA), the proposed method stores the observed traffic parameter at specific locations in the network instead of as a linear combination of latent variables. Furthermore, a CX-compression scheme identifies the important links in the network and reveals the relationships between these links and the rest of the network, leading to practical benefits for traffic management. More importantly, such an approach could be useful for real-time traffic monitoring operations. In the following we discuss two attractive applications of CX-based compressed representation, namely compressed sensing and compressed prediction.

### 2.3.3 Compressed Sensing

So far, we have assumed that the matrix $X_h$ is stored together with $C_h$ leading to the compression of matrix $A_h$. In this scenario, the matrix $X_h$ is computed for a given data matrix $A_h$ and a column matrix $C_h$. Alternatively, one may precompute a matrix $X_h$ and re-use the same matrix to infer $A$ for any given $C$. Although we still need data from all the links to precompute $X_h$, this operation can easily be performed offline. Hence, during online operations, the system would only require data from a small number of sensors. We refer to this scenario as compressed sensing. It is noteworthy that low-dimensional PCA models cannot be used for compressed sensing since PCA requires data from all sensors for both offline and online (real-time) operations.

The underlying assumption of the proposed method is that the traffic conditions are stationary, so that a fixed matrix $X$ allows us to accurately reconstruct the original data matrix $A$ from $C$ \cite{85, 4, 86, 87, 75}. Therefore, we can estimate the present network state ($\hat{\alpha}_i^j$) as

$$\hat{\alpha}_i^j = c_i^j X_h \quad \forall \ i \in \{k, \ldots, m\}, \quad (2.8)$$
where $\hat{\alpha}_i^p \in \mathbb{R}^{1 \times n}$ is a row vector which represents the current state of the entire network (see Fig. 2.1a) for test data ($i = k, \ldots, m$). Row vector $c_p^i \in \mathbb{R}^{1 \times c}$ contains the information about the current traffic conditions at $c$ specific locations in the network (see Fig. 2.1). Matrix $X_h \in \mathbb{R}^{c \times n}$ is the relationship matrix, learned from a training data set. We define the compression ratio for compressed sensing as:

$$CR_p = \frac{c}{n}.$$  \hfill (2.9)

Large traffic networks contain a diverse set of road segments. We want to explore whether homogeneous subnetworks can improve the overall performances of compressed sensing. We divide the traffic network into $s$ mutually exclusive subnetworks (based on category of the road segment) such that $\alpha_p = [\alpha_1, \alpha_2, \ldots, \alpha_s]$ where $\alpha_i \in \mathbb{R}^{1 \times n}, \forall i \in \{1, \ldots, s\}$. Then, we perform compressed sensing for each subnetwork separately. At last, we merge the results of the clustered subnetworks to infer the traffic state of the entire network. Although different choices of temporal and/or spatial clustering can be applied, we consider simple clustering based on different road categories in this study.

The overall performance of the proposed compressed sensing method is sensitive to the “compressibility” of the network and “non-stationarity” in the traffic data. For compression, we represent the traffic data as a product of two low-rank matrices, i.e., the subnetwork data matrix and the most appropriate relationship matrix. As the compression is lossy, we expect the reconstructed matrix $\hat{A}_h$ to be different from the original matrix $A_h$. The issue of non-stationarity is due to the fact that matrix $X_h$ is inferred from training (historical) data instead of the current data. The matrix $A_p$ ($A_p = [\alpha_1^k \ldots \alpha_m^p]^T$) is not available, and the goal is to infer that matrix by extrapolating the matrix $C_p$ ($C_p = [c_1^k \ldots c_m^p]^T$) according to the CX decomposition. Obviously, the matrix $X_p$ cannot be extracted from the current data $A_p$, since the matrix $A_p$ is not available. Instead we determine $X_h$ from the training data set. Since traffic is not perfectly stationary, this approximation will induce an additional reconstruction error. We refer to it as the error due to non-stationarity.
Figure 2.1: Compressed sensing. Uncompressed (top) network contains all segments. Compressed (bottom) network comprises only carefully selected links.
of traffic spatial relationships. To quantify this error, let us call \( B = C_p X_p \) the reconstruction of the data matrix \( A_p \), assuming the latter is available to compute the CX decomposition. The reconstruction \( \hat{A}_p \) \( (\hat{A}_p = [\hat{\alpha}_p^k ... \hat{\alpha}_p^m]^T) \) in the scenario of compressed sensing is less accurate, since we need to replace \( X_p \) (determined from the test data matrix \( A_p \)) by \( X_h \) (determined from training data matrix \( A_h \)). The mean squared error (MSE) incurred for compressed sensing can be written as:

\[
\frac{1}{rn} \| A_p - \hat{A}_p \|^2_F = \frac{1}{rn} \| (A_p - C_p X_p) - (C_p X_h - C_p X_p) \|^2_F, \\
= \frac{1}{rn} \| (A_p - B) - (\hat{A}_p - B) \|^2_F, \\
= \frac{1}{rn} \left( \sum_{i=k}^{m} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2 + \sum_{i=k}^{m} \sum_{j=1}^{n} (\hat{a}_{ij} - b_{ij})^2 \\
- 2 \sum_{i=k}^{m} \sum_{j=1}^{n} (a_{ij} - b_{ij})(\hat{a}_{ij} - b_{ij}) \right),
\]

where \( r = (m - k + 1) \) represents the number of time instants in test data matrix \( A_p \). The first component of the error corresponds to the compressibility of the network and the second component is due to the non-stationarity of spatial patterns within the network (see (2.12)). The third component of the error refers to the correlations between the aforementioned error components (see (2.12)). To make this interpretation more explicit, we rewrite (2.12) as:

\[
\text{MSE}_{\text{est}} = \text{MSE}_{\text{com}} + \text{MSE}_{\text{ns}} - 2\xi_{\text{est}},
\]

where \( \xi_{\text{est}} \) is correlation coefficient between compressibility and non-stationarity. We will analyze the behavior of these errors for different compression ratios in Section 2.5.3.
Figure 2.2: The idea of the compressed prediction. We carefully select a subset of the road segments from the network (left). Then we explicitly predict the future traffic conditions only for the subset of links (bottom) and extrapolate results throughout the network (right).
2.3.4 Compressed Prediction

In the previous section, we inferred the condition of the entire traffic network by observing traffic conditions at a small subset of links (see Fig. 2.1 or left part of Fig. 2.2). Here we will extend this approach to prediction; we aim to predict the state of the entire traffic network (see upper-right part of Fig. 2.2) from the predicted state of a small subset of links (see bottom-right part of Fig. 2.2). We recall that low-dimensional models generated by PCA cannot be utilized for this task since PCA requires information for all links in the network. Instead, we use a state-of-the-art algorithm to predict the traffic speed only for a selected subset of locations (see bottom part of Fig. 2.2). Then, we utilize the proposed method to extrapolate the predictions to the rest of the network using the precomputed relationship matrix (see right part of Fig. 2.2). This can be written as:

$$\hat{\alpha}^i_f = \hat{c}^i_f X_h, \quad \forall \ i \in \{k, \ldots, m\}, \quad (2.14)$$

where $\hat{c}^i_f \in \mathbb{R}^{1 \times c}$ is the row vector containing the predicted values of the traffic variable at the selected locations and at the $i^{th}$ time instant, $\hat{\alpha}^i_f \in \mathbb{R}^{1 \times n}$ contains the predictions for all locations at $i^{th}$ time instant, and $X_h \in \mathbb{R}^{c \times n}$ is the relationship matrix (see Fig. 2.2). If the predictions $\hat{c}^i_f$ would be identical to the true values $c^i_f$, then the problem boils down to compressed sensing, which we discussed in the previous section. In practice, however, the predictions have some inaccuracies. Therefore, we can write $c^i_f = \hat{c}^i_f + \Delta c^i$ where $\Delta c^i$ represents the prediction error for the subnetwork at time $i$. Furthermore, let $D = C_f X_h$ be the estimated network profile, during the entire observational period, without any prediction error in $C_f$ ($C_f = [c^k_f c^m_f]^T$). Then, the MSE between the predicted $A_f$ ($A_f = [\hat{\alpha}^k_f \ldots \hat{\alpha}^m_f]^T$)
and the true future values $A_f$ ($A_f = [\alpha_f^k ... \alpha_f^m]^T$) can be written as:

$$\frac{1}{rn} \| A_f - \hat{A}_f \|_F^2 = \frac{1}{rn} \| (A_f - C_f X_h) - (\hat{C}_f X_h - C_f X_h) \|_F^2,$$

$$= \frac{1}{rn} \| (A_f - D) - (\hat{A}_f - D) \|_F^2,$$

$$= \frac{1}{rn} \left( \sum_{i=k}^{m} \sum_{j=1}^{n} (a_{ij} - d_{ij})^2 + \sum_{i=k}^{m} \sum_{j=1}^{n} (\hat{a}_{ij} - d_{ij})^2 
- 2 \sum_{i=k}^{m} \sum_{j=1}^{n} (a_{ij} - d_{ij})(\hat{a}_{ij} - d_{ij}) \right),$$

where the first component of the error corresponds to the non-stationarity of spatial patterns within the network and the second component is due to inaccurate predictions. The third component of the error shows the correlations between these two error components. We refer to (2.17) as MSE for compressed prediction. We rewrite (2.17) in more explicit form:

$$MSE_{total} = MSE_{est} + MSE_{pred} - 2\xi_{pred},$$

where $\xi_{pred}$ refers to correlation between non-stationarity and predictability. By substituting (2.13) in (2.18), we obtain:

$$MSE_{total} = MSE_{com} + MSE_{ns} + MSE_{pred} - 2\xi_{est} - 2\xi_{pred}.$$  

Hence, the total error of compressed prediction can be decomposed into four error components: (i) error due to compression; (ii) error due to changes in spatial relationships (non-stationarity); (iii) error due to inaccurate predictions; (iv) correlations among the previous error components.

Compressed prediction provides a significant reduction in computational complexity by explicitly predicting the traffic variables for only a small subset of road segments in the network. Compressed prediction involves two computations: (i) prediction of...
the traffic conditions at representative locations in the network and (ii) extrapolation of the predicted values to the entire network. In the former, the computational complexity depends on the underlying prediction algorithms, and is proportional to the number of locations $c$ in the subnetwork. The second step (extrapolation) requires a single matrix-vector multiplication with complexity $O(cn)$. Therefore, the total complexity is of order $O(c + cn)$. In practice, the predictions at each link in the subnetwork are computationally complex. By contrast, the extrapolation can be executed much faster. Hence, by performing prediction only for a small subnetwork, the computational complexity can be drastically reduced. The reduction is proportional to the compression ratio, i.e., the ratio of the total number of links and the number of links in the subnetwork.

### 2.4 Experimental Setup

In this section we explain the traffic data set considered in this study, and describe how we analyzed it for our different applications (compression, compressed sensing, and compressed prediction). We also briefly review support vector regression, which we use in our CX-based prediction approach. At last, we introduce several measures to assess the proposed methods.

#### 2.4.1 Data Set

We consider the nationwide traffic network in Singapore, comprising 17,967 road segments. The Land Transportation Authority (LTA) of Singapore collects raw traffic data in the network from multiple sources such as loop detectors and probe vehicles. The Traffic Management of LTA analyzes these signals and matches them to the corresponding road segments. Then, estimation models are used to assess the space mean speed for each segment in the network during the 5-min sampling interval (see Fig. 2.3). The traffic network contains diverse types of roads, belonging to different categories (Local roads, . . . , Expressways). The data set contains
the average speed at each link of the transportation network for a period of three months (August - October 2011). Each segment had less than 5% of missing values. We performed imputation by applying the Low Dimensional CP Weighted OPTimization (LDCP-WOPT) imputation method as it is able to deal with the large data set [88, 89].

We represent the data set in the form of a matrix as explained in Section 2.3. We use data for a period of three months to evaluate the performance of compression, for three different sampling schemes: random, energy, and SVD. For compressed sensing and compressed prediction, we need training data to: (i) determine the sub-network of \( c \) links, corresponding to the matrix \( C \) (see blue rectangles in Fig. 2.3); (ii) learn relationships between the subnetwork, as defined in (i), and the entire network; (iii) train the predictors. To this end, we use the speed data of the months August and September, 2011 for training (see red rectangle in Fig. 2.3). The remaining data is used to evaluate the performance of compressed sensing and compressed prediction (see green rectangle in Fig. 2.3). For compressed prediction, we predict traffic variable at specific locations using a baseline predictor (see blue shaded rectangles in Fig. 2.3). We apply SVR (support vector regression) for prediction, since it is commonly used [38, 11, 17, 16]. In the baseline case, we apply SVR to each link individually. We refer to this case as uncompressed prediction. In compressed prediction, we apply SVR only to a subnetwork and next extrapolate the predictions to the entire network.

Although any prediction algorithm can be applied, we consider here support vector regression (SVR), which we briefly review in the following.

### 2.4.2 Support Vector Regression

Support vector regression (SVR) is a data-driven prediction algorithm that is often applied for traffic data [11, 90, 65]. It efficiently uses current and past values of a traffic parameter at a particular location in order to predict future value. Let us assume that we want to predict the future speed value \( y_{jk} \) at time \( t_j + k\delta_t \) for link \( s_i \). We write
Figure 2.3: *Left:* City-scale network of Singapore with 17,967 road segments of different categories, from freeways to local feeders. *Right:* Corresponding input data matrix that contains training (red) and testing (green) sets. The data is collected with the sampling interval of 5 min. Selected links (compressed network state) are represented as blue rectangles and they have been explicitly predicted (see blue shaded squares/rectangles).
that as \( y_{jk} = z(s_i, t_j + k\delta_t) \), where \( t_j \) is the current time. In our analysis, we define the corresponding input feature vector as \( x_j = [d(t_j), h(t_j), z(s_i, t_j) \ldots z(s_i, t_j - k\delta_t)]^T \) where \( d(t_j) \) and \( h(t_j) \) is the day and hour, respectively, of the particular \( t_j \).

\( z(s_i, t_j) \ldots z(s_i, t_j - k\delta_t) \) are current and \( k \) past speed values for the particular link. We aim to infer relationship between \( y_{jk} \) and \( x_j \) such that

\[
y_{jk} = f_k([d(t_j), h(t_j), z(s_i, t_j) \ldots z(s_i, t_j - k\delta_t)]).
\]

We organize the training data set as \( r \) input-output pairs \( \{(x_j, y_{jk})\}_{j=1}^r \) which we use to train SVR and infer non-linear relationships.

SVR non-linearly maps the input speed data into a higher dimensional feature space \( \Phi \) [91, 92]. In order to avoid explicit mapping in \( \Phi \), SVR utilizes the Kernel trick. The Kernel trick replaces dot products in the feature space by the relation

\[
\kappa(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j),
\]

where \( \kappa \) is the desired kernel function [92]. In our analysis, we consider a Radial Basis Function (RBF) kernel

\[
\kappa_{x_i, x_j} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\gamma^2}\right)
\]

which is highly effective in mapping non-linear relationships [93]. Therefore, the function \( f_k \) will be [91, 92]:

\[
f_k(x) = \sum_{j=1}^r (\alpha_j - \alpha_j^*) x \cdot x_j + b. \tag{2.20}
\]

where \( \alpha_j, \alpha_j^* \) are the Lagrange multipliers. We use (2.20) to train SVR and perform speed prediction. For SVR implementation we utilize the Matlab package LIB-SVM [94]. In our analysis, we deploy \( \nu \)-SVR for \( \nu = 1 \) and default values for other parameters [94].

### 2.4.3 Performance Measures

We now briefly explain the three performance measures considered in this paper: percent root mean distortion (PRD), mean absolute percentage error (MAPE), and
mean squared error (MSE). PRD is a commonly used performance metric for low-dimensional estimation of matrices and tensors [95, 96, 97]. PRD is also often referred as Relative error [95, 96, 97]. Percent root mean distortion (PRD) can be considered as a measure of energy loss in the reconstructed network profile as compared to the original profile. We use percent root mean distortion (PRD) error to evaluate the reconstruction error of low-dimensional models [98]. The percent root mean distortion (PRD) quantifies the reconstruction error:

\[ \text{PRD}(\%) = \frac{\| A - CX \|_F}{\| A \|_F}. \] (2.21)

An alternative measure of the reconstruction error is the mean absolute percentage error (MAPE) defined as:

\[ \text{MAPE} = \frac{1}{mn} \sum_i^n \sum_j^m \frac{|a_{i,j} - \hat{a}_{i,j}|}{a_{i,j}}, \] (2.22)

where \( a_{i,j} \) and \( \hat{a}_{i,j} \) are true and predicted value, respectively.

We use the mean squared error (MSE) to assess the impact of several error components on the overall error. MSE is defined as:

\[ \text{MSE} = \frac{1}{mn} \| A - CX \|_F^2. \] (2.23)

### 2.5 Results

In this section we discuss the temporal patterns, inferred by Energy and SVD sampling strategies. Then, we access the performance of the proposed low-dimensional models for three different applications: (1) compression; (2) compressed sensing; and (3) prediction.
2.5.1 Temporal Patterns in Urban Traffic Networks

First, we explore the temporal patterns in large traffic network by analyzing the weights of different column selection strategies. Fig. 2.4 and Fig. 2.5 show the importance (selection probability) of each time instant as determined by the Energy and SVD sampling strategies, respectively. The colors represent the selection probabilities, such that black corresponds to a value of one. On the $x$-axis, we display different days for the month of August 2011. We limit ourselves here to the first four weeks (28 days). The $y$-axis shows different time instant for each day, starting from midnight (00:00 AM). Each time instant represents an interval of 5 minutes.

Fig. 2.4 shows the selection probability computed by the Energy sampling method. The Energy sampling strategy gives significant importance to time instants at which the average traffic speed is high. Fig. 2.4 shows that large values of the traffic speed occur during the early morning and late night hours. These are usually the time periods with the lowest traffic. We also observe that the traffic tends to be smooth during the weekends ($6^{th}$, $7^{th}$, $13^{th}$ day, etc.). Interestingly, we observe a slightly unique pattern on the $9^{th}$ of August, which is a public holiday in Singapore. For other (week) days the congestion is quite stable on network wise level, from early AM peak hours till late PM peak hours (see Fig. 2.4). For the sake of simplicity, we represented the missing entries as zero in the data set. These instances are represented by blue color (see Fig. 2.4). Energy sampling strategy correctly assigned the least importance to such time instants.

We show the temporal patterns for the SVD sampling strategy in Fig. 2.5. The SVD sampling method assigns higher weights to these time instants whose corresponding traffic patterns are rarely observed within the observational period. From Fig. 2.5, we can see that the traffic is mostly irregular during the morning transition period, when traffic conditions change from free flow traffic to high-density traffic during the morning peak hours. During the rest of the day, the traffic patterns are more stable, and therefore, the corresponding time instants receive smaller weights. We also observe that time instants during evening peak hours are given higher impor-
Figure 2.4: Assigned weights (importance) by Energy sampling method.
Figure 2.5: Assigned weights (importance) by SVD sampling method.
Fig. 2.5 shows that the SVD sampling strategy can also reveal uncommon traffic patterns such as the one during the public holiday on the 9th of August. At the top of Fig. 2.5, green regions appear only on Sundays and the public holiday, suggesting that traffic conditions during these days follow unusual patterns. Similarly, anomalies such as missing data can also be easily spotted: SVD sampling strategy also assigns least importance to time periods with missing data.

![Figure 2.6: Performance of CX method, in terms of the percent root mean distortion (PRD), for compression and different sampling strategies. The SVD sampling scheme provides the best performance among the proposed strategies. Therefore, for compressed sensing and compressed prediction, we will consider SVD sampling scheme.](image)

Figure 2.6: Performance of CX method, in terms of the percent root mean distortion (PRD), for compression and different sampling strategies. The SVD sampling scheme provides the best performance among the proposed strategies. Therefore, for compressed sensing and compressed prediction, we will consider SVD sampling scheme.
2.5.2 Compression

We now investigate the CX method for compression of traffic data. We apply random, energy and SVD sampling strategies to find the appropriate set of columns. For each strategy, we repeat sampling 5 times and report the average reconstruction accuracy. Our benchmark is Principal Component Analysis (PCA), as it is considered as the optimal linear transformation. Fig. 2.6 shows the compression performance of different methods and sampling schemes. As expected, PCA slightly outperforms the proposed method in terms of compression error. This is in agreement with Theorem 1, which relates the CX decomposition to SVD. However, as we pointed out earlier, the low-dimensional model obtained by PCA cannot be applied for compressed estimation and prediction, since it requires data from all links. By contrast, the low-dimensional models generated by the CX method are perfectly suitable for both applications, which is the main advantage of our approach compared to PCA.

Let us also analyze the efficiency of different sampling schemes. Amongst the proposed sampling strategies, the so-called SVD scheme provides the best results. Fig. 2.7a depicts the assigned importance for the SVD sampling method. From that figure, it can be seen that the SVD scheme assigns higher selection probability to roads with large traffic speed variations. Fig. 2.7b shows the assigned importance for energy sampling method. Unlike the SVD sampling strategy, the energy scheme mostly relies on high-speed expressways segments and thereby it almost entirely ignores the variations across non-primary roads. The energy sampling method is more suitable for traffic parameters (e.g., number of OD trips) that simultaneously can exhibit large energy and variations [82].

The random sampling scheme yields similar performance to the SVD scheme, especially for high compression ratios (see Fig. 2.6). However, for low compression ratios, the SVD strategy outperforms the random sampling scheme.

As the SVD sampling method outshines the other two sampling schemes, we will only consider the SVD sampling method from now on. An important question is whether the SVD sampling scheme leads to subnetworks that are stable over time
Table 2.1: Overlap (%) among the $k$ links with the highest selection probability (calculated by the SVD sampling method) in the three months of traffic data.

(i.e., the selected subsets of links are identical/similar over time). To assess the stability of the subnetwork generated by the SVD sampling method, we applied this method to each of the three months (August, September, October, 2011) of traffic data separately. For each month, we sort the road segments in descending order according to the assigned probability by the SVD sampling method. Hence, the most representative roads are at the top of these lists. Next we select the first $k$ links of each list, with $k$ corresponding to 10%, 25%, and 50% of the links in the network. If the subnetwork is stable across time, the three short lists of top-$k$ links should have many links in common. The results of this analysis are summarized in Table 2.1, where the percentage of common links is provided. As can be seen from this Table, most links in the subnetwork are consistently selected for all three months, suggesting that the SVD sampling method results in a subnetwork that is stable over time.

Let us now explore the subnetworks obtained by SVD sampling. Fig. 2.8 shows the road categories of the whole network and selected subnetworks. Subnetworks mostly contain the road segments of a lower hierarchy level ("Slip" and "Other"). These road segments exhibit high variations. On the other hand, selected subnetworks...
contain a small percentage of expressways i.e., “CAT A” roads (see Fig. 2.8). Nearby expressway segments exhibit similar behavior and hence they can be modeled by fewer components, in comparison with the roads of lower hierarchy level which exhibit more irregular traffic patterns.
Figure 2.7: Assigned importance to each road segment in the network by SVD (top) and the energy sampling method (bottom).
2.5.3 Compressed Sensing

We now investigate the case of compressed sensing. We aim to reconstruct the average speed at each link in the entire network by collecting data from a small subset of roads. The relationship matrix $X_h$ is determined from the training set (data from Aug-Sep, 2011), and the reconstruction error is assessed on the test set (data from October, 2011). Fig. 2.9 shows the reconstruction accuracy of the proposed method for three different approaches: In the first approach we select the subset of road segments according to the SVD sampling scheme (see solid line in Fig. 2.9). In the second approach we cluster the network according to the category of the road. For each cluster, we select the subset of the road segments using SVD sampling scheme. Then, we perform compressed sensing for each cluster separately (see dashed line in Fig. 2.9). In the third approach we use the identical set of
roads as defined in the second approach to perform network estimation. Unlike in the second approach, we do not perform any clustering here (see dotted line in Fig. 2.9). Intuitively, the difference between the second and the third approach shows the gain obtained by network clustering. Fig. 2.9 indicates that applying the compressed sensing method to different road categories leads to better estimation performance for the entire network. As expected, the reconstruction accuracy of all three approaches increases with the size of subnetwork. Let us now investigate the error of compressed sensing in more details. In our analysis, we consider the subset of links as defined in the SVD scheme without clustering.

The overall compressed sensing (estimation) error is caused by information loss due to compression of traffic data and changes in traffic behavior between training and testing periods. Table 3.3 shows the MSE of the individual error components, the

---

Figure 2.9: Accuracy of compressed sensing method for different approaches and compression ratios in terms of Percent Root mean Distortion (PRD).
<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE (Compression)</td>
<td>0.01</td>
<td>2.18</td>
<td>7.27</td>
<td>11.95</td>
<td>15.93</td>
</tr>
<tr>
<td>MSE (Non-Stationarity)</td>
<td>19.77</td>
<td>29.07</td>
<td>31.20</td>
<td>29.23</td>
<td>27.35</td>
</tr>
<tr>
<td>ξ(Correlation)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total MSE</td>
<td>19.78</td>
<td>31.25</td>
<td>38.47</td>
<td>41.18</td>
<td>43.28</td>
</tr>
</tbody>
</table>

Table 2.2: MSE of the proposed method for application of compressed sensing and for different compression ratios.

correlation between the two errors components and the total MSE, for different compression ratios. As it can be seen from Table 3.3 the non-stationarity of the traffic data is the main contributor to the estimation error. As expected, the error associated with the compressibility of traffic data increases with the compression ratio. Furthermore, Table 3.3 also shows that there is no correlation between compressibility and non-stationarity of traffic data.

### 2.5.4 Compressed Prediction

The third application of the proposed CX-based method is traffic prediction. In compressed prediction, we use the future state of a small subset of roads to predict future traffic condition for the whole network. We also consider the traditional (baseline) approach where the speed for each road segment is explicitly predicted. For both approaches we use the identical SVR settings as explained in Section 2.4.2. Fig. 2.10 depicts the prediction accuracy of the proposed and traditional methods for different prediction horizons and various compression ratios. As expected, the compressed method has slightly larger PRD error than the traditional approach. This additional error decreases with decreasing compression ratio. Also, the additional error decreases for large prediction horizons (see Fig. 2.10). Naturally, it is difficult to predict for larger horizons even with traditional approach (see dashed line in Fig. 2.10e and Fig. 2.10f). Hence, in such cases the error due to prediction tends to become the dominant component.
Figure 2.10: Prediction performance of the proposed and traditional prediction methods for 5 (top), 10 (middle) and 30 minutes (bottom) prediction horizons. The figures show the PRD (left) and the MAPE (right) error.
Figure 2.11: Prediction performance of the proposed and traditional prediction methods for 5 (top) and 10 minutes (bottom) prediction horizons in the case of workdays (M-F). The figures show the PRD (left) and the MAPE (right) errors.

Fig. 2.11 shows the temporal variations in the performance of proposed and traditional prediction methods. The results are averaged across all work days (Monday-Friday) and entire network. The performance of the proposed and traditional methods follow the similar pattern. As expected, traffic conditions during peak hours are difficult to predict even with traditional approach (see Fig. 2.11). We suggest that during these hours one should consider to explicitly collect data from larger portion of the network. Opposite to this, during non-peak hours less data is required to provide acceptable prediction performance. Hence, adaptive sampling should be considered by traffic practitioners if they to decide to employ the proposed method.

Fig. 2.12 shows the prediction performance for different road categories for the (pro-
Table 2.3: MSE of the proposed method for application of compressed prediction and for different compression ratios.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE (Estimation)</td>
<td>19.78</td>
<td>31.25</td>
<td>38.47</td>
<td>41.18</td>
<td>43.28</td>
</tr>
<tr>
<td>MSE (Prediction)</td>
<td>33.73</td>
<td>28.23</td>
<td>24.15</td>
<td>20.95</td>
<td>18.61</td>
</tr>
<tr>
<td>ξ (Correlation)</td>
<td>5.80</td>
<td>6.01</td>
<td>5.52</td>
<td>4.83</td>
<td>4.42</td>
</tr>
<tr>
<td>Total MSE</td>
<td>41.91</td>
<td>47.46</td>
<td>51.58</td>
<td>52.47</td>
<td>53.05</td>
</tr>
</tbody>
</table>

The proposed approach of compressed prediction provides substantial reduction in computational complexity by explicitly predicting the variables at a small representative set, followed by (fast and efficient) extrapolation to the entire network. This reduction in computation time is obtained at the expense of a small increase in the prediction error (see Fig 2.10). The computation times for the compressed and traditional methods are reported in Table 2.4. For the purpose of benchmarking, we tested the compressed and uncompressed prediction algorithms on a 2.67 GHz MacPro server on a single core with 32GB of random-access memory (RAM). Note that the MacPro server has 32 cores, and therefore, the total computational time...
Figure 2.12: Prediction performance of the proposed and traditional prediction methods for different road categories and 5 min prediction horizon.
Table 2.4: Computation time (in seconds) of the compressed method for 5 minute prediction horizon. The traditional approach requires 91.63 sec to perform prediction for the whole network.

for updating the predictions for a single horizon in the entire network is about 3s for the uncompressed method, compared to 0.3s for the compressed method with a compression factor of 10. We assume that training phases are performed offline for both methods. Prediction time for compressed method involves the time to predict traffic variable for a subset of links and time required to perform network wide extrapolation. As Table 2.4 shows, the latter can be neglected. Consequently, the required computation time for compressed prediction is proportional to the number of the road segments in $C$. Consequently, the reduction in computational complexity is approximately proportional to the compression ratio (see Table 2.4).

## 2.6 Discussion

In Section 2 we proposed a method that allows us to make a favorable trade off between prediction accuracy and computational complexity. The method is capable of making online traffic predictions for large traffic networks. The proposed method extrapolates traffic information from a “small” portion of a network to the entire network using a straightforward matrix multiplication. These computations can easily be performed on portable devices (such as smartphones or devices on-board) that nowadays have significant computational resources. We investigate this option in more details in Section 3.
Chapter 3

Implementation of CX Models on Smartphones

Summary - The role of smartphones in traffic applications is typically limited to the front end interface. Although smartphones have significant computational resources, which are most likely to increase further in the near future, most of the computations are still performed on servers. In this chapter, we study the computational performance of centralized, decentralized and hybrid architectures for Intelligent Transportation Systems applications. We test these architectures on various Android devices. For implementation, we consider the Android Software Development Kit (SDK) and the Android Native Development Kit (NDK). Numerical results show that recent smartphones take less than one second to estimate the speed for each road segment in a network of 10,000 links from speed measurements at 1,000 links. The proposed decentralized architecture significantly reduces the overhead of the communication network and paves the way for new cooperative traffic applications and operations.
3.1 Introduction

Smartphones currently play a vital role in everyday traffic management operations. Being equipped with global positioning system (GPS) and Internet access, mobile phones are frequently used to collect the location and the (instantaneous) speed of vehicles [34]. Internal sensors of smartphones such as accelerometers and GPS devices can further provide information about the behavior of drivers and the road conditions [32]. High penetration rates and powerful visualization capabilities of smartphones facilitate delivery of valuable traffic information to the end users (e.g., drivers) [99].

In most traffic applications, smartphones are used to: (i) collect traffic data and send this information to the server for further processing; (ii) deliver the traffic information (e.g., estimated travel time) to the driver once the server performs all required computations. Hence, all of the computational operations are performed on servers, while the role of smartphones is often restricted as a front end interface. We refer to this as centralized system architecture which is traditionally deployed for traffic applications on smartphones. Although a typical smartphone possesses significant computational resources, which are most likely to increase in the future, this potential has hardly been explored in traffic applications. The only exceptions are a few safety related applications [42] where all the computations are internally performed with the help of native programming languages.

In this study, we explore how traffic speed estimation and prediction, and travel time prediction can be implemented in a decentralized manner on smartphones, as opposed to a centralized server. Although this method can be applied to many other different domains, we focus on ITS applications. We use a column-based (CX) matrix decomposition method to allocate certain computations from the server to the smartphones. The CX method approximates any arbitrary matrix as a product of a subset of its columns and an extrapolation matrix. The extrapolation matrix contains the relationship functions (coefficients) that expresses every column of the matrix in terms of the basis provided by a small subset of the original columns.
The extrapolation matrix is inferred from the training data set, for a given subset of columns. This subset of the columns can be selected according to different criteria [79]. The CX method has proven to be efficient in traffic networks since traffic parameters (e.g., speed, volume) across different roads tend to be related [75]. The CX method has been deployed for applications of compression, estimation and prediction where the traffic conditions in the entire network (represented as a matrix) are assessed using the information from a few road segments (columns of the matrix) [100]. However, this method has previously not been deployed and systematically tested in an Android environment.

For the purpose of the study, we have developed an Android app that aims to assess the traffic conditions of the entire network from the data of a subset of the road segments. The smartphone app relies on the column based (CX) matrix decomposition method. In other words, the developed app estimates the traffic conditions in the entire network through vector-matrix multiplications where: (i) the row vector contains the traffic data (e.g., speed) for a subset of the links; (ii) the extrapolation matrix (or relationship matrix) encompasses relationship functions between this subset of the links (defined in (i)) and the entire network [100]. The extrapolation matrix can be pre-computed on the server and stored in the phone memory. In this case, the server sends the traffic data for the predefined subset of the links to the smartphones where the extrapolation is performed. We refer to this as the Decentralized Non Adaptive (D-NA) mode of operations. The D-NA architecture aims to reduce the overhead of the communication network since only the subset of the predefined links is explicitly monitored.

In the D-NA mode of operations, we assume that traffic data, for the predefined subset of road segments, will be available during all time instants. However, this may not be a reasonable assumption due to malfunctions of fixed sensors and the nature of mobile sensors (e.g., probe vehicles) [101]. In such cases, we can still use the server to collect data from a subset of available sensors and send it to the smartphone. Then, the smartphone can compute the corresponding extrapolation matrix for the obtained data and use it to perform network extrapolation. We refer...
to this as the Decentralized ADaptive (D-AD) mode of operations. The proposed D-AD architecture is a stepping stone toward cooperative (or server-free) mode of operations, since information about traffic conditions may be provided directly by other probes.

We use the decentralized modes of operations for other application domains, beyond estimation, such as traffic and travel time predictions. In these applications, the server performs traffic prediction for a subset of the links and sends the predicted data to the smartphone where the extrapolation is performed. Similar to the application of network estimation, we can also compute the relationship matrix on the server (D-NA mode) or the smartphones (D-AD mode). With the help of the relationship matrix, the smartphone app can compute the future conditions for the entire network and different prediction horizons. This information can then be used for travel time calculations of potential routes. For each road segment of these routes, the algorithm estimates when the driver will be traveling through it, and chooses the prediction horizon accordingly. Then, the smartphone infers the link travel time by dividing the predicted traffic speed by the length of the segment. Finally, the smartphone computes the total travel time as a sum of the link travel times.

We evaluate the execution time of the decentralized architecture for different Android platforms and smartphone devices using traffic data from the Singapore network. For development platforms, we consider the commonly used native development kit (NDK) and software development kit (SDK). We also consider different generations of smartphone devices, and we evaluate the performance of different smartphones with the help of the Android emulator [102]. We measure the execution time of the app by inserting monitoring functions into the Android operating system. Since the execution time depends upon the size of the underlying network, we assess the app performance in three test network comprising 2,156 (highway network), 5,000 and 10,000 links.

Our experiments show that the proposed modes of operations represent promising
alternatives to the commonly used centralized framework. The decentralized mode of operations can be efficiently deployed if the development platform of the traffic app relies on the native development kit (NDK) platform where the physically complex operations are coded in native languages. In particular, the NDK development platform outperforms the traditional software development kit platform (SDK) by 93-95% in the case of the complex problems. Furthermore, the NDK execution time of the app is highly acceptable for all tested modes of operations, even in the case of large traffic networks. The experimental results also show that memory requirements of the traffic app are satisfactory in most of the tested cases.

Nowadays, traffic applications are increasingly relying on various sensors to collect data about mobility conditions. These applications then use these large data sets to improve user’s mobility experience. The centralized architecture, however, can potentially become a bottleneck in terms of scalability, connectivity and service delivery, in the future. At the same time, smartphones offer great untapped potential to explore decentralized and hybrid architectures for traffic related applications. These architectures can also prove useful for device-to-device communication and vehicle-to-vehicle communications. In this study, we consider the problem of traffic state estimation in the context of above mentioned architectures. Our main contributions in this regard are as follows: We compare the computational performance of centralized vs decentralized architecture by using smartphones. We also discuss the choice of different development platforms for Android OS for traffic related applications.

### 3.2 Column-based Method for Traffic Applications on Smartphone

In this section, we explain how we use the column based (CX) matrix approximation to allocate certain computations from the server to smartphones, for applications of traffic speed estimation and prediction, and travel time prediction.
3.2.1 Column-based (CX) Matrix Approximation

The column based method approximates a matrix $A \in \mathbb{R}^{m \times n}$ as the product of low dimensional matrices $C \in \mathbb{R}^{m \times c}$ and $X \in \mathbb{R}^{c \times n}$, such that $\hat{A} = CX$ [81]. Matrix $C$ contains $c$ columns of the matrix $A$ (see Fig. 3.1a). The relationship (or extrapolation) matrix $X$ expresses every column of $A$ in terms of the basis provided by the columns of $C$ [81]. For given matrices $C$ and $A$, we compute the matrix $X$ as:

$$X = C^+ A,$$

where $C^+$ is the Moore-Penrose pseudo-inverse of matrix $C$ [84].

There are a few sampling strategies to select $c$ columns [79]. The SVD based sampling technique, which assigns higher selection probability to the columns with larger variations, has the best performance (see Section 2.2.1). The SVD strategy assigns the selection probability to each column $\{P_{ai}\}_{i=1}^{n}$ in proportion to the Euclidean norm of the top $k$ right singular vectors of the matrix [81]:

$$P_{ai} = \frac{1}{k} \sum_{j=1}^{k} v_{ij}^2 \quad \forall \ i \in \{1, \ldots, n\},$$

where $v_{ij}$ is the $i$-th coordinate of $j$-th right singular vector.

The other selection strategy that performs reasonably well is random sampling (see Section 2.2.1). This algorithms assigns equal selection probabilities to each column of the matrix $A$. We refer to [81] for detailed information about the low-dimensional network representation and different sampling methods.

3.2.2 CX Method for Traffic Speed Estimation

We start by explaining how we use column based (CX) low-dimensional models to assess current traffic conditions in the entire network through straightforward vector matrix multiplication. For this purpose, we consider the traffic data in the form of
Figure 3.1: Top: An example of a matrix (blue) and a subset of columns (red). The explicitly observed red shaded values are used to infer the blue shaded values. Bottom: An analogous example of traffic network (blue) with the predefined subset of locations where the traffic is explicitly monitored (red).
a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where the columns of the matrix $\{\mathbf{a}_i\}_{i=1}^n$ contain traffic data from different roads $\{s_i\}_{i=1}^n$. In that way we can write the traffic data matrix as $\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_n]$. Fig. 3.1b shows an arbitrary traffic network with $n$ road segments. Suppose that the matrix $\mathbf{C}$ contains the observed traffic conditions at $c$ specific locations in the network, such that $\{\mathbf{c}_1, \ldots, \mathbf{c}_c\} \subseteq \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$. These $c$ locations are shown as red rectangles in Fig. 3.1b where $\mathbf{c}_i$ is a row vector that contains the traffic parameter (e.g., speed) at these $c$ specific locations at time $i$. Then, at time $i$ we can estimate the traffic conditions of the entire network ($\mathbf{\alpha}_i \in \mathbb{R}^{1 \times n}$) as:

$$\hat{\mathbf{\alpha}}_i = \mathbf{c}_i^\top \mathbf{X}, \quad (3.3)$$

where $\mathbf{X} \in \mathbb{R}^{c \times n}$ is the relationship matrix that maps the subnetwork associated with $\mathbf{C}$ to the entire network represented by $\mathbf{A}$.

We estimate the current traffic conditions of the entire network by taking the product of traffic data for the subset of the links (given in the form of a row vector) and the relationship matrix (see (3.3)). Since the latter remains constant at any time $i$ (and for a given subset $\mathbf{c}$) it motivates us to explore whether network extrapolation can be executed on a smartphone. We propose to store the extrapolation matrix in the phone memory and use it to perform extrapolation once the traffic data (for given subset of the links) is fetched from the server. We refer to this as the Decentralized Non Adaptive (D-NA) method since the relationship matrix is pre-computed on the server in an offline manner. The proposed framework will significantly reduce the communication load since only a few links need to be explicitly monitored (see Fig. 3.1b).

The D-NA method assumes that traffic data for the predefined subset of the links in the network is available all the time. However, this may not be a reasonable assumption since the traffic sensors are prone to faulty operations and damages (see Fig. 3.2) [101]. One alternative is to use the server to re-compute the extrapolation matrix for the available subset of the links (see green circles in Fig. 3.2) and send it to the smartphone. However, this is not a feasible solution since the relationship
matrix has to be transmitted for each new subset of the explicitly observed links. Another alternative is to re-compute the appropriate extrapolation matrix $X_*$ on the smartphone as:

$$X_* = C_*^+ A,$$

(3.4)

where $C_*$ contains the historical traffic information for the subset of any $c_*$ locations where the traffic data is currently available (see green circles in Fig. 3.2). In this case, the historical traffic data for the entire network, given as matrix $A$, has to be stored in the phone memory, as part of the smartphone app (see (3.4)). Finally, we estimate the traffic conditions of the entire network as $\hat{\alpha}^i = c_*^i X_*$. We refer to this as the Decentralized ADaptive (D-AD) scenario since the relationship matrix is calculated in an adaptive manner, using the smartphone resources.

We deploy the proposed method for the more realistic scenario where only up-to-date traffic estimates of several potential routes are provided. We estimate the current traffic conditions of only these links ($j = 1, 2, ..., L$) that lie along the potential
routes, as follows:

\[ \hat{\alpha}_j^i = c^i x_j, \quad \forall \ j \in \{1, \ldots, L\}, \quad (3.5) \]

where \( x_j \) is \( j^{th} \) column of the relationship matrix \( X \). In this case, the list of \( L \) segments has to be either stored in the phone memory (for the commonly used routes by a user) or obtained from the server. The approach \( (3.5) \) can be deployed within any of the described decentralized modes of operations.

It is noteworthy that \( c \) (in D-NA case) and \( c^* \) (in D-AD case) are subsets of \( \hat{\alpha} \) and this helps to reduce the computational load of the matrix multiplication as follows. Let us assume that \( c_j \ (j \in \{1, 2, \ldots, c\}) \) is \( j^{th} \) element of \( c \) and \( \ell^{th} \) \((\ell \in \{1, 2, \ldots, n\}) \) entry of \( \hat{\alpha} \). The \( \ell^{th} \) column of the extrapolation matrix \( X \) has all zeros except the \( \ell^{th} \) entity which is equal 1. We improve the computation efficiency by storing the \( c \) positions of these columns, instead of performing \( c \) vector-vector multiplications.

### 3.2.3 CX Method for Speed Prediction

Similar to \( (3.3) \), we can obtain the future traffic conditions of the entire network for the \( k^{th} \) prediction horizon as:

\[ \hat{\alpha}^{i+k\Delta t} = \hat{c}^{i+k\Delta t} X, \quad (3.6) \]

where \( \Delta t \) is the sampling interval (e.g., 5 minutes). The row vector \( \hat{c}^{i+k\Delta t} \) contains the predicted traffic speed for \( c \) selected locations and \( k^{th} \) \((k = 1, 2, \ldots, p) \) prediction horizon. These predicted values \( \hat{c}^{i+k\Delta t} \) are computed on the server by means of a state-of-the-art prediction algorithm (e.g., support vector regression (SVR)). Since the extrapolation matrix \( X \) remains unchanged for different prediction horizons, we can use \( (3.6) \) to perform network extrapolation for multiple prediction horizons \((k_{i+\Delta t} \ldots k_{i+p\Delta t})\). In this case the matrix \( \hat{C} \in \mathbb{R}^{p \times c} ([\hat{c}_{i+\Delta t} \ldots \hat{c}_{i+p\Delta t}]^T) \)
has to be fetched from the server and sent to the smartphone. Similar to the application of traffic speed estimation, the extrapolation matrix can be either pre-computed on the server or re-computed on the smartphone; leading to the already described D-NA and D-AD scenarios, respectively.

### 3.2.4 CX Method for Travel Time Prediction

In the following we extend the idea of compressed prediction to additional practical applications. More precisely, we rely on the compressed prediction method for inferring the future traffic conditions along several potential routes. Next we use the speed predictions to predict the travel times along these routes.

Similar to (3.5), we predict the traffic speed of the link \( j \) and horizon \( k \) as follows:

\[
\hat{\alpha}_{j}^{i+k\Delta t} = \hat{\mathbf{c}}^{i+k\Delta t} \mathbf{x}_{j},
\]

Matrix \( \mathbf{X} \) can be computed on either the server (D-NA scenario) or smartphone side (D-AD method). By fetching the matrix \( \hat{\mathbf{C}} \) from the server and multiplying each row of the matrix \( \hat{\mathbf{C}} \) with the corresponding columns of \( \mathbf{X} \), we can compute the expected traffic conditions at different time instants. We estimate the prediction horizon (\( k \)) when the driver will be traveling through the link \( s_{j} \) as follows [103]:

\[
k_{s_{j}} = \left\lfloor \frac{1}{\Delta t}(T_{o} + \sum_{i=1}^{j-1} \frac{l_{i}}{\hat{v}_{i,k}}) \right\rfloor + 1,
\]

where \( k_{s_{j}} \) is an integer (\([1, 2, \ldots, p]\)). The length of the link \( s_{i} \) is \( l_{i} \). By \( \hat{v}_{i,k} \) we denote the predicted speed for the link \( s_{i} \) and prediction horizon \( k \) (\( k \leq k_{s_{j}} \)). The variable \( T_{o} \) represent the time offset between the current time and trip starting time. We assume that the server performs route selections by means of a state-of-the-art routing algorithm, and sends the ordered list of \( L \) road segments, along the selected route, to the smartphone. As an alternative, the lists of road segments (of commonly used routes) may be pre-stored in the smartphone memory. For each link in the list,
we use (3.8) to compute the prediction horizon $k$. Then, we apply the computed $k$ in (3.7) to obtain a link travel time. Hence, the complexity of the travel time computations for the selected route involves $L$ vector-vector multiplications (where each vector has $c$ elements) and straightforward addition of link travel times.

We use the smartphone’s resources to predict the travel time and inform the driver about the most likely traffic conditions of the particular link at time when he is expected to travel across this link. This information can help the drivers in deciding the route, mode of transportation and the departure time [104]. As part of future work, we will investigate the option to (partially) perform routing on the smartphone.

### 3.3 System Infrastructure

In the following section, we explain the different modes of operations and development platforms that we use to evaluate the computational performance of smartphone devices for the applications of traffic speed estimation and prediction, and travel time prediction.

#### 3.3.1 Mode of Operations

Fig. 3.3 shows the list of operations that are performed on the server and smartphone, for centralized and decentralized modes of operations. In the centralized (or traditional) approach the server performs all the required computations; i.e., collecting, processing and sending the traffic data to the smartphone. The smartphone is not involved in any calculations in this scenario, instead it only handles the visualization. Specifically, the smartphone overlays the obtained data on the top of underlying road map (see Fig. 3.5).

In case of the decentralized operation mode, we deploy non adaptive (D-NA) and adaptive (D-AD) scenarios (see Section 3.2.2) by allocating certain computations
### Figure 3.3: List of the computations performed on the server (orange) and smartphone (green), for different operation modes (red). Memory requirement of the traffic app is given in blue.

<table>
<thead>
<tr>
<th>Operation Modes</th>
<th>Server Operations</th>
<th>Smartphone Operations</th>
<th>Smartphone Memory Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Acquistion</td>
<td>Computation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entire Network</td>
<td>Extrapolation Matrix</td>
<td>Extrapolation Matrix</td>
</tr>
<tr>
<td></td>
<td>SVD Sampling</td>
<td>Network Extrapolation</td>
<td>Historical Data</td>
</tr>
<tr>
<td></td>
<td>Uniform Sampling</td>
<td>Travel-Time Calculation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prediction</td>
<td>Data Visualization</td>
<td></td>
</tr>
<tr>
<td>Centralized</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Adaptive</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
from the server to the smartphone devices. In particular, the server collects the traffic data for the subset of the road segments in the network and performs predictions for these roads by means of a state-of-the-art prediction algorithm [11, 65]. Next the server sends the collected and predicted data to the smartphone, which uses its own resources to perform the network extrapolation, estimate the travel time along the requested route, and visualize the computed data. We refer to this as the Decentralized Non Adaptive (D-NA) operation mode since the relationship matrix $X$ is computed in advance for the predefined subset of the links and stored in the phone memory (see Fig. 3.3). Alternatively, we can use the computational resources of the smartphone to compute the relationship matrix for any subset of road segments, followed by network extrapolation, travel time calculations and data visualization (see Fig. 3.3). To compute the relationship matrix $X$ we store the historical traffic matrix $A$ in the phone memory (see (3.4)). We refer to this as the Decentralized ADaptive (D-AD) scenario since the relationship matrix is computed in an adaptive fashion.

In this study, we evaluate the execution time and memory requirement of the traffic app for the different modes of operations and ITS applications. Although any Android platform can be used to build the app, we rely on the commonly used SDK and NDK approaches which we briefly explain in the following.

### 3.3.2 Android Platforms

We have developed an app that estimates or predicts the traffic speed by means of the CX method, and then overlays the traffic information on the top of the street map. The proposed smartphone app has two components. The first component conducts the vector-matrix operations on the traffic data. The second component performs visualization, where road segments are colored according to the inferred or predicted traffic speed (see Fig. 3.5). We have implemented apps in two environments: SDK and NDK. In Fig. 3.4, we depict the data flow in both implementations.

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE
former, we use the traditional SDK development platform to perform the computations and visualization operations, with the help of Java computation libraries and the ArcGIS module (see Fig. 3.4a). In the latter, we use the Java Native Interface (JNI) to call the native applications and libraries in Java code. We use these applications and libraries, written in C/C++, to perform vector-matrix operations. Once the computations have been performed, JNI sends the data to the Java environment where the visualization tasks are performed using the ArcGIS module (see Fig. 3.4b). ArcGIS runtime for Android is used to implement geographic information system (GIS) features like map layers, geocoding, location services, etc. [105]. GIS metadata including geographic locations of the road segments (links) is stored on the smartphones as part of the application package.

3.4 Experimental Setup

In this study, we consider the nationwide traffic network in Singapore that contains highways and arterial road segments whose average length is 85\textit{m}. The variable of interest is the average traffic speed, i.e., the mean speed of all vehicles which traverse a road segment during the given sampling interval of 5 minutes. The Land Transportation Authority provided us experimental data for a period of three months (August - October 2011). We selected 10,000 highway and arterial links which had less than 5\% of missing values. We imputed the missing data by means of the Low Dimensional CP Weighted OPTimization (LDCP-WOPT) imputation method as it can deal with large data sets [88, 89]. The relative imputation error is negligible (\sim 2\%) since only a few percent of data (\sim 3\% on average) is missing [88].

We use this data set for the following two purposes: (i) to evaluate the performance of column based (CX) low-dimensional models for the applications of traffic speed estimation and prediction, and travel time prediction; (ii) to compute the execution time of the proposed app for these applications and different modes of operations. Since the size of the network significantly impacts the required execution time, we investigate the smartphone performance for three different networks:
Figure 3.4: Major components of the traffic app for SDK (top) and NDK (bottom) scenarios. The computation part is coded using either the SDK (orange) or NDK programming platform (red). The graphics part is coded with the help of the GIS SDK module (green).
(i) a highway traffic network that consists of 2,156 links (see Fig.3.5); (ii) a traffic network consisting of 5,000 road segments; (iii) a traffic network containing 10,000 road segments (see Fig.3.5).

![City-scale traffic network of Singapore with 10,000 links of the arterial (grey) and highway (other colors) categories. The highway network contains 2,156 road segments, divided into 8 highways. Each highway is marked with different color and used as a testing route for travel time calculations.](image)

The proposed app aims to inform drivers about the expected travel time and update them with up to date traffic estimates during the trip. Hence, the app is not required to be frequently run and, therefore, the battery drainage should not be an issue. However, the precise evaluation of the energy consumption of an app is still a challenging task [106]. We will investigate the battery depletion of the proposed app as part of our future work.

### 3.4.1 Performance of Low-dimensional Models

We divide the data set into training and test subsets. The training subset contains the speed data of the months August and September, 2011. In case of D-AD mode of operations, we use speed data of one week (within two months), leading to the
eight mutually exclusive training subsets. From the training set we determine the subnetwork of \( c \) links (see (3.2) and Section 3.2.1), where \( c = n^{-1}CR \), \( \forall \ CR \in \{2, 4, \ldots , 10\} \). Moreover, from the training subset we generate relationship matrices (see (3.1) and (3.4)), and train predictors. All the predictors are trained on the training data for two months. We apply support vector regression (SVR) for prediction as it is a commonly used and effective approach [38, 11, 90, 19, 16]. The test subset contains the speed data of the month October 2011. We use the test subset to assess the accuracy of the applications. In case of D-AD mode of operations, the reported accuracy refers to the average performance of eight different training subsets.

We apply SVD and uniform sampling strategies to identify the \( c \) links in the network. The SVD selection strategy is deployed for the D-NA scenario where the server sends traffic data for a predefined subset of the locations to the smartphone. The uniform selection strategy is deployed for the D-AD scenario since the traffic data is obtained from arbitrary sets of links (without control of the user).

### 3.4.1.1 Traffic Speed Estimation

We collect traffic information at \( c \) locations and extrapolate this information through the network using the relationship matrix (see (3.3)), inferred from the training data set.

### 3.4.1.2 Speed Prediction

We predict the traffic variable at \( c \) locations using a baseline (SVR) predictor and extrapolate these predictions through the network using the relationship matrix (see (3.6)). As benchmark, we perform prediction by means of the same SVR algorithm at every single link in the network, as opposed to only the links in the (random) subnetwork. This is our baseline method.
3.4.1.3 Travel Time Prediction

We use the predicted traffic information (for multiple prediction horizons) to assess the travel time along each direction of 8 highways in the network (see Fig. 3.5). As benchmark, we consider commonly used historical and instantaneous travel time methods [104].

3.4.1.4 Performance Measure

We compute the percent root mean distortion (PRD) to evaluate the performance of the applications. The PRD quantifies the error as:

\[
\text{PRD}(\%) = \frac{\| T - \hat{T} \|_F}{\| T \|_F}.
\] (3.9)

For the applications of traffic estimation and speed prediction, \( T \) and \( \hat{T} \) are matrices that contain the true and inferred speed values, respectively for the entire network and all testing time instants. In case of traffic estimation \( \hat{T} = CX \) while for the application of the traffic prediction \( \hat{T} = \hat{CX} \). For the application of travel time prediction, \( T \) and \( \hat{T} \) are matrices that contain the true and predicted travel times, respectively for 16 test routes in the network and all testing time instants.

3.4.2 Computation Time

We test the computational performance of the smartphones for the above ITS applications. We evaluate the execution time of the app for different modes of operations, development platforms, and smartphone devices using the Android emulator. The Android emulator accurately approximates the execution time of the proposed app for different smartphone devices (see Fig. 3.6) [102]. The execution time is measured by inserting the monitoring functions into the Android operating system [102]. Our development platforms rely on either the NDK+SDK framework or only the SDK.
platform (see Fig. 3.4). In the former, the computation components are coded in
NDK framework using C/C++ libraries while the visualization part is executed
through the SDK. In the latter, traffic app is built in commonly used SDK environ-
ment. Table 3.1 shows the specifications of different smartphone devices that are
used in the experiments, as well as the experimental environment.

![Graph showing computation time](image)

Figure 3.6: Computation time of the emulator (shaded bars) and the corresponding
smartphone device (solid bars) for 6 randomly selected test cases (x-axis).

3.5 Results

In this section we analyze the performance of the low-dimensional models and exe-
cution time of smartphone app for the applications of traffic speed estimation and
prediction, and travel time prediction.
### Table 3.1: Technical characteristics of the tested smartphones and the server.

<table>
<thead>
<tr>
<th></th>
<th>Galaxy S2</th>
<th>Galaxy S3</th>
<th>Nexus 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Chip</strong></td>
<td>Exynos 4210</td>
<td>Exynos 4412</td>
<td>Snapdragon 800</td>
</tr>
<tr>
<td><strong>Processor</strong></td>
<td>2 Cores (1.2Ghz)</td>
<td>4 Cores (1.4Ghz)</td>
<td>4 Cores (2.2Ghz)</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>1024 MB</td>
<td>1024 MB</td>
<td>2048 MB</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>8GB</td>
<td>16GB</td>
<td>16GB</td>
</tr>
<tr>
<td><strong>Android Version</strong></td>
<td>4.2</td>
<td>4.3</td>
<td>4.4.2 (kitkat)</td>
</tr>
<tr>
<td><strong>Server OS</strong></td>
<td>Microsoft Windows 8.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IDE</strong></td>
<td>Eclipse v10</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sun Java Version</strong></td>
<td>1.7.067</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Android SDK</strong></td>
<td>Revision 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Android NDK</strong></td>
<td>Revision 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ARC GIS Version</strong></td>
<td>g++ 4.8.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Android Emulator</strong></td>
<td>Genymotion 1.6 with ARM patch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.5.1 Performance of the CX Models for Traffic Applications

We use data from the Singapore highway network to evaluate the performance of the CX-based models. We sample the road segments according to the SVD and uniform sampling methods, since these techniques are deployed in D-NA and D-AD scenarios, respectively. For each sampling strategy and compression ratio, i.e., the ratio of the number of links in the subnetwork \( c \) and the total number of links \( n \), we repeat the experiments five times and report the mean value. The standard deviation around the mean (of these five runs) is typically less than .001 due to significant homogeneity of the highway traffic network.
Table 3.2: PRD error [%] of the applied low-dimensional approach for applications of estimation and for two sampling strategies.

<table>
<thead>
<tr>
<th>Application and sampling strategy</th>
<th>Training subset</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation (SVD)</td>
<td>2 months</td>
<td>2.80  5.13  7.20</td>
</tr>
<tr>
<td>Estimation (uniform)</td>
<td>2 months</td>
<td>2.95  5.31  7.67</td>
</tr>
<tr>
<td>Estimation (uniform)</td>
<td>1 week</td>
<td>5.15  7.11  9.06</td>
</tr>
</tbody>
</table>

3.5.1.1 Traffic Speed Estimation

Table 3.2 shows the estimation accuracy of the sampling techniques for different compression ratios. As expected, the SVD-based sampling strategy outshines the uniform sampling where all the columns (roads) have the same probability of being selected. The minor difference in performance is a consequence of the high homogeneity of the test network that only contains the highway segments. As can be seen from Table 3.2, there is a slight reduction in estimation performance of the D-AD mode of operations as a consequence of reduced training subset.

3.5.1.2 Speed Prediction

Fig. 3.7 shows the prediction accuracy of traditional and compressed methods, for different compression ratios and various prediction horizons. The traditional (baseline) prediction algorithm explicitly predicts the speed for each road segment through support vector regression (SVR) [11, 90, 19, 16]. In the case of compressed methods, we test two scenarios: (i) D-NA scenario where the roads segment are sampled using the SVD sampling strategy and extrapolation matrix is inferred from the training set that contains speed data for two months (see dashed lines in Fig. 3.7); (ii) D-AD scenario where the links are selected using the uniform sampling strategy and relationship matrix is inferred from the training set that contains speed data for one week (see solid lines in Fig. 3.7). Table 3.3 shows the required computation times for the compressed and traditional methods when all experiments are run on a 2.67
Figure 3.7: The performance of compressed and traditional prediction methods. In the case of compressed prediction, the roads are sampled using the SVD and random sampling strategies which are deployed in D-NA and D-AD modes of operations, respectively. The roads are sampled from either two months of training data (D-NA mode) or one week of the training subset (D-AD mode).

GHz MacPro server on a single core and 32GB of random-access memory (RAM). The results indicate that column based (CX) low-dimensional methods provide significant reduction in computational complexity by explicitly dealing with the traffic variable for only a small subset of road segments in the network (see Table 3.3). This reduction in the computational cost comes at the expense of a negligible increase in error (see Fig. 3.7). Similar to the application of traffic estimation, the degradation in prediction accuracy of the D-AD mode of operation is associated with the limited training data set.
### Table 3.3: Computation time (in seconds) for compressed and traditional prediction methods. Network extrapolation is performed on the server.

<table>
<thead>
<tr>
<th>CR</th>
<th>2</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR prediction</td>
<td>5.51</td>
<td>2.75</td>
<td>1.10</td>
</tr>
<tr>
<td>Extrapolation</td>
<td>0.0042</td>
<td>0.0025</td>
<td>0.0015</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>5.52</td>
<td>2.76</td>
<td>1.10</td>
</tr>
<tr>
<td>Traditional method</td>
<td>10.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Savings</strong></td>
<td>49.77%</td>
<td>74.89%</td>
<td>89.99%</td>
</tr>
</tbody>
</table>

3.5.1.3 **Travel Time Prediction**

In the following, we extend the idea of compressed prediction for the application of travel time prediction. Usually, drivers are interested in traffic conditions along the potential routes more than in traffic data for the entire network. This fact motivates us to explore how compressed prediction can be used to infer future conditions and estimate travel time along the potential routes for provided start and end points as well as start time of the trip. The ordered list of links between start and end points is obtained from the server. For these links we apply (3.8) to estimate the prediction horizon when the driver will be traveling across them (see Section 3.2.4).

We tested this method along 16 routes, i.e., for each direction of 8 highways in the network (see Fig. 3.5). As a benchmark, we also test the historical and instantaneous methods for travel time calculations [103]. Table 3.4 shows the average accuracy of the proposed method (for different compression ratios) and benchmark methods. As can be seen from Table 3.4, the proposed method outperforms the historical and instantaneous models and shows that taking speed predictions into account leads to more accurate travel time estimation. Detailed analysis of the quality of travel time estimation using speed predictions can be found in our another study [107].
### Table 3.4: Performance of the proposed and benchmark methods for travel time estimation.

<table>
<thead>
<tr>
<th>PRD (%)</th>
<th>CR=1</th>
<th>CR=2</th>
<th>CR=4</th>
<th>CR=10</th>
<th>Instantaneous</th>
<th>Historical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.43</td>
<td>2.75</td>
<td>2.82</td>
<td>3.24</td>
<td>4.48</td>
<td>6.44</td>
</tr>
</tbody>
</table>

3.5.2 Computation Time of the App

The performance of the CX methods encourage us to explore the option where network extrapolation is performed on the smartphone which possesses significant computational power. This decentralized framework can diminish the burden of communication in traditional (centralized) system architecture. In the following, we analyze the execution time of smartphone app for the applications of traffic speed estimation and prediction, and travel time prediction.

3.5.2.1 Traffic Speed Estimation

Table 3.5 shows the execution times of different smartphone devices and Android platforms to perform network extrapolation (for different compression ratios) and display traffic information on the road map once the extrapolation is performed. The reported results refer to the highway network in Singapore and decentralized non adaptive (D-NA) mode of operations. From Table 3.5 it can be seen that the NDK development platform requires significantly less time than the commonly used SDK. Since in both scenarios the visualization task is performed in the same manner, the significant time savings are obtained for computational tasks. Table 3.5 shows that newer Android models require less computation time to perform network extrapolation as a consequence of the increased computational resources (see Table 3.1). Table 3.5 also shows the size of the corresponding extrapolation matrix that needs to be stored on the phone, as part of traffic app. The elements of the extrapolation matrix have type double. As can be seen from Table 3.5 memory requirement and app execution time (in the case of the NDK platform) are acceptable from the user point of view. As the NDK platform clearly outshines the SDK platform, we will
only consider the NDK programming platform from now on. Similarly, we will only consider the smartphone Nexus 5 in the rest of this chapter.

In the following, we present the performance of the D-NA mode of operations in the case of traffic networks that contain 5,000 and 10,000 road segments (see Table 3.6). Here, the server sends traffic data for the subset of the links to the smartphone where data extrapolation is performed with the help of the extrapolation matrix $X$.

Table 3.6 shows the promising execution time of the D-NA mode of operations, especially for higher compression ratios. However, the memory requirement of traffic app may make it less user friendly since the large extrapolation matrix $X$ has to be stored in the phone memory (see Table 3.6).
We now investigate decentralized adaptive (D-AD) mode of operations. Unlike in the D-NA case, the traffic data comes from the subset of locations which are only revealed in real-time and will change from one time instant to another. Consequently, the pre-defined relationship matrix cannot be used here. We deploy the NDK development platform to calculate the relationship matrix $X$ and perform network extrapolation in the case of three test networks. For computation of the matrix $X$, we use the historical data for the entire network, stored in the phone memory and fetched every time when a new subset of data is received. The stored historical data matrix encompasses one week of traffic data (2016 time instants) since the similar traffic patterns can be observed during the same week days [79]. Table 3.7 shows the execution time of the traffic app for different sizes of the underlying traffic network. The computation time for the extrapolation matrix accounts for a significant portion of the total execution time, especially for the lower compression ratios and larger traffic networks (see Table 3.7). Hence, the D-AD mode of operations might be efficiently deployed in the following cases: (i) in small and moderate traffic networks (e.g., a few thousand links) regardless of the compression ratio; (ii) in large traffic networks in the case of higher compression ratios. In the case of large traffic networks and lower compression ratios (e.g., CR=2) the smartphone has to compute the pseudo-inverse of a large matrix which is still an intensive task for the existing smartphone resources (see Table 3.7). Table 3.7 also provides the memory requirements of the historical traffic data, stored in the phone memory. The stored matrix contains the historical speed data for one week, which are stored as integer values.

### 3.5.2.2 Speed Prediction

We now explore the execution time of the decentralized non adaptive (D-NA) mode of operations for the application of compressed prediction. The predictions for $c$ road segments and $p$ prediction horizons are obtained from the server in the form of matrix $\hat{C} \in \mathbb{R}^{p \times c}$. To obtain traffic conditions in the entire network for the $k^{th}$ prediction horizon we multiply the $k^{th}$ row of $\hat{C}$ with the extrapolation matrix $X$. 

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE
<table>
<thead>
<tr>
<th>Number of links in the network, and size of the data matrix A</th>
<th>Execution time, required to compute / perform:</th>
<th>Compression ratio (CR)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Extrapolation matrix</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D-NA scenario</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D-AD method</td>
<td></td>
</tr>
<tr>
<td>2,156 (12.4 Mb)</td>
<td>3,077</td>
<td>1,312</td>
</tr>
<tr>
<td></td>
<td>859</td>
<td>698</td>
</tr>
<tr>
<td></td>
<td>4,378</td>
<td>1,897</td>
</tr>
<tr>
<td>5,000 (28.2 Mb)</td>
<td>16,881</td>
<td>6,088</td>
</tr>
<tr>
<td></td>
<td>3,077</td>
<td>1,312</td>
</tr>
<tr>
<td></td>
<td>19,958</td>
<td>7,400</td>
</tr>
<tr>
<td>10,000 (57.6 Mb)</td>
<td>221,489</td>
<td>27,443</td>
</tr>
<tr>
<td></td>
<td>6,321</td>
<td>2,964</td>
</tr>
<tr>
<td></td>
<td>227,810</td>
<td>30,407</td>
</tr>
</tbody>
</table>

Table 3.7: The execution time [ms] of the app for the D-AD scenario and different networks. The total execution time includes: (i) computation of the relationship matrix; (ii) network extrapolation; and (iii) data visualization.

stored in the phone memory. The app execution time for the single prediction horizon is equivalent to the execution time for the application of speed estimation which we discussed earlier (see Tables 3.5 and 3.6). Naturally, execution time for \( p \) multiple prediction horizons will be \( p \) times larger than corresponding times given in Tables 3.5 and 3.6.

Similar to the application of traffic estimation, we assess the execution time of the D-AD mode of operations as the sum of two components: (i) time required to compute the relationship matrix; (ii) time required to execute corresponding D-NA scenario (see Table 3.7).

3.5.2.3 Travel Time Prediction

We use (3.8) to assess the prediction horizon \( (k) \) when the driver will be traveling along the link (see Section 3.2.4). Then we substitute the computed \( k \) in (3.7) to
predict a link travel time. We repeat these computations for each of the \( L \) links that belong to the assigned route. Table 3.8 shows the app execution time for different sizes \( (L) \) of the test routes in the Singapore highways network using the NDK development platform and Nexus 5. The reported results refer to the D-NA mode of operations. Similar results are obtained for other two test networks that contain 5,000 and 10,000 links. As can be seen from Table 3.8, smartphones are capable of performing travel time calculations, even along the routes that contain hundreds of road segments, corresponding to several kilometers of route length.

In case of the D-AD mode of operations, we assess the execution time of the app as the sum of two components: (i) time required to compute the relationship matrix (see Table 3.7); (ii) time required for travel time calculations and data visualization (see Table 3.8).

### 3.6 Discussion

In Section 3 we presented our research on novel non-centralized traffic applications, by allocating certain computations from the server to user mobile devices. We proposed a theoretical foundation (by extending the idea proposed in Section 2) and evaluated the system infrastructure. The results (in terms of computation time) are quite promising and should reveal more possibilities and applications in the near future.

Intuitively, the proposed method, based on compressed network state, relies on the fact that traffic follow common patterns. However, this might not be always the case. For instance, during the inclement weather and heavy accidents the traffic...
might exhibit some pattern that have not been observed in the past and thereby the proposed method can not be explicitly applied. To partially tackle these issues, we investigate the impact of the rainfall on prevailing traffic conditions in Section 4.
Chapter 4

Impact of Rainfall on Traffic

Summary - Weather conditions tend to have measurable impact on traffic conditions of the roads. This relationship is commonly studied for specific time periods at the network level. Furthermore, existing studies typically use high-resolution traffic data which may not be available across the entire network and especially during adverse weather conditions. In this study we explore the impact of rainfall intensity on low-resolution speed data during different time periods throughout the day. We also investigate whether this additional information about rainfall may improve the prediction accuracy of data-driven models for individual roads. To do so, we incorporate the information about the rainfall intensity into ensemble methods and support vector machines. As a benchmark we only consider temporal features to predict near-future traffic conditions during rainy weather. Numerical results for 4079 road segments in Singapore confirm that rainfall impacts traffic conditions in terms of decreasing the driving speed. This reduction increases with the rain intensity and with an increase in demand. The results also show that additional rainfall data enhances the prediction accuracy for certain links; while for the others the rainfall information is not that useful.
4.1 Introduction

With the recent improvements in sensor technology, traffic-related information has been collected from multiple sources with a high temporal resolution. Significant research has focused on the incorporation of weather parameters in traffic modeling and estimating the impact of adverse weather conditions on road traffic and driving behavior. It has been proven that different weather factors such as temperature [47, 48], precipitation [49, 50] and visibility [51, 52], tend to reduce the roadway capacity and increase the risk of vehicle collisions. In particular, Agarwal et al. found that highway capacity declines up to 17% depending on rain intensity [55]. This decrease in throughput leads further to the degradation of other key traffic stream parameters such as volume and speed [49].

Rainfall intensity has been frequently identified as an external factor that has the highest impact on driving conditions and network performance [49, 53, 54, 55, 56, 57]. Quantifying such an impact could help transportation operators to better manage the safety and efficiency of transportation systems during intense rainfall accidents. For instance, the use of variable speed limits (VSLs) during the rainy conditions can increase the safety by providing the drivers with the information about the speeds that are appropriate for the prevailing conditions [108]. Similarly, weather-responsive signals are capable of generating appropriate timing plans to accommodate non-recurrent fluctuations in traffic demand and traveling speeds, explicitly caused by adverse weather [109]. At the core of these and many other weather-responsive strategies, lie an accurate assessment of the expected degradation in traffic parameters caused by the inclement/rainy weather.

Despite the fact that rainfall affects traffic performance significantly, there are only a few studies that explicitly investigate the inclusion of rainfall information into short-term traffic forecasting. Tsirigotis et al. [58] assess the effects of rainfall intensity on the predictability of traffic speed using vector autoregressive moving average models with explanatory variables. The authors concluded that information about the rainfall only marginally improves the prediction performance. Furthermore, the
authors emphasized the need for considering the large database with intense rainfall incidents in traffic forecasting applications. In another study, Butler et al. [62] applied neural networks to predict traffic volume by including weather conditions as exogenous variables. The authors revealed that additional rainfall information decreased the forecasting accuracy for traffic volumes. In addition, the authors suggested that weather information should be observed at finer intervals than 1h in traffic applications. The scope of the studies above is restricted to the special cases such as single intersection and short corridors. Moreover, the investigated data is frequently collected for a short period and / or aggregated in one-hour increments.

This study has two primary objectives: first, to investigate the impact of the rainfall on the various levels of congestion; and secondly, to determine whether including information about the rainfall can improve the performance of the state-of-the-art prediction algorithms. To do so, we first built a large database comprising of information about the rainfall intensity and average speed for 4,079 road segments in the large-city network of Singapore. Singapore was considered for analysis as it has a tropical rainforest climate where the rain can be particularly heavy and persistent throughout the year (see Fig. 4.1). The speed and rainfall data are collected with a sampling rate of 5 minutes during period of 7.5 months. The rainfall intensity is collected from the National Environmental Agency (NEA) of Singapore website in the form of an image [110]. We use Optical Character Recognition (OCR) and Doppler Radar reflectivity techniques to accurately estimate corresponding time instant and average rainfall intensity for all links in the network, respectively. Land Transportation Authority of Singapore provided us traffic information for 4,079 road segments in the form of speed band. For instance, speed band one (SB1) would mean that the average speed on certain road segment is less than 20 km/h. Similarly, speed bands 2, 3, and 4 refer to the speeds between 20 − 40 km/h, 40 − 60 km/h and speeds higher than 60 km/h, respectively. We interconnect these speed bands with the levels of congestion by assigning SB1 to heavy congestion and SB4 to low congestion.

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE
We start by assessing the network congestion level for rainy and corresponding non-rainy $5 - \text{min}$ intervals throughout the day. We compute the network congestion level for a given $5 - \text{min}$ interval as the mean value of the average congestion level of these segments which at least once exhibit rain during that $5 - \text{min}$ interval, within the observational period. In the case of inclement weather, we only consider
these time instants when the rainfall intensity, at particular segment, is positive and above a certain threshold. Similarly, to compute the network congestion level for dry weather we only consider these instances when the corresponding rainfall intensity is 0. We conduct an analysis for various periods across the day and different groups of days such as weekday and weekend. We also perform an investigation for individual road categories and different rainfall thresholds. Numerical results show that the impact of the rainfall on the network congestion level significantly varies across the day for both weekday and weekends. As expected, this impact increases with an increase in demand and rainfall threshold. The results also suggest that the impact of the rainfall on the congestion level is more severe for urban arterial than for highways.

In the second part of the study we combine speed information with the rainfall data to train and test state-of-the-art forecasting methods such as Support Vector Machines and boosting methods. We restrict our attention to these instances when the rainfall intensity is positive. As a benchmark, we perform prediction using only provided traffic information. We compare the performance of these two sets of predictors to assess the impact of additional weather information. We conduct our analysis for different periods of the day and rainfall thresholds. Numerical results show that for certain links additional rainfall information marginally improves the prediction accuracy. The number of these links, where the rainfall information is useful, significantly varies with the prediction method and time of the day.

To the best of our knowledge, this is the first study that estimates the effect of the rainfall in a large traffic network and on a high temporal resolution. Moreover, this study can be regarded as a pioneer in the integration of high-resolution weather data into the mechanism of state-of-the-art algorithms for short-term traffic forecasting.

### 4.2 Data Collection and Processing

In this section, we describe different steps involved in the collection and processing of traffic and rainfall data sets.
4.2.1 Traffic data

The Land Transportation Authority (LTA) of Singapore collects raw traffic data in the network from multiple sources such as loop detectors and probe vehicles (see Chapter 2.4.1 for more details). The provided speed information, for the evaluated period, was only available in the form of a discrete variable with range $1 - 4$ (see Section 4.1). Please note that we refer to these discrete states as speed bands or levels of congestion in an equivalent way. We represent the speed data set in the form of a matrix $A$ ($A \in Q^{m \times n}, Q = \{1, 2, 3, 4\}$). The columns of the matrix $\{a_i\}_{i=1}^{n}$ contain traffic data from different roads $\{s_i\}_{i=1}^{n}$. Rows represent time instants $\{t_i\}_{i=1}^{m}$ at which the traffic data is recorded. Each matrix cell $(a_{ij})$ represents one of the four speed bands (or levels of congestion) at location $s_j$ during the interval $(t_i - T, t_i)$ where $T$ is the sampling period. In this analysis we consider speed data for 4079 road segments for a period of 7.5 months ($A \in Q^{65880 \times 4079}$). Each of these links have less than 5% of missing values data; and the missing values are not taken in consideration in this study. Fig. 4.2 shows the experimental network, used in the analysis. For the purpose of this study we categorize the roads into two groups: (i) highway links and (ii) non-highway links that include arterial and low-priority roads (see Fig. 4.2).

4.2.2 Rainfall Data

National Environmental Agency (NEA) of Singapore uses automated weather instruments to collect information about the rainfall from more than 60 weather stations, densely distributed across the country (see Fig 4.3). The rainfall data is published on NEA internet cite in the form of an image every 5-15 minutes (see Fig. 4.3) [110]. This image contains the information about the rainfall intensity for each location in the network, used precipitation scale, time stamp, and the map of the highway network that helps us to integrate precisely weather and traffic data (see Fig 4.3). We collected rainfall information during the 7.5-months period that corresponds to
the speed data set. We organized rainfall (weather) information in the form of a matrix \( B \in \mathbb{R}^{65880 \times 4079} \). Each matrix cell \((b_{ij})\) in the matrix \( B \) represents the estimated rainfall intensity at the location \( s_j \) during the interval \((t_i - T, t_i)\).

We estimate rainfall intensity at the location \( \{s_i\}_{i=1}^n \) as follows: first we overlay the rainfall and traffic road maps to estimate the location of the link \( s_i \) on the rainfall map (see Fig. 4.4). To derive rainfall rate at a single pixel of the map visualizations, we use Doppler Radar reflectivity:

\[
    r = a(10^d)^b
\]

(4.1)

where \( r \) is rainfall rate in millimeters per hour, \( d \) is reading from the image visualization and \( a \) and \( b \) are parameters, used from the similar study [54]. The rainfall intensity at certain segment \( s_i \) is approximated by the average of rainfall intensities in the area of that segment (see the right side of Fig. 4.4).

Since the rainfall data is irregularly reported (especially during the heavy raining intervals) we use Optical Character Recognition (OCR) to extract the time compo-
Figure 4.3: Locations of rainfall stations in Singapore (upper left). Rain map is continuously downloaded from NEA’s website (bottom) [110]. Information about the rainfall is updated every 5 – 15 minutes on average. Corresponding precipitation scale is also provided on NEA’s website (upper right corner) [110].

ments from each of the collected images (see the top right corner of Fig. 4.4). OCR is the process of converting text images into text in ASCII format. As a preprocessing step we binarize and scale the text images as suggested by other authors [111]. Then we apply the OCR and take in further analysis only these images that fulfill the following requirement: $0 < (t_i - t_i^*) < \sigma$, where $t_i$ is the time when the image appeared in our database (or time instant when the picture is collected from the internet). $t_i^*$ is the corresponding output from the OCR and $\sigma$ is an empirically determined parameter (15 minutes).

The occasional irregularities in reporting rainfall maps lead to the unavoidable miss-
ing values in weather data sets. If the number of consecutive time instants with no rainfall data is less than 3 (i.e., there is no data for previous either 5 or 10 minutes) then we impute the missing data by means of straightforward interpolation that relies on the available rainfall information for the neighboring time instants. To this end, we arrange data in two matrices, speed matrix $A$ and rainfall matrix $B$.

4.3 Data Analysis

In this study we aim to investigate the impact of the rainfall on the level of congestion and explore whether the information about the rainfall can be used to improve the prediction performance of data-driven models.

4.3.1 Impact of the Rainfall on Level of Congestion

We start by accessing the average level of congestion in the network for rainy and dry weather conditions. We first divide the traffic and weather data sets into Weekday and Weekend subsets ($t_k \in \{\text{Weekend}, \text{Weekday}\}$). Then we divide each group of data into 288 mutually exclusive subsets ($t_s \in \{1, 2, \ldots, 288\}$) in such a way that
all data within the same subset is recorded at identical 5-min interval, during the observational period of 7.5 months. Our underlying assumptions rely on a fact that traffic exhibits different patterns during the day and on different days; and, thereby, the impact of the rainfall is expected to be also different. For each $t^k_s$ subset we compute the average network congestion level during the inclement weather as a mean value of the average congestion level of the subnetwork $C_{t^k_s} \subseteq \{a_1, a_2, \ldots, a_n\}$ where the rainfall intensity $b_{i,j} (i \in t^k_s)$ is at least once above the threshold $\ell$. We write that as follows:

$$V_{t^k_s}^\ell = \frac{\sum_{j=1}^{c_{t^k_s}} \left( \sum_{i=1}^{r_{t^k_s}(j)} \frac{a_{i,j}}{r_{t^k_s}(j)} \right)}{c_{t^k_s}}, \quad \forall i \in t^k_s \text{ such that } b_{i,j} > \ell, \quad (4.2)$$

where the $c_{t^k_s}$ is the size of subnetwork $C_{t^k_s}$ that corresponds to certain $t^k_s$ and threshold $\ell$. The variable $r_{t^k_s}(j)$ shows how many times, for particular $t^k_s$, the the rainfall intensity (for link $j$) is above the threshold $\ell$.

Similarly, we compute the network congestion level for dry weather $V_{t^k_s}^B$ as the average level of congestion of the identical subset of the links $C_{t^k_s}$ if there is no rain ($b_{i,j} = 0, \forall i \in t^k_s$). We write that as follows:

$$V_{t^k_s}^B = \frac{\sum_{j=1}^{c_{t^k_s}} \left( \sum_{i=1}^{r_{t^k_s}(j)} \frac{a_{i,j}}{r_{t^k_s}(j)} \right)}{c_{t^k_s}}, \quad \forall i \in t^k_s \text{ such that } b_{i,j} = 0, \quad (4.3)$$

where $r_{t^k_s}(j)$ is the total number of instances within the $t^k_s$ when the rainfall intensity (for certain link $j$) is 0.

4.3.2 Impact of the Rainfall on Prediction Performance

In the following section we explain how we incorporate the information about the rainfall, as an exogenous variable, in the prediction of congestion level at a certain segment. Let say that we want to predict the future level of congestion $a_{j+kT,i}$ at time $t_j + kT$ and for link $s_i$ ($i \in \{1, 2, \ldots, n\}$) where the $t_j$ is the current time, $k$
refers to prediction horizon and $T$ is sampling interval. The future level of congestion $a_{j+kT,i}$, that we aim to predict, depends on the time of the day ($t_j$), current/past congestion levels of that segment ($a_{c,i}$, where $c \in \{j, j-1, \ldots, j-k\}$) and weather conditions (in terms of rainfall intensity $b_{j,i}$). We write that as follows:

$$a_{j+kT,i} = f_k([d(t_j), h(t_j), a_{j,i}, a_{j-1,i}, \ldots a_{j-k,i}, b_{j,i}]) \quad (4.4)$$

where $d(t_j)$ and $h(t_j)$ are the day and hour, respectively, of the particular $t_j$. Please note that in our setup, the rainfall has a near-immediate impact on the traffic, which is alignment with the methodology of relevant studies [54]. The function $f_k$ can be approximated by means of any state-of-the-art prediction algorithm. In this study, we apply support vectors machine (SVM) for classifications and ensemble methods which we briefly review in the following.

### 4.3.2.1 Support Vector Machine

Support vector machine (SVM) is a data-driven prediction algorithm that is often applied for traffic applications [112, 11, 90, 65]. In this study we use multiclass SVM since the provided traffic speed belongs to one of four categories (1 – 4). The multiclass SVM problem is frequently decomposed into multiple binary classification problems [113].

The binary classification aims to find a suitable hyperplane (i.e., decision boundary) to separate the classes. The equation of this hyperplane is:

$$w^T x + b = 0, \text{ where } w \in \mathbb{R}^n, b \in \mathbb{R}. \quad (4.5)$$

where $w$ is known as the weight vector.

SVM tries to search for such hyperplane which maximizes the margin between the two classes and minimizes the training error [114]. This leads to the following optimization problem:
\[
\min_w L(w) : \frac{1}{2} w^T w + C \sum_i \epsilon_i, \quad (4.6)
\]
\[
s.t. : y_i (w^T x_i + b) \geq 1 - \epsilon_i \text{ and } \epsilon_i \leq 0 \quad \forall \ x_i, \quad (4.7)
\]

where \( L(\beta) \) is a function that needs to be minimized and \( y_i \) represents each of the labels of the training examples. The variable \( C \) is a cost associated with the error of training data points and \( \epsilon_i \) are slack variables.

To achieve better performance linear classifiers are frequently replaced with the non-linear. In this case all \( x_i \) are replaced with \( \phi(x_i) \), where \( \phi \) provides the higher-dimensional mapping that leads to standard SVM formulation:

\[
\min_w L(w) : \frac{1}{2} w^T w + C \sum_i \epsilon_i, \quad (4.8)
\]
\[
s.t. : y_i (w^T \phi(x_i) + b) \geq 1 - \epsilon_i \text{ and } \epsilon_i \leq 0 \quad \forall \ x_i. \quad (4.9)
\]

This is a well-known problem of Lagrangian optimization that can be solved using Lagrange multipliers to obtain the weight vector \( w \) and the optimal hyperplane [115]. For more detailed information about SVM, please refer to [115].

### 4.3.2.2 Ensemble Methods

Ensemble methods are data-driven algorithms that have been widely used for applications such as prediction and face detection [116, 117, 118, 119]. These techniques heavily rely on underlying assumption that collection of weak classifiers can be a better predictor of a correct answer than the single classifier. The classifier (or learner) is often observed as an hypothesis about the true function \( f \) where \( y_i = f(x_i) \).

In this study, we applied adaptive boosting (AdaBoost) and Bootstrap aggregating (Bagging) since they are frequently used to solve multiclass problems.
In the case of Adaptive boosting, we define the predictor \((H)\) as a weighted sum of a set of the weak classifiers\((h)\). We write that as:

\[
H(x) = \alpha^1 h^1(x) + \alpha^2 h^2(x) + \ldots \alpha^n h^n(x),
\]

where \(\{\alpha^i\}^n_i\) is the importance of the \(i\)-th weak learner \(h^i\) and \(x (x \in \{x_1, x_2, \ldots, x_N\})\) is a training set comprising of \(N\) sampling points. At each time step \(i\) we pick a weak classifier \(h^i\) that minimizes the error rate \(\varepsilon_i\), given as:

\[
\varepsilon_i = \sum_{j: h^i(x_j) \neq y_j} w^i_j,
\]

where \(w^i_j\) is a weight of the training point \(x_j\) at time step \(i\). We calculate \(w^i_j\) as follows:

\[
w^i_j = \frac{w^{i-1}_j}{z} e^{-\alpha^{i-1} h^{i-1}(x_j)y(x_j)}, \text{ where }
\]

\[
w^1_j = \frac{1}{N}, \text{ and, }
\]

\[
\alpha^{i-1} = \frac{1}{2} \ln \frac{1 - e^{i-1}}{e^{i-1}}.
\]

\(h^{i-1}(x_j)y(x_j)\) is 1 for correctly assigned points and -1 otherwise. The variable \(z\) is normalization factor that guarantees that \(\sum_{j=1}^{N} w^i_j = 1, \forall i \in \{1, \ldots, n\}\). It is noteworthy that the weights of misclassified training points are enhanced so that the next (weak) learner is forced to focus on these data points (see Eq 4.12).

In the case of Bagging, we independently learn \(n\) base (or “weak”) classifiers, each of them using \(N' (N' < N)\) randomly sampled data points. Then, for each example in the test set, we apply all classifiers and select the final output of the model by majority vote technique. The Bagging ensemble method is based on assumption that majority of weak learners are correct most of the time and that they make mistakes mostly at different places of data set [120, 119].
4.3.2.3 Experimental Setup

In this study we predict a level of congestion at a certain link for a 5-minute prediction horizon \((k = 1\) in Eq. 4.4). For training and evaluation of prediction algorithms we only consider these data points when the corresponding rainfall intensity \((b_{j,i}\) in Eq. 4.4) is above the certain threshold \((\ell)\). We run the prediction models in the cases of any and heavy rains, with the rainfall thresholds of \(\ell = 0\) and \(\ell = 2.5\), respectively [58]. Intuitively, an increase in \(\ell\) would lead to the reduction in training and testing data sets that correspond to inclement weather conditions \((b_{i,j} > \ell)\); while the data set for dry weather conditions \((b_{i,j} = 0)\) remains unchanged. Please note that in our analysis the sum of training and testing data points, for any link in the network, is greater than 1000, for \(\ell = 0\) and 800 in the case of heavy rain \((\ell = 2.5)\). The prediction performance is evaluated using the 5-fold cross validation. As a benchmark method, we run the predictions without considering the information about the rainfall \((b_{j,i})\) as input feature in (4.4). This formulation allows us to analyze the impact of rainfall information on the prediction accuracy of data-driven prediction models.

The reported accuracy represents the ratio of accurately predicted points for a certain link and the total number of test points for that link. We analyze the prediction performance for different rainfall thresholds, periods of the week and times of the day in Section 4.4.2.

4.4 Results

In the following we discuss the effect of the rainfall on a level of congestion and analyze the benefits of using the information about the rainfall in short-term traffic prediction.
Figure 4.5: Average speed band (level of congestion) in the network during rainy and non-rainy time instants for weekday (top) and weekend (bottom).
4.4.1 Impact of the Rainfall on the Level of Congestion

Fig. 4.5 shows the average speed during rainy and corresponding non-rainy time instants. Reduction in average speed (across the network), influenced by the rainfall, can be seen even in low-resolution traffic data (see Fig. 4.5). The reduction is higher for the periods with congested traffic; which is in alignment with the results from relevant studies. Fig. 4.5 shows two distinctive peak periods during a typical weekday and non-clear peak during the weekend (see blue lines in Fig. 4.5a and Fig. 4.5b). The graph of average network speed during weekdays is smoother than for weekends (see blues and green lines in Fig. 4.5a and Fig. 4.5b) as a consequence of approximately 2.5 larger data set. Please note that surprising late weekend peak in Fig. 4.5b might be due the fact that we are dealing with the vibrant traffic network (such as Singapore) when significant number of users either coming back from a shopping or going out.

Let us introduce three performance measures that we use to quantify the impact of rainfall on low resolution traffic data. First, we define the sum of the speed band reduction (SSbR) over the period as follows:

\[ SSbR = \sum_{s=t_s}^{t_s+n-1} \left( V^f_{t^s_k} - V^B_{t^s_k} \right), \]  

(4.15)

where \( t_s \) refers to the starting time and \( n \) is the number of 5-min intervals within the considered period. Since observational periods might have different length we introduce the mean speed band reduction (MSbR) over the period as follows:

\[ MSbR = \frac{SSbR}{n} = \frac{1}{n} \sum_{s=t_s}^{t_s+n-1} \left( V^f_{t^s_k} - V^B_{t^s_k} \right). \]  

(4.16)

Finally, we define relative speed band reduction (RSbR) over the period of time as follows:

\[ RSbR = \frac{1}{n} \sum_{s=t_s}^{t_s+n-1} \left( \frac{V^f_{t^s_k} - V^B_{t^s_k}}{V^B_{t^s_k}} \right). \]  

(4.17)
Table 4.1 shows the numerical values of the introduced performance measure for a different types of the roads and rainfall intensities. The roads in the network are divided into highway and non-highway (“other”) segments while the rainfall is categorized as either light ($\ell \leq 2.5 \text{ mm/h}$) or heavy ($\ell > 2.5 \text{ mm/h}$) [58]. Naturally, the heavy rainfall has a greater effect on traffic data than light rainfall. As it can be seen from Table 4.1 the reduction in traveling speed is greater for arterial streets than highways. One of the reasons for this is the fact that we are dealing with the low-resolution speed data where the lower speed bands (e.g., SB1-3) are more prone to small changes in traffic conditions than the higher speed bands (e.g., SB4) which are frequently observed on highways. Interestingly, Table 4.1 shows that overall reductions in speed bands during the weekends are greater from these during the weekdays for both light and heavy rainfall. One of the reasons can be clearly seen in Table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>Weekday</th>
<th>Weekday</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highway</td>
<td>Others</td>
</tr>
<tr>
<td>Light Rainfall</td>
<td>SSbR</td>
<td>-22.874</td>
</tr>
<tr>
<td></td>
<td>MSbR</td>
<td>-0.079</td>
</tr>
<tr>
<td></td>
<td>RSbR</td>
<td>-0.021</td>
</tr>
<tr>
<td>Heavy Rainfall</td>
<td>SSbR</td>
<td>-29.755</td>
</tr>
<tr>
<td></td>
<td>MSbR</td>
<td>-0.103</td>
</tr>
<tr>
<td></td>
<td>RSbR</td>
<td>-0.027</td>
</tr>
</tbody>
</table>

Table 4.1: Impact of the rainfall intensity on average congestion level in the network for different types of the roads, periods of the week and rainfall thresholds.

Table 4.2 shows the numerical values of the introduced performance measure for the different periods of the day. These periods are selected according to the workday...
traffic patterns and classified into four distinctive groups as follows: (i) Morning peak period, (ii) daily non-peak period (iii) evening peak period and (iv) night traffic (see vertical red lines in Fig. 4.5). Table 4.2 shows that degradation in speed band is higher for time periods with the higher demand. In the case of weekday, as expected, these periods are evening and morning peak hours. Regarding weekends, the highest impact of rainfall is during evening and night hours when an increasing number of recreational and shopping related trips is realized in a vibrant city such as Singapore.

### 4.4.2 Impact of the Rainfall on Prediction Performance

We evaluate prediction algorithms for a small portion of the network (15%) to select the best subset of input parameters, which is then used for network-wise prediction.
The parameter selection is done for the proposed and benchmark prediction methods. Fig. 4.6 shows the prediction accuracy of the SVM method for different combinations of $C$ (cost) and $\gamma$ [121]. We set $C = 32$ and $\gamma = 0.5$ for network-wise prediction. Similarly, we set the number of trees to be 80 and default values for other parameters in the contexts of Adaboost and Bagging ensemble methods for both proposed (with rainfall data) and benchmark methods [122].

Table 4.3 shows that incorporation of rainfall information leads to marginal degradation in network-wise prediction performance for each of the evaluated algorithms, regardless the rainfall intensity. One reason might be the fact that prediction methods assign significant importance to the present and near past traffic values, especially for (evaluated) lower prediction horizons. Table 4.3 also shows the size of the subnetworks (in terms of number of links) where the incorporation of rainfall data increases / reduces the prediction performance of traffic data. Fig. 4.7 shows the magnitude of these changes in performance for different prediction methods and rainfall intensities. The increase of link-wise prediction performance is marginal for all evaluated algorithms and rainfall thresholds. Similar conclusions apply for these cases where the incorporation of rainfall data leads to degradation in prediction performance.

In our analysis we observed these cases where the rainfall information is useful for certain periods of the day while for others it either degrades or does not influence the prediction performance. Fig. 4.8 shows that incorporation of the rainfall data might increase prediction performance of Bagging ensemble method up to 10% if only the specific period of the day (e.g., AM peak) is considered. One reason might be the fact that impact of the rainfall on traffic significantly varies across the day. Another reason may lie in the reduced data set, which is significantly smaller than the corresponding data sets considered in Fig. 4.7e and Fig. 4.7f.

Fig. 4.7 and 4.8 show that impact of the rainfall on prediction performance may vary across the day and among different segments in the network. Although the incorporation of rainfall data, in general, marginally reduces the network-wise prediction;
<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>Rainfall Threshold</th>
<th>Integration of Rainfall data</th>
<th>Accuracy (%)</th>
<th># of links where the rainfall data is useful / not useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>$\ell = 0$</td>
<td>Yes</td>
<td>86.11%</td>
<td>1345</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>86.14%</td>
<td>1694</td>
</tr>
<tr>
<td></td>
<td>$\ell = 8$</td>
<td>Yes</td>
<td>86.17%</td>
<td>1310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>86.18%</td>
<td>1615</td>
</tr>
<tr>
<td>Adaptive Boosting</td>
<td>$\ell = 0$</td>
<td>Yes</td>
<td>85.43%</td>
<td>310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>85.44%</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>$\ell = 8$</td>
<td>Yes</td>
<td>85.45%</td>
<td>319</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>85.47%</td>
<td>394</td>
</tr>
<tr>
<td>Bag Boosting</td>
<td>$\ell = 0$</td>
<td>Yes</td>
<td>84.26%</td>
<td>462</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>85.62%</td>
<td>3557</td>
</tr>
<tr>
<td></td>
<td>$\ell = 8$</td>
<td>Yes</td>
<td>84.32%</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>85.67%</td>
<td>3567</td>
</tr>
</tbody>
</table>

Table 4.3: Prediction performance of SVM and boosting methods. Two sets of models, with and without information about the rainfall, have been evaluated for different rainfall thresholds. Bolded value (see last column) shows the size of the subnetwork where the information about the rainfall improves the forecasting performance of traffic data. Similarly, the values in italic show the number of links where the additional information of about the rainfall degrades the prediction accuracy of traffic data.

This additional information proved to be useful for certain links in the network, especially during the particular periods of the day. This fact should be carefully considered in relevant studies that investigate the impact of the weather parameters on prediction performance of traffic data.

### 4.5 Discussion

In Section 4 we investigated the impact of the rainfall on traffic conditions by analyzing a sizable rainfall data set. We were able to (i) reveal reduction in traffic speed...
due to increase in rainfall intensity; and (ii) identify certain segments in the network where the information about the rainfall improves the prediction performance of forecasting algorithms. However, the low-resolution of traffic data significantly limits our investigation. In order to reach more firm conclusions, the high-resolution traffic information is needed.
Figure 4.6: SVM grid search method for parameter selection. The \( \{C, \gamma\} \) tuple that leads to the highest prediction performance of SVM method is \( \{32, 0.5\} \).
Figure 4.7: Histogram of the links where rainfall data is useful (blue) and not useful (green) for AdaBoost prediction method. Results are presented for different prediction methods (rows) and rainfall intensities (columns).
Figure 4.8: Histogram of the links where rainfall data is useful (blue) and not useful (green) for bagging prediction method. Results are presented for AM peak (top), PM peak (middle) and night hours (bottom), during weekdays (left) and weekends (right).

NANYANG TECHNOLOGICAL UNIVERSITY  SINGAPORE
Chapter 5

Conclusion and future directions

5.1 Conclusion

In this thesis, we studied the problems of estimation and prediction of large traffic networks. We addressed these problems in the contexts of scalability and system implementation. We also put significant attention to an external factor such as inclement weather since it might severely affect the traffic operations in the network. So far, the management centers collect traffic data from all the sensors and use it to develop the explicit prediction models for every segment in the network. Numerous studies have focused on developing highly accurate prediction methods without explicit consideration of scalability and deployment of the proposed solutions in practical traffic networks. The solutions proposed by other studies are often implemented in a traditional way where all operations are performed on the central server and only final outputs are sent to all users. Moreover, the impact of the external factors is often investigated for small test networks and short periods of time. By contrast, we proposed generic algorithms that can model traffic patterns in heterogenous city-scale networks. The proposed method significantly improves the scalability of traffic operations by explicitly observing only a small portion of the data. Also, the proposed solution can be embedded into different architectures in a straightforward manner. We also proposed decentralized and hybrid architectures.
and reveal their great potential for traffic applications. In addition, we performed a comprehensive analysis of the effect of the rainfall on traffic conditions and prediction performance in the context of large traffic networks and extensive data sets. Our main contributions are as follows:

- In chapter 2, we proposed low-dimensional methods for traffic applications such as compression, estimation and prediction. We first obtained a suitable low-dimensional representation of large-scale road networks. We used this low-dimensional representation to compress the traffic data in an intuitive yet accurate way. We also utilized low-rank representation of traffic data to infer meaningful temporal patterns in urban networks. Finally, we used a low-dimensional representation to pave the way for real-time traffic applications such as compressed sensing (estimation) and compressed predictions. We assessed the current and future traffic conditions in the entire network using only information from carefully selected locations in that network. In this way, the proposed method significantly improves the scalability of traffic applications at the expense of slightly increased prediction error. We decomposed the prediction error into several components and investigated the relationship between them. We also suggested strategies to reduce the overall and component-wise errors. We analyzed the performance of the proposed method for various road categories such as expressways, arterial roads, access roads and slip roads.

- In chapter 3, we evaluated different architectures for the implementation of the proposed low-dimensional method on smartphones. We developed a traffic app and used it to assess the computation times of centralized and decentralized modes of operations for applications of traffic speed estimation and prediction, and travel time prediction. We estimated travel-time in the network using the predicted speed values, obtained by the proposed compressed prediction method. We considered different modeling platforms, smartphone devices and test networks. We proposed a decentralized architecture to significantly reduce the overhead of the already busy communication networks. Moreover,
our numerical results revealed great potential of the non-centralized modes of operations for cooperative traffic applications.

- In chapter 4, we studied the impact of the inclement weather on traffic conditions and performance of state-of-the-art prediction algorithms. We leveraged our analysis on an extensive amount of rainfall and traffic data, collected on $5 - \text{min}$ intervals. We run our investigation for different road categories and rainfall intensities. Furthermore, we also assessed the impact of the inclement weather on traffic conditions and prediction performance for data collected during weekdays and weekends. We found that the impact of the rainfall varies across the day and it might be significant even for weekends. We also revealed that incorporation of rainfall data might be useful for traffic prediction of some segments, while it marginally reduced the network-wise prediction performance. Finally, we showed that impact of the rain on prediction performance varies across the day and among evaluated prediction methods.

### 5.2 Future Directions

In this dissertation, we evaluated the impact of the rainfall intensity on traffic conditions in large and diverse networks. However, traffic conditions might be also affected by other external factors such as incidents (see Fig. 5.1). Different incidents can disrupt traffic in different ways. For instance, minor accidents (e.g., vehicle breakdown) on a lower-priority road during off peak and dry weather might not affect traffic drastically. On the other hand, an accident on the highway during peak hours and inclement weather might have a severe impact on traffic. Furthermore, the question that might arise from this example concerns the contribution of the inclement weather to accident occurrence and duration (see Fig. 5.1). In summary, assessing the influence of inclement weather on severity and occurrence of traffic accidents followed by an estimation of the impact of these incidents on traffic can prove to be highly useful for management systems.
Figure 5.1: Concept diagram showing the relationships between weather, accidents and traffic. In this thesis we study the impact of rainfall on traffic conditions (see solid arrow). For future work, we aim to explore the impact of the inclement weather on traffic incidents as well as impact of these incidents on traffic performance (see dashed arrows).

Figure 5.2: Smartphones offer a significant computation power which can be utilized to perform certain operations, allocated from the server. In particular, routing might be (partially) performed on smartphones.

Another interesting direction would be to extend the role of smartphones in traffic applications (see Fig. 5.2). We revealed that smartphones can play the significant role in estimating the current and future traffic conditions and assessing the
travel time for a given route in the network. We also showed that feasibility of non-centralized traffic applications, implemented on smartphones, highly depends on the size of the underlying network. However, with a continuous increase in their computation power, smartphones would most likely overcome these limitations. Moreover, this increase in computation power would reveal new possibilities in the frameworks such as smart-car and cooperative Intelligent Transportation Systems. For instance, an interesting option would be to (partially) perform routing on smartphones (see Fig. 5.2). Another useful application would be to use the computation power of smartphones to process and visualize traffic information, explicitly obtained from other ad hoc probes in the network. This “server-free” mode of operations can prove helpful in reducing the network bandwidth and workload of traffic management centers. Furthermore, this scenario can prove useful for novel device-to-device applications in the framework of traffic operations.
Bibliography


