Data-driven Solutions for Addressing Two Pressing Urban Sustainability Challenges: Air Pollution Reduction and Traffic Management

by

Shiva Radhakrishnan Iyer

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January, 2022

Professor Lakshminarayanan Subramanian

Acknowledgements

The work produced in this dissertation would not have been possible without the constant guidance, mentorship, support and care from my advisor, Lakshmi Subramanian. His constant push for excellence in research and publishing has instilled me the confidence of aiming for the highest quality of work and the most prestigious of venues for publishing my work. I am thankful to my committee members – Aurojit Panda, Anirudh Sivaraman, Yasir Zaki and Edwin Gerber – for their many useful and lasting comments on my thesis and research. I thank NYU GSAS, Lakshmi and Edwin for providing financial support through the course of the journey.

I am grateful to all my collaborators and co-authors over the years – Ananth Balashankar, for his efforts in the air pollution work; Ankit Bhardwaj, for his efforts in a related work on air pollution detection from images; Sriram Ramesh, for his efforts in the traffic jams work; William Aeberhard, for his assistance in the air pollution project, his many mathematical ideas on spatiotemporal modeling, and implementing the STHM; Sujoy Bhattacharyya, Giuditta Rusconi and Sonya Suter, for their tireless efforts in managing the air pollution project in Delhi over the course of many months, leading talks with Kaiterra, and getting our pollution network up and running on the ground in Delhi; Muhammad Khan, Luis Diez, Yasir Zaki and Talal Ahmad for their efforts in our mmWave congestion control project; Jerome White, for his help with the analysis of traffic jams in New York City; Ulzee An, for his assistance in the traffic forecasting problem and implementing early versions of the MPRNN model; Soumie Kumar and other students in her class, for their assistance in development of our mobile sensing testbed, called MUSIC, for mo-

bile monitoring of air quality; Kate Boxer for her help with traffic data analysis and the MUSIC application; and many others. I am indebted to many of these people, without who the works presented here would not have seen the light of the day. I am also grateful for the lasting personal and professional connections I have made as a result of these collaborations.

I am pleased to have had the fortune to work with Rohini Pande and Anant Sudarshan, two of my most prominent collaborators over the years on the Delhi pollution monitoring project. It started with an internship opportunity at EPoD India at IFMR Lead in Delhi, where I spent the summer of 2016, after my first year, as a data scientist intern. Over a period of 5 years, I have had the pleasure of gaining immense knowledge and mentorship from them while working on the pollution project. As someone who is particularly passionate about making academic research impactful to the general community, I enjoyed working amid a diverse group of researchers and practitioners at EPoD, from who I have learnt a lot of practical knowledge. I am also thankful to many of the interns I have worked with for shorter periods of time, including Sameeksha Jain, Prateek Behera and others. I am also hugely thankful to members of the EPoD staff, including Tamara, Anuradha and Niharika, for always making me feel welcome at the EPoD office in Delhi. I had spent a total of 8-9 months working out of their office over three visits, and I have enjoyed every moment of working there.

I am thankful to Anandan, Rahul Panicker, Nikhil Jagtiani and the many other wonderfully talented and passionate people at Wadhwani AI, where I had the fortune of spending the summer of 2018 as an intern, exploring one of my passions, which was AI for Social Good. Through my time at Wadhwani AI, I gained firsthand exposure to the frontline healthcare system in India, its complexity and the embedded challenges in it. And in the summer of 2019, I had the opportunity to be an intern at Nokia Bell Labs, an amazing place for research, where I was fortunate to have got to meet many smart people, including two Nobel Laureates! I am thankful to my mentors there, Jinfeng Du, Dmitry Chizhik and Reinaldo Valenzuela, all the staff and my fellow interns, for an enjoyable summer internship. I particularly thank Sameer Sharma, for providing the best

accommodation an intern could have ever hoped for. The home was just a stone throw away, and Sameer's mother prepared the most delicious meals. I thank Sameer also for the good company and the help he provided on multiple occasions in making my overall internship experience comfortable.

Over here at NYU, I have had the fortune to know and interact with many bright students – Sebastian Angel, Sunandan Chakraborty, Cheng Tan, Siddharth Krishna, Abhinav Tamaskar, Varun Chandrasekaran, Ashwin Venkataraman, Talal Ahmad, Fatima Zarinni, Yan Schvartzshnaider, Ananth Balashankar, Mukund Sudarshan, Srikar Varadaraj, Nisarg Patel, Kevin Choi, Ankit Bhardwaj and Sriram Ramesh. I particularly thank my labmates Ashwin, Talal and Varun for the close friendships and good moments we have shared. I am thankful to all the professors who have taught me in various courses that I have taken – Prof. Mike Walfish, Prof. Victor Shoup, Prof. Subhash Khot, Prof. Michael Overton, Prof. Yann LeCun and Prof. Ralph Grishman. I am thankful to Prof. Jinyang Li and the other faculty and students in the NYU Systems Group, for providing opportunities to present my works on numerous occasions and giving valuable feedback at each time. The weekly meetings, designed primarily to help students hone their presentation skills, have helped me immensely. Through exposure to the works of many other students on a regular basis and the critical analyses and discussions that followed each talk, I have benefited as a researcher, while also making connections with many other students on the floor.

I am very grateful to the NYU CS PhD administration staff and faculty, particularly Santiago Oded, Rosemary Amico, Prof. Oded Regev and Prof. Thomas Wies, for their constant support and encouragement. I am particularly thankful to Leslie Cerve, our floor administrator, an able, loving and caring lady, who went above and beyond to make our lives easier on the floor. She assisted with every little issue, several times outside administrative matters too. She would pay attention to the tiniest of details such as restocking the tea bags in the pantry with different flavors each time. Most of all, to us young students, she was like a mother and a friend, always

brimming with positivity and optimism, being receptive to our needs, willing to lend an ear and sharing her experiences. I am also grateful to Hong Tam for his assistance on many occasions with administrative matters.

My acknowledgements would not be complete without mentioning my flatmates, Rahul, Digvijay and Nisarg, and other friends, Aravind, Thy and others. Rahul and Digvijay were the best flatmates I could have asked for, and I am glad to have found a friend in Aravind, who gave me a lot of life advice at different times. Having these wonderful people at home and outside work made life so much easier and enjoyable than it could have been. I will always remember the almost infinite number of conversations with Rahul about everything under the sun (that somehow invariably ended up one of politics, religion or languages), the many games of chess and badminton, the many fun cooking sessions at home and the occasional hikes and dinners. I am also thankful to Navatman, particularly Sahi and Samyukta, among several others with who I have made friends over the course of several months of our shooting of the Mahabharata film, and the numerous friends I have made at the Hindu temple at Flushing NY, who have all collectively helped me rediscover a passion for Indian classical arts.

I also thank my mentors and advisors at IIT Madras and TCS Innovation Labs–Ravindran Balaraman, Venkatesh Sarangan, Arun Vasan, Anand Sivasubramaniam–for their support at the time of applying for the program. Finally, but not the least, I thank my family–my parents, my grandmother and my wonderful brother Rahul–for their enduring love, encouragement and care, my friend and soon-to-be partner Mathangi for her patience and support in my final year, and the grace of the Almighty.

Abstract

Data Science and AI-driven solutions are abound today for a large variety of practical applications. With a continuing focus on urban development and sustainability, in this thesis, I present our attempts in addressing two prominent urban challenges – urban air pollution control and road traffic congestion management. For both these applications, we have developed novel methods, such as the message-passing recurrent neural network, for predictive analytics and inference in collaboration with economists, public policy experts and ICTD researchers.

The city of Delhi has 32 air quality monitors over an area of about 900 sq km, but we do not have information on fine-grained variations in air quality in the city in order to reason about citizen exposure and identify hotspots. We have installed 28 low-cost sensors, many of them concentrated in the south Delhi region. We have identified many hotspots by studying spatio-temporal variations from the data, further motivating the need for fine-grained sensing. And ultimately, we designed a novel model combining geostatistics and deep learning that is able to make spatio-temporal pollution forecasts by the hour with an MAPE of about 10% across all locations.

Urban traffic management is another pressing challenge in an era where we observe increasing urbanization and industrialization. Simply building new lanes and larger roads is not enough – we need to go back to formula and understand how jams happen, and how we can effectively implement traffic control. In the first of our works, we show that road networks can experience traffic jams over prolonged periods, as high as 20 hours sometimes, due to sudden traffic bursts

over short time scales. We illustrate this using real data from two different cities – New York and Nairobi. We provide a formalism for understanding the phenomena of traffic collapse and sudden jams. In the second work, we devise a novel model called the message-passing neural network for modeling the propagation of congestion within a road network and forecasting congestion. The MPRNN achieves the lowest mean error of < 0.3 mph when predicting ahead in 10 minute intervals, for up to 3 road segments ahead (message passing across 3 hops). Finally, in the third work, we describe an algorithm for signal control in free-flow road networks, inspired from congestion control in computer networks. Our proposed method significantly enhances the operational capacity of free-flow road networks in the real world by several orders of magnitude (between 3× and 5×) and prevents congestion collapse.

Contents

List of Figures

List of Tables

Part I

Data Science for Countering Air Pollution

1 Introduction

Across the developing world, urbanization and economic growth have led to the emergence of heavily polluted towns that are at once the focus of local economic growth, and ground-zero for one of the most urgent public health challenges the world faces today [Alpert et al. 2012]. Over 4.5 billion people live in parts of the world where the average ambient concentration of fine particulate matter, the leading cause for poor air quality, is above 20 $\mu g/m^3$, twice the maximum level the World Health Organization (WHO) considers safe [WHO et al. 2006]. In 2016, the annual average level of airborne fine particulate matter ($PM_{2.5}$) in Delhi, the capital of India, was 142 $\mu g/m^3$, 14 times higher than the level deemed safe by the WHO and 4 times higher than the least stringent interim target of 35 $\mu g/m^3$ [WHO et al. 2006]. The issue is most severe in the winter season, with $PM_{2.5}$ concentration levels in ambient air soaring beyond 1000 $\mu g/m^3$ on some days. One of the most cited macro sources of air pollution in Delhi, the burning of crop residue in the farmlands of the neighboring states of Punjab and Haryana post the harvest season [Bikkina et al. 2019; Cusworth et al. 2018], contributes only 17-26% of the total pollution in the winter, and about 7-12% in the summer [Sharma and Dikshit 2016]. The remaining is attributed to a myriad of localized sources within the city, such as road dust, vehicular traffic, domestic emissions (from domestic activities like waste burning and cooking), construction and demolition activities, etc [Apte et al. 2017; Sharma and Dikshit 2016]. In Delhi, municipal solid waste (MSW) is burned across the city, especially during the winter months. Over 20% of Delhi's air pollution is estimated to come from waste burning and building construction, both of which are decentralized

and dispersed sources of pollution [Sharma and Dikshit 2016]. These activities create a large number of local spatial and temporal hotspots, many of which are unaccounted for when making policy. Local spatial peaks can cause exposure peaks potentially hurting a vulnerable population like those in urban slums out of proportion as compared to the average level. Even if air pollution levels did not vary over time, levels of human exposure may vary widely depending on lifestyle choices, localities and routes frequented [Pant et al. 2017]. Thus, identifying fine-grained hotspots that vary over both space and time will prove to be very valuable for targeted interventions.

Delhi has a network of air quality monitoring stations operated by three different public bodies – Central Pollution Control Board (CPCB), Delhi Pollution Control Committee (DPCC) and the Indian Meteorological Department (IMD) – that provides quality-controlled public data. The spatial monitoring resolution of this network, hereafter referred to as the "public network" or the "government network", despite being the densest such network in an Indian city, is still not fine enough to capture the more localized causes of high pollution that inform us about citizen exposure. Apart from the significant and documented local sources mentioned before, we find evidence for even smaller and regular localized spatial pollution hotspots (Figures 1.2) that contribute to the overall poor air quality and citizen exposure. In one experiment in 2018, we traversed repeatedly through the roads of two different neighborhoods in Delhi (area 4-5 km^2 each) over a period of four hours while carrying portable handheld air quality monitors. We were able to attribute some of these local hotspots to various local activities such as open food carts, open ironing shop (with the traditional coal iron press), incense burning, roadside construction and earth-moving activity. Some of these sources generated $PM_{2.5}$ concentration in the air upwards of 100 $\mu g/m^3$ while being present up to 10 feet away from those sources. Neither of these neighborhoods contained a government monitoring station in its vicinity, thus validating our hypothesis that localized but significant sources of pollution do not get captured by the existing public network of monitors.

The reason these observations are important is because localized sources of pollution affect

narrow sections of the population that frequent those locations. There are currently 32 of these public monitors spread over a ground area of about 858 km^2 i.e. one sensor for an area of approximately 26 km^2, which is a rather dense network for city-level air quality monitoring, but sparse for a reasoning about citizen exposure at the microscale. One reason for the lack of a denser network is equipment cost. Regulatory-grade monitors, such as the ones used in the public monitoring stations, are expensive high-end instruments—the cost of one fixed automated station can reach about 12 million Indian Rupees (200,000 USD) to set up with a 10% yearly operational and maintenance cost [Hindustan Times 2017; MOEF 2018]. We therefore augment this public network with our own denser network of low-cost noisier air quality sensors in a smaller region of the city, resulting in a total of 60 sensors (Figure 1.1).

In March 2018, we began the deployment of a network of 28 low-cost sensors, many of them concentrated in the south Delhi area, in collaboration with Kaiterra [1], a company that makes low-cost air quality monitors and air filters. The sensors use light-scattering technology to measure PM concentrations, every few minutes, which we averaged to the hour to smoothen the data and remove noise. The data continues to be reported until the present day, January 2022. However, for the purposes of our analyses and the spatiotemporal modeling, we chose the data over a two-year period from May 2018 to May 2020. The main reason for this is that since the onset of the COVID-19 pandemic in April 2020, regular maintenance and service of these sensors have been difficult, resulting in reduced data unavailability. These low-cost sensors both validated the air quality from existing public monitors and provided insights on the air quality at locations without any public monitors.

In our study, we find that hotspots may be short-lived, long-lived, spatially focused or spread out, and may be transient or recurring. By plotting data at various temporal resolutions, we notice these different patterns. By placing more air quality monitors on the ground, we observed other types of hotspots that are lesser known (Figure 1.3), such as transient hotspots, longer lived but

[1] https://kaiterra.com/

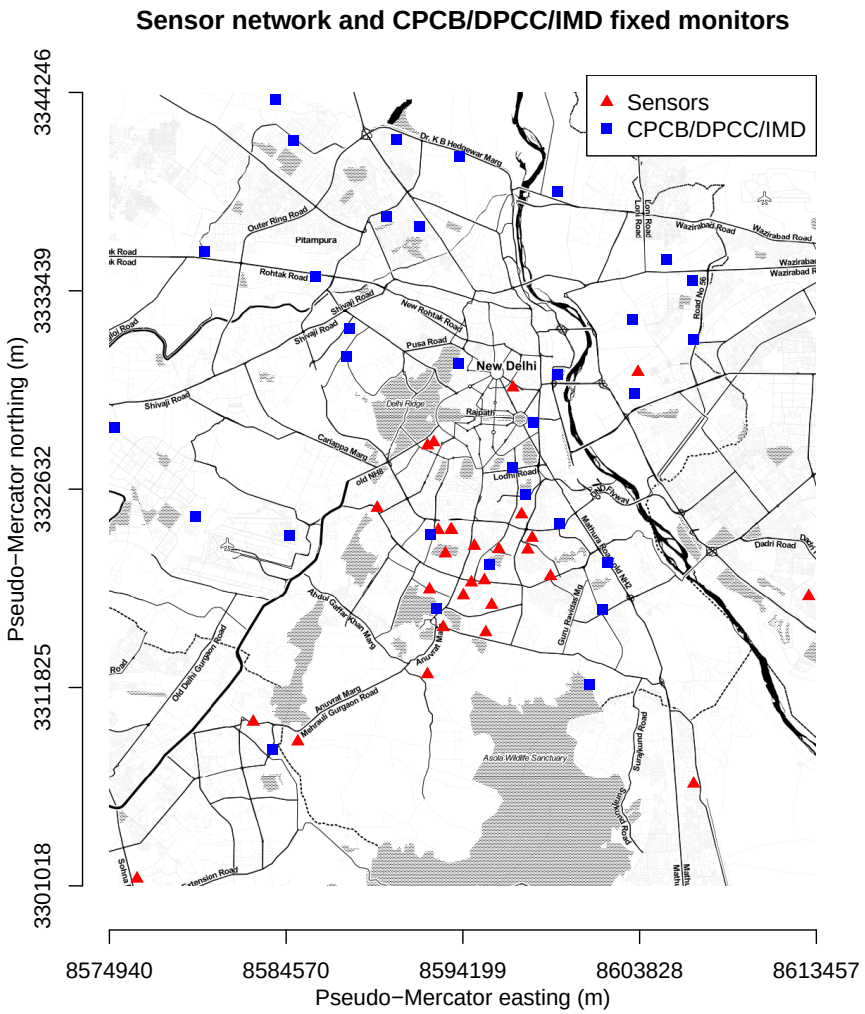
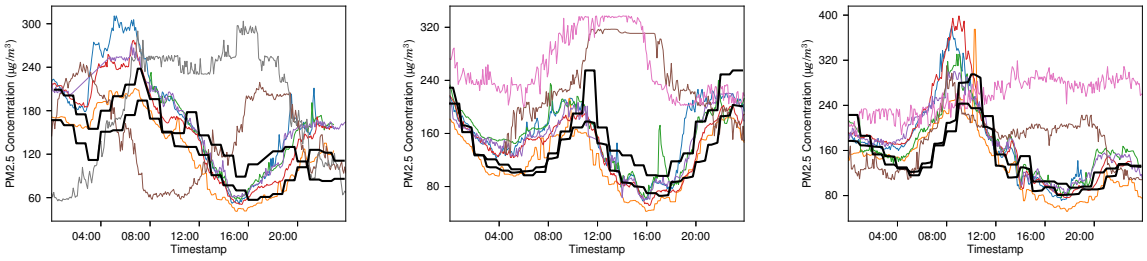


Figure 1.1: List of all the sensors for our study. The 32 public fixed monitoring stations by the government are shown in blue, and the 28 low-cost sensors that we have installed are shown in red. (The figure actually shows 61, because there is 1 additional public monitoring station at East Arjun Nagar, but there has been no PM data in the entire 30-month period of consideration, and therefore we exclude it from our analysis.



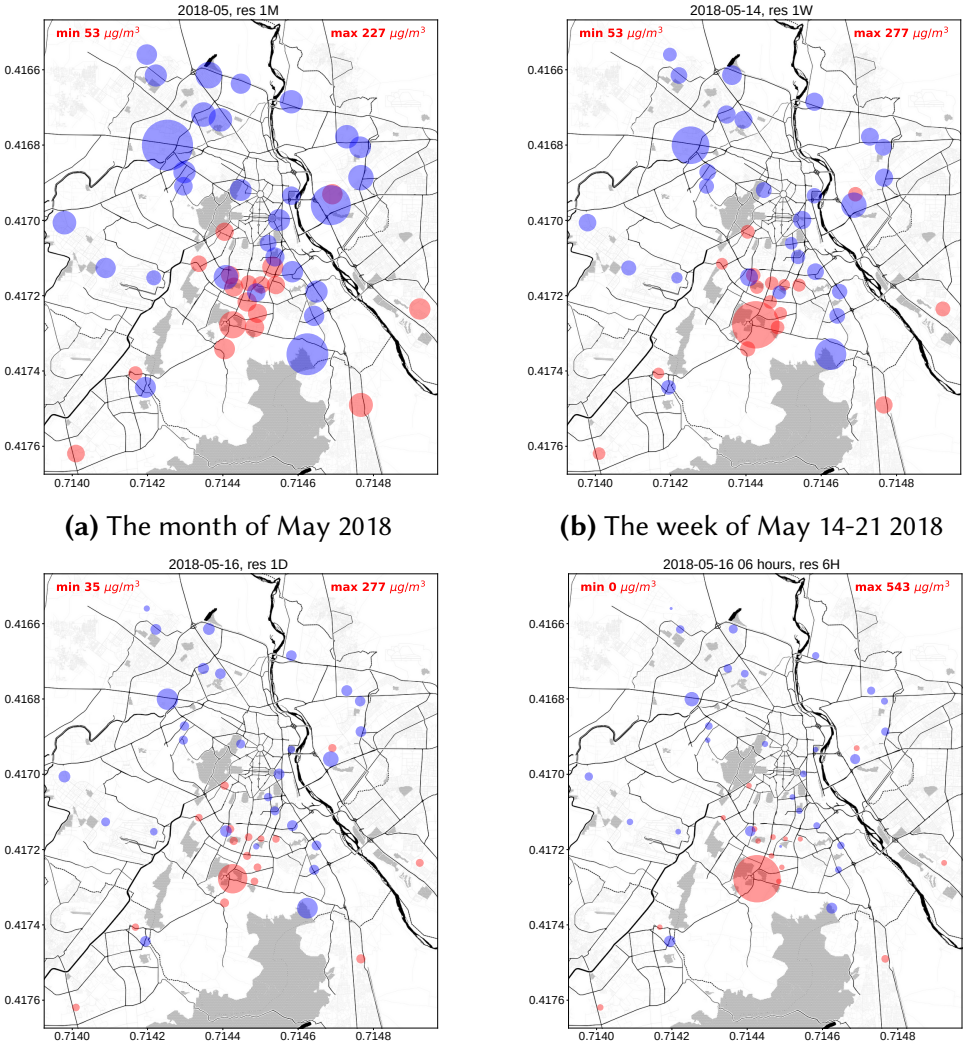
(a) Jan 30, 2018 **(b)** Feb 1, 2018 **(c)** Feb 2, 2018

Figure 1.2: The plots show a sample of readings in a small neighborhood containing about 7 of our low-cost sensors and 2 public stations for three different days. Sustained fine-grained spatial variations that are not captured by the public monitoring stations are captured by the denser low-cost sensor network. The bold dotted lines show the readings from the closest public monitors, and the other lines show the readings from the rest of the sensors.

more spatially focused and so on. The variation across space can be fairly large too, as evidenced by the heatmap showing the readings from all (low-cost and high-cost public) the monitors in our study.

Then, we describe a methodology to model and forecast urban air quality at a fine-grained level using our dense and noisy *low-cost sensor network*. Pollution forecasting in cities with dense populations can be critical for generating fine-grained policy recommendations and public health warnings [Shaddick et al. 2020; Rao et al. 2021; Geng et al. 2021]. The scale of accurate sensor based monitoring required to achieve this can come at a huge cost and thus inhibit building a dense fine-grained pollution sensing map. There are two main questions we seek to answer through this work – *i)* how can we use a network of low-cost and portable air quality monitors in order to build a fine-grained pollution heatmap in a city that provides accurate forecasting?, *ii)* does it help to augment existing monitoring networks by the local governments with low-cost air quality sensors? We develop a hybrid model that combines state-of-the-art approaches for spatio-temporal modeling such as a spatio-temporal hierarchical model (STHM) and a message-passing recurrent neural network (MPRNN) that models spatio-temporal interactions to predict an accurate air pollution field.

The deployment of low-cost particulate matter sensors to replace or augment reference grade



(a) The month of May 2018

(b) The week of May 14-21 2018

(c) The day of May 16, 2018

(d) The quarter 6am-Noon of May 16, 2018

Figure 1.3: Heatmaps that show PM concentrations (in $\mu g/m^3$) at various locations in space, at different temporal resolutions. More transient hotspots are observed at finer temporal resolutions such as 6H and 12H, but more longer-lived hotspots are observed at coarser resolutions such as 1D and 1W. The four maps show the maps at various "zoom" levels in time in the month of May 2018, thus identifying that pollution hotspots occur at varying levels of granularity.

pollution air quality monitoring systems has been studied extensively recently, and have addressed issues of calibration [Liu et al. 2019, 2020; Giordano et al. 2021], design [Tryner et al. 2021; Prakash et al. 2021], data selection [Bi et al. 2022] and personal exposure quantification [Zusman et al. 2020; Mahajan and Kumar 2020]. However, building a highly accurate large scale fine-grained pollution sensing and monitoring map that leverages the size of a pollution network has been largely unexplored. Specifically, modeling the behavior of noisy low-cost sensors in cities with high pollution and population density has not been studied previously, with recent state-of-the-art mapping approaches providing errors only in the range of 30-40% [Spyropoulos et al. 2021; Chu et al. 2020]. This high error lends the pollution sensing map unusable for policy making and air quality hazard detection. We build on prior work and model the pollution network in its entirety, with prediction models at each sensor location based on a recurrent neural network model dependent on sensor-reading messages sent from near-by sensor locations.

Our adaptive statistical approach can incorporate data from several noisy and low-cost sensors and provides an attractive and more viable alternative. We employ a hybrid data-cum-model driven approach, in which first we fit a spatio-temporal hierarchical model (STHM) to impute missing values of sensors (due to power and network outages) and create a "baseline" spatio-temporal field, then fit a cubic spline function that captures the daily and hourly mean trends, and then finally train a message-passing graph neural network that incorporate spatial priors and temporal trends from nearby sensors' data, to fit the residuals after the spline correction. We aim to forecast a given sensor's readings of the concentration of fine particulate matter ($PM_{2.5}$) measured in $\mu g/m^3$ using historical data of up to 8 hours from all the sensors in the network. We make this choice because the primary advantage of low-cost sensors lies in their ability to provide a large number of noisy measurements. By learning the variability of each of these noisy measurements through message passing neural networks which have the ability to model each sensor separately, we learn to not only separate the signal from the noise, but build an accurate sensing network of low-cost sensors that achieves 10% Root Mean Squared Error (RMSE) in forecasting

up to one hour in advance over a fine-grained spatio-temporal grid as compared to baseline modeling approaches that provide 30% RMSE. By using a sparse network of sensors, whose signals are shared through neural network embeddings, we learn to capture the information from nearby sources that might affect the readings of nearby sources (e.g. factory) and ignore the ones which are heavily localized (e.g. food cart). Such an accurate fine-grained pollution sensing map ($\leq$ 10% MAPE) is usable by policy makers in deciding which neighborhoods of the city need interventions to improve the air quality and population health. Estimating such models provides a way to efficiently use information from several monitors to make predictions over a fine-grained grid, with the ability to seamlessly and flexibly incorporate low-cost sensors in developing countries.

We model the spatio-temporal forecasting problem as a graph prediction problem, where we predict a value at every node at a certain time using as input the historical values from neighboring nodes. In our setting, each sensor location $v \in \mathcal{V}$ is a node in an undirected graph. Assuming that air pollutants diffuse uniformly in all directions and exert their influence throughout our region of interest, in this case the greater Delhi region, we make the graph complete, where an edge exists between every pair of nodes. The end goal is to train a model that predicts at any node, the pollution level, measured in terms of the concentration of fine particulate matter $PM_{2.5}$, at time t given one or more readings from neighboring locations prior to t. The first step is to interpolate the gaps in the data. We use a geostatistics model for this task, called the Spatio-temporal Hierarchical Model (STHM). Then we fit a cubic spline based on daily trends at each sensor location, and then finally train a Message-Passing Recurrent Neural Network (MPRNN) (§2.1.3) to predict residuals over the baseline. In order to account for the amount of influence based on the pairwise distances, we include the Euclidean distance between sensors as part of our feature embedding in our message-passing formulation. We test this model by predicting values at locations where sensors, and therefore ground truth information, are present, but the model is generalized enough to be used to predict at locations where there is no ground truth data available. If $y_{v,t}$ is the reading of the sensor at location v, at timestamp t, and $\hat{y}_{v,t}$ is our

corresponding prediction, the forecasting model aims to minimize the mean absolute percentage loss:

$$MAPE = \sum_{v} \sum_{t} \frac{|\hat{y}_{v,t} - y_{v,t}|}{y_{v,t}} \tag{1.1}$$

2 Building Spatio-temporal Pollution Maps

2.1 Methods

Our pollution forecasting model for estimating the $PM_{2.5}$ particulate matter concentration across space and time consists of three important steps. Given the variations in data availability across our pollution sensors, the first step of our method uses a standard Spatio-Temporal Hierarchical Model (STHM) to estimate the missing data. Our STHM model is a standard statistical modeling framework from geostatistics that combines multiple sources of information, accommodates missing values and computes predictions in both space and time. A detailed description of the STHM model is provided in the appendix (ref §A.1). Based on daily variation patterns observed at each of the pollution sensors, the second step in our method estimates a three-way cubic spline at each sensor location, one for each disjoint 8-hour interval in a 24-hour period (12 am to 8 am, 8 am to 4 pm and 4 pm to 12 am), representing three different patterns in the $PM_{2.5}$ variations. The cubic splines for each sensor represented a *baseline* level of $PM_{2.5}$ concentration. The cubic splines may provide a good approximation to the overall average daily variations across sensors but do not capture short term spatio-temporal variations represented by the residual errors in the baseline. The final step of our method is to train a Message-Passing Recurrent Neural Network (MPRNN) across the pollution monitoring points to estimate the residual errors from neighboring

sensors. A short and general background to recurrent neural networks (RNN) is provided in the supplementary section (ref §A.2).

2.1.1 Data

The data used for the modeling the air pollution levels in Delhi was sourced from a combination of 32 local government monitorsand a network of 28 low-cost sensors deployed by us in various locations of Delhi from May 2018 to May 2020. The average availability of each of these sensors are about 90% and 30% over the measured period respectively. This disparity is attributed to a variety of factors such as disconnection for periodic necessary calibration, network outages and periodic servicing of sensors. The sensors are calibrated against the government sensors, by conducting a longitudinal comparison study by measuring in proximity to the location of the government monitoring centers. The locations and their summary statistics of the sensors by location is given by the Tables 2.1 and 2.2, and are shown visually in the box plots in Figure 2.1.

2.1.2 Cubic Splines

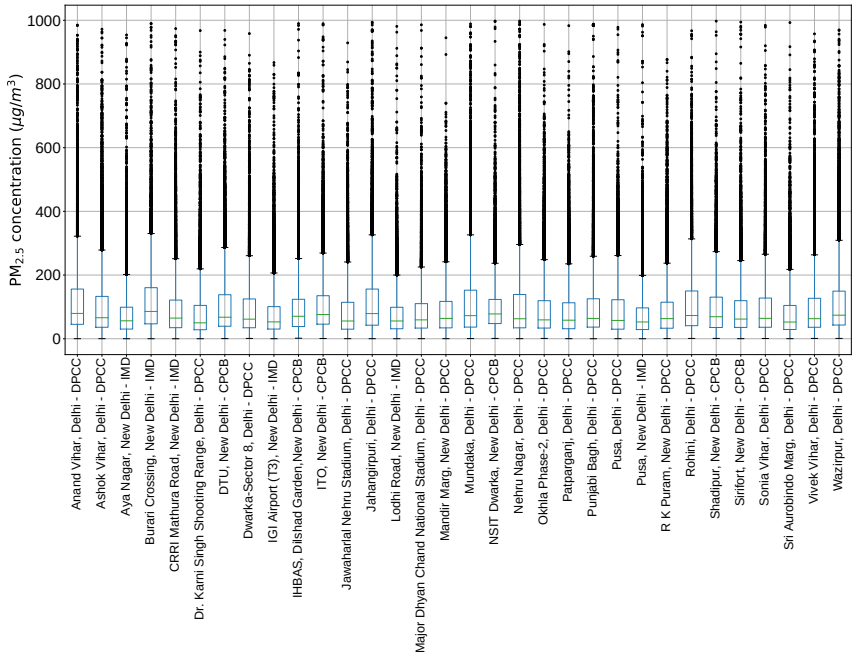
We observe that on a daily basis, depending on the time of the day and the location, there is a low-frequency component that makes up an approximate "baseline level" of PM concentration. Based on this observation, we fit a piecewise polynomial function, called a spline, to model this low-frequency component. We divided a single day into a number of epochs and fit a spline for each epoch. Prior to implementing the cubic splines, we observed that the residual errors from the MPRNN model exhibits different errors at different times in the day. We then proceeded to fit cubic splines based on the daily spatio-temporal patterns per sensor and per location. For example, if our prediction error follows a temporal pattern of say, higher prediction error in the morning, while lower in the afternoon, we can leverage this fitting separate splines for morning and afternoon to subtract out this component. The spline can be of any order, but given our

Location	Count	Minimum	Maximum	Median	Mean	Std Dev
Anand Vihar, Delhi - DPCC	18911	0.2	985.0	79.5	122.1	120.0
Ashok Vihar, Delhi - DPCC	21278	0.2	972.0	66.0	105.9	110.6
Aya Nagar, New Delhi - IMD	20368	0.1	954.0	56.3	76.9	74.4
Burari Crossing, New Delhi - IMD	9593	0.1	989.7	85.7	125.8	121.2
CRRI Mathura Road, New Delhi - IMD	21242	0.0	973.8	64.8	97.0	97.7
Dr. Karni Singh Shooting Range, Delhi - DPCC	19908	0.1	967.5	50.0	82.1	87.1
DTU, New Delhi - CPCB	20854	0.5	968.2	67.6	105.9	102.3
Dwarka-Sector 8, Delhi - DPCC	21615	1.0	958.2	61.5	95.7	94.1
IGI Airport (T3), New Delhi - IMD	20640	0.1	867.2	52.9	79.8	79.7
IHBAS, Dilshad Garden,New Delhi - CPCB	20913	1.6	989.6	70.5	95.3	86.6
ITO, New Delhi - CPCB	19804	0.8	989.2	76.0	107.1	94.2
Jawaharlal Nehru Stadium, Delhi - DPCC	21251	0.2	929.0	55.8	90.3	95.0
Jahangirpuri, Delhi - DPCC	21414	0.2	994.0	79.0	119.3	114.3
Lodhi Road, New Delhi - IMD	20248	0.1	980.8	55.8	77.3	72.5
Major Dhyan Chand National Stadium, Delhi - DPCC	21680	0.2	985.8	59.2	86.5	81.3
Mandir Marg, New Delhi - DPCC	20839	0.3	945.0	63.8	90.5	85.0
Mundaka, Delhi - DPCC	19654	0.5	988.5	72.5	116.6	120.2
NSIT Dwarka, New Delhi - CPCB	21601	1.0	997.5	77.9	97.6	76.8
Nehru Nagar, Delhi - DPCC	21692	0.2	997.5	62.8	110.2	121.9
Okhla Phase-2, Delhi - DPCC	21474	1.0	987.0	59.2	94.3	95.6
Patparganj, Delhi - DPCC	21602	0.2	983.0	58.2	89.7	90.0
Punjabi Bagh, Delhi - DPCC	21049	0.0	988.0	63.8	103.0	112.5
Pusa, Delhi - DPCC	19142	0.2	978.0	57.5	91.9	92.6
Pusa, New Delhi - IMD	19770	0.1	986.2	52.7	77.0	77.7
R K Puram, New Delhi - DPCC	19383	0.5	877.2	63.2	92.9	93.6
Rohini, Delhi - DPCC	21310	1.0	967.0	73.0	116.7	115.2
Shadipur, New Delhi - CPCB	20937	1.0	997.2	69.2	97.6	89.3
Sirifort, New Delhi - CPCB	20735	0.2	994.2	61.8	92.8	89.9
Sonia Vihar, Delhi - DPCC	21176	0.8	984.0	64.0	99.4	97.5
Sri Aurobindo Marg, Delhi - DPCC	20116	0.2	992.8	52.5	80.6	80.4
Vivek Vihar, Delhi - DPCC	21344	1.0	957.2	63.0	101.2	104.7
Wazirpur, Delhi - DPCC	21401	1.0	969.8	74.0	118.7	117.7

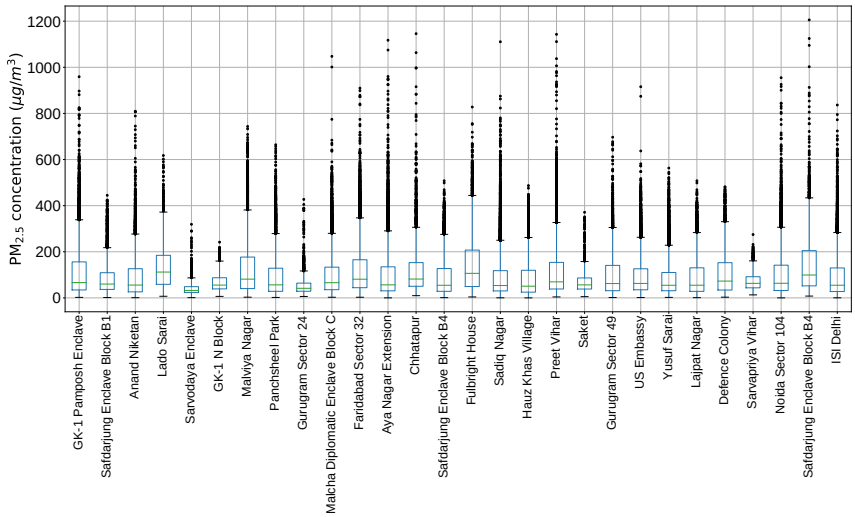
Table 2.1: Summary Statistics of Government Pollution Monitors – number of readings, minimum, maximum, median, mean and standard deviations of the $PM_{2.5}$ concentrations detected

Location	Count	Minimum	Maximum	Median	Mean	Std Dev
GK-1 Pamposh Enclave	10173	2.2	959.0	66.3	113.7	113.9
Safdarjung Enclave Block B1	3804	1.5	444.7	60.0	85.9	74.5
Anand Niketan	8655	0.9	809.3	55.6	91.4	92.5
Lado Sarai	1733	7.0	617.5	112.0	135.6	99.8
Sarvodaya Enclave	810	1.0	319.0	32.2	45.1	39.8
GK-1 N Block	954	6.5	241.7	55.5	67.4	41.5
Malviya Nagar	9253	3.0	743.4	81.2	125.3	117.0
Panchsheel Park	6959	1.9	664.1	56.4	97.0	100.2
Gurugram Sector 24	2776	6.2	427.1	41.5	52.2	40.4
Malcha Diplomatic Enclave Block C	6274	2.8	1047.4	65.9	103.0	100.1
Faridabad Sector 32	13975	2.8	909.8	81.0	119.1	104.5
Aya Nagar Extension	8003	0.1	1117.4	56.5	102.8	113.7
Chhatapur	2753	9.6	1145.8	81.7	125.6	120.5
Safdarjung Enclave Block B4	6998	1.0	507.8	54.7	89.3	84.0
Fulbright House	5103	4.1	827.7	106.4	145.6	124.7
Sadiq Nagar	20079	0.3	1110.8	53.6	90.7	93.4
Hauz Khas Village	4404	0.0	487.0	50.8	87.6	88.0
Preet Vihar	11223	4.2	1142.9	69.5	114.9	114.5
Saket	859	5.3	371.1	56.3	75.1	61.1
Gurugram Sector 49	10246	1.4	696.8	62.2	97.3	90.5
US Embassy	10952	1.4	915.7	62.7	95.3	87.1
Yusuf Sarai	7220	2.1	563.0	54.7	88.0	86.2
Lajpat Nagar	6959	1.0	507.8	54.9	89.9	84.2
Defence Colony	4721	3.0	481.5	72.8	105.3	92.7
Sarvapriya Vihar	1009	13.0	274.7	63.1	72.6	41.5
Noida Sector 104	15586	0.1	954.8	63.3	99.5	94.4
Safdarjung Enclave Block B4	4538	7.8	1205.5	99.2	143.7	124.6
ISI Delhi	16075	0.3	836.4	54.8	91.0	89.5

Table 2.2: Summary statistics of low-cost pollution sensor network – number of readings, minimum, maximum, median, mean and standard deviations of the $PM_{2.5}$ concentrations detected



(a) Government network



(b) Our low-cost network

Figure 2.1: Boxplots of distributions from the monitors

residual error patterns, but we found that piecewise cubic spline works best.

Suppose at time t and location v, the raw PM value is given by $y_{v,t}$. Then, the piecewise spline to predict y, with time period p is given by:

$$\hat{y}_p(v,t) = \alpha_{v,p} * t^3 + \beta_{v,p} * t^2 + \kappa_{v,p} * t + \nu_{v,p} \tag{2.1}$$

Note that the chosen parameters per sensor $\alpha_{v,p}, \beta_{v,p}, \kappa_{v,p}, \nu_{v,p}$, where $p \in$ {"morning", "afternoon", "evening"}, depend on the patterns in our residual errors and are fit accordingly to minimize the root mean-squared residual error:

$$RMSE(v) = \sum_t \sum_p \sqrt{(y(v,t) - \hat{y}_p(v,t))^2} \tag{2.2}$$

2.1.3 Message-Passing Recurrent Neural Network

Message-Passing Recurrent Neural Network (*MPRNN*), based on [Iyer et al. 2020; Gilmer et al. 2017], is a neural network architecture that is applied on a graph in order to predict values at each node in the graph. This approach enables to us incorporates spatial interactions between each pair of nodes as "messages" that are broadcast from every node to its neighbors. Each node has a modified version of a Long Short Term Memory (LSTM) network that iterates between message-passing and the recurrent computations.

Suppose $y_{v,t}$ is a quantity of interest at node v and time t, for which we would like to build a predictive model. Mathematically, we would like to learn a function $\mathcal{F}$ such that, $y_{v,t+1} = \mathcal{F}(v_1, y_{v_1,t}, v_2, y_{v_2,t}, \ldots; v_j \in \mathcal{V})$ where the set $\mathcal{V}$ denotes the set of all the nodes in the graph. A recurrent neural network unit is assigned to each node in the graph, with each node v maintaining a hidden state $h_{v,t}$ at time t. Through a message-passing phase and a time-recurrent phase, our model infers the next hidden state, $h_{v,t+1}$ from which the PM value at v is decoded. A message-

passing operation allows one segment to observe the hidden state of its neighboring segments.

The computation proceeds in five steps, as five layers of the neural network. In the first phase, the observation phase, the input observations $Y_t = \{y_{v,t} | v \in \mathcal{V}\}$ at time t are encoded into $h_{v,t}$ by the observation operation O_v. In the second and third phases, one or more iterations of messaging (M) and updating (U) operations are performed to propagate the observations in the graph. In the fourth phase, for each node, a time-recurrent operator T_v utilizing an LSTM unit takes as input the final hidden state $h_{v,t}$ and predicts the next hidden state $h_{v,t+1}$. The final phase is the readout operation R_v, which decodes the hidden state to produce the output value to be predicted $\hat{y}_{v,t+1}$. These five steps are shown below. The message function takes as input the hidden states of a pair of nodes v and n and the Euclidean distance between them, $d_{v,n}$ as the influence of the pollution at a given location on the pollution at another location would depend on the distance between them. Hence, we include the distance in the embedding.

$$h_{v,t} = O_v(h_{v,t-1}, y_{v,t}) \tag{2.3}$$

$$m_{v,t} = \sum_{n \in V - v} M(h_{v,t}, h_{n,t}, d_{v,n}) \tag{2.4}$$

$$h_{v,t} = U(h_{v,t}, m_{v,t}) \tag{2.5}$$

$$h_{v,t+1} = T_v(h_{v,t}) \tag{2.6}$$

$$\hat{y}_{v,t+1} = R_v(h_{v,t+1}) \tag{2.7}$$

For a selection of nodes $\mathcal{W}$ in the graph, the components of the model $\{O_w, M, U, T_w, R_w, | w \in \mathcal{W}\}$ are defined. During inference, the states $H_t = \{h_{w,t} | w \in \mathcal{W}\}$ are maintained at each time step. The hidden state for each segment is initialized at $t = 0$ randomly during training and evaluation $h_{v,0} \sim \mathcal{N}(0, 1)$.

2.1.4 Implementation

We used the data from May 1, 2018, to Nov 1, 2019, a period of 18 months, as the training period. The number of samples we had for training were 166979 from our low-cost sensor network, and 371806 from the government network, resulting in a total of 538785 samples. The model was trained at each sensor location, using as input data from all the other monitors except itself, over the entire training period. We used the Adam optimizer [Kingma and Ba 2015] with a learning rate of 0.001, and ran the training for 30 epochs to ensure a robust and well-trained model. To validate the model, we used the remaining 6 months data from Nov 1, 2019, to May 1, 2020. The number of ground truth samples available in this period were 20408 and 91493 in the low-cost network and government network, respectively, resulting in a total of 111901 samples. However, only 12 out of the 28 low-cost sensors were operational in the testing phase, since many of them had not been serviced properly, partly owing to the COVID-19 pandemic. The testing error reported under Results (§5.4), therefore, shows the predictions tested at 12 low-cost sensor locations and 32 government monitors, a total of 44 locations combined. The MPRNN is implemented using the *Deep Graph Library* [Wang et al. 2019] and PyTorch [Paszke et al. 2019] in Python.

2.2 Results

Our data consists of $PM_{2.5}$ concentration data averaged to the hour from the 28 low-cost sensors and the 32 government monitors, a total of 60 monitors, collected over a period of 24 months, from May 1, 2018, to May 1, 2020. We use the until Oct 30, 2019 for training (75%) and hold out the remaining (25%) for testing. We report two criteria – the root mean squared error (RMSE) and the mean absolute percentage error (MAPE). We evaluate our models on the data from the combined set of our 28 low-cost sensors and the 32 government monitors, as well as separately on each set. For each of these locations, we compare our model-based predictions with the ground

truth of the measurement of the pollution sensor.

We contrast our combined model with two alternative modeling approaches in order to set a baseline to benchmark the MPRNN model performance. The first one is the STHM itself, a state-of-the-art spatio-temporal modeling methodology. When the STHM is used solely for the prediction, it performs poorly, as it does not model unknown non-linear spatial dependencies due to dispersion and so on.The second baseline is an alternative neural network formulation that collects information from a specified number (K) of nearest neighbors to a location L, and feeds them into a trained recurrent neural network, to predict the value at L. Unlike the MPRNN, this model does not account for explicit spatial influence between every pair of sensors, thus allowing us to see how a more simplified multi-variate non-linear model might perform. We call this model the k-Nearest Neighbor (k-NN) Spatial Neural Network (refer §A.3 for more details on this model).

Model	Our sensors		Govt monitors		Combined	
	RMSE	*MAPE*	*RMSE*	*MAPE*	*RMSE*	*MAPE*
STHM	29.5	33.2%	38.3	32.7%	31.4	37.8%
k-NN Neural Network	38.8	35.7%	69.7	52.6%	54.2	51.6%
MPRNN	37.1	34.4%	65.2	51.3%	56.3	51.6%
Per-Sensor Spline	25.1	32.8%	60.4	49.1%	47.3	36.5%
STHM + Spline	21.8	25.8%	27.2	24.9%	24.2	26.2%
k-NN Neural Network + Per-Sensor Residual Spline	11.6	16.3%	18.1	13.4%	12.8	14.7%
MPRNN + Per-Sensor Residual Spline	9.8	10.2%	13.2	11.7%	10.4	12.6%
Per-Sensor Spline + Residual MPRNN	10.1	10.5%	14.7	12.2%	10.7	13.5%
Per-Sensor Spline with STHM imputation + MPRNN	9.5	9.4%	12.6	10.5%	10.1	9.6%
MPRNN with STHM imputation + Average Residual Spline	10.1	9.8%	13.2	10.9%	11.2	10.3%

Table 2.3: RMSE and MAPE of prediction of PM concentrations, averaged across all the sensor locations. The RMSE is in units of $\mu g/m^3$. The best performing model is shown in boldface. The Per-sensor spline with STHM imputation followed by the use of MPRNN to estimate residual errors performs the best and has significantly lower RMSE and MAPE than any of the models that do not combine these steps. Using just a cubic spline or STHM or MPRNN in isolation results in a significant increase in the RMSE and MAPE errors. Replacing the per-sensor spline with an average spline does not significantly affect the RMSE and MAPE errors. The STHM model is primarily useful in filling in missing values and only provides a minor improvement to the MPRNN + per-sensor spline model. Another baseline method where we replace the MPRNN with k-Nearest Neighbors increases the MAPE and RMSE errors.

Overall, the MPRNN model with imputed data using STHM along with the spline correction

provides a very highly accurate estimation of the PM concentration level across all locations (ref Table 2.3). The best performing model is able to predict $PM_{2.5}$ concentrations with an average RMSE of 10.1 $\mu g/m^3$ and MAPE of 9.6% across all the locations and over the testing period. While estimating a spline per location provides the best predictive performance, we note that using an average spline across all observed locations only marginally increases the RMSE and MAPE errors. The average spline is computed after averaging the data over all the locations.

Across all locations, the median RMSE and MAPE are 9.15 $\mu g/m^3$ and 8.64% respectively (ref Figure 2.2(b)). The best case values are 4.28 $\mu g/m^3$ and 5.57% respectively, and the worst case values are 24.1 $\mu g/m^3$ and 19.64% respectively. The location where we have minimum MAPE is at a location in Green Park, a very busy area of south Delhi, further validating the need for fine-grained pollution sensing in a large city like Delhi.

2.2.1 Effect of Spline Correction

The 3-way cubic spline fit shows a common trend of baseline pollution rising steadily up to 8 am, then decreasing up to 4 pm and then increasing again until midnight. We note that this is the composite polynomial model of the PM concentrations in an average day (ref Figure 2.3(d)). The median error of this model is about 40 $\mu g/m^3$ at each of the three windows, 12 am-8 am, 8 am-4 pm and 4 pm-12 am, and this is reduced to about 10 $\mu g/m^3$ post the neural network model fit on the residuals. Figures 2.3 and 2.4 show the per-sensor splines and the average spline in detail. Not only do the per-sensor splines vary widely across space, we notice that regions with significantly high spline residual errors like the sensors A838, E8E4 and 2E9C in Figure 2.4(a), are all located in central locations of Delhi with well established commercial activity like Connaught Place, Sardarjung Enclave and Lado Sarai respectively. Further, in Figure 2.4(b), the outliers with significantly high residual error splines among the government monitoring stations are Patparganj DPCC, Punjabi Bagh DPCC and DKSSR DPCC. While Patparganj is situated next to an industrial area, Punjabi Bagh is a well-known residential locality with established commercial activity centers,

and DKSSR, short for Dr. Karni Singh Shooting Range, is a shooting range located in the outskirts of Delhi next to an interstate highway. The diversity of these splines across various geographical regions further indicate the need to model fine-grained pollution profiles in seemingly remote as well as central locations of Delhi. We also note that the average spline can sufficiently operate for bootstrapping at locations where we do not have enough sensor data to begin with.

For the most part, locations that exhibited high residual errors after MPRNN fit continued to show high error (relative to other locations) even after spline correction, even though the magnitude of the residual decreases. This phenomenon is partially explained by the high baseline values of the sensors with high residual errors, that is often coupled with high variance in measurement.

2.2.2 Effect of Network Size

The fewer the monitors we used in our hybrid model, the greater was the final prediction performance. As figure 2.5 shows, with only one monitor in the network, the predictive errors are about 35 $\mu g/m^3$ and 20 $\mu g/m^3$, respectively, for the low-cost sensor network and government network. However, as we include data from more nodes in the network, final prediction error drops sharply to about 15% and then gradually tails off at about 10%. The error flattens out about 30 sensors, which is approximately the number of sensors of each type that we have in our experiment. We infer that having an even denser deployment likely adds little value to the predictive performance.

2.3 Discussion

Our contributions are significant when compared to the recent and fast-growing literature that explores the use of distributed sensor networks to gather information on air pollution and other meteorological variables in urban contexts [Liu et al. 2018; Chambliss et al. 2021; Liang et al. 2021; Ferraro and Agrawal 2021; Ludescher et al. 2021]. Clements et al. [Clements et al. 2017] provide

a comprehensive review of many such works. In the last few years, researchers have sought to learn more about how pollution sensing systems of low-cost sensors may be deployed in urban contexts [Jiao et al. 2016; Lin et al. 2015; Shusterman et al. 2016; Moltchanov et al. 2015; Sun et al. 2016; Tsujita et al. 2005; Gao et al. 2015]. With the exception of Gao et al. [Gao et al. 2015], who examine the performance of fine particulate sensors in Xi'an in China, most of these deployments have occurred in areas with significantly lower air pollution than the city of Delhi in India. In this work, we provide evidence of modeling a fine-grained low-cost pollution sensing map from a highly polluted city like Delhi. Gao et al. [Gao et al. 2015] also point out that low-cost $PM_{2.5}$ sensors may perform worse in very low pollution environments, suggesting that they may be relatively more useful when particulate concentrations are high. While their study focused on Xi'an, a large city (area 3,898 mi^2) with only 8 low-cost sensors, we dramatically increase the density of the deployment by 28× in Delhi (area 573 mi^2) with 28 sensors. Further, the large longitudinal dataset we have been able to capture over 2 years as compared to prior work which captured at most a few weeks of data, allows us to model long-term seasonal changes and train more complex neural network models that can adapt to seasonal and daily patterns and produce significantly low RMSE. Related approaches in this space can be broadly classified into three groups – spatial interpolation approaches, land-use regression and dispersion models Xie et al. [Xie et al. 2017], Jerrett et al. [Jerrett et al. 2005]. In the case of dispersion models, they assume that an appropriate chemical transport model is identified along with their parameter values, and a high-quality emissions inventory. In the case of land-use regression models, having access to environmental characteristics that significantly influence pollution is critical. This additional data is often suited for longer range predictions, as the geographical and meteorological data vary over a longer temporal and coarser spatial grids [Yeh et al. 2020]. For instance, in the US EPA dispersion model, parameters are estimated on grid cell squares with a length in the order of a few kilometers [U. S. Environmental Protection Agency (EPA) 2021], while the parameters are used for inference of meteorological outputs at spatial resolutions of up to 500 m. Our approach, in contrast, relies

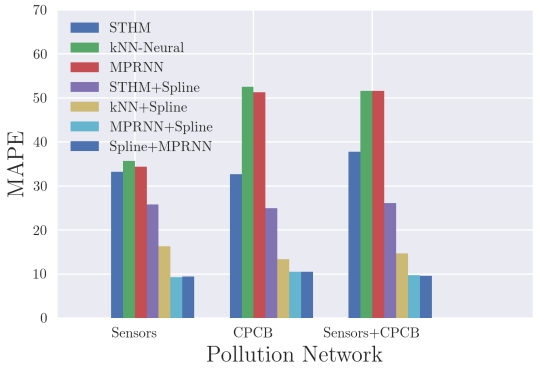
on fine-grained positioning of low-cost sensors and makes the case for crowdsourcing pollution sensing. This way, we demonstrate that a collective effort from citizens using low cost sensors can actually help in building a high quality pollution sensing map.

The low MAPE and RMSE across all monitors in Delhi provided by our Per-Sensor Spline+MPRNN with STHM imputation model are significant as it means that our model can detect hazardous air quality with high precision. The RMSE error is significantly lower than the observed variance in $PM_{2.5}$ concentrations in a day, making it useful for short-term and intraday analyses as well. The WHO air quality standards prescribe that $PM_{2.5}$ levels should not exceed 10 $\mu g/m^3$ and 35 $\mu g/m^3$ at an annual and daily average levels, while the Indian Government air quality standards prescribe 40 $\mu g/m^3$ and 60 $\mu g/m^3$ respectively. We note that for the 60 sensors, Delhi has exceeded these prescribed levels 371 out of the 641 days on a daily level, across 2 years of our measurement. The 9.6 % MAPE error that we are able to achieve, corresponds to the ability to detect hazardous air quality as per Indian government standards with 93.5% precision and 90.8% recall. This further indicates that the low error rate we have obtained leads to an almost exact forecasting of hazardous air quality. This enables citizen-driven sensing where pollution sensor readings can be crowdsourced and effective policy interventions like clean energy policies that penalize construction sites that have $PM_{2.5}$ levels more than 25% higher than the nearest monitoring center can be operationalized [1]. Specifically, the improvement in forecasting power is achieved in specific pollution hotspots like bus stations, markets, etc. (Figures 2.2(c), 2.2(d)). In addition, we can provide transparency of the overall average pollution of the city [2] and contribute towards increasing the co-benefits of clean energy policies [Qian et al. 2021; Tibrewal and Venkataraman 2021]. The development of fine-grained pollution sensing maps at low-costs can further catalyze the deployment of such monitoring networks in other polluted cities, where the pollution networks are sparse. With citizens procuring, deploying and modeling pollution of cities accurately, this

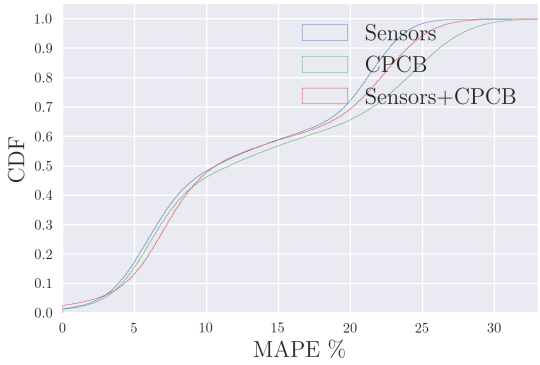
[1] https://indianexpress.com/article/cities/delhi/dust-management-committee-recommends-air-quality-monitors-at-large-delhi-construction-sites-7437599/

[2] https://www.downtoearth.org.in/blog/air/delhi-s-air-quality-and-number-games-76214

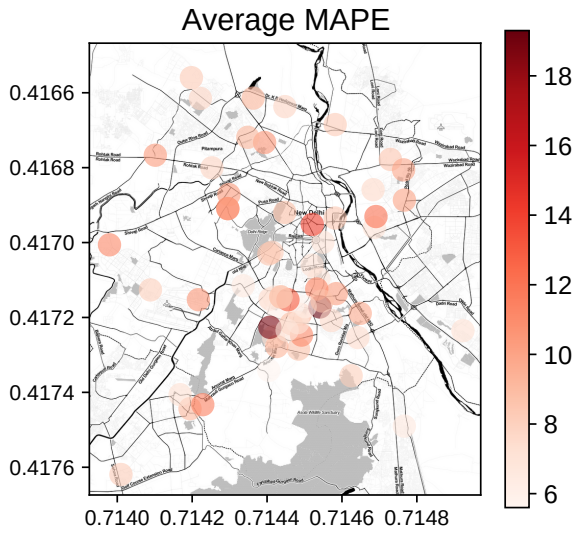
work provides a way forward for developing high-quality fine-grained pollution sensing maps.



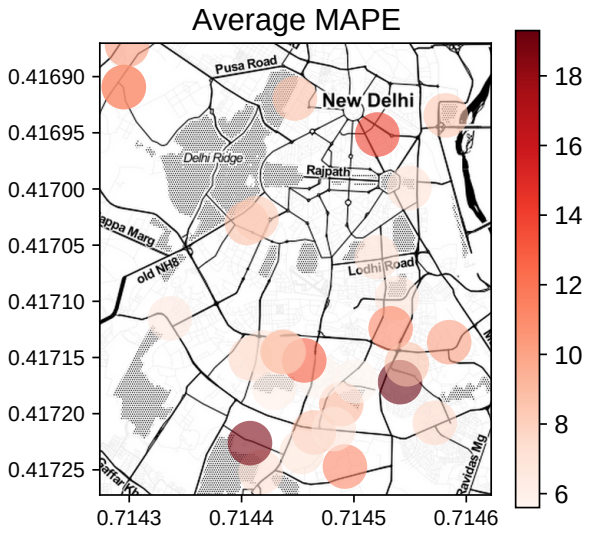
(a) Bar plot comparing our methodology with other competing approaches. We note that modeling spatiotemporal interactions using a neural network such as MPRNN and accounting for intra-day periodic patterns in the form of spline corrections together make a big difference in the performance.



(b) Distribution of MAPE for the best performing model shown in Table 2.3 across all the locations shown as a cumulative density function (CDF).

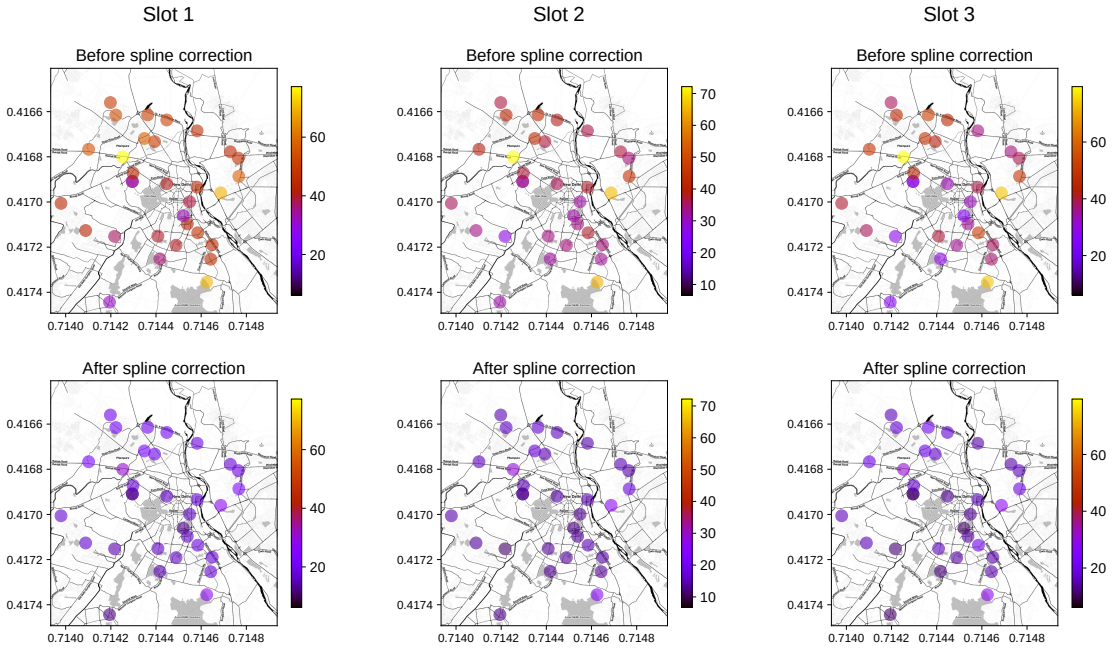


(c) Prediction errors of the best performing model (MPRNN+Spline) at every monitoring location on the map



(d) Errors of the final prediction zoomed into the regions with highest concentration of sensors (New Delhi and South Delhi)

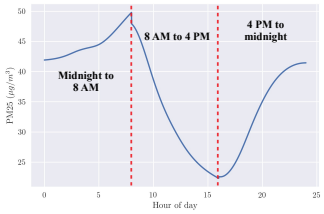
Figure 2.2: Prediction errors of $PM_{2.5}$ during the test period (Nov 1, 2019 - May 1, 2020) shown as the Mean Absolute Percentage Error (MAPE) of the ground truth and predicted $PM_{2.5}$ concentration. In this period, the $PM_{2.5}$ concentration values ranges between 0 and 1000 $\mu g/m^3$, and average value being ~130 $\mu g/m^3$



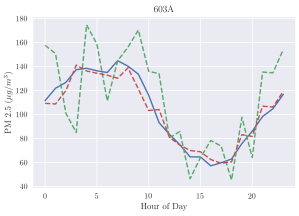
(a) Slot 1 (12 AM - 8 AM)

(b) Slot 2 (8 AM - 4 PM)

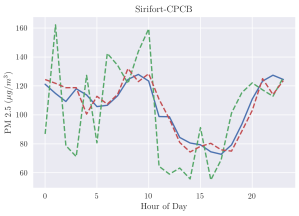
(c) Slot 3 (4 PM - 12 AM)



(d) Composite cubic spline correction consisting of three splines fitted for three non-overlapping parts of the day – midnight to early morning (12 AM to 8 AM), midday (8 AM to 4 PM) and evening to midnight (4 PM to 12 AM)

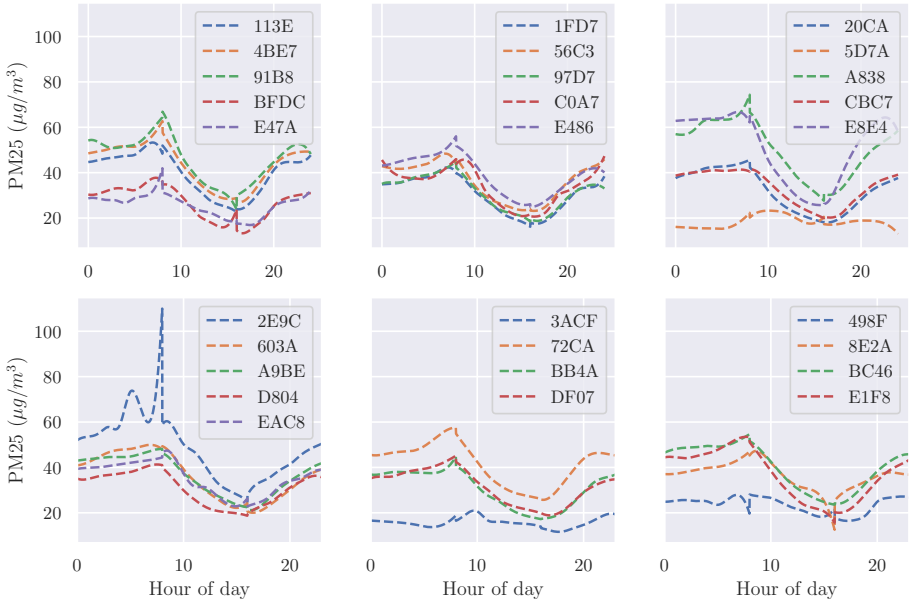


(e) Ground truth $PM_{2.5}$ (blue), along with MPRNN prediction (green) and final prediction after spline correction (red) at one of our sensor locations in Chanakyapuri in New Delhi.

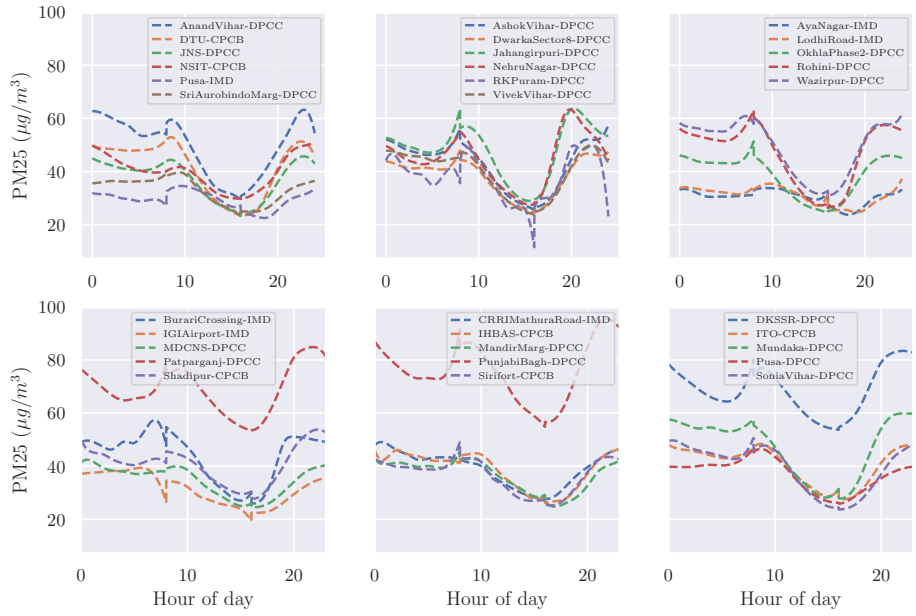


(f) Ground truth $PM_{2.5}$ (blue), along with MPRNN prediction (green) and final prediction after spline correction (red) at the CPCB monitor at Sirifort in South Delhi.

Figure 2.3: This figure aims to show the interpretation of the spline correction, and its effect on the residual. The top two rows show the distribution of the residuals (in PM units of $\mu g/m^3$) over space, before and after the spline correction. Three different splines were fitted over the residuals in three different time slots in the day. We observe that for the most part, locations that exhibited high residual errors after MPRNN fit (in the upper quantiles of the residual error distribution) continued to show high error (relative to other locations) even after spline correction, even though the magnitude of the residual does decrease. This phenomenon is partially explained by the high baseline values of the sensors with high residual errors, that is often coupled with high variance in measurement.



(a) Splines for each of the 28 low-cost sensors (4 digit sensor IDs are shown for brevity rather than the full location names)



(b) Splines for each of the government pollution monitors

Figure 2.4: The daily variations in the splines learned for each of the sensors show that there are temporal patterns which when incorporated into a prediction model can significantly improve prediction accuracy. Each color shows a different sensor location. Each plot shows about 4-5 sensor locations for the sake of readability. The first six plots show the low-cost sensor locations, and the next six show the government monitors.

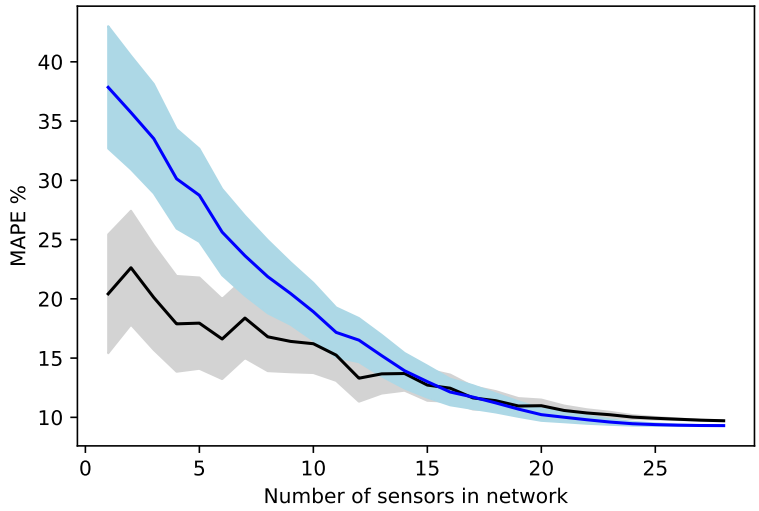


Figure 2.5: This figure shows the impact of sensor network size on forecasting error. The blue line shows the errors for our low-cost sensors, and the black for the government monitors. We see that the more sensors we use in our model, the better the performance of the model in terms of the prediction error. The error flattens out about 30 sensors, which is approximately the number of sensors of each type that we have in our experiment. We infer that having an even denser deployment likely adds little value to the predictive performance.

Part II

Data Science for Better Road Traffic Analysis

3 Introduction

Poor road traffic management can trigger extended periods of traffic congestion as is witnessed in most parts of the world. As per Texas Transportation Institute's 2009 Mobility report [Texas Transportation Institute], congestion in the US has increased substantially over the last 25 years with massive amounts of losses pertaining to time, fuel and money. In the top 10 cities with the worst levels of congestion in the world, the average number of hours wasted per commuter per year is over 150 hours [Friedman 2020]. When the number of hours wasted exceeds about 35 hours per year, it is observed to affect the economy negatively [Badger 2013]. This kind of prolonged traffic congestion persists in many large urban cities [TomTom International BV], especially in developing regions, with poorly managed road networks and freeways and this has remained an important barrier to economic development in these regions. Reducing traffic jams improves quality in life also in the form of improved air quality. A report on the air quality study in Delhi [Sharma and Dikshit 2016] attributes nearly 20% of PM concentration in the air to traffic.

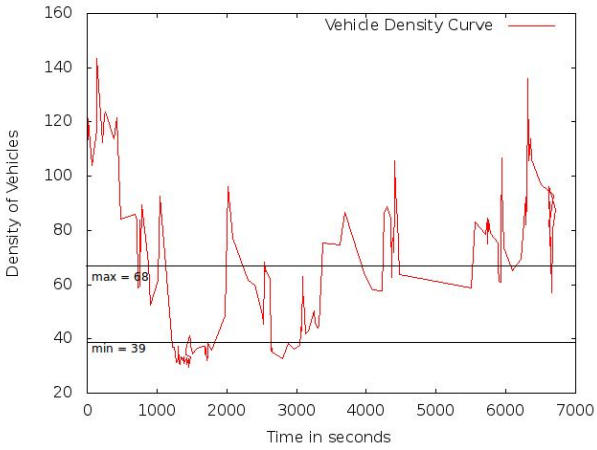
Traffic congestion can be both good and bad [Badger 2013]. A certain level of congestion, especially in the dense urban regions, may indicate economic prosperity and thriving economic development, as in many major cities. However, commuters wasting away their times on the freeways is the bad congestion. We explore the issue of sudden congestion occurrences in such free-flow settings, such as highways, major junctions and freeways. Contrary to the conventional belief that traffic congestion is triggered due to excessive traffic, traffic jams for elongated time-periods can actually be triggered by small traffic bursts over small time scales [Jain et al.

2012]. The underlying cause of the traffic jam is not due to the lack of road capacity but due to a "spiralling effect" triggered by a small burst that pushes the road traffic network to a low-capacity operational capacity equilibrium point. This equilibrium point is highly stable that the only way to recover from this is to dramatically reduce the input flow into the traffic network and drain the congested network. This phenomenon occurs because traffic links exhibit a traffic curve behavior where the capacity of a link is variable dependent on the traffic density on any link; any input flow beyond the optimal operational rate over short time that triggers the density beyond a critical threshold automatically triggers a spiralling effect resulting in a traffic jam. We refer to any jam caused this way as a *sudden traffic jam*.

Traffic collapse results when the traffic density on a link exceeds a certain threshold. The operational free-flow exit rate of the link, which determines how quickly the link is drained, varies with the traffic density [May 2000]. Each traffic link reaches an optimal capacity at a corresponding optimal operating density, beyond which the exit rate rapidly drops. In New York City, the exit point of the Williamsburg bridge on the Manhattan side (figure 3.1). A traffic light immediately follows the bridge, and a large number of vehicles are regularly stuck there for several minutes especially during the morning and evening rush hours. The graph shown in figure 3.1(b) is a plot of the traffic congestion at the exit point of the Williamsburg bridge. The y-axis is the measure of the vehicle density, or the traffic density, on the road. The x-axis represents the time in seconds after 14:50:40 when the traffic was observed. By manual inspection of the camera feed, we are able to determine the *min* and *max* densities as well. The max density is a density value above which there is complete congestion, whereas the *min* is a value below which there is no congestion at all. In this particular example, the min and max values were determined to be 68 and 39. The congestion can be mitigated if the input rate of the vehicles into the bridge is controlled before it reaches the tipping point for congestion collapse.

First, we explore the concept of sudden traffic jam, provide a formal definition and illustrate occurrences of sudden jams in two cities – New York and Nairobi. We also show a connection

(a) Still from a CCTV camera from the NYC DoT showing congestion at the end of the Williamsburg bridge on the Manhattan side



(b) Traffic density on the link shown in the image as a function of time

Figure 3.1: Traffic on the Williamsburg bridge in New York City

to the traffic curve, and how we can potentially predict a sudden jam based on road segment characteristics. Further, we present a new take on the problem of traffic forecasting from sparse but easily available and accessible data such as mobility traces of public transportation vehicles. Many popular traffic prediction applications that are used for travel time prediction today such as Google Maps and Waze rely on large amounts of crowdsourced data from human mobile phone users on the road [Barth 2009]. This crowdsourced approach is typically not useful or reliable in all but the big cities [Yuniar 2018], since data is either unavailable or stale. Such an approach is also difficult to implement in regions with strict regulations on free data access and privacy concern among the public.

We demonstrate in this traffic prediction work that it is possible to forecast road traffic conditions, in particular the level of congestion, from sparse data collected from significantly fewer input sources. Our input sources are public transportation buses fitted with location trackers. A significant difference from more traditional takes in this problem is the *sparsity* of the data in time and space. On the space front, we note that buses typically ply only on select road segments in a city in a predictable and repeatable fashion, covering only a fraction of all the roads in a city.

On the time front, we note that many related works use either taxi traces [Lv et al. 2018; Cui et al. 2018; Achar et al. 2018] or data from specialized instrumentation called loop detectors [Guo et al. 2019; Yu et al. 2018; Li et al. 2018]. These data sources supply data nearly continually throughout the day by very nature of design, whereas buses do not ply with as much temporal frequency and spatial coverage at late nights as much as during the day times.

Accurate modeling of traffic flow and congestion requires additional features apart from vehicle traces that directly impact traffic flow, such as the number of lanes, frequency of stop lights and pedestrian density. However, the public transit authorities may either not collect such data or may not make them available to us. Without the full set of features, forecasting road conditions strays further from simulating a closed system and thus traffic patterns appear highly non-linear. In this work, we motivate Message-Passing RNN (*MPRNN*) which reduces confounding effects through spatial awareness and models interactions between road segments such that forecasts are resilient to unreliable local measurements. Using the MPRNN, we are able to make longer-term forecasts of traffic speeds in both space and time using only limited input data from a small number of road segments.

Our contributions in this work are three-fold from a performance point-of-view. First is the novel application of the message-passing neural network formulation in the context of traffic congestion forecasting and mapping. We demonstrate improved prediction performance, as well as better and faster modeling of spatial interactions using the MPRNN formulation. Second, we show, for the first time, competitive forecasting results over a working day period (approx 12 hours). The MPRNN is able to predict next step traffic speeds with an impressively low error less than 0.3 mph, and forecast over longer periods with an minimum error of about 1.8 mph. Third, we demonstrate the ability to forecast speeds at road segments that are not immediately adjacent to observed road segments ("spatial" forecasting). In fact, we are able to forecast speeds in a segment using limited data from segments up to about one kilometer (0.6 mile) away.

Finally, we propose a methodology for signal control for efficient traffic flow. We observe

that a significant percentage of traffic congestion occurs on free-flow networks (i.e. without explicit signal control) such as highways. The most common such scenario is the merging of n lanes into m lanes where $n > m$. This reduction in the link capacities results in congestion when the input rate exceeds a certain threshold value. By monitoring such choke points and implementing automated signal control that utilizes known traffic density information, we aim to create a distributed system of traffic signals that work to increase the overall operational capacity of road traffic networks and reduce instances congestion collapse. There has been a large body of work in estimating traffic conditions on the roads using a variety of methods such as sensors, cameras and so on [Sen et al. 2010; Sen et al. 2011; Sen et al. 2012, 2013; Aditya et al. 2016]. Our work builds on works like these, as we depend on the knowledge of the traffic density at the input links.

There are various traffic management applications that are used or have been used in traffic engineering, monitoring and control, such as the VII California initiative and the Berkeley Highway Laboratory (BHL) testbed by the California Partners for Advanced Transportation Technology (PATH) at Berkeley, TRANSYT 7F program by the McTrans Center at the University of Florida, Synchro and SimTraffic software by Trafficware, Quadstone Paramics, a leading microscopic traffic and pedestrian simulation software used by many planning professionals, and many others. The Texas Transportation Institute (TTI) has developed signal control and optimization programs for signalized arterials, such as the enhanced versions of PASSER, that are widely used in local, state and federal levels for controlling the signal timing information. The McTrans center at the University of Florida has made available a host of applications related to road traffic management, such as the Highway Capacity Software (HCS) for planning and operational level analyses, and the microscopic traffic simulation software package TSIS-CORSIM containing a suite of applications including TRANSYT 7F, a signal timing optimization program.

Our idea for traffic signaling networks differs in spirit from these and several other pieces of work in the Intelligent Transportation Systems community in the sense that may be viewed as

a network-wide generalization of the "synchronizing green signals" concept. Our design takes a local view of the problem where traffic signals within a small geographic area locally coordinate among themselves in a decentralized manner, relying only on local traffic state for controlling signals, without making any assumptions about the traffic flow characteristics. Our work borrows ideas from Internet congestion control with two differences – links have variable capacities and there are no packet drops. Our signaling protocol is motivated by the concept of backpressure algorithms [Mahajan et al. 2002; Tassiulas 1995] where every traffic signal exerts a backpressure on upstream traffic signals to reduce the input rate, keep link buffers from reaching capacity and thus prevent a congestion collapse.

We describe our signaling control protocol for free-flow traffic and show results on simulation. We have written our own traffic simulator for the purpose of evaluation, simulate the protocol application on two real-world free-flow road networks in the world and show significant improvements in the operational capacity. We are able to achieve throughput increase by several orders of magnitude (between 3× and 5×), and prevention congestion collapse in the face of bursty traffic conditions. Our protocol provides strong theoretical guarantees. We have proved that scheduling based on local information is sufficient to achieve an optimized global scheduling solution. Finally, our protocol can also be easily applied in practice, using CCTV cameras on traffic signals or other such mechanism, to estimate traffic density.

4 | Understanding Traffic Collapse and Traffic Jams

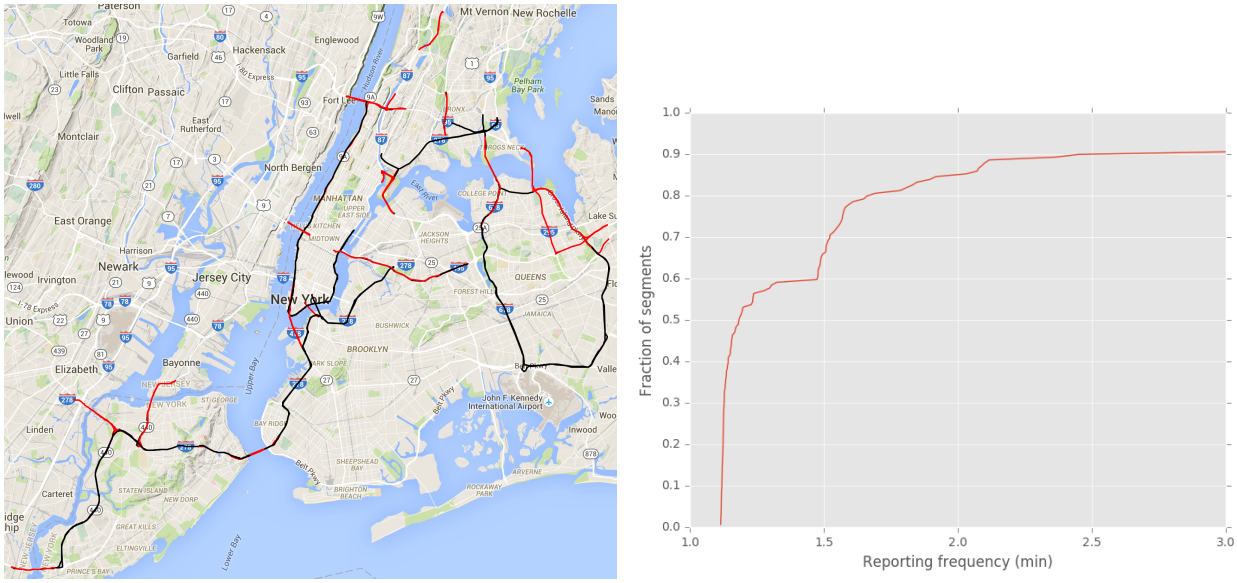
4.1 Materials

4.1.1 New York City data

Data comes from the City of New York Department of Transportation (NYCDOT). The department provides a website[1] that publishes traffic information for various traffic segments throughout the five boroughs of New York City; there are currently 153 such segments. Each segment has a name, a unique identifier, and location data. The name is a string describing the segment as most resident travelers would, "FDR, north 25th at 63rd street," for example. The location data consists of a polyline, suitable for understanding the geography of the segment and placing it on a map.

Along with basic information, measurements come with a timestamp and two types of speed data: the average vehicular speed over the segment, and the average time required to traverse the segment. In theory, each signal is continuous—its timestamp representing a snapshot. In practice, and due to limitations in polling, the information is updated each minute. Thus, although the timestamp is at the granularity of seconds, it can be resampled to the granularity of minutes without loss of generality.

[1] https://www1.nyc.gov/html/dot/html/about/datafeeds.shtml



(a) Measured road segments in across the New York City area. Segments in black are those with reporting times of 75 seconds or less. *(use color in print)*

(b) Distribution of reporting times.

Figure 4.1: Available segments in the loop detector data feed from NYC DoT. The left plot shows the segments on the map, color-coded by the reporting frequency, and the right side shows the distribution of reporting frequencies.

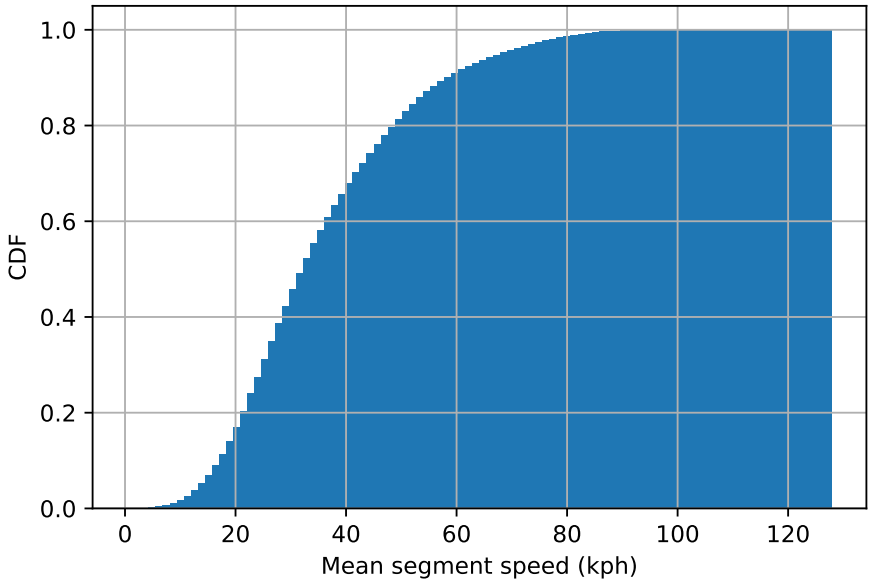


Figure 4.2: Cumulative distribution of hourly traffic speeds across all the segments in the city

Data presented in this article was collected between November 2014 through April 2016. Thus, in theory each segment should contain approximately 740,000 measurements. However, the reporting reliability at each segment varies,[2] so not all segments have reports for each minute. Further, periodic system downtime, on both the reporting and collection end, prevented complete continuous polling. Inter-polling statistics—the average amount of time between traffic measurements—is one way of describing a segments' reliability. The average inter-polling time across all segments was approximately 7.85 minutes, however that number is dominated by a few very unreliable segments as the standard deviation is almost 63 minutes. In the best case, there is a measurement every 66.88 seconds; in the worst, every 12.72 hours. Figure 4.1(b) presents a distribution of average reporting times across the signals.

4.1.2 Nairobi data

We analyzed the Uber Movements data for Nairobi city in Kenya to study sudden traffic jams [Uber Technologies]. The data has hourly average speeds in 4817 road segments in the city. Every segment or way in the dataset is defined as the stretch of road between two junctions. Each road segment has multiple nodes which are used to characterize the stops along the way, and the dataset consists of average speed values every hour for every pair of source and destination nodes. To calculate the average speed in a segment, we average all the individual speeds across all consecutive pairs of source-destination nodes in the segment. Each segment corresponds to a *way* in OpenStreetMap [OpenStreetMap contributors 2017], denoted by a *way_id* , and therefore we are able to look up these segments on a map. We analyzed the dataset from January 2018 to December 2019. Figure 4.2 shows the distribution of average speeds and max speeds of 4817 segments. As we can see from the data, only 12% of the average speed is exceeding the speed limit of 50 km/hr [Hodge]. And the average length of such a road segment is about 80 m long [Nesbitt and Dara-Abrams 2017]. For most of the analysis in this work, except for the clustering of segments, we only choose a subset of 2903 segments for which data is available for at least 18 hours in a day. For the clustering, we imposed an even more strict requirement that each selected segment should have at least one data point for every hour of the day. This gave us only 2005 segments. Additionally, many of the segments did not have contiguous data at the hour. For such segments, we did a partial imputation of the gaps in the data, up to a maximum of 3 consecutive hours only, using simple linear interpolation. If there were gaps longer than 3 hours in the data, only the first consecutive 3 hours were filled, and the remaining were left empty.

[2]On average, a segment consists of approximately 513,000 measurements (SD$\approx$ 179, 000 minutes).

4.2 Theory and Empirical Approximations

4.2.1 Traffic Curve and Traffic Collapse

A transportation network is a collection of *segments* or *links*, where a segment/link consists of a set of geographic coordinates representing a polyline, and a collection of *observations*. Observations are a family of average speeds along and/or travel time across the polyline at an instance in time: $\{(x, y)_t\}_{t\in\mathbb{N}}$, where x and y are speed and travel time, respectively. For convenience, travel time is dropped from the formalization, allowing the speed at time t for a given segment x to be denoted as simply x_t. Every finite stretch of road, a *link*, can be associated with a *traffic density*, or the fraction of the link capacity that is occupied by vehicles, at a given time. This may equivalently be expressed by the *buffer size or buffer capacity* (B_ℓ), which is the number of vehicles in the link. The *exit rate* or *exit capacity* (C_ℓ) of a link is defined as the number of vehicles exiting the link per unit time. The traffic curve captures the variation between these two parameters [Jain et al. 2012]. At high traffic densities (indicating traffic jams), links have very low operational exit capacities and at low densities, the exit rate varies linearly with the density (figure 6.1). We define the optimal operating points of a traffic curve based on *optimal exit rate* C^*_ℓ where the exit rate is the highest and the corresponding B^*_ℓ, the *optimal buffer size* at C^*_ℓ. Based on the traffic curve, one can define the maximum exit rate of a link as a function of the current buffer capacity as shown in Figure 6.1. The maximum exit rate is the maximum number of vehicles that can exit the buffer per-unit time, $C_\ell(B_\ell(t))$.

Now, consider the case where the input rate is larger than the optimal exit rate for a short time period, causing the link buffer to grow. Once the buffer size increases beyond the optimal value B^*_ℓ, the exit rate begins to decrease, leading to a more rapid increase of the buffer size, which further perpetuates the cycle, until a point is reached when the buffer is full and the exit rate is at its lowest possible value. This is called a *traffic collapse*. A very common real world example is

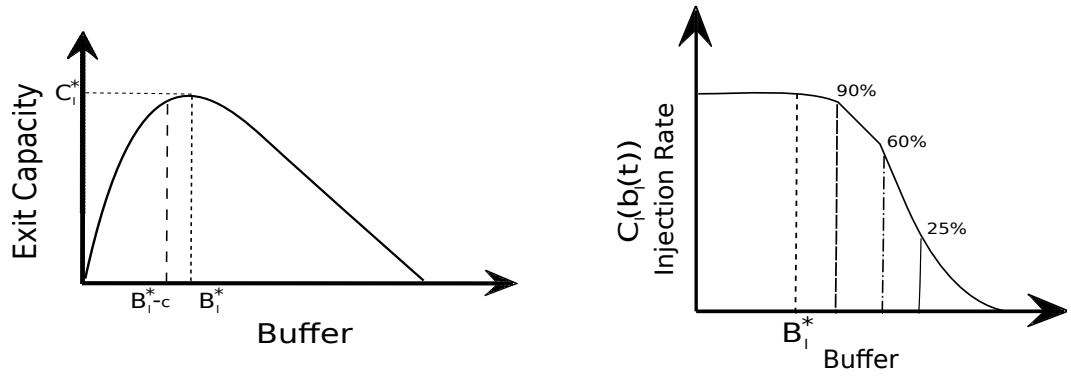


Figure 4.3: Traffic curves showing the instantaneous exit rate (left) and the maximum exit rate (right) as functions of the buffer size

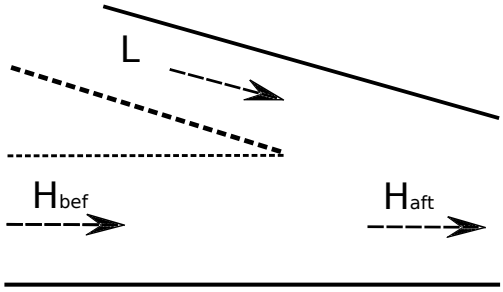


Figure 4.4: Merging of two freeways

two freeways merging into a single freeway. A simple example is illustrated in figure 4.4 where vehicles in L are merging with the stream of vehicles on H. This simple example can be viewed in two ways: two lanes in the same freeway merging into a single lane or two separate freeways merging or a single lane merging into a freeway. To visualize this problem from the perspective of traffic curves, consider three links in the setup: (a) H_{bef} representing, a small segment of H (covering a short distance of up to 0.5 miles) before the merge point; (b) a small segment L before the merge point; (c) H_{aft}, representing a small segment of H after the merge point. Each of the links can be associated with their corresponding traffic curves. Since we are dealing with a discrete version approximation using traffic curve, we should choose reasonable lengths to have meaningful buffer values for the links. The above traffic merging can be viewed using a simple 3-link topology where H_{bef} and L merge into H_{aft}, and each segment has an associated traffic curve. We primarily concentrate on two specific parameters of H_{aft}:$C_l^*(H_{aft})$ and $B_l^*(H_{aft})$. If the

sum total of the exit rates of H_{bef} and L is always less than the optimal exit rate $C_l^*(H_{aft})$, then the merging never faces a congestion problem. If, however, the sum of the input rates of L and H_{bef} is larger than $C_l^*(H_{aft})$, then the buffer size of H_{aft} grows. If the buffer of H_{aft} grows beyond $B_l^*(H_{aft})$, then the exit rate of H_{aft} begins to drop thereby, triggering the spiraling effect.

4.2.2 Identifying Sudden Jams

The phenomenon of traffic collapse as defined in the previous section leads to a *traffic jam*, which is a prolonged state of very slow movement of vehicles on the road. A road segment is said to be in a state of *complete jam* when the density is maximum and the average speed of vehicles in the segment is 0 i.e. the vehicles have come to a standstill. Above a certain threshold of density, or equivalently, below a certain threshold of the average speed of the flow of vehicles in the segment, the segment can be said to be *approaching* a jam. The choice of either threshold is based on the range of possible speeds in the segment. We define *sudden jam* qualitatively as the state when we approach a jam quickly. That is, if the traffic collapse happens over a very short time period (typically within a matter of minutes), then we call it a sudden jam. This can happen due to a rapid build-up in the buffer size, in turn resulting in a rapid drop in vehicle speeds in the segment. Sudden jams are very common in congested cities all over the world [Jain et al. 2012].

Formally, a sudden jam may be defined as follows. Consider a point in time t and in segment i, and two intervals around t, m and n where $m < n$. The *prediction* window is the interval $(t, t+m)$, and the *target* window is $[t+m, t+n]$. Further, let the *observation* window be an interval prior to t that is the same size as the target window: $[t-(n-m), t]$. A *sudden jam* at time t is a condition in which acceleration between the target and observation windows is less-than, or equal-to, some

threshold α. Equation 4.1 shows the mathematical definition of the sudden jam function.

$$s_t(x, m, n) = \begin{cases} 1 & \text{if } \frac{1}{(m-t)(n-m)} \left(\sum_{i=t+m}^{t+n} x_i - \sum_{i=t-(n-m)}^{t} x_i \right) \leq \alpha \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Essentially, there is a sudden jam if the average speed of the observation window differs from the average speed of the target window, hence the division by the window size $n - m$. The additional factor in the denominator, $m - t$, is the size of the target window, converting the derivation to one of acceleration, and allowing α to be expressed as gravitational units. This not only allows the metric to be consistent with the literature on sudden braking events [Harbluk et al. 2007; Simons-Morton et al. 2009], but removes biases toward segments with faster free-flows. By "normalizing" with respect to deceleration, the impact that slow down has on the passenger remains relatively constant. We apply this definition in our New York dataset, where we have speed data from very selected road segments, especially freeways, bridges and tunnels, every minute, collected using loop detector instrumentation, from the local transportation department (NYC DOT) [New York City Department of Transport]. The results are shown in the next section §5.4.

4.2.3 Estimating speed-density curve

In practice, sudden jams happen over very short timescales, such as within 5 minutes [Jain et al. 2012], and we might not always have speed or density data at that fine granularity in order to be able to analyze them. In such situations, we cannot apply the formula from equation 4.1. Additionally, we might not have information about the actual vehicle density on the road. In such cases, we can make certain reasonable assumptions and estimate the speed-density curve from the speed data alone, from which we can identify key threshold points for determining sudden jams. We employ this approach to show evidence of sudden jams in our Nairobi dataset, since we have speed data in Nairobi only every hour. We are able to identify potential sudden jams by

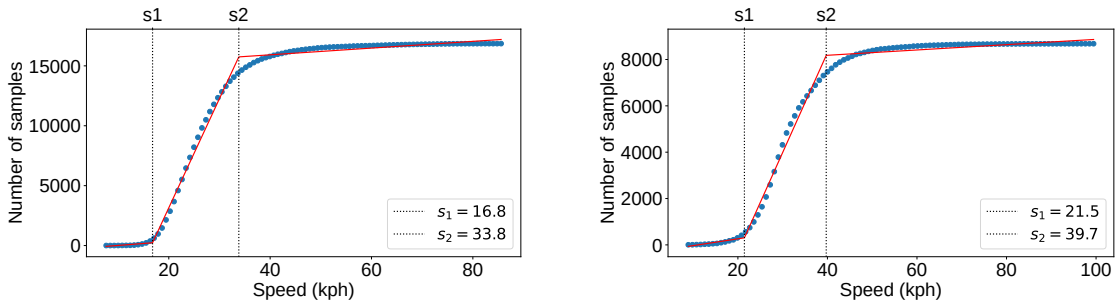


Figure 4.5: Examples of most segments (about 2600 out of the selected 2903) where there were two clear break points in the CDF of observed speeds. We named these break points as s_1 and s_2. The s_1 speed would be approximately the point at which the traffic crosses the "threshold" density for a jam. These break points were obtained by fitting a piecewise linear model to the CDF function.

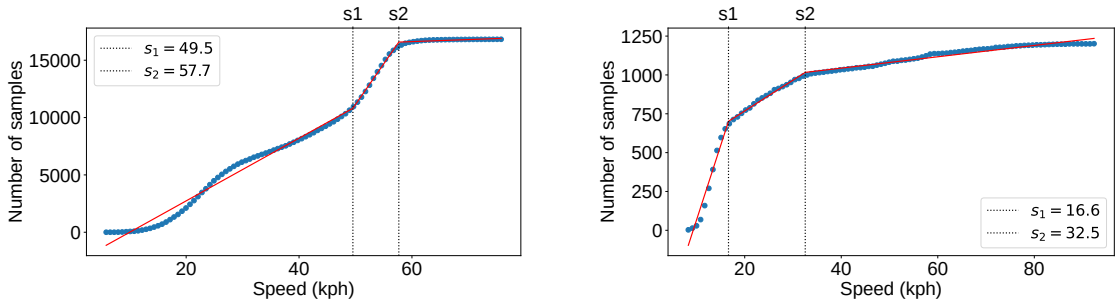


Figure 4.6: Examples of some segments (about 300) where the speed CDFs were different. There were no two clear break points, hence not a well-defined threshold density to define a jam. Note that this is *not* an artifact of the amount of available data in these segments – some of these segments had even more data available than those in the first set.

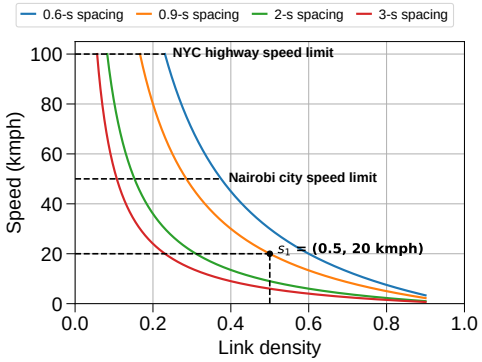
defining a suitable threshold on the vehicle speed in a segment and observing any sudden drops in the segment.

Now we describe a methodology to compute the speed-density relationship, as in figure 6.1, from observed hourly speed data. We begin by making a few assumptions. The first is that drivers try to keep moving as fast as possible at all times, keeping with the pace of the rest of the traffic, while driving safely and not exceeding the speed limit. There is typically a rule that is taught to student drivers, called the "3-second rule", which means that every driver should maintain a certain distance to the vehicle ahead such that it takes 3 seconds to cover that distance at the current speed of traffic. Based on a study conducted on typically driver reaction times [Chen

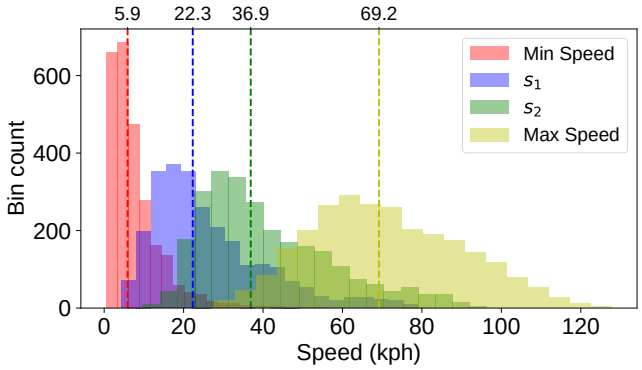
and Wang 2007], the absolute minimum spacing required between vehicles is between 0.6 and 0.9 seconds, while the typical spacing is 2 seconds and the recommended spacing is 3 seconds. As recommended by the authors in the study, we pick 0.9 seconds as the lower limit, since 0.6 seconds is at the absolute limit for adequate human response to avert an accident. Therefore, we assume that every driver maintains a spacing of 0.9 seconds (on average) to the vehicle ahead. The second assumption is that we make, without a loss of generality, is that the traffic is only composed of cars.

The *jam density* is the upper limit on free-flowing traffic density, beyond which the road segment is said to be in the state of a jam [May 1990]. Although there is a bit of variation on this threshold density in literature [Knoop and Daamen 2017; Wu 2002], the theory according to May [1990] gives a value for jam density of 185 to 210 vehicles per kilometer per lane, which translates to between 51% and 58% lane occupancy, assuming an average car length of 4.5 meters. We assume the threshold for a jam to be 50% occupancy of the road segment. This corresponds to a spacing of a single car between every two cars. If every driver were to maintain 0.9 second spacing, the maximum possible speed at 50% density is 20 kph. If every driver instead maintained 2 or 3 second spacing, the corresponding speeds at the same density would be 9 kph and 6 kph, respectively. We thus plot the max possible speed (assumed to be equal to the max exit rate) as a function of density, for each of these cases – 0.6-second spacing, 0.9-second spacing, 2-second spacing and 3-second spacing (figure 4.7(a)). If the density exceeds 50%, then we are said to be in a jam.

In reality, the perception of a "jam" would depend on the particular road segment. On a highway, where speed limits may normally be upwards of 100 kph, drivers may perceive the state of driving at less than, say even 40 kph, to be in a state of a jam. Whereas on a city road, the limit is much lower. For most well-behaved segments, this limit would be approximately the first break point in a cumulative distribution of the observed speeds, denoted as s_1, as shown in figures 4.5. We deduce this s_1 for each segment as follows. Given the hourly speed data in a road segment



(a) Estimated speed-density plot assuming, without significant loss of generality, that roads contain only cars and every car is 5 m long. Based on our assumptions, we expect a jam to occur when the average speeds drop to below 20 kph for most roads, and this corresponds with our observed distribution of s_1.

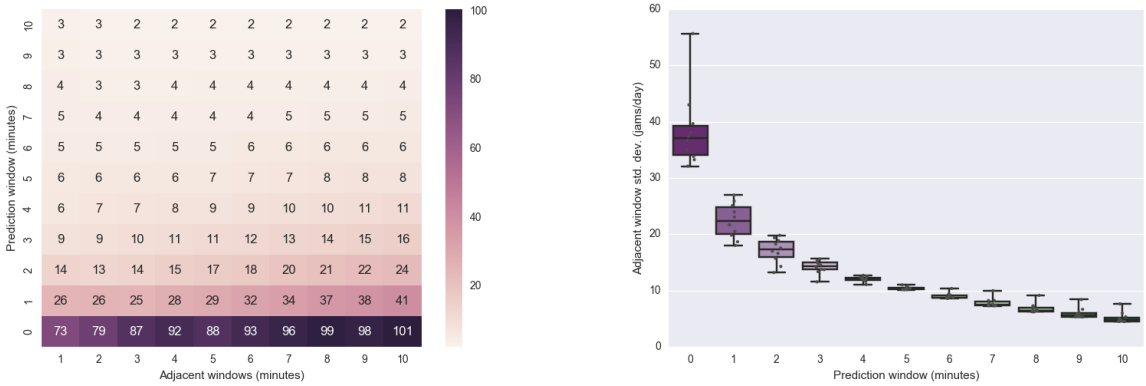


(b) Distribution of the maximum, minimum, s_1 and s_2 speeds across the chosen segments. The medians for each of the four quantities are shown at the top of the plot. We observe that the median value of s_1 corresponds to the expected value of 20-25 kph from our empirical calculations. In practice, it would be even lower, as can be seen from the *mode* of the distribution of s_1, which is less than 20 kph.

Figure 4.7: Estimation of the speed-density curve and the jam threshold. *(use color in print)*

over a period of time, we first plot a distribution of all the measured/observed average vehicle speeds as a cumulative density, which shows the distribution of speeds across all the segments in Nairobi observed over the entire time period of two years. We then fit a 3-way piecewise linear function to this CDF using the *pwlf* library in Python [Jekel and Venter 2019] to obtain the two break points, s_1 and s_2. Figure 4.2 shows the distribution of the minimum, maximum, s_1 and s_2 speeds for 2590 segments which are "well-behaved", in the sense of having two clear break points. We note that the median value of s_1 is about 20 kph, corresponding to the speed at the jam density with 0.9 inter-vehicle spacing. For most urban roads in Nairobi County, the speed limit is 50 kph [Ndubi 2019], while it is higher on some highways, and we observe this in the distribution of the maximum speeds.

There were about 300 segments which were not well-behaved in this sense and displayed speed distributions that were very different. Two examples are shown in figure 4.6. For these segments, we assumed a jam density speed of 20 kph, since we are unable to compute the s_1



(a) Average number of jams, per day, per segment, across varying prediction and target windows. Observation and target windows were held equal, denoted here simply as "adjacent".

(b) Average standard deviation of per day, per segment jam counts

Figure 4.8: Sudden jam characterization across all segments in the network

from the distribution. We note that this is not an artifact of the available data, as there were both segments with more than 10000 data points in two years that did not show well-behaved CDF and segments with just 200 data points that showed two clear break points.

4.3 Results

4.3.1 New York City

4.3.1.1 Sudden jams in practice

When characterizing sudden jams, signals with average reporting rates of 90 seconds or less were used. Further, segments had to have a polyline component. These two conditions reduced the total set of 153 segments to 98. Finally, to be eligible for jam classification, the observation and prediction components of a given window cannot contain missing data.

Recall that a window consists of three components—observation, prediction, and target. A window is considered a sudden traffic jam if the average speed between the observation and

target portions decreases by more than a particular rate. "Prediction window" is thus the time between the observation and target portions. Figure 4.8 presents an overview of frequency using a fixed $\alpha = -0.002$, and window sizes varying from 1 to 10 minutes. The two window portions—observation and target—remained equal throughout. As such, they are denoted simply as "adjacent." The frequency of sudden jams depends not only on the characteristics of the segment, but on parameters to Equation 4.1; in particular, the size of the window and the choice of α.

Figure 4.8(a) outlines the inverse relationship between sudden jam frequency and respective window sizes on a sample segment. Prediction windows range from 0 to 10, where 0 signifies a comparison between adjacent points in time. The adjacent windows range from 1 to 10, as a lower bound of 0 would mean a comparison between non-existent windows. Darker cells denote higher occurrences, with each cell annotated with the number of occurrences. The number of sudden jams increases as the size of the adjacent window increases and as the size of the prediction window decreases. The increase across prediction windows is exponential, while the increase across the adjacent windows is linear. With respect to the adjacent windows, this relationship is likely due to variability in average speed. Using observation and prediction windows of size one means that the model is taking into consideration "unsmoothed" values. In some cases, these observations may not be entirely representative of actual conditions; a result of measurement anomalies, for example. As window sizes increase, spurious values can augment average speed such that a sudden jam is harder to determine. With respect to prediction windows, the inverse relationship is largely a result of the manner in which sudden traffic is calculated: larger prediction windows require a larger decrease in speed between the observed and target windows to be classified as traffic events.

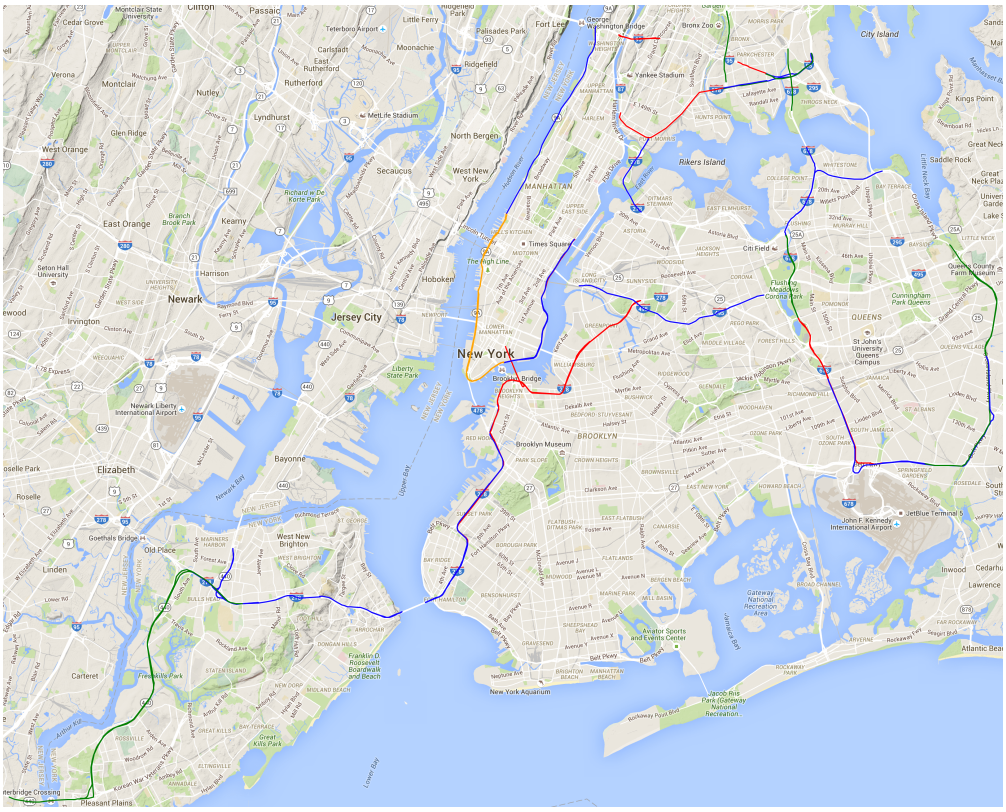
Figure 4.8(b) presents the average standard deviation at varying adjacent window sizes across all segments considered. The large number of traffic incidents shown in Figure 4.8(a) for smaller adjacent window segments is dominated by a relatively small number of road segments. Measurements across mid to large windows, while less in number, are spread more evenly.

4.3.1.2 Characterization of Segments

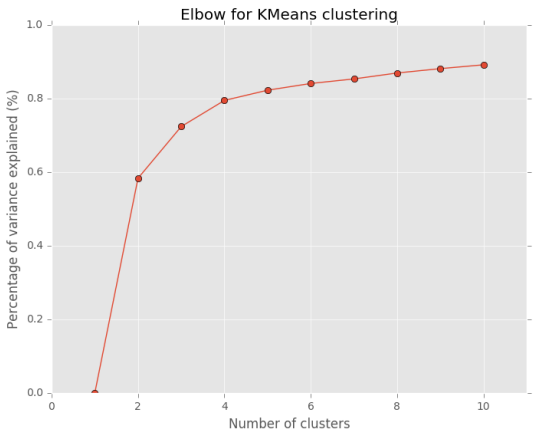
Each road segment exhibits a unique behavior with respect to average speed and the variability of those averages. This signature can be described by grouping measurements into their respective hour of the day. Thus, each hour represents its own distribution. To formalize segment classifications in this way, each segment is described as a vector of 48 elements, where each pair of elements represents the median speed and the standard deviation of that speed, for the respective hour. Then, we performed an unsupervised clustering of the segments. Figure 4.9 outlines the results of such clustering using the k-means method. Figure 4.9(b) shows the "elbow" curve for various amounts of clustering—four *seems* best based on the elbow plot[3] Figure 4.9(c) shows the distribution of segments when using four clusters. The segments are plotted using their average daily speed versus the average standard deviation of those daily speeds. Finally, Figure 4.9(a) shows the position of each clustered segment on the New York City map. Segments that are not operational are not displayed.

To get an idea of what the speed signatures across clustered segments, Figure 4.10 shows an example segment from each cluster. Clusters 2 and 3 have low speed variation, but their average speeds are near seemingly respective free flows. Some segments in these clusters are – the lower half of the Henry Hudson Parkway in Manhattan and the West Shore Expressway in Staten Island. Clusters 0 and 1 show high variance—potentially good for sudden jams. There are more segments in these clusters – Brooklyn-Queens Expressway, Verrazano-Narrows Bridge, Van-Wyck Expressway along with other bridges and tunnels in New York. These clusters are characterized by lower average speeds and high variance, indicating a higher frequency of sudden jams.

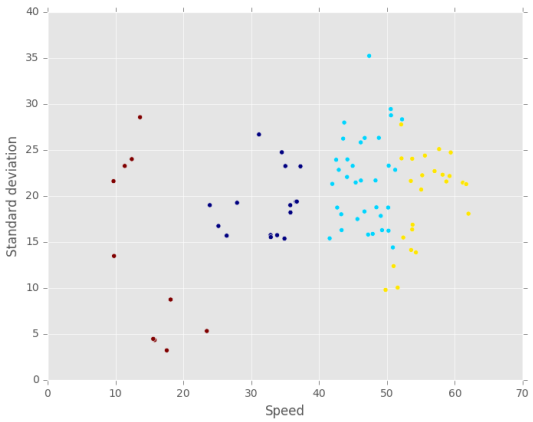
[3] But this could be highly dependent on what we are doing with these clusters.



(a) Operational road segments clustered by color. Cluster 0 corresponds to red, Cluster 1 to blue, Cluster 2 to green, and Cluster 3 to orange. *(use color in print)*

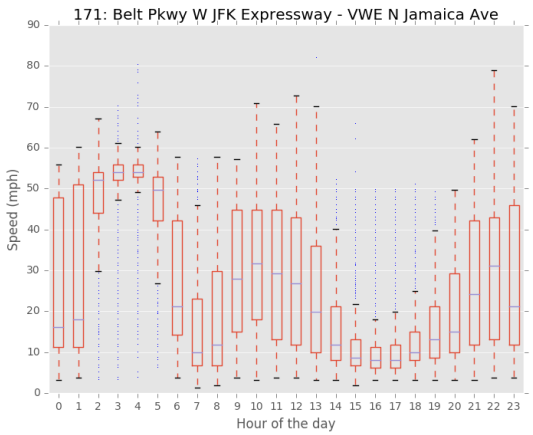


(b) Elbow curve to determine the number of suitable clusters.

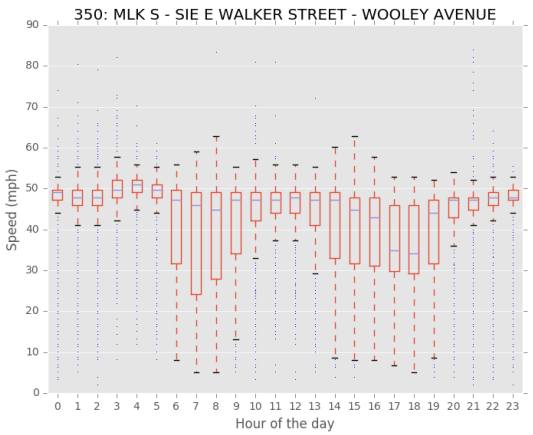


(c) Distribution of road segments. Segments were reduced from 48 dimensions to two for plotting purposes. *(use color in print)*

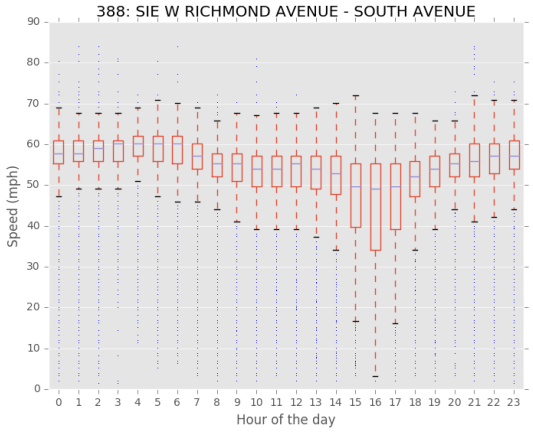
Figure 4.9: Cluster outline.



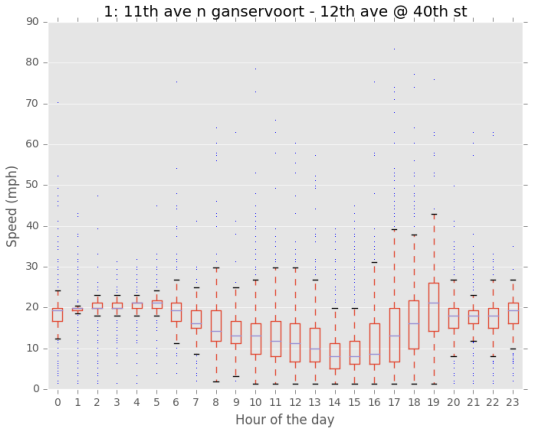
(a) Cluster 0: Segments with moderate to high average free-flow speeds and large changes in those free flow speeds throughout the day.



(b) Cluster 1: Segments with moderate to high average free-flow speeds, with changes to free flow speeds at some point during the day that are not as significant as segments in Cluster 0.



(c) Cluster 2: Segments exhibiting higher average speeds, with relatively little variation throughout the day.



(d) Cluster 3: Segments with low average speeds and exhibit little variation.

Figure 4.10: Cluster examples.

Way ID	Total hours in jams	Max hours in a jam	Mean hours in a jam	Mode hours in a jam	Mode hour of day
555497908	417	7	2.6	1.0	16:00
9931279	2246	10	2.3	1.0	06:00
678371493	1559	11	2.0	1.0	15:00
364279376	844	6	1.8	1.0	16:00
678371494	1521	6	1.8	1.0	16:00
4724017	2677	10	2.5	1.0	16:00
336067605	1842	10	2.7	1.0	17:00
39573541	688	10	1.4	1.0	15:00
580233744	715	5	1.5	1.0	17:00
4742016	1237	10	2.0	1.0	17:00

Table 4.1: Sample of 10 segments from the 6 junctions, showing total hours in jams, the max number of hours for which a jam has lasted, the mean hours for which a jam has lasted, the mode hours (most commonly observed jam duration) and the mode hour of day (most commonly observed time of day when jam occurs).

4.3.2 Nairobi

4.3.2.1 Sudden jams in practice

Applying the s_1 threshold on every one of the 2903 valid segments in the city, we observe that the total number of hours spent in jams by all the traffic in the city over the two-year period is about 2188201 hours, which translates to about 3000 hours per day, summed across all the segments. Per segment, the average number of hours spent in jams is about 1.2 hours per day (Figure 4.11), and the maximum is about 8 hours at a time. Figure 4.12(a) shows the mode[4] number of hours for which a jam lasts across the whole dataset. We notice that for the most part, jams last about 1 hour or less (about 2300 instances), but the next highest number of instances is for jams lasting 4 hours (about 200 instances). Figure 4.12(b) shows the distribution of number of hours of jam over times of a day. As one would expect, this is a bimodal distribution, with a spike during the morning rush traffic and another spike in the evening rush. We do observe longer jams in the evening though, based on the data.

We define three different phases of traffic based on our observations – *Phase 0:* heavy traffic, speed $< s_1$; *Phase 1:* light traffic, $s_1 <$ speed $< s_2$; *Phase 2:* no traffic, speed $> s_2$. Table 4.1 shows

[4]the most frequently occurring observation in a dataset; this may not be unique

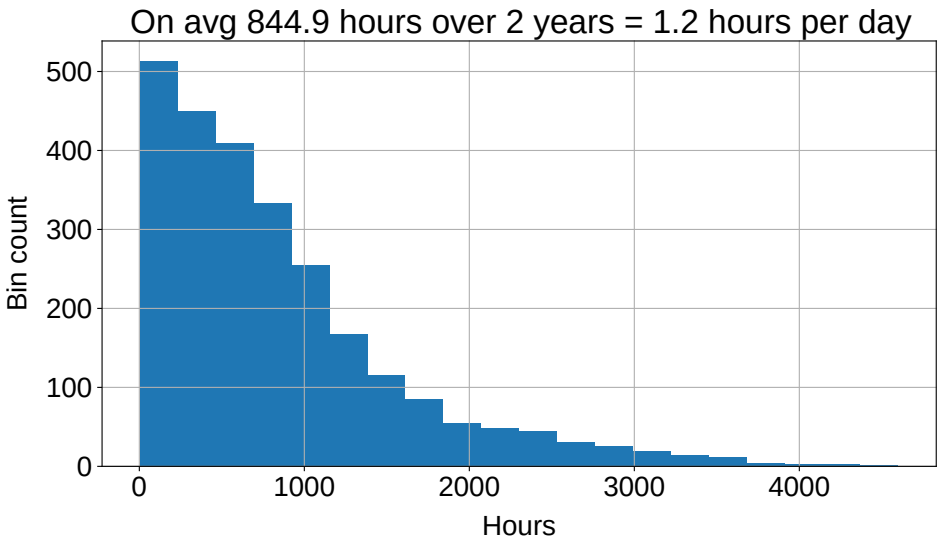
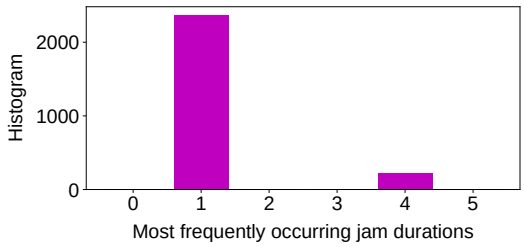
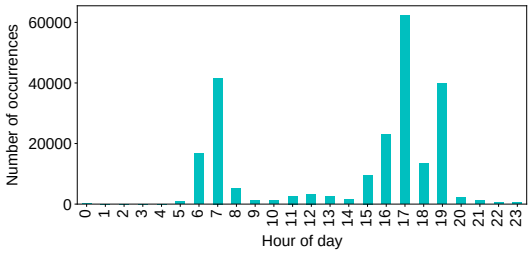


Figure 4.11: Total number of hours spent by the traffic in Nairobi in jams, when jam is defined as a drop in average vehicle speed below s_1



(a) Most frequently observed jam duration across all the segments. For each segment, we applied the condition that speed less than s_1 indicates a jam, made a list of all jams (in terms of how long was the jam in hours), computed the modes of the list and then finally plotted the counts of each mode value. Hence, this plot shows, for example, that for most segments (approx 2200), the mode number of hours in jam is 1 hour. The next highest is 4 hours for about 200 segments.



(b) Total number of hours in jam across the city when the jam begins at different times of the day. As one would expect, this is a bimodal distribution, with a spike during the morning rush traffic and another spike in the evening rush. We do observe longer jams in the evening though, based on the data.

Figure 4.12: How long do the jams last in Nairobi and what times do they occur?

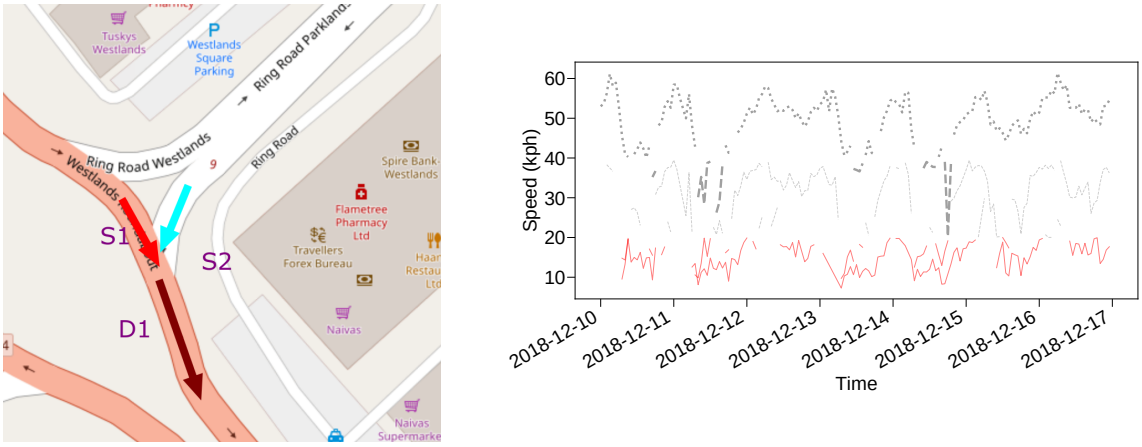


Figure 4.13: Junction A, a 2-1 merge *(use color in print)*

statistics for six representative junctions in the city, covering a variety of settings – T-junctions in the city, highway merges and roundabouts, and figures 4.13 to 4.18 show some sample snapshots for the six segments on various days. The faint lines show the input segments in to the output segment(s) of interest, which are shown with thicker lines. The red color shows a jam, which is when the speed in a segment drops below the s_1 for that segment. While the low and medium congestion clusters do not exhibit many instances of sudden jams, we observe numerous instances in the high congestion cluster, and also jams that last for several hours at a time. The snapshots show how the average speeds in the segments vary through the course of the day.[5] We note large number of instances of prolonged congestion (i.e. jam) in the sink segment in junctions B, D and E. In the roundabout junctions C and F, we observe jams at one or two output sink segments. We observe that in many cases, the jams persist over 2 to 3 hours. In the junctions B, D and E particularly, which are all 2-1 merges, we observe that the average speed in the output segment is below the respective s_1 continuously for more than several hours on multiple days.

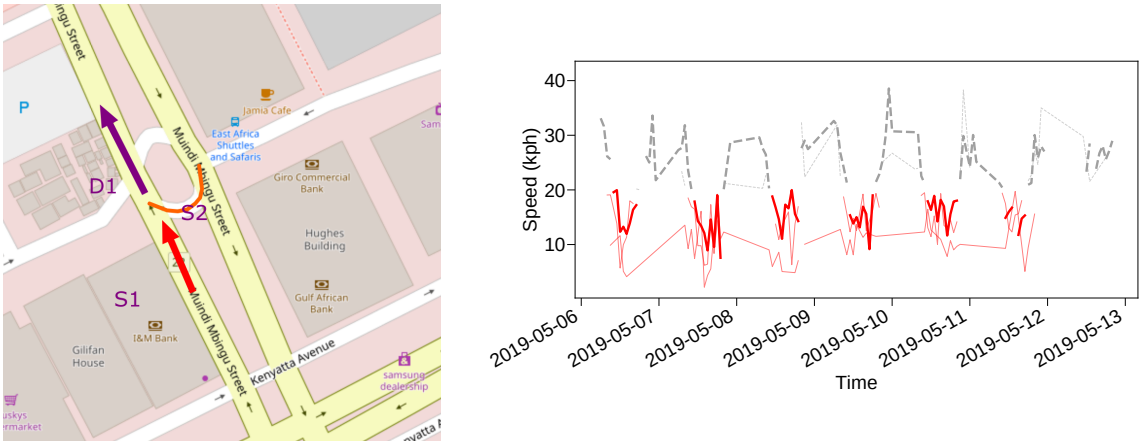


Figure 4.14: Junction B, a 2-1 merge at a U-turn *(use color in print)*

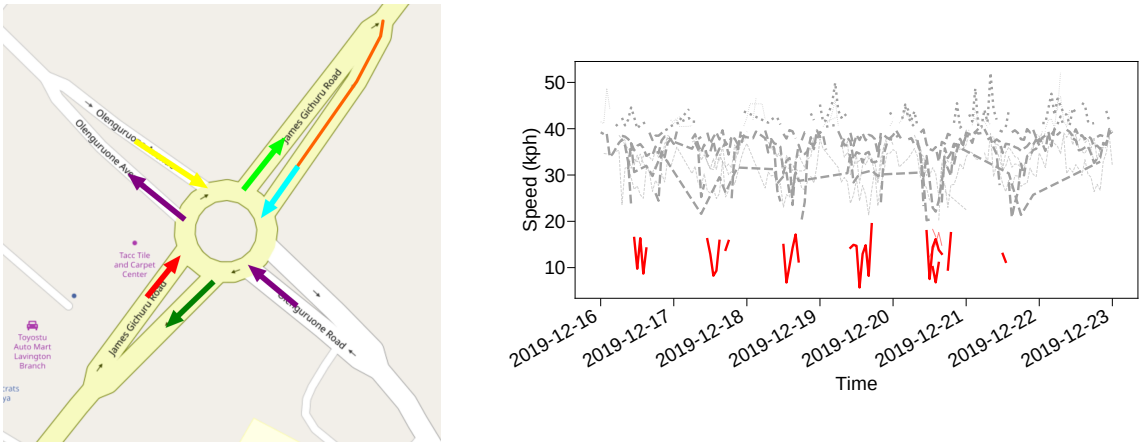


Figure 4.15: Junction C, a roundabout with 4 sources and 4 sinks *(use color in print)*

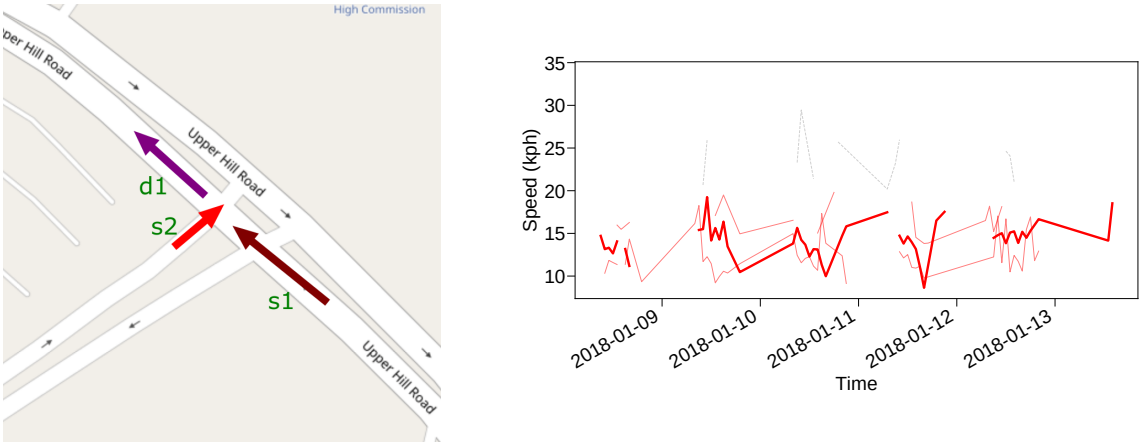


Figure 4.16: Junction D, a 2-1 merge at a T-junction *(use color in print)*

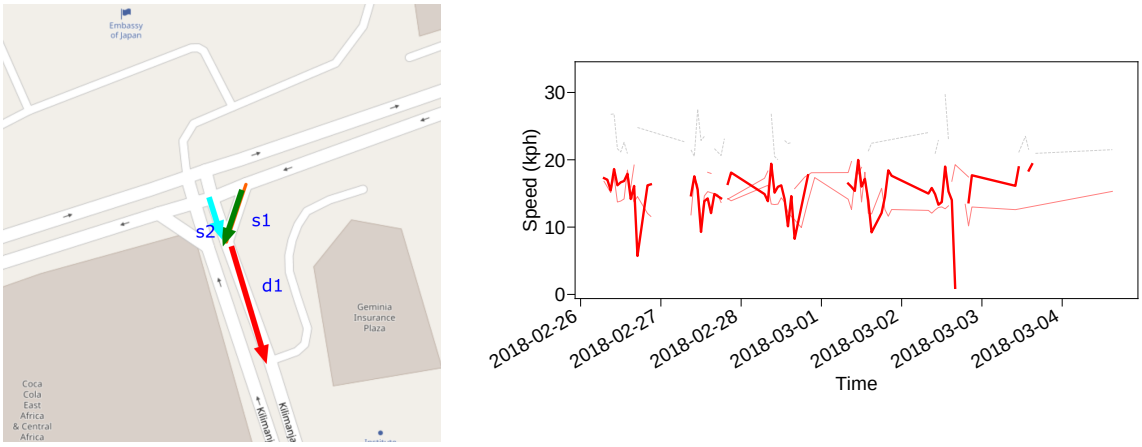


Figure 4.17: Junction E, a 2-1 merge at another T-junction *(use color in print)*

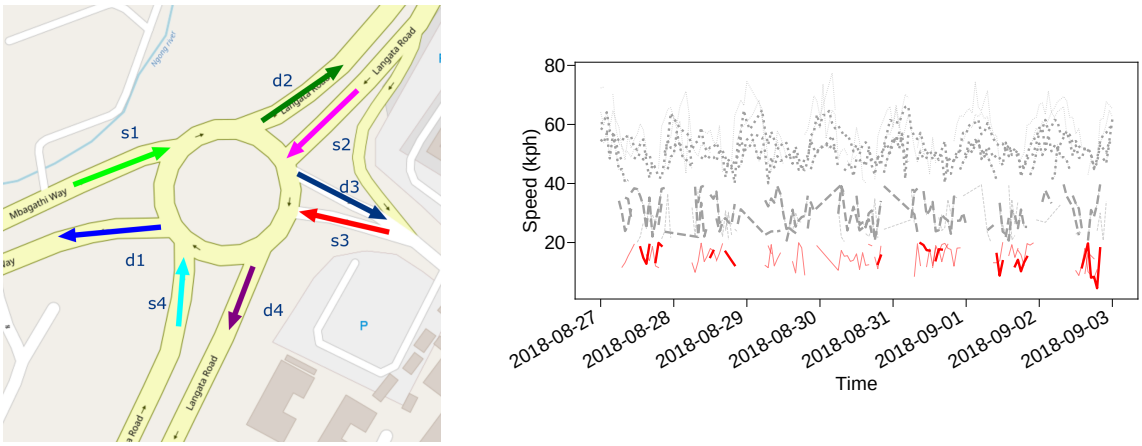
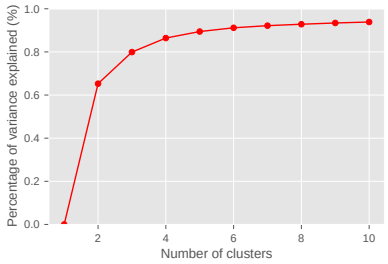
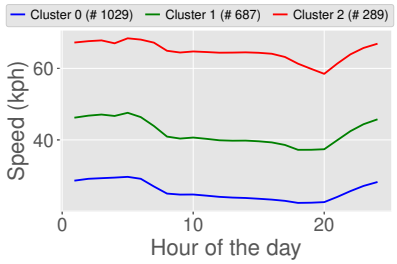


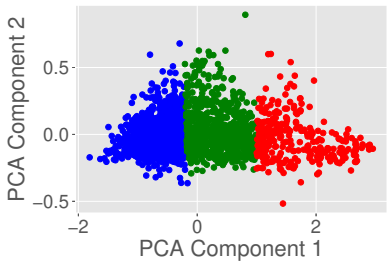
Figure 4.18: Junction F, another roundabout with 4 sources and 4 sinks *(use color in print)*



(a) Elbow plot of the clustering of road segments in Nairobi



(b) Centroids of the high, medium and low speed clusters



(c) Scatter plot of the top two components after a Principal Components Analysis (PCA) of all the 4817 segments

Figure 4.19: Clustering of the road segments using 48-dimensional feature vectors consisting of mean and standard deviation of speeds every hour, for 24 hours. The spread across the primary PCA component is quite significant, that it makes sense to cluster the segments based on average observed speed. The elbow for the clustering is at 3, shown in the first figure. The next two figures show the centroids and the scatter over the 2-D space of the top two principal components. *(better use color in print)*

4.3.2.2 Characterization of Segments

If the small number of segments in New York fall into 4 different clusters, then we should be able to observe a similar type of behavior in Nairobi as well. We performed an unsupervised clustering similar to what was done in New York. We averaged the segment speeds across all the segments over the two-year period, constructed a 48-dimensional feature vector for each segment ("way" in OpenStreetMap) consisting of average and standard deviations of the observed speeds at each hour. We only considered those segments that had at least one data point for each hour of the day in the two-year period. Out of 4817 segments, only 2005 satisfied this condition, which became our dataset. When we did a Principal Components Analysis (PCA) and plotted the top two most significant components, we observed a small clustering, especially at the higher end of the speed distribution, indicated in figure 4.19(c). To bring out these clusters, we performed k-Means clustering and figure 4.19(a) shows the elbow plot for various values of k. We see that the elbow is at 3, compared to 4 for New York. These correspond to low-, medium- and high-

[5]For the sake of brevity, we only show handpicked snapshots for each of the junctions that best illustrate the different phases. We will be happy to share our analyzed data in greater detail as needed for reproduction of our results.

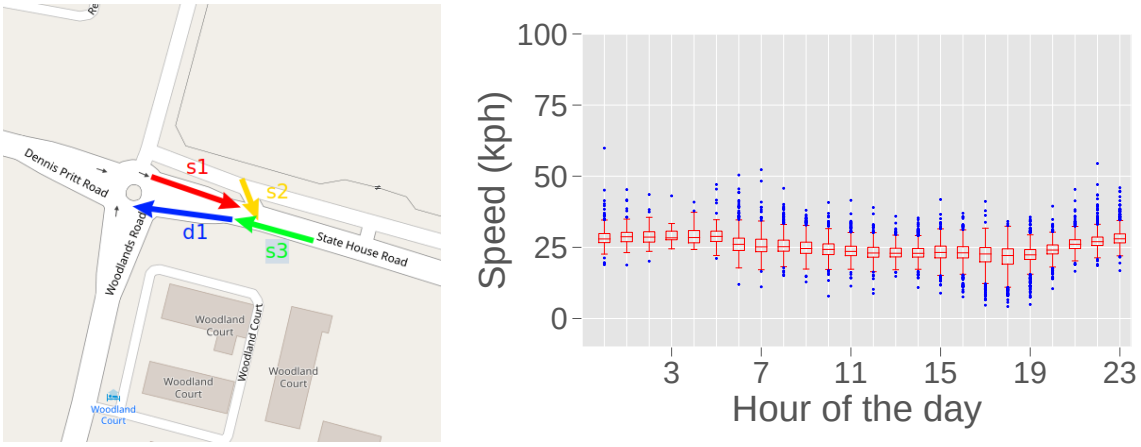


Figure 4.20: High congestion cluster – State House Road (a roundabout with three other input segments)

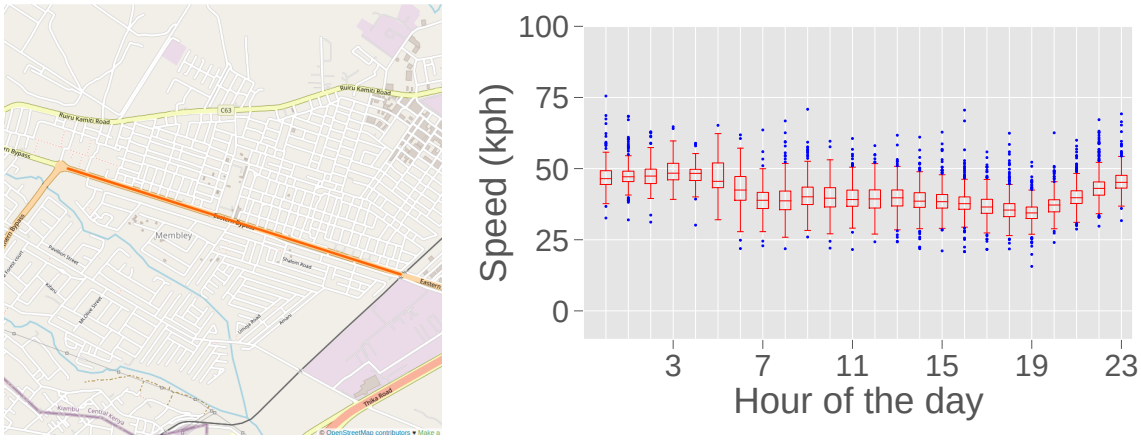


Figure 4.21: Medium congestion cluster – Eastern Bypass (a long arterial road with about 30 "tributary/distributary" roads that enter/exit the main road)

congestion segments. The three clusters centroids are shown in figure 4.19(b). It is interesting to note that up to 1029 segments (20%) are in the low speed cluster, with average speeds of less than 20 kph through the entire day, going down as low as even 5 kph. This is a significant number of segments.

To understand why we observe this, we investigated the segments closest to the cluster centroids, as shown in figures 4.20 to 4.22. The figure shows, for three distinct types of segments, the location on the map and the box-whisker plot of the speed distribution through the entire two-year period. The high congestion cluster is a roundabout at the intersection of Dennis Pritt

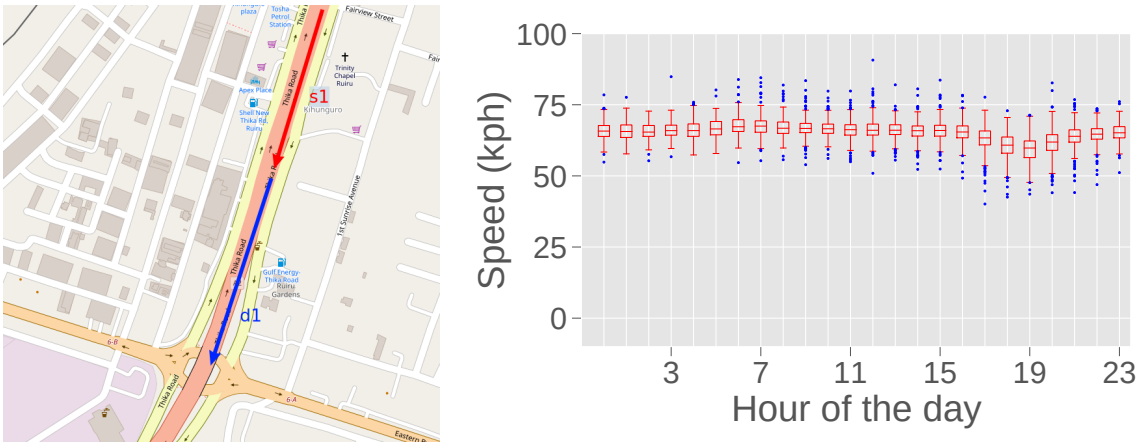
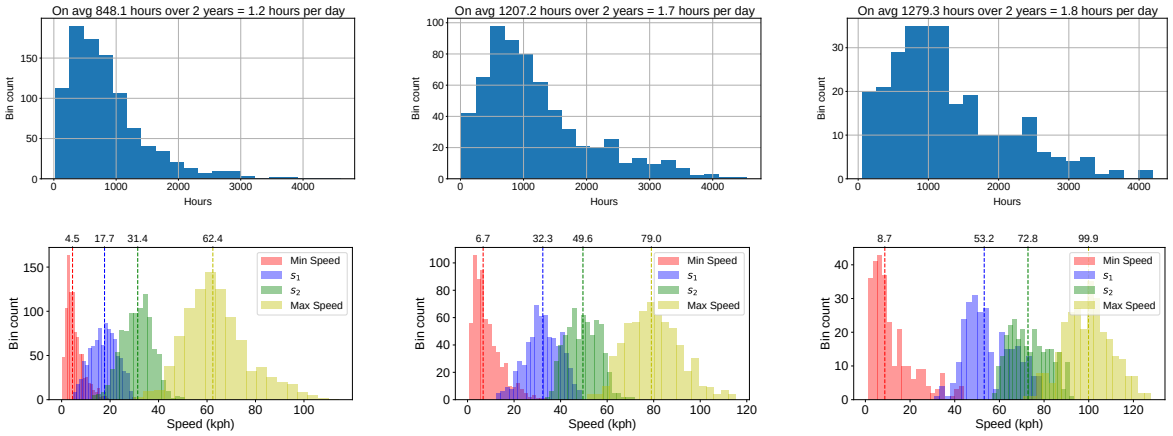


Figure 4.22: Low congestion cluster – Thika Road / Eastern Bypass (a limited access highway)

Road and Woodlands Road, with median speed around 25 kph. The medium congestion cluster is a section of the Eastern Bypass road, an arterial road in the city, with median speed between 45-50 kph. This section is interesting because although the Eastern Bypass is a wide arterial road, the presence of many "tributary" and "distributary" style small roads entering and exiting the main road affects the traffic in the main road. Each of these input road segments would be accounted for in the traffic curve for the Eastern Bypass road, thereby bringing down the overall movement speed. And the low congestion cluster is Thika Road, possibly a highway, from the access restrictions that we can observe on the map, with a median speed of 70 kph.

The clusters are marked by differences such as – the number of hours for which sudden jams last and the time of the day at which the jams occur. Figure 4.23 shows the cluster-wise differences in the statistics, including of the break points in the speed distributions. Quite surprisingly, we observe *more hours* of jams in the high speed cluster than in the lower speed clusters. Across the entire two-year period, the traffic in Nairobi spends about 845 hours stuck in jams in every segment on average, which is about 1.2 hours per segment. When divided across clusters, the numbers change a bit. For the lowest speed clusters, the statistic is still the same. However, for the medium and high speed clusters, the numbers are 1207 hours and 1280 hours respectively, which translates to about 1.7 hours per day and 1.8 hours per day respectively. This is because

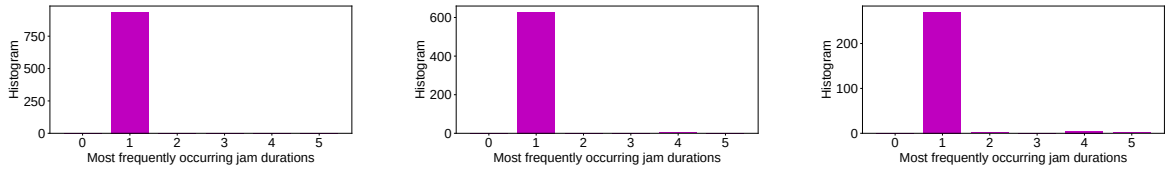


(a) Low speed cluster **(b)** Medium speed cluster **(c)** High speed cluster

Figure 4.23: Variations in the statistics across clusters due to variations in the s_1 threshold. Notice the variations in the s_1 and maximum speeds in the segments in these clusters. *(use color in print)*

of the variation in the s_1 threshold – for larger roads that have higher movement speeds, this threshold is a little higher than for the smaller roads. For traffic that normally moves at 100 kph, a slowdown to even 50 kph would be a "jam" since this could dramatically alter travel time predictions.

The clusters also show marked differences in terms of the most frequently observed jam duration, and the times of day when jams occur. Although across the board, most jams last about 1 hour, little less or little more, cluster 1 and 2 have a greater proportion of jams lasting 3, 4 and 5 hours. This seems to suggest that cluster 0, the high-congestion cluster, does not show as high a proportion of instances of sudden jams which last for several hours, in comparison to clusters 1 and 2, the medium- and high-speed clusters. Cluster 0 shows only 5 instances of jams lasting 4 hours for 936 instances of jams lasting 1 hour. In comparison, clusters 1 and 2 show 5 instances each of jams lasting 4 hours, for 628 and 270 instances of jams lasting 1 hour, respectively. Clusters 1 and 2 also have greater proportions of jams lasting 2, 3 and 5 hours. In terms of the hour of day, all of them show bimodal distributions. Clusters 0 and 1 show similar profiles, with the highest number of instances at 5 pm, whereas cluster 2 shows more jams at 7 pm. Cluster 1, the medium cluster, also has a sizable number of instances at 7 am, possibly indicating the class of

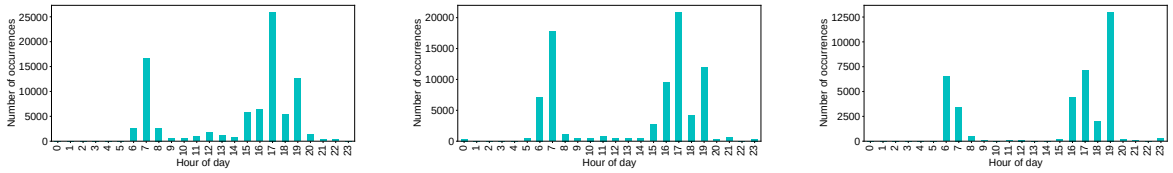


(a) Cluster 0 (low-speed / high-congestion cluster)

(b) Cluster 1 (medium-speed / medium-congestion cluster)

(c) Cluster 2 (high-speed / low-congestion cluster)

Figure 4.24: How long do the jams last in Nairobi in each type of cluster? Distribution of the mode of number of hours (in each segment) for which jams last for the different clusters.



(a) Cluster 0 (low-speed / high-congestion cluster)

(b) Cluster 1 (medium-speed / medium-congestion cluster)

(c) Cluster 2 (high-speed / low-congestion cluster)

Figure 4.25: At what times do jams occur? Total number of hours in jam across the city when the jam begins at different times of the day for the different clusters.

roads taken by most commuters.

4.4 Discussion and Conclusion

From the previous section, we note many occurrences of jams in various types of road segments, both in New York and Nairobi. Specifically, we observe more instances in jams in highways, as shown in figure 4.24. This is evident from other published works in literature. In one study published in 1996, the authors study traffic jams in the Autobahn in Germany in 1992 and observe that traffic jams can move through highways for long periods of time over long distances [Kerner and Rehborn 1996]. In one case, traffic jam moved for a period of 50 minutes over the entire stretch of 13.1 km of highway, and in another case, an almost stationary moving jam existed on a highway. This shows that highways and other free-flow segments are rather susceptible to jams, and this can be attributed to the sudden bursts of traffic entering the highway, causing the

density to shoot up briefly. And once the threshold density has been crossed, the queue builds up rapidly, resulting in a jam that lasts for a long time. In another paper, the authors have shown that traffic jams on highways are a familiar phenomenon, and explain how jams can occur without any bottleneck, merely due to fluctuations in the vehicle density that can grow out of control and cause a jam that propagates through a highway [Sugiyama et al. 2008].

How do we mitigate these jams? Again, turning to the traffic curve from §4.2.1, the key idea is to be aware of the jam density and the threshold speed in a segment. Real time instrumentation, such as the one in New York, can inform drivers through navigation apps about impending congestion in a segment. Such knowledge can also be used in signaling at important points such as n-1 merges, where we observe the most instances of congestion resulting in prolonged jams.

Thus, to conclude, the key takeaways from our work are the collections of observations about traffic jams – they can appear in various scenarios, for various durations and can potentially result in prolonged congestion of vehicles on the road, on many occasions lasting several hours at a stretch. Sudden jams that appear due to an increase in vehicle density beyond the threshold *jam density* are particular types of jams that last for long periods of time. We have provided a traffic curve formalism for understanding the phenomena of traffic collapse leading up to sudden jams, and a formula for sudden jams in terms of the drop in acceleration. For more sparse data such as in Nairobi, where we do not have minute-on-minute data to use the formula, we have proposed an alternative approach to estimating this exit rate curve (speed-density curve) from the speed data and some basic assumptions. We compute a loose upper limit on the speed at the jam density in a road segment from the speed distribution. This speed, called s_1, the first break point in the CDF distribution, ranges between 10 and 40 kph, with median approx 20 kph. The value of 20 kph also corresponds approximately to the value of the speed at the threshold jam density of 185 to 210 vehicles per km per lane from literature.

Then we turn our attention to the data. Using loop detector data in New York City for about 1.5 year, from November 2014 to April 2016, we observe that sudden jams are extremely frequent,

occurring in the 100s every 10 minutes. In Nairobi, we use Uber movement speed in order to understand traffic jams in Nairobi. We have data over a period of two years from January 2018 to January 2020 at over 4800 road segments. We observe that on an average, vehicles are stuck in jams for about 1.2 hours on every segment every day.

Further, we performed an unsupervised clustering to show that segments may be categorized into multiple types based on the average speeds and variance in speeds. In New York, we observed four clusters, and in Nairobi we observed three. Sudden jams occur frequently in Nairobi too, especially in the 2-1 merge scenarios, many resulting in prolonged jams.

5 | Predicting and Forecasting Traffic

5.1 Related Works

The problem of short-term road traffic forecasting is an area replete with studies, as summarized in an excellent and long review of the area [Vlahogianni et al. 2014], with discussions and references to more than 200 works. [Seo et al. 2017] is also another good review. Traditional approaches to travel time prediction such as ARIMA models and their variants, and Kalman filters, work well to estimate next-step future values in time series, and have been used with some moderate success in short-term traffic flow prediction [Lippi et al. 2013; Liu et al. 2011]. But as more recent works have repeatedly shown [Guo et al. 2019; Lv et al. 2018; Li et al. 2018; Cui et al. 2018; Yu et al. 2018], ARIMA and similar approaches do not model spatial dependencies between connecting road links sufficiently, and thus not do not yield the best predictive performance. The congestion state in a road segment depends strongly on the states upstream as well as downstream. Second, ARIMA methods are extremely poor at long-term forecasting. And third, they assume that the time series data is stationary, which cannot be expected of the real traffic speeds. The most recent works cited here attempt to address many of these issues using different deep neural net architectures, but evaluated on datasets of a different nature viz. taxi traces and loop detector data, that are denser and richer than our dataset. We utilize a simpler architecture that works with sparse datasets such as ours, and yielding performance comparable to [Guo et al. 2019].

5.2 Dataset and Graphical Representation

The Metropolitan Transportation Authority (MTA) in New York City provides a raw datalog of locations reported continually by the MTA buses. The available information is not only sparse in space, but coarse. Each bus reports a timestamp, distance traveled in the trip, and a *stop code* referring to the next bus stop. We define a *segment* as the portion of a bus route between two consecutive bus stops. Data entries are received at an arbitrary interval close to < 1 minute, sometimes observed as low as 30 seconds. We utilized a downloadable historical dump [NYC MTA 2014] of all bus locations over a continuous period of 90 days in 2014 in Manhattan. The data contains information for about 42 different bus routes, covering over 685 bus stops, across the borough of Manhattan.

In the data preparation step, speeds are computed for each individual bus from the distance and timestamp information. The speeds from multiple buses are then aggregated at each segment at 10 minute intervals to create a time series for each segment. As a result of this procedure, traffic speed data is obtained at far greater spatial coverage than what would be obtained from loop detectors, which are usually placed only on arterial and peripheral roads. A **segment graph** is constructed with the sparse data to observe the approximate flow of traffic between bus stops at a point in time. Each node in the graph represents a segment and holds the average speed in a 10 minute interval. It is a directed graph, and there exists an edge between two segments if they share a common bus stop or, equivalently, if they are adjacent to one another. Figure 5.1 shows a graph of the traffic segments in Manhattan.

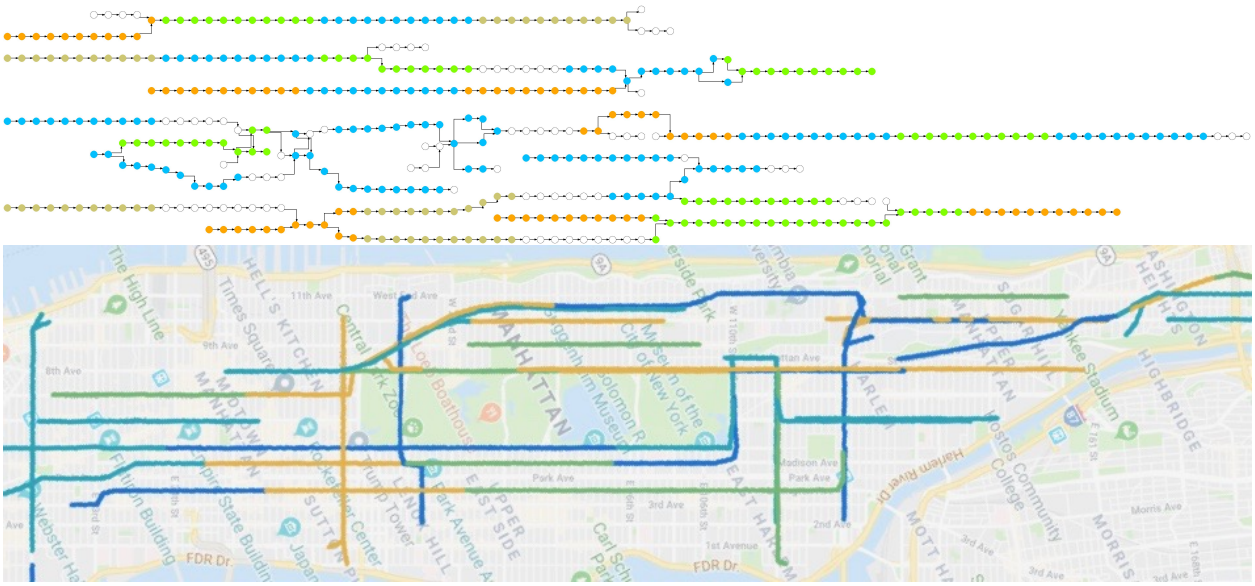


Figure 5.1: Traffic graph defined over Manhattan divided into chunks of 5-hop subgraphs on which models are evaluated (each color is a subgraph). 5-hop subgraphs consist of all segments within five upstream and five downstream hops based around a selected root segment.

5.3 Approach

5.3.1 Mixed-Adjacency Recurrent Neural Net

We define a Mixed-Adjacency Recurrent Neural Net (*MXRNN*) to investigate the capability of Recurrent Neural Nets to disambiguate interactions between input measurements belonging to different nodes with no knowledge of their spatial connectivity. This serves as a baseline performance for us when studying neural network approaches. Measurements in each subgraph are flattened into a 1-dimensional vector by fixing the order of nodes. As the initial choice of ordering can be arbitrary, no prior adjacency information is given to MXRNN. In other words, MXRNN consists of an RNN for each subgraph, with readings from all input segments flattened out before feeding into the RNN. The spatial interactions of traffic and dependencies are learned during training. We use Long-Short Term Memory (LSTM) cells which aid in detecting long-term dependencies [Hochreiter and Schmidhuber 1997a] and have been shown to work well for time

series prediction models.

5.3.2 Message-Passing Recurrent Neural Net

We introduce greater supervision and regularization steps to improve generalization ability from the naïve MXRNN in the form of a "graphically aware" message-passing neural net called the **Message-Passing Recurrent Neural Net** (*MPRNN*). The MPRNN is a deep neural network architecture with differentiable operations which iterate message-passing among connected nodes in our traffic graph and a recurrent architecture to detect temporal effects. Through tunable model parameters, MPRNN explicitly controls the breadth of information propagation between connected nodes in the graph and the flow of information based on the directionality of node adjacencies (§5.2). In our results, we observe the introduction of message-passing improves generalization over baseline models (§5.4.2).

Departing from Graph Convolution [Gilmer et al. 2017], a state-maintaining messenger and LSTM unit is defined for each node $v \in V$ in the traffic graph where one node is solely responsible for learning the traffic patterns in that node given neighboring states. The set of neighbors, $N(v)$, of a node v is the set of adjacent road segments. In evaluation, each node converses in messages with their immediate neighbors based on the current observations, and then updates the internal states $\{h_t(u) \mid u \in V\}$ in its LSTM unit. At time $t + 1$, the speed is predicted from the internal state. Message-passing operates by allowing one node to observe the hidden state of its neighbors. Performing multiple iterations of message-passing allows the propagation of information beyond immediate neighbors.

5.3.3 Training

For both the MPRNN and MXRNN, the initial hidden state for each segment at $t = 0$, $h_0(v)$, is initialized randomly during training and evaluation, sampled from $\mathcal{N}(0, 1)$, since the previous traffic

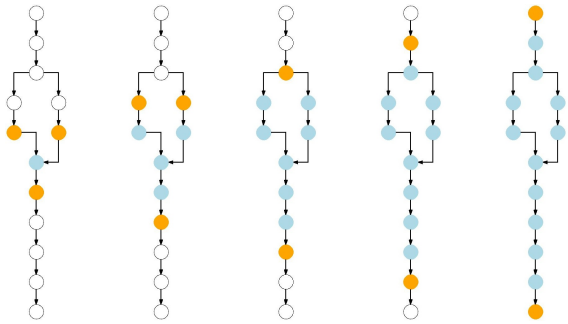


Figure 5.2: *Procedural Training:* Parameters trained for smaller subgraphs are transferred to initialize training progressively larger subgraphs of $k = 1 \ldots 5$.

conditions are unknown. In training, loss is computed using *Mean-Squared Error (MSE)* to obtain a regression on predicted speeds. For all the LSTM cells, we picked 24 as the *history length*, or the length of unrolled LSTM network. Empirical challenges exist with back-propagating the gradient over exceedingly long histories, as gradients are distributed among more parameters [Hochreiter 1991]. Also, in the context of traffic flow, we also do not expect effects to be propagated significantly more than a few hours ahead in time. A history length of 24 corresponds to 4 hours of history (at one sample every 10 minutes), which is neither too small nor too large.

We follow a training curriculum where the model is defined and trained on smaller subgraphs then on increasingly larger graphs with transferred weights for previously trained nodes. Figure 5.2 shows the procedure as the number of hops-k increases. For the first 8 epochs, only the parameters defined on nodes newly introduced from $k' - 1 \rightarrow k'$ hops receives gradient update. Then, all parameters are fine tuned in a final epoch. This training procedure was determined to obtain quicker convergence for large hops and also to preserve well-performing local parameters from becoming diluted. As all segments were ensured to have $> 50\%$ data availability, a roughly equal volume of continuous data measurements of 18 days in the latter of the data collection period were reserved for testing and the remaining 72 days were used in training.

Specific to the MPRNN implementation, one iteration of message-passing dictates the propagation of one node's state to its neighbors. Multiple iterations of message-passing allows broader

Model	k=1	k=2	k=3
Root-mean squared error (RMSE) (mph)			
Linear	0.505 ± 0.7	0.647 ± 0.8	0.752 ± 1.1
MXRNN	0.283 ± 0.3	0.291 ± 0.4	0.672 ± 1.8
GCN [Kipf and Welling 2016]	-	0.294 ± 0.3	0.294 ± 0.3
ASTGCN [Guo et al. 2019]	0.286 ± 0.3	0.274 ± 0.3	**0.272 ± 0.2**
MPRNN	**0.273 ± 0.4**	**0.260 ± 0.3**	**0.272 ± 0.3**

Table 5.1: Next timestep prediction ($t+1$) accuracy evaluated on a reserved period of 18 continuous days. RMSE measurements are shown first, with Pearson Correlation Coefficient below.

spread of information with tradeoffs in train-evaluation time. A fixed number of iterations occur within each time-step, thus 3 iterations were deemed appropriate in our context. As a result, newly obtained hidden states are intended to propagate only during the next timestep for segments more than 3-hops away.

5.4 Results

5.4.1 Next timestep Prediction

As a first test of performance, we predict speeds in the immediate next timestep, at a distance of up to 3 hops (3 bus stops) away. This is approximately 1 kilometer. In this, we show the performance of each model in predicting the traffic speed in the test segment, 10 minutes (i.e. one timestep) ahead, given 4 hours (24 steps) of prior history. Performances are compared across a simple linear regression model with a bias term, the Mixed-Adjacency RNN (*MXRNN*), and our proposed Message-Passing RNN (*MPRNN*), and two state-of-the-art methods that employ Graph Convolution Networks, all trained and evaluated with history length 24.

As shown in Table 5.1, the MPRNN performs very well with a *Root Mean-Squared Error* (*RMSE*) of 0.272 mph with input from up to $k = 3$ neighboring hops, while the naïve LSTM approach, MXRNN, is comparable for low k, but then rapidly degrades in performance, indicating a lack of ability to predict at farther locations. The GCN [Kipf and Welling 2016] is a simple GCN that is

used for supervised learning, and the ASTGCN [Guo et al. 2019] is a very recent work that beat many other state-of-the-art methods in traffic flow prediction[1].

The best-fit error rates are obtained by MPRNN, consistently demonstrating error lower than the other methods. It is noted that the MXRNN shows a sharp degradation at $k = 3$. While an outlier, the inconsistency in prediction behavior becomes much more apparent in forecasting where the MXRNN exhibits higher error rates with high variance. In general, we attribute the improvement in prediction to the regularization introduced in message-passing as spatial consistency is maintained along the true layout of traffic. This type of regularization is achieved by the graph convolutional approaches as well, which are also resilient to the addition of more hops in the input. In fact, the ASTGCN performs *better* as more hops are added. But the problem appears in forecasting, which is a much more expensive computational operation, owing to the number of parameters and variables involved. Thus the GCN based methods, which are much more complex than the other methods, take prohibitively long times to produce forecasting results, and hence we do not consider those results in our comparison.

For all implementations, we experience a tradeoff in accuracy when training for larger k-hops. At the benefit of observing information from more segments, the number of parameters increase and the training objective requires the model to fit neighboring data measurements. However, despite this, benefit of training for larger k enables forecasting over much broader segments of road.

5.4.2 Forecasting

Forecasting performance for subgraphs in the directed traffic graph of Manhattan are assessed given known values at entry and exit nodes. Experiments are performed for varying sizes of the subgraphs of hops $k = 1 \ldots 3$ where the tested models must propagate known observations over an increasing sequence of intermediate segments for which observations are not available.

[1]Traffic *flow* prediction is a different problem that traffic speed prediction; flow refers to volume of traffic

Model	k=1	k=2	k=3	k=4
Root-mean squared error (RMSE) (mph)				
MXRNN	2.59 ± 1.6	2.71 ± 1.5	2.74 ± 1.8	2.85 ± 1.2
MPRNN	1.83 ± 0.3	2.13 ± 0.5	2.29 ± 0.6	2.35 ± 0.6
Pearson Correlation Coefficient (PCC)				
MXRNN	0.59 ± 0.0	0.53 ± 0.1	0.47 ± 0.1	0.47 ± 0.1
MPRNN	0.76 ± 0.0	0.64 ± 0.0	0.51 ± 0.1	0.43 ± 0.0

Table 5.2: Forecasting performance for increasing hops $k = 1 \ldots 4$ averaged across 47 graphs which consist the traffic graph of Manhattan. Variance is reported over the individual graphs.

Model	k=1	k=3	k=5	k=7	k=9
Root-mean squared error (RMSE) (mph)					
MXRNN	3.06	3.28	4.41	5.24	4.27
MPRNN	2.75	3.26	3.25	3.27	3.24
Pearson Correlation Coefficient (PCC)					
MXRNN	0.37	0.33	0.34	0.01	0.08
MPRNN	0.50	0.30	0.19	0.16	0.13

Table 5.3: Forecasting error for a broader range of hops $k = 1 \ldots 10$ for a the single segment plotted in Figure 5.1 (the intersection of *54th St and 7th Ave*)

All forecasting errors were assessed for the reserved period of 18 days. Table 5.2 shows the forecasting errors (RMSE) across all subgraphs in Manhattan for $k = 1 \ldots 3$. In addition to the RMSE metric, the Pearson Correlation Coefficient (*PCC*) [Rodgers and Nicewander 1988], which characterizes the quantifies the correlation between the predicted and real values, is shown. For one specific segment, we also show the errors for larger values of k up to 9 in table 5.3. For the first available time step of each day, both MXRNNs and MPRNNs were initialized with zeros (mean of random initialization during training). Forecasting then proceeded for all subsequent traffic speeds throughout the day until the last available measurement.

As can be seen from the tables, the MPRNN outperforms the MXRNN in all cases, and particularly when using larger number of hops. While the ability to forecast future values degrade for all models, there is a more gradual degradation in error for the MPRNN (Table 5.3), where for hops $k = 9$, the MPRNN still maintains an RMSE of 3.24 mph while using the MXRNN produces an error of 4.27 mph with indications that error will continue to increase with more hops. We re-

iterate that the MPRNN architecture only defines message propagation which is consistent with the spatial layout of the traffic graph, as opposed to MXRNN which is not regularized by spatial information and thus defines arbitrary correlations with fully-connected layers over any input measurements.

5.5 Conclusion

Highly parameterized neural nets have been applied successfully to data that is sporadic and unidimensional, but abundant and easily collected at the same time. We examine historical travel data of public transit vehicles in New York City which currently sees use solely to check bus arrival times for commuters. We aggregate bus speeds into a traffic segment graph that represents the relationships between road segments. The MPRNN architecture is defined on the graphical representation of traffic flow which leverages the interaction between pairs of traffic segments where flow patterns of individual segments are subject to highly variable factors. Forecasting performance is assessed on traffic segments spanning the entirety of Manhattan. Benchmarks are presented in fine tuning and in close comparison with a more naïve implementation of MPRNN, called MXRNN, resulting in a final model which produces meaningful forecasts several hours ahead in time.

6 Implementing Signal Control for Traffic

6.1 Signaling Control Protocol

Every finite stretch of road, a *link*, is associated with a *buffer size* (B_ℓ), which is the number of vehicles in the link at a certain time. Traffic collapse results when buffer size in a link (the number of vehicles) exceeds a certain threshold. The instantaneous exit rate of the link (C_ℓ), which is the rate of exit of vehicles out of the link, varies with the traffic density [May 2000]. The *traffic curve* (figure 6.1) captures the variation between these two parameters [Jain et al. 2012]. Notice that the maximum exit rate C^*_ℓ is obtained for an optimal value of buffer size B^*_ℓ. When the buffer size exceeds this optimal value, the exit rate drops sharply.

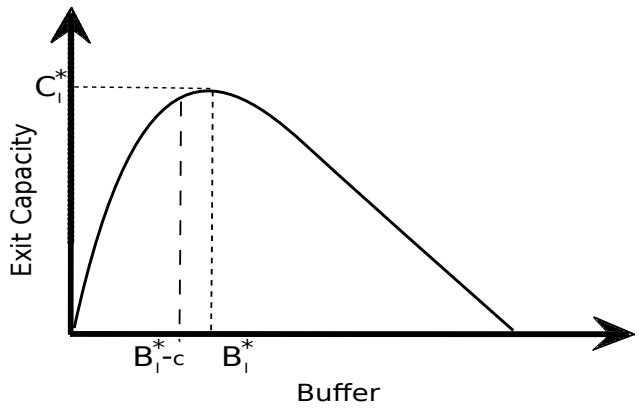


Figure 6.1: Traffic curve showing the instantaneous exit rate as a function of the buffer size

Our protocol utilizes this dependency to control input rate into the system for a desired exit rate. For our protocol, we propose the addition of two types of processing points to existing traffic systems – *monitoring points* and *signaling points*. The goal of a monitoring point is to constantly monitor the *buffer size* of each link (number of vehicles in the link) and report it to the signaling point. The signaling point, which is present for each junction or choke point in the road network, operates the red and the green signals at the junction according to the proposed protocol, which utilizes the buffer information from the neighboring monitoring points.

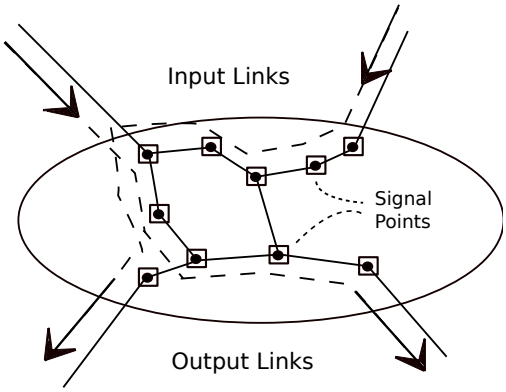


Figure 6.2: Link model

Figure 6.2 shows a sample link model, where a road network is described in terms of links and signal points. The protocol uses a backpressure algorithm that uses the local traffic information in the neighboring signaling points to manipulate the traffic signal. For example, suppose traffic flows from signal point A to signal point B. B has to communicate local traffic variables to A, and based on this information, A adapts its signal lights to change the flow of traffic. So, the information is sent from a forward signal point (B) to a backward signal point (A), which depicts the applied *backpressure* to the flow of traffic.

6.1.1 Formal description

Suppose that $\mathcal{L} = \{1, 2, \ldots, L\}$ is the set of all road links. We shall assume that each vehicle exiting the link ℓ joins the link ℓ' with probability $p_{\ell\ell'}$ and leaves the network with probability

$1 - \sum_{\ell'} p_{\ell\ell'}$. The protocol has two main parts. The first part is the rate control at the entry points and the second part concerns the traffic signal scheduling. Effectively, all the decisions are done every 'signaling epoch'. Let $\tau \in \mathbb{N}$ denote the index of the epoch and let $b_\ell(\tau)$ denote the buffer size (number of vehicles) of link ℓ in the beginning of epoch τ. The protocol will use this information to make decisions for epoch $\tau + 1$.

Rate control: At link ℓ, exogenous vehicles are admitted as per rate $\min(w_\ell/b_\ell(\tau), C_\ell/2)$, where w_ℓ is a fixed positive constant. Assuming unit size epoch, this means allowing $\min(w_\ell/b_\ell(\tau), C_\ell/2)$ exogenous vehicles to enter[1]. This corresponds to restricting the maximal rate to $C_\ell/2$ to avoid oversubscription. In reality, only a few network links will have exogenous traffic where such rate control is needed. Most links serve internal traffic only. For example, in figure 6.2, there are only four links through which exogenous traffic enters the network.

Traffic scheduling: At each traffic junction, the competing links are scheduled using the local link buffer information in three steps – *i)* if there is link ℓ' with $p_{\ell\ell'} > 0$ and the buffer size $b_{\ell'}(\tau) \geq B^*_{\ell'} - c$, link ℓ is non-schedulable in epoch $\tau + 1$; *ii)* among remaining schedulable links, to schedule vehicles, they are assigned priorities as described next. The vehicles with *highest priority* move to the next link or move out of the system; *iii)* the vehicles, scheduled to move out of a link ℓ, exit at rate C_ℓ. This scheduling policy makes sure that the buffer capacity almost reaches the *optimal buffer capacity* B^*_ℓ, allowing it to reach $B^*_\ell - c$ for a small constant $c > 0$. The constant c sees to it that the buffer capacity never reaches critical limit.

Priority: At each intersection we need to schedule certain number of vehicles from each of the input links to move to the output links. At each step of the epoch time, the highest priority

[1]This corresponds to utilizing utility function $V_\ell(z) = w_\ell \log z$ in optimization ($V_\ell : \mathbb{R}_+ \to \mathbb{R}$ is the utility function of link ℓ so that its utility is $V_\ell(x_\ell)$ when exogenous rate is x_ℓ. An example of a reasonable utility function arises from proportional fair allocation (corresponds to the solution of Nash bargaining game) with $V_\ell(z) = w_\ell \log z$ for fixed positive weight/constant w_ℓ.)

vehicle is scheduled. Priority of each vehicle P_v is defined below.

$$P_v = \left(b_\ell(\tau) - \sum_{\ell' \in \mathcal{L}} p_{\ell\ell'} b_{\ell'}(\tau) \right) \tag{6.1}$$

As the vehicle's route is unknown, we compute the priority based on the difference in the input buffer size and probability of vehicle moving to a particular output link ℓ' times the buffer size of ℓ'. We need to have a predefined set of probability routing matrix for each intersection, so that we can use this map to compute the priority of each vehicle. If the vehicle route is known, then P_v reduces to $b_\ell(\tau) - b_{\ell'}(\tau)$, the difference between the current input and output link buffers. We choose k eligible vehicles in each link that are near the intersection and compute priorities of each of these vehicles. The highest priority vehicle moves to the output link ℓ'.

Eligibility criteria: The k vehicles are chosen based on the eligibility criteria. We compute the eligibility of the vehicles (to move from the input link to the output link) across all the input links. A vehicle is eligible to be scheduled, if its timestamp v_t is less than the schedule time t. Based on this metric, some subset of vehicles in each input link are chosen to be eligible for schedule.

Scheduling and signal timing control: After choosing the top priority vehicles to be scheduled, we compute the minimum schedule time for each vehicle given by, $\frac{1}{C_\ell}$; it means the vehicle moves out of link ℓ with an exit rate of C_ℓ. The fraction of time required to schedule each vehicle is added to the epoch time, $\tau = \tau + \frac{1}{C_\ell}$. This minimum schedule time gives the fraction of time the signal needs to be on, to schedule that particular vehicle. This min schedulable time is added for each top priority vehicle in every input link. When this is done, we know for what fraction of the epoch time τ should we schedule each input link ℓ. The signal is turned on (green) for an epoch τ, so that the scheduled vehicles move from ℓ to ℓ'. This marks the end of an epoch and the protocol goes back to choosing some k vehicles near the intersection based on some eligibility criteria and then compute their priorities. At every step of the protocol, if $b_{\ell'}(\tau) \geq B^*_{\ell'} - c$, then

no vehicle can move to ℓ' in that epoch. This helps in avoiding congestion in the output links.

We summarize the protocol in Algorithm 1.

Algorithm 1 Signaling Control Protocol

while input links not empty **do**
  Compute eligibility of each vehicle v in each input link $\ell \in \mathcal{L}$
  if $v_t < t$ **then**
    vehicle eligible
  else
    vehicle not eligible
  end if
  while epoch time $\tau < 1$ **do**
    Choose the set of eligible vehicles v_e in $\mathcal{L}$.
    Compute priority P_v of each eligible vehicle in v_e.
    Choose the highest priority vehicle $v \in v_e$.
    Apply back pressure by adapting the epoch time τ.
    $\tau = \tau + \frac{1}{C_\ell}$
    if $\tau \geq 1$ **then**
      Turn on the signal up to time τ
      Move each $v \in v_e$ from ℓ to ℓ'
    end if
  end while
end while

6.2 Theory

Here we summarize somewhat idealized version of the protocol for the purpose of analysis. To that end, let us first start with the rate control for the exogenous traffic. As per the protocol description, the rate control restricts the rate of injection to link ℓ, $x_\ell(\tau)$ equals $\min\left(\frac{w_\ell}{b_\ell(\tau)}, \frac{C_\ell}{2}\right)$. For choice of utility function $V_\ell(x) = w_\ell \log x$, this corresponds to the solution of the following optimization problem.

$$\text{maximize} \quad V_\ell(x) - b_\ell(\tau)x \quad \text{over} \quad 0 \leq x \leq C_\ell/2. \tag{6.2}$$

The traffic links are scheduled by assigning priority to vehicles. Specifically, a vehicle on link ℓ at time τ has priority equal to $b_\ell(\tau) - \sum_{\ell' \in \mathcal{L}} p_{\ell\ell'} b_{\ell'}(\tau)$. Such an assignment can be thought of as selecting a traffic signaling ρ among all possible allowed options Λ at time τ. Equivalently, ρ can be though of as a solution of the following optimization problem.

$$\begin{aligned} \text{maximize} \quad & \sum_{\ell \in \mathcal{L}} \rho_\ell \left(b_\ell(\tau) - \sum_{\ell' \in \mathcal{L}} p_{\ell\ell'} b_{\ell'}(t) \right) \\ \text{subject to} \quad & \rho \in \Lambda \ \ \forall \ell \in \tau. \end{aligned} \tag{6.3}$$

Under the above protocol, the evolution of queue-sizes $b_\ell(\cdot)$ for $\ell \in \mathcal{L}$ can be described as follows.

$$\frac{db_\ell(\tau)}{d\tau} = \left[x_\ell(\tau) + \sum_{\ell'} p_{\ell'\ell} \rho_{\ell'}(\tau) - \rho_\ell(\tau) \right]^+_{b_\ell(\tau)},$$

where by $[y]^+_x$ we mean

$$[y]^+_x = \begin{cases} y & \text{if } x > 0, \\ \max(0, y) & \text{if } x = 0. \end{cases}$$

The above described protocol solves the following fair resource allocation problem as $\tau \to \infty$.

$$\begin{aligned} \text{maximize} \ & \sum_\ell V_\ell(x_\ell) \quad \text{over} \ \ x = (x_\ell) \in \mathbb{R}^L_+ \\ \text{subject to} \ & x_\ell + \sum_{\ell' \in \mathcal{L}} p_{\ell'\ell} \rho_{\ell'} \le \rho_\ell \\ & \rho_\ell \le C_\ell / 2, \quad \forall \ell \in \mathcal{L} \\ & \rho \in \Lambda \end{aligned} \tag{6.4}$$

In above, Λ denotes the convex hull of the set of all traffic junction actions. That is, Λ denotes the effective capacity of the road traffic network. In summary, the protocol implemented attempts to achieve resource allocation among all completing incoming traffic so that to maximize resource utilization while being fair by maximizing the network's utility. We show a proof of optimality below.

6.2.1 Proof of Optimality

We start with the optimization problem of interest:

$$\begin{aligned} &\text{maximize } \sum_{\ell} V_\ell(x_\ell) \quad \text{over } \; x = (x_\ell) \in \mathbb{R}_+^L \\ &\text{subject to } x_\ell + \sum_{\ell' \in \mathcal{L}} p_{\ell'\ell}\rho_{\ell'} \leq \rho_\ell \\ &\qquad \rho_\ell \leq C_\ell/2, \quad \forall \ell \in \mathcal{L} \\ &\qquad \rho \in \Lambda \end{aligned} \tag{6.5}$$

We shall only dualize the first constraint of equation 6.5 and let b_ℓ be the corresponding dual variable with $b_\ell \geq 0$ for $\ell \in \mathcal{L}$. The corresponding Lagrangian and dual functions are below.

$$F(x, b) = \sum_{\ell} V_\ell(x_\ell) - b_\ell \left(x_\ell + \sum_{\ell' \in \mathcal{L}} p_{\ell'\ell}\rho_{\ell'} - \rho_\ell \right) \tag{6.6}$$

$$\begin{aligned} D(b) = \text{maximize} \quad & F(x, b) \quad \text{over } \; x \in \mathbb{R}_+^L \\ \text{subject} \quad \text{to} \quad & \rho \in \Lambda, \\ & \rho_\ell \leq U_\ell, \; \forall \ell \in \mathcal{L} \end{aligned} \tag{6.7}$$

Now it is easily checked that (6.7) decomposes into two separate optimization problems. First, for each $\ell \in \mathcal{L}$,

$$\text{maximize} \quad V_\ell(x_\ell) - b_\ell x_\ell \quad \text{over} \quad x_\ell \geq 0 \tag{6.8}$$

And second,

$$\begin{aligned} \text{maximize} \quad & \sum_\ell \rho_\ell \left(b_\ell - \sum_{\ell' \in \mathcal{L}} p_{\ell\ell'} b_{\ell'} \right) \\ \text{subject to} \quad & \rho \in \Lambda, \;\; \rho_\ell \leq U_\ell \;\; \forall \ell \in \mathcal{L} \end{aligned} \tag{6.9}$$

Clearly, problem (6.8) can be solved separately for each link $\ell \in \mathcal{L}$. The problem (6.9) is coupled by the constraint $\rho \in \Lambda$. But it is rather limited, since each traffic junction's signal operation is done independently. Therefore, the problem (6.9) gets decoupled as follows: over the traffic junctions, the actions ρ are chosen so that they maximize

$$\begin{aligned} \text{maximize} \quad & \sum_\ell \rho_\ell \left(q_\ell - \sum_{\ell' \in \mathcal{L}} p_{\ell\ell'} b_{\ell'} \right) \\ \text{subject to} \quad & \rho \in \Lambda, \;\; \forall \ell \in \tau. \end{aligned} \tag{6.10}$$

Clearly, (6.10) can be solved very much locally at the traffic junction. Let $x(b), \rho(b)$ denote the optimal values of x, ρ as per (6.8) and (6.10). Indeed, if b is an optimal dual assignment then $x(b)$ is optimal exogenous rate that solves the above optimization problem. To reach optimal b in an iterative manner, we describe a simple dual *subgradient* algorithm. The idealized version of the algorithm is given by

$$\frac{db_\ell}{d\tau} = \left[x_\ell + \sum_{\ell'} p_{\ell'\ell}\rho_{\ell'} - \rho_\ell \right]^+_{b_\ell}. \tag{6.11}$$

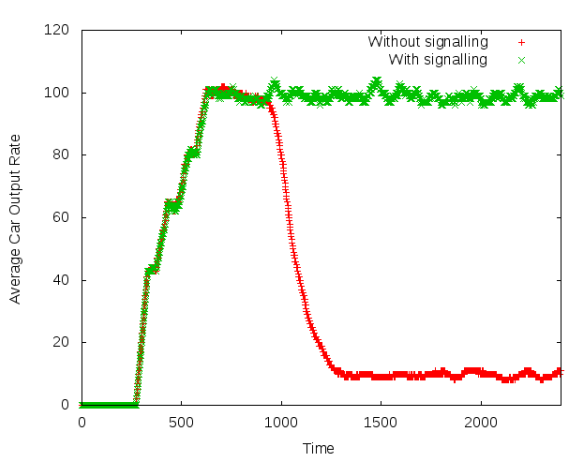
Essentially (6.11) tries to maintain the feasibility of $x_\ell \sum_{\ell'} p_{\ell'\ell}\rho_{\ell'} \leq \rho_\ell$. Thus, the primal dual algorithm given by (6.8), (6.10) and (6.11) leads to solution of the optimization. Note that this primal dual algorithm is precisely the idealized protocol description given above. This establishes that our protocol solves the optimization problem as claimed by our main result.

6.3 Evaluation

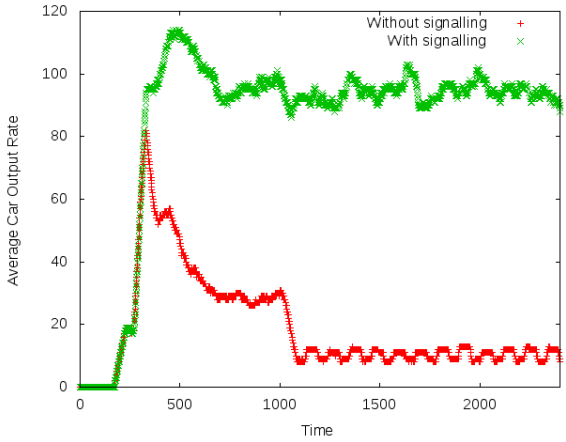
We evaluate the protocol on two real world road networks exhibiting free flow behavior – *i)* Bay Bridge in the San Francisco Bay Area, USA, and *ii)* Wilkinson Road-Murray Town junction in Freetown, Sierra Leone. We demonstrate how the protocol provides significant enhancements to the operational capacity in comparison to the status quo with no signalized control. For each network, we conducted three simulations (three variations), each 2400 seconds long, with progressively increasing input traffic burst rates.

Bay Bridge: The Bay Bridge is a very complex road topology in which there are multiple input and output links (figure 6.4). The points marked J/M represent the junctions and the merges in the topology. These would then be replaced by the new traffic signaling system. This topology has 5 sources, 3 sinks, 13 links and 6 J/Ms.

It is observed as in Figure 6.3(a) that without the proposed signaling system, the throughput rises to a value of 100 cars per second and then falls to 10 cars per minute approximately. It remains at that state for the remaining part of the simulation. However, with the signaling system, the throughput reaches a maximum value of 100 cars per minute and remains at that state for the rest of the simulation period.



(a) Throughput at Bay Bridge



(b) Throughput at Wilkinson Town-Murray Road Junction

Figure 6.3: Throughput plots for real world free flow networks

Wilkinson Road-Murray Town Junction: The Wilkinson Road-Murray Town junction (figure 6.5), is a heavily congested junction located in Freetown in Sierra Leone. Due to heavy traffic which flows in this junction, it becomes congested very frequently and regularly, creating a bottleneck in the system.

Initially, the burst rates of all the links is kept at 30 cars per minute. It is observed that in the existing signaling system, as shown in figure 6.3(b) the throughput rises to around 80 cars per minute and then gradually starts dropping. It finally settles down to around 5 cars per minute. However, with the new signaling system, the throughput rises to around 110 cars per minute and remains at this same level till the very end.

6.4 Conclusion

We have proposed a novel traffic signaling network concept for free-flow traffic that is inspired from the idea of backpressure-based congestion control in the Internet. The proposed protocol uses only localized information but guarantees a global optimality. By simulations on real world road networks, we observe that the proposed signaling control algorithm is able to achieve sig-

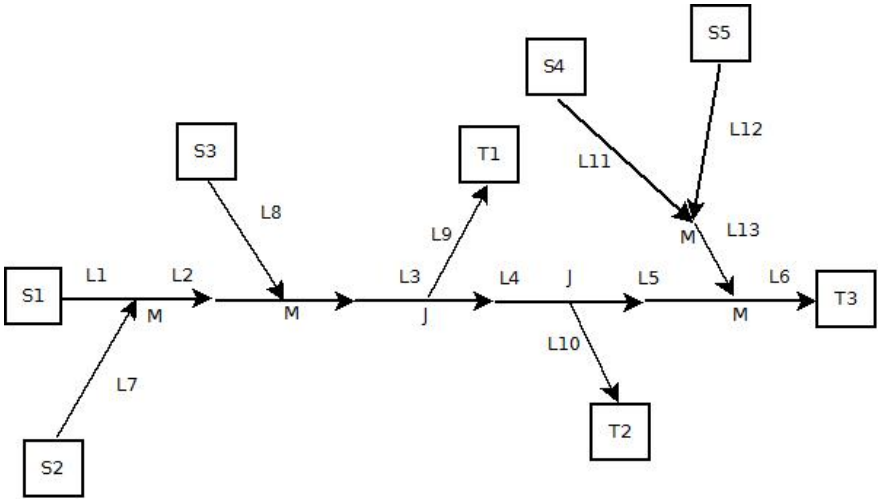


Figure 6.4: Local topology around the Bay Bridge in the San Francisco Bay area, as an illustration to show *control* points where signals may be inserted/controlled to prevent collapse

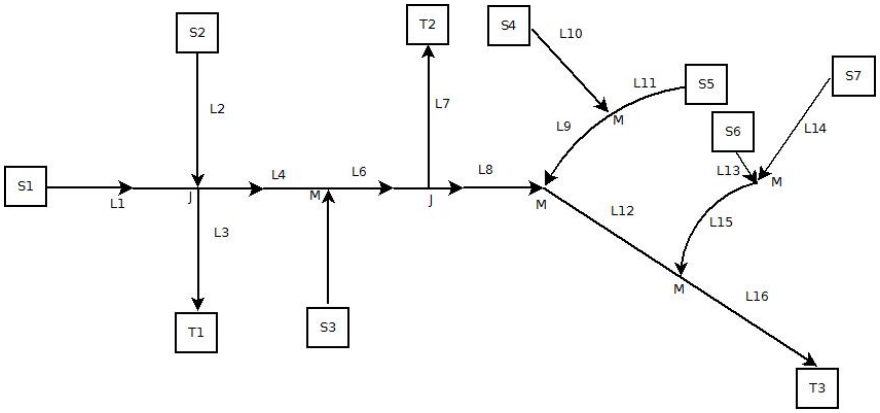


Figure 6.5: Wilkinson Road- Murray Town Junction

nificant increase in throughput compared to the free-flow case without any signaling.

A | Appendix

A.1 Spatio-temporal Hierarchical Model

The Spatio-Temporal Hierarchical Model (STHM) is a statistical modeling framework from geostatistics. It combines various sources of information, accommodates missing values and computes predictions in both space and time. This statistical model is hierarchical in that it distinguishes between observed variables, such as the actual PM measurements, and underlying processes that are not directly observed. In a state space terminology the latter are known as unobserved states, while in some statistics literature they are known as random effects or latent variables (see e.g. Harvey. [Harvey 1989]). The hierarchy is explicit since the model is defined through multiple levels of equations, where a higher level typically involves variables conditioned on the variables defined at deeper levels. This allows the important identification of two sources of error: measurement error which applies to the observations, usually at the highest hierarchical level, and process error which enters the specification of the dynamics of the underlying processes at deeper levels. We refer to Cressie and Wikle [Cressie and Wikle 2011] for details about the relevance of this hierarchical framework for spatio-temporal modeling, how it is currently considered state-of-the-art, and for links to the geostatistics literature. In particular, optimal spatial prediction (often referred to as Kriging) pertains to the prediction of underlying processes and not of the noisy measurements and is addressed below.

Successful recent applications of such hierarchical models for the spatio-temporal modeling

of air pollution include [Cameletti et al. 2011], [Cameletti et al. 2013] and [Beloconi et al. 2018]. Our model proposed below generally follows these references for the dynamics in space and time of an underlying specified random field modeled as a Gaussian process.

Let $\mathcal{D} \subseteq \mathbb{R}^2$ denote the spatial domain of interest and $Y(\boldsymbol{s}, t)$ denote the $PM_{2.5}$ concentration in $\mu g/m^3$ measured at location $\boldsymbol{s}$ and time t. The location vector $\boldsymbol{s} = (s_1, s_2)^\top \in \mathcal{D} \subseteq \mathbb{R}^2$ consists of geographical coordinates s_1 and s_2 in the plane following a map projection, such as the easting or northing in km according to the Universal Transverse Mercator (UTM) coordinate system, so that a notion of distance $h(\boldsymbol{s}, \boldsymbol{s}') \in \mathbb{R}$ can be unequivocally defined. Following Cameletti et al. [Cameletti et al. 2011, 2013], the top hierarchical level of our STHM is specified by the following measurement equation.

$$\log Y(\boldsymbol{s}, t) = \boldsymbol{z}(\boldsymbol{s}, t)^\top \boldsymbol{\beta} + \sum_{j=1}^{J} \alpha_j B_j(t) + X(\boldsymbol{s}, t) + \epsilon(\boldsymbol{s}, t), \tag{A.1}$$

Here $t = 1, 2, \ldots$ is a discrete representation of timestamps (regardless of the actual temporal resolution of the data), $\boldsymbol{z}(\boldsymbol{s}, t)$ is a p-vector of covariates which defines deterministic (fixed) effects along with the corresponding coefficient $\boldsymbol{\beta}$, $B_j(t)$ for $j = 1, \ldots, J$ is a set of specified (periodic) basis functions used to model seasonality effects along with the corresponding basis coefficients α_j, $X(\boldsymbol{s}, t)$ is a mean zero Gaussian process whose dependence structure in space and time is specified at the second hierarchical level, and the $\epsilon(\boldsymbol{s}, t)$'s are measurement error terms assumed independent and identically distributed as Gaussian with mean zero and constant variance σ_ϵ^2 (the latter known as the "nugget" effect in the geostatistics literature). As a result, the $\log Y(\boldsymbol{s}, t)$'s are independent conditionally on $X(\boldsymbol{s}, t)$, for all $\boldsymbol{s} \in \mathcal{D}$. The modeling on the natural logarithm scale ensures the positivity of Y. We note that although deterministic effects enter as a linear combination, any auxiliary information can be part of $\boldsymbol{z}(\boldsymbol{s}, t)$. For instance, outputs from a dispersion model predicting the propagation of fine particles from various environmental

inputs would enter the STHM through $\boldsymbol{z}(\boldsymbol{s},t)^\top\boldsymbol{\beta}$. In the application of this model to our Delhi sensor data, we however place ourselves in a data-poor situation with no extra information other than the air pollution measurements themselves, so that no auxiliary deterministic effects are estimated and $\boldsymbol{z}(\boldsymbol{s},t)^\top\boldsymbol{\beta} = 0$ for all observations. We model daily seasonality with $J = 6$ quadratic B-spline bases over four disjoint time intervals: [00:00–06:00), [06:00-12:00), [12:00–18:00) and [18:00–00:00). This implies $J - 1 = 5$ fixed knots to facilitate interpretation of periodic patterns throughout the day. The corresponding α_j coefficients are estimated from the data, but only $J - 2 = 4$ are free since we enforce two constraints on the continuity and differentiability of the resulting linear combination at the boundary at midnight. The intercept term (constant mean level) is included in the B-splines linear combination.

The secppond hierarchical levels describe the temporal dynamics and spatial dependence structure of the underlying stochastic process X. The process equation describes a stationary autoregressive (AR) process of first order through time:

$$X(\boldsymbol{s},t) = \phi X(\boldsymbol{s},t-1) + \delta(\boldsymbol{s},t), \tag{A.2}$$

for $t = 1,2,\ldots$ and $\boldsymbol{s} \in \mathcal{D}$, where the constraint on the AR coefficient $|\phi| < 1$ ensures stationarity, and the process error δ is distributed as Gaussian with expectation zero. The δ terms are temporally independent but spatially dependent:

$$\mathrm{Cov}[\delta(\boldsymbol{s},t),\delta(\boldsymbol{s}',t')] = \begin{cases} 0 & t \neq t' \\ C(h(\boldsymbol{s},\boldsymbol{s}');\gamma,\sigma_\delta) & t = t', \end{cases}$$

where C is a (positive-definite) spatial covariance function; we set it to the stationary and isotropic exponential spatial covariance function for simplicity:

$$C(h(\boldsymbol{s}, \boldsymbol{s}'); \gamma, \sigma_\delta) = \sigma_\delta^2 \exp(-h(\boldsymbol{s}, \boldsymbol{s}')/\gamma),$$

where $h(\boldsymbol{s}, \boldsymbol{s}') = \sqrt{(s_1 - s_1')^2 + (s_2 - s_2')^2}$ is the Euclidean distance between locations $\boldsymbol{s}$ and $\boldsymbol{s}'$, σ_δ^2 is the process variance for $h(\boldsymbol{s}, \boldsymbol{s}') = 0$, and γ regulates the steepness of the exponential decay of the covariance with increasing distance. The initial states $X(\boldsymbol{s}, 0)$ follow the stationary distribution, i.e. a Gaussian distribution with mean zero and covariance matrix given by $C(h(\boldsymbol{s}, \boldsymbol{s}'); \gamma, \sigma_\delta)/(1 - \phi^2)$ for $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{D}$.

Overall, this STHM involves $(p + 8)$ fixed parameters: $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \sigma_\epsilon^2, \phi, \gamma, \sigma_\delta^2)^\top$,. The dynamic spatial field X will be predicted given an estimate of $\boldsymbol{\theta}$. With the Delhi data, there is no $\boldsymbol{\beta}$ to estimate so that $p = 0$ and only 8 parameters are estimated.

Model fitting: For a sample of T time points and n locations, let $\boldsymbol{Y}(t) = (Y(\boldsymbol{s}_1, t), \ldots, Y(\boldsymbol{s}_n, t))^\top$ and $\boldsymbol{X}(t) = (X(\boldsymbol{s}_1, t), \ldots, X(\boldsymbol{s}_n, t))^\top$. Following a state space terminology as in [Harvey 1989], the measurement equation (A.1) specifies the conditional distribution of $\boldsymbol{Y}(t)|\boldsymbol{X}(t)$, while the process equation (A.2) specifies the Markovian dynamics of $\boldsymbol{X}(t)|\boldsymbol{X}(t-1)$. Taken together, they define the joint likelihood function

$$L(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{x}) = f(\boldsymbol{x}(0)) \prod_{t=1}^{T} f(\boldsymbol{y}(t)|\boldsymbol{x}(t)) f(\boldsymbol{x}(t)|\boldsymbol{x}(t-1)),$$

where, by some slight abuse of notation, we use f to denote probability density functions with the arguments lifting any ambiguity. This joint likelihood comprehensively summarizes the model, but is not useful per se since the dynamic underlying process X is not observed. Thus, we base estimation and inference on the marginal likelihood

$$L(\boldsymbol{\theta}; \boldsymbol{y}) = \int \cdots \int L(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{x})\, d\boldsymbol{x}(0) \cdots d\boldsymbol{x}(T),$$

where the high-dimensional integrals admit no closed form and need to be approximated. In our implementation of this STHM within the R package Template Model Builder [TMB; Kristensen et al. 2016], we rely on the Laplace approximation for integrals to evaluate such marginal likelihood. In addition, the TMB package makes use of automatic differentiation, which allows for fast and efficient evaluation of the gradient of the Laplace-approximated marginal likelihood with respect to $\boldsymbol{\theta}$ [Griewank and Walther 2008]. This is well suited for the optimization problem defining the (Laplace-approximated) maximum likelihood estimator of $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta}; \boldsymbol{y}).$$

We note that missing values in $\boldsymbol{y}$ are automatically accounted for (and accommodated) in this likelihood optimization, up to a reasonable extent. Indeed, the observation contribution to the joint likelihood represented by $f(\boldsymbol{y}(t)|\boldsymbol{x}(t))$ is not required to strictly follow the exact same $(\boldsymbol{s}, t)$ indices as the underlying X process, in particular it may contain gaps in time. The "reasonable extent" understood here is meant to exclude extreme cases, where as few as one or two locations only are available at a given time point; for such time points, a spatial model is simply not identifiable.

Interpolation: Given $\hat{\boldsymbol{\theta}}$, we predict the underlying dynamic spatial field by maximizing the joint log-likelihood

$$\hat{\boldsymbol{X}}(0), \ldots, \hat{\boldsymbol{X}}(T) = \arg\max_{\boldsymbol{x}} \log L(\hat{\boldsymbol{\theta}}; \boldsymbol{y}, \boldsymbol{x}). \tag{A.3}$$

This maximization can also be invoked for out-of-sample prediction, either for a new location, for forecasting in time, or for both simultaneously. That is, the $PM_{2.5}$ forecasting at time $t+1$ for a new location $\boldsymbol{s}'$ is given by $\exp\left(\boldsymbol{z}(\boldsymbol{s}', t+1)^\top \hat{\boldsymbol{\beta}} + \sum_{j=1}^{J} \hat{\alpha}_j B_j(t) + \hat{X}(\boldsymbol{s}', t+1)\right)$, where $\hat{X}(\boldsymbol{s}', t+1)$ is indeed obtained by maximizing the joint log-likelihood for given parameter estimates. To be precise, such a forecast is not a prediction of the noisy measurement $Y(\boldsymbol{s}, t)$, but really a prediction of the true underlying $PM_{2.5}$ concentration represented by $\exp\left(\boldsymbol{z}(\boldsymbol{s}', t+1)^\top \boldsymbol{\beta} + \sum_{j=1}^{J} \alpha_j B_j(t) + X(\boldsymbol{s}', t+1)\right)$. Our predictions can thus be seen as Bayesian posterior modes, while spatial prediction by Kriging typically corresponds to posterior expectation, see Chapter 4 of Cressie et al. [Cressie and Wikle 2011] for a discussion.

For a given time point t, plotting the predicted $PM_{2.5}$ concentrations as a smooth map by simple interpolation over the n measured locations likely gives rise to visual artifacts and distortions if either n is too small or if the measured locations are not spread evenly over $\mathcal{D}$. Such artifacts happen with clusters and empty spaces, as in the application to the Delhi data. The STHM provides a natural way to "fill-in" the spatial domain $\mathcal{D}$ with predictions at extra locations according to equation (A.3). Regarding the choice of extra locations, rather than constructing an inefficient regular grid of points, we follow here Lindgren et al. [Lindgren et al. 2011] by using a constrained Delaunay triangulation as implemented in the R package Integrated Nested Laplace Approximation [INLA; Rue et al. 2009]. This triangulation method allows us to tessellate $\mathcal{D}$ with triangles such that their minimum interior angle is maximized under the constraint that measured locations correspond to vertices. This (constrained) maximin property ensures that the density of vertices somewhat follows the density of measured locations (i.e. smaller triangles where sensors are clustered) while maintaining an even spread in empty areas, including beyond the convex

hull of all measured locations. The predictions at these extra locations can be integrated within the fitting of the model, since they are equivalent to missing values in $\boldsymbol{y}$ (locations where no observation is available).

A.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) belong to a broader family of deep neural networks which are general function approximators [Hornik et al. 1989]. In this experiment, the purpose of using a deep model is to model the complex nonlinear dependencies between the input and output without the need to impose an explicit physical model. We specifically choose RNNs because they are well suited for modeling sequential or time series data. The working of an RNN can be described simply by the function Φ in the equation $y_t, h_t = \Phi(y_{t-1}, h_{t-1})$, where y_t refers to a label or value that is predicted by the network at time t, and h_t is an internal state that represents the "memory" of the network at time t. Given sequential data of the form $y_0, \ldots, y_t$, Φ is applied repeatedly to predict label y_i, state h_i and so on until time t. The initial internal state h_0 is assumed to be zero in most applications. The number of such recursive computations (equivalently, the number of cells in the unraveled RNN) defines the length of the *history* that is used in the learning process to predict the value at $t+1$. However, while RNNs provide a semantic framework for prediction of sequential data, they provide no innate mechanism in deciding when the internal state h should be modified. This challenge is addressed by Long-Short Term Memory (LSTM) cells [Hochreiter and Schmidhuber 1997b] which explicitly facilitates the persistence or re-initialization of the internal state vector h over real sequential data. The use of LSTM cells in RNN architectures have been empirically shown to improve predictive power in temporal data because of their ability to learn long-term dependencies, and hence we employ them in our model.

A.3 K-Nearest Neighbor Spatial Neural Network

Another type of neural network that we have used to model spatio-temporal interactions for predictive modeling, in order to benchmark the MPRNN performance and contrast with it, is a more simplified neural network model in which messages are not explicitly passed between pairs of nodes, but rather the sensor readings from a set of neighboring monitors to a location v at time t are directly used as input. Each node runs a neural network with an LSTM unit for predicting future values, similar to the MPRNN. The pairwise distances and relative positions are encoded in the feature vector along with the input sensor readings. At each node, only a certain number of closest neighbors are used as input to model the air quality at that location, in contrast to the MPRNN where all the other sensors are used. We call this model the *K-Nearest Neighbor (K-NN) Spatial Neural Network* since the input for the prediction task at a location v is the set of sensory readings from the K nearest monitors to v and their relative locations in terms of distances and positions, where K is a positive number.

The equation denoting the function to approximate is similar to that for the MPRNN, except that we restrict the number of input sensor locations to K, where K is a parameter: $y_{v,t} = \mathcal{F}(v_1, y_{v_1,t}, v_2, y_{v_2,t}, \ldots, v_K, y_{v_K,t}))$ In this equation, v_j denotes the j^{th} nearest neighbor to v. K is the maximum number of neighboring sensors that can provide input sensory data. Note that the set of nearest neighbors $v_1, v_2, \ldots v_K$ and K itself are functions of time, since at time t, these are the sensors at which data is available and the number of such sensors, respectively. For each input sensor at location v_j, we add as feature the triple of the sensor reading, the geodesic distance between v and v_j, the compass bearing of v_j with respect to v. The length of the feature vector is thus $3K$, where K is the number of available sensors at that time.

History length: While a recurrent neural network is capable of predicting labels on a rolling basis, computing and backpropagating a loss function through an arbitrarily long history is not feasible. As a result, our recurrent neural network is trained on segments of fixed history length.

That is, the state of the neural network is persisted through a number of points equal to the history length, and then reset. We experimented with several different reasonable history lengths during training and chose a history of 8 hours (32 measurements, at one reading per 15 minutes). Each training example consisted of a block of history length $H = 32$ collected at time step t for a chosen set of sensors except for the target sensor.

Training: Since neural networks take only fixed length input feature vectors, our formulation required the training of several models, one for each value of K. We trained a total of 10 models, for K from 1 to 10. By combining these 10 different models, we obtain a *master model*, that generalizes to predicting $y_{v,t}$ at any location v at a given time, regardless of the number of available neighboring input sensors at the time, using data from up to a maximum of 10 available input sensors. For each value of K, and for each sensor v in our set, we extracted blocks of available data of length $H = 32$ through the entire year from the K nearest neighbors to v. Then we merged all the blocks together for each version and each value of K, thus giving us a large dataset of training samples mapping K sensor readings to output sensor values over the history length H (i.e. a sample consisted of a block of dimensions $H \times K$ or $H \times 3K$ depending on the version). The list of samples was then shuffled prior to feeding into the neural network to reduce the chances of the optimization algorithm becoming stuck in local optima and and also increase the test prediction performance. We repeat this mechanism for every value of K, giving us totally 10 models for each version.

We needed at least 20 epochs for convergence. With a total of at least 2000 batches for every value of K, the training for each K and version took prohibitively large amount of time (several days). Hence we resorted to reduction of training time by selecting only a smaller number of samples from the entire corpus of samples for each K and version The shuffling of training samples thus allowed us to "effectively" reduce our training time and yet not lose generality since the shuffling ensured that data from all round the year was utilized for training. We used the `pytorch` [pyt] library in Python for implementing the neural network and the Adam opti-

mizer [Kingma and Ba 2015] for training.

Bibliography

Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. https://pytorch.org/.

Avinash Achar, Venkatesh Sarangan, Rohith Regikumar, and Anand Sivasubramaniam. Predicting vehicular travel times by modeling heterogeneous influences between arterial roads. In *The 32nd AAAI Conference on Artificial Intelligence*, 2018.

Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. I-pic: A platform for privacy-compliant image capture. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 235–248, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342698. doi: 10.1145/2906388.2906412. URL https://doi.org/10.1145/2906388.2906412.

Pinhas Alpert, Olga Shvainshtein, and Pavel Kishcha. AOD trends over megacities based on space monitoring using MODIS and MISR. *American Journal of Climate Change*, 01(03):117–131, 2012. doi: 10.4236/ajcc.2012.13010.

Joshua S. Apte, Kyle P. Messier, Shahzad Gani, Michael Brauer, Thomas W. Kirchstetter, Melissa M. Lunden, Julian D. Marshall, Christopher J. Portier, Roel C.H. Vermeulen, and Steven P. Hamburg. High-Resolution Air Pollution Mapping with Google Street View Cars: Exploiting Big Data. *Environmental Science & Technology*, 51(12):6999–7008, June 2017. ISSN 0013-936X.

doi: 10.1021/acs.est.7b00891. URL https://doi.org/10.1021/acs.est.7b00891. Publisher: American Chemical Society.

Emily Badger. How traffic congestion affects economic growth. https://www.bloomberg.com/news/articles/2013-10-22/how-traffic-congestion-affects-economic-growth, 2013.

Dave Barth. The bright side of sitting in traffic: Crowdsourcing road congestion data, Aug 2009. URL https://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html.

A. Beloconi, N. Chrysoulakis, A. Lyapustin, J. Utzinger, and P. Vounatsou. Bayesian geostatistical modelling of PM10 and PM2.5 surface level concentrations in Europe using high-resolution satellite-derived products. *Environment International*, 121:57–70, 2018.

Jianzhao Bi, Nancy Carmona, Magali N. Blanco, Amanda J. Gassett, Edmund Seto, Adam A. Szpiro, Timothy V. Larson, Paul D. Sampson, Joel D. Kaufman, and Lianne Sheppard. Publicly available low-cost sensor measurements for pm2.5 exposure modeling: Guidance for monitor deployment and data selection. *Environment International*, 158:106897, 2022. ISSN 0160-4120. doi: https://doi.org/10.1016/j.envint.2021.106897. URL https://www.sciencedirect.com/science/article/pii/S0160412021005225.

Srinivas Bikkina, August Andersson, Elena N. Kirillova, Henry Holmstrand, Suresh Tiwari, A. K. Srivastava, D. S. Bisht, and Örjan Gustafsson. Air quality in megacity Delhi affected by countryside biomass burning. *Nature Sustainability*, 2(3):200–205, March 2019. ISSN 2398-9629. doi: 10.1038/s41893-019-0219-0. URL https://www.nature.com/articles/s41893-019-0219-0. Number: 3.

M. Cameletti, R. Ignaccolo, and S. Bande. Comparing spatio-temporal models for particulate matter in Piemonte. *Environmetrics*, 22(8):985–996, 2011.

M. Cameletti, F. Lindgren, D. Simpson, and H. Rue. Spatio-temporal modeling of particulate matter concentration through the SPDE approach. *AStA Advances in Statistical Analysis*, 97(2): 109–131, 2013.

Sarah E. Chambliss, Carlos P.R. Pinon, Kyle P. Messier, Brian LaFranchi, Crystal Romeo Upperman, Melissa M. Lunden, Allen L. Robinson, Julian D. Marshall, and Joshua S. Apte. Local- and regional-scale racial and ethnic disparities in air pollution determined by long-term mobile monitoring. *Proceedings of the National Academy of Sciences*, 118(37), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2109249118. URL https://www.pnas.org/content/118/37/e2109249118.

Yuan-Lin Chen and Chong-An Wang. Vehicle safety distance warning system: A novel algorithm for vehicle safety distance calculating between moving cars. In *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, pages 2570–2574, 2007. doi: 10.1109/VETECS.2007.529.

Hone-Jay Chu, Muhammad Zeeshan Ali, and Yu-Chen He. Spatial calibration and pm 2.5 mapping of low-cost air quality sensors. *Scientific reports*, 10(1):1–11, 2020.

Andrea L Clements, William G Griswold, Jill E Johnston, Megan M Herting, Jacob Thorson, Ashley Collier-Oxandale, and Michael Hannigan. Low-cost air quality monitoring tools: From research to practice (a workshop summary). *Sensors*, 17(11):2478, 2017.

N. Cressie and C. K. Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, Hoboken, NJ, 2011.

Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *arXiv preprint arXiv:1802.07007*, 2018.

Daniel H Cusworth, Loretta J Mickley, Melissa P Sulprizio, Tianjia Liu, Miriam E Marlier, Ruth S DeFries, Sarath K Guttikunda, and Pawan Gupta. Quantifying the influence of agricultural

fires in northwest india on urban air pollution in delhi, india. *Environmental Research Letters*, 13(4):044018, mar 2018. doi: 10.1088/1748-9326/aab303. URL https://doi.org/10.1088%2F1748-9326%2Faab303.

Paul J. Ferraro and Arun Agrawal. Synthesizing evidence in sustainability science through harmonized experiments: Community monitoring in common pool resources. *Proceedings of the National Academy of Sciences*, 118(29), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2106489118. URL https://www.pnas.org/content/118/29/e2106489118.

Jordan Friedman. 10 cities with the worst traffic in the world. https://www.usnews.com/news/cities/slideshows/cities-with-the-worst-traffic-in-the-world?slide=12, 2020.

Meiling Gao, Junji Cao, and Edmund Seto. A distributed network of low-cost continuous reading sensors to measure spatiotemporal variations of pm2. 5 in xi'an, china. *Environmental pollution*, 199:56–65, 2015.

Guannan Geng, Yixuan Zheng, Qiang Zhang, Tao Xue, Hongyan Zhao, Dan Tong, Bo Zheng, Meng Li, Fei Liu, Chaopeng Hong, et al. Drivers of pm2. 5 air pollution deaths in china 2002–2017. *Nature Geoscience*, 14(9):645–650, 2021.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1263–1272, 2017.

Michael R. Giordano, Carl Malings, Spyros N. Pandis, Albert A. Presto, V.F. McNeill, Daniel M. Westervelt, Matthias Beekmann, and R. Subramanian. From low-cost sensors to high-quality data: A summary of challenges and best practices for effectively calibrating low-cost particulate matter mass sensors. *Journal of Aerosol Science*, 158:105833, 2021. ISSN 0021-8502. doi: https://doi.org/10.1016/j.jaerosci.2021.105833. URL https://www.sciencedirect.com/science/article/pii/S0021850221005644.

A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition, 2008.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, 2019.

Joanne L. Harbluk, Y. Ian Noy, Patricia L. Trbovich, and Moshe Eizenman. An on-road assessment of cognitive distraction: Impacts on drivers' visual behavior and braking performance. *Accident Analysis & Prevention*, 39(2):372–379, 2007. ISSN 0001-4575. doi: 10.1016/j.aap.2006.08.013.

A. C. Harvey. *Forecasting, structural time series models, and the Kalman filter*. Cambridge University Press, Cambridge, UK, 1989.

Hindustan Times. Delhi to get 10 more pollution monitoring stations by next winter, 2017. URL https://www.hindustantimes.com/delhi-news/delhi-to-get-10-more-pollution-monitoring-stations-by-next-winter/story-Ipal3yRk71AMNrPCtW86iL.html. [Online; accessed 14-May-2018].

S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997a.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, nov 1997b. doi: 10.1162/neco.1997.9.8.1735.

Kellie Hodge. Guide to driving in Kenya. https://www.rhinocarhire.com/Drive-Smart-Blog/Drive-Smart-Kenya.aspx.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, jan 1989. doi: 10.1016/0893-6080(89)90020-8.

S. R. Iyer, U. An, and L. Subramanian. Forecasting sparse traffic congestion patterns using message-passing rnns. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3772–3776, 2020. doi: 10.1109/ICASSP40776.2020.9052963.

Vipin Jain, Ashlesh Sharma, and Lakshminarayanan Subramanian. Road traffic congestion in the developing world. In *Proceedings of the 2nd ACM Symposium on Computing for Development*, ACM DEV '12, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312622. doi: 10.1145/2160601.2160616. URL https://doi.org/10.1145/2160601.2160616.

Charles F. Jekel and Gerhard Venter. *pwlf: A Python Library for Fitting 1D Continuous Piecewise Linear Functions*, 2019. URL https://github.com/cjekel/piecewise_linear_fit_py.

Michael Jerrett, Altaf Arain, Pavlos Kanaroglou, Bernardo Beckerman, Dimitri Potoglou, Talar Sahsuvaroglu, Jason Morrison, and Chris Giovis. A review and evaluation of intraurban air pollution exposure models. *Journal of Exposure Science and Environmental Epidemiology*, 15(2):185, 2005.

Wan Jiao, Gayle Hagler, Ronald Williams, Robert Sharpe, Ryan Brown, Daniel Garver, Robert Judge, Motria Caudill, Joshua Rickard, Michael Davis, et al. Community air sensor network (cairsense) project: evaluation of low-cost sensor performance in a suburban environment in the southeastern united states. *Atmospheric Measurement Techniques*, 9(11):5281, 2016.

B. S. Kerner and H. Rehborn. Experimental features and characteristics of traffic jams. *Physical*

Review E, 53(2):R1297–R1300, February 1996. doi: 10.1103/PhysRevE.53.R1297. URL https://link.aps.org/doi/10.1103/PhysRevE.53.R1297. Publisher: American Physical Society.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Victor L. Knoop and Winnie Daamen. Automatic fitting procedure for the fundamental diagram. *Transportmetrica B: Transport Dynamics*, 5(2):129–144, 2017. doi: 10.1080/21680566.2016.1256239. URL https://doi.org/10.1080/21680566.2016.1256239.

K. Kristensen, A. Nielsen, C. W. Berg, H. J. Skaug, and B. M. Bell. TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5):1–21, 2016.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Yutong Liang, Deep Sengupta, Mark J. Campmier, David M. Lunderberg, Joshua S. Apte, and Allen H. Goldstein. Wildfire smoke impacts on indoor air quality assessed using crowdsourced data in california. *Proceedings of the National Academy of Sciences*, 118(36), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2106478118. URL https://www.pnas.org/content/118/36/e2106478118.

C Lin, J Gillespie, MD Schuder, W Duberstein, IJ Beverland, and MR Heal. Evaluation and calibration of aeroqual series 500 portable gas sensors for accurate measurement of ambient ozone and nitrogen dioxide. *Atmospheric Environment*, 100:111–116, 2015.

F. Lindgren, H. Rue, and J. Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society, Series B*, 73(4):423–498, 2011.

M. Lippi, M. Bertini, and P. Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, June 2013. ISSN 1524-9050.

Hai-Ying Liu, Philipp Schneider, Rolf Haugen, and Matthias Vogt. Performance assessment of a low-cost pm2. 5 sensor for a near four-month period in oslo, norway. *Atmosphere*, 10(2):41, 2019.

Tianjia Liu, Miriam E. Marlier, Ruth S. DeFries, Daniel M. Westervelt, Karen R. Xia, Arlene M. Fiore, Loretta J. Mickley, Daniel H. Cusworth, and George Milly. Seasonal impact of regional outdoor biomass burning on air pollution in three indian cities: Delhi, bengaluru, and pune. *Atmospheric Environment*, 172:83–92, 2018. ISSN 1352-2310. doi: https://doi.org/10.1016/j.atmosenv.2017.10.024. URL https://www.sciencedirect.com/science/article/pii/S1352231017306854.

Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xing Xie. Discovering spatio-temporal causal interactions in traffic data streams. In *17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2011)*, August 2011.

Xiaoting Liu, Rohan Jayaratne, Phong Thai, Tara Kuhn, Isak Zing, Bryce Christensen, Riki Lamont, Matthew Dunbabin, Sicong Zhu, Jian Gao, David Wainwright, Donald Neale, Ruby Kan, John Kirkwood, and Lidia Morawska. Low-cost sensors as an alternative for long-term air quality monitoring. *Environmental Research*, 185:109438, 2020. ISSN 0013-9351. doi: https://doi.org/10.1016/j.envres.2020.109438. URL https://www.sciencedirect.com/science/article/pii/S0013935120303315.

Josef Ludescher, Maria Martin, Niklas Boers, Armin Bunde, Catrin Ciemer, Jingfang Fan, Shlomo Havlin, Marlene Kretschmer, Jürgen Kurths, Jakob Runge, Veronika Stolbova, Elena Surovyatkina, and Hans Joachim Schellnhuber. Network-based forecasting of climate phenomena. *Proceedings of the National Academy of Sciences*, 118(47), 2021. ISSN 0027-8424. doi: 10.1073/pnas.1922872118. URL https://www.pnas.org/content/118/47/e1922872118.

Zhongjian Lv, Jiajie Xu, Kai Zheng, Hongzhi Yin, Pengpeng Zhao, and Xiaofang Zhou. LC-RNN: A deep learning model for traffic speed prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3470–3476. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communication Review*, 32:62–73, 2002.

Sachit Mahajan and Prashant Kumar. Evaluation of low-cost sensors for quantitative personal exposure monitoring. *Sustainable Cities and Society*, 57:102076, 2020. ISSN 2210-6707. doi: https://doi.org/10.1016/j.scs.2020.102076. URL https://www.sciencedirect.com/science/article/pii/S2210670720300639.

A.D. May. *Traffic Flow Fundamentals*. Prentice Hall, 1990. ISBN 9780139260728. URL https://books.google.com/books?id=JYJPAAAAMAAJ.

Adolf May. *Traffic Flow Fundamentals*. Prentice Hall, 2000.

MOEF. National clean air programme india, 2018.

Sharon Moltchanov, Ilan Levy, Yael Etzion, Uri Lerner, David M Broday, and Barak Fishbain. On the feasibility of measuring urban air pollution by wireless distributed sensor networks. *Science of The Total Environment*, 502:537–547, 2015.

Erick Kombo Ndubi. List of nairobi city roads where motorists are not allowed to exceed 50 kph speed limit. https://www.tuko.co.ke/318251-list-nairobi-city-roads-motorists-allowed-exceed-50kph-speed-limit.html, 2019.

David Nesbitt and Drew Dara-Abrams. Osmlr traffic segments for the entire planet by mapzen. https://www.mapzen.com/blog/osmlr-2nd-technical-preview/, 2017.

New York City Department of Transport. Data feeds. https://www1.nyc.gov/html/dot/html/about/datafeeds.shtml.

NYC MTA. MTA Bus Time Historical Data. http://web.mta.info/developers/MTA-Bus-Time-historical-data.html, 2014.

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

Pallavi Pant, Gazala Habib, Julian D. Marshall, and Richard E. Peltier. PM2.5 exposure in highly polluted cities: A case study from New Delhi, India. *Environmental Research*, 156:167–174, July 2017. ISSN 0013-9351. doi: 10.1016/j.envres.2017.03.024. URL http://www.sciencedirect.com/science/article/pii/S0013935116313329.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Jai Prakash, Shruti Choudhary, Ramesh Raliya, Tandeep S. Chadha, Jiaxi Fang, and Pratim Biswas. Real-time source apportionment of fine particle inorganic and organic constituents at an urban site in delhi city: An iot-based approach. *Atmospheric Pollution Research*, 12(11): 101206, 2021. ISSN 1309-1042. doi: https://doi.org/10.1016/j.apr.2021.101206. URL https://www.sciencedirect.com/science/article/pii/S1309104221002701.

Haoqi Qian, Shaodan Xu, Jing Cao, Feizhou Ren, Wendong Wei, Jing Meng, and Libo Wu. Air pollution reduction and climate co-benefits in china's industries. *Nature Sustainability*, 4(5): 417–425, 2021.

Narasimha D Rao, Gregor Kiesewetter, Jihoon Min, Shonali Pachauri, and Fabian Wagner. Household contributions to and impacts from air pollution in india. *Nature Sustainability*, pages 1–9, 2021.

Joseph Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988. ISSN 00031305.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society, Series B*, 71(2):319–392, 2009.

R. Sen, P. Siriah, and B. Raman. Roadsoundsense: Acoustic sensing based road congestion monitoring in developing regions. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 125–133, 2011. doi: 10.1109/SAHCN.2011.5984883.

Rijurekha Sen, Bhaskaran Raman, and Prashima Sharma. Horn-ok-please. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, page 137–150, New York, NY, USA, 2010. Association for Computing Machinery. ISBN

9781605589855. doi: 10.1145/1814433.1814449. URL https://doi.org/10.1145/1814433.1814449.

Rijurekha Sen, Abhinav Maurya, Bhaskaran Raman, Rupesh Mehta, Ramakrishnan Kalyanaraman, Nagamanoj Vankadhara, Swaroop Roy, and Prashima Sharma. kyun queue: A sensor network system to monitor road traffic queues. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, page 127–140, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311694. doi: 10.1145/2426656.2426670. URL https://doi.org/10.1145/2426656.2426670.

Rijurekha Sen, Andrew Cross, Aditya Vashistha, Venkata N. Padmanabhan, Edward Cutrell, and William Thies. Accurate speed and density measurement for road traffic in india. In *Proceedings of the 3rd ACM Symposium on Computing for Development*, ACM DEV '13, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318563. doi: 10.1145/2442882.2442901. URL https://doi.org/10.1145/2442882.2442901.

Toru Seo, Alexandre M. Bayen, Takahiko Kusakabe, and Yasuo Asakura. Traffic state estimation on highway: A comprehensive survey. *Annual Reviews in Control*, 43:128–151, 2017.

G Shaddick, ML Thomas, P Mudu, G Ruggeri, and S Gumy. Half the world's population are exposed to increasing air pollution. *NPJ Climate and Atmospheric Science*, 3(1):1–5, 2020.

Mukesh Sharma and Onkar Dikshit. Comprehensive study on air pollution and green house gases (GHGs) in Delhi. Technical report, IIT Kanpur, Jan 2016. URL https://cerca.iitd.ac.in/uploads/Reports/1576211826iitk.pdf.

Alexis A Shusterman, Virginia E Teige, Alexander J Turner, Catherine Newman, Jinsol Kim, and Ronald C Cohen. The berkeley atmospheric co 2 observation network: initial evaluation. *Atmospheric Chemistry and Physics*, 16(21):13449–13463, 2016.

Bruce G Simons-Morton, Marie Claude Ouimet, Jing Wang, Sheila G Klauer, Suzanne E Lee, and Thomas A Dingus. Hard braking events among novice teenage drivers by passenger characteristics. In *International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design*, volume 2009, pages 236–242, 2009.

Georgios C. Spyropoulos, Panagiotis T. Nastos, and Konstantinos P. Moustris. Performance of aether low-cost sensor device for air pollution measurements in urban environments. accuracy evaluation applying the air quality index (aqi). *Atmosphere*, 12(10), 2021. ISSN 2073-4433. doi: 10.3390/atmos12101246. URL https://www.mdpi.com/2073-4433/12/10/1246.

Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, mar 2008. doi: 10.1088/1367-2630/10/3/033001. URL https://doi.org/10.1088/1367-2630/10/3/033001.

Li Sun, Ka Chun Wong, Peng Wei, Sheng Ye, Hao Huang, Fenhuan Yang, Dane Westerdahl, Peter KK Louie, Connie WY Luk, and Zhi Ning. Development and application of a next generation air sensor network for the hong kong marathon 2015 air quality monitoring. *Sensors*, 16(2):211, 2016.

Leandros Tassiulas. Adaptive back-pressure congestion control based on local information. *IEEE Transactions on Automatic Control*, 40:236–250, 1995.

Texas Transportation Institute. 2009 Urban Mobility Report. http://mobility.tamu.edu/ums/.

Kushal Tibrewal and Chandra Venkataraman. Climate co-benefits of air quality and clean energy policy in india. *Nature Sustainability*, 4(4):305–313, 2021.

TomTom International BV. Full ranking 2020. https://www.tomtom.com/en_gb/traffic-index/ranking/.

Jessica Tryner, Mollie Phillips, Casey Quinn, Gabe Neymark, Ander Wilson, Shantanu H. Jathar, Ellison Carter, and John Volckens. Design and testing of a low-cost sensor and sampling platform for indoor air quality. *Building and Environment*, 206:108398, 2021. ISSN 0360-1323. doi: https://doi.org/10.1016/j.buildenv.2021.108398. URL https://www.sciencedirect.com/science/article/pii/S0360132321007952.

Wataru Tsujita, Akihito Yoshino, Hiroshi Ishida, and Toyosaka Moriizumi. Gas sensor network for air-pollution monitoring. *Sensors and Actuators B: Chemical*, 110(2):304–311, 2005.

U. S. Environmental Protection Agency (EPA). AERMOD implementation guide. *Office of Air Quality Planning and Standards*, (EPA-454/B-21-006), July 2021. URL https://gaftp.epa.gov/Air/aqmg/SCRAM/models/preferred/aermod/aermod_implementation_guide.pdf.

Uber Technologies. Uber movement speeds. https://movement.uber.com/.

Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43, Part 1:3–19, 2014. ISSN 0968-090X. doi: 10.1016/j.trc.2014.01.005. Special Issue on Short-term Traffic Flow Forecasting.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019. URL https://www.dgl.ai/.

WHO, UNAIDS, et al. *Air quality guidelines: global update 2005*. World Health Organization, 2006.

Ning Wu. A new approach for modeling of fundamental diagrams. *Transportation Research Part A: Policy and Practice*, 36(10):867–884, 2002. ISSN 0965-8564. doi: https://doi.org/10.

1016/S0965-8564(01)00043-X. URL https://www.sciencedirect.com/science/article/pii/S096585640100043X.

Xingzhe Xie, Ivana Semanjski, Sidharta Gautama, Evaggelia Tsiligianni, Nikos Deligiannis, Raj Rajan, Frank Pasveer, and Wilfried Philips. A review of urban air pollution monitoring and exposure assessment methods. *ISPRS International Journal of Geo-Information*, 6(12):389, Dec 2017. ISSN 2220-9964. doi: 10.3390/ijgi6120389. URL http://dx.doi.org/10.3390/ijgi6120389.

Christopher Yeh, Anthony Perez, Anne Driscoll, George Azzari, Zhongyi Tang, David Lobell, Stefano Ermon, and Marshall Burke. Using publicly available satellite imagery and deep learning to understand economic well-being in africa. *Nature communications*, 11(1):1–11, 2020.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, July 2018.

Resty Woro Yuniar. Google maps: a lost cause for indonesian drivers, Apr 2018. URL http://www.scmp.com/week-asia/society/article/2139712/you-have-not-arrived-why-google-maps-lost-cause-indonesian-drivers.

Marina Zusman, Cooper S. Schumacher, Amanda J. Gassett, Elizabeth W. Spalt, Elena Austin, Timothy V. Larson, Graeme Carvlin, Edmund Seto, Joel D. Kaufman, and Lianne Sheppard. Calibration of low-cost particulate matter sensors: Model development for a multi-city epidemiological study. *Environment International*, 134:105329, 2020. ISSN 0160-4120. doi: https://doi.org/10.1016/j.envint.2019.105329. URL https://www.sciencedirect.com/science/article/pii/S0160412019321920.