Road Traffic prediction using Programmable IoT Platform

Ish Kumar Jain
ishjain@nyu.edu
New York University

***Abstract*— We present a programmable IoT platform that can be used in general for any distributed sensor network. It constitutes several IoT devices distributed in a geographical area and are connected to a central controller via Internet. We analyzed the performance of the network model in Mininet emulator. The major part of this work is based on an application of our IoT platform for road traffic inference. We used mobile phones as sensors placed in many vehicles running on city roads to collect the location information of the vehicles and send it to the back-end server. The key idea that distinguishes our work from literature is that the programmable server/controller can send control messages to the client (mobile phones) regarding the operation of sensors or installation of software on the fly. For example, the commands can be start/stop sensing, sampling rates *etc*. This helps in better management and durability of sensors, especially when the sensors are battery hungry and using limited data plans. The server implemented in a cloud can perform the traffic prediction and inference to build the traffic map of a city using sampled and sparse data. Initial result shows promises of implementation of such system on the ground.**

I. INTRODUCTION

Technically, the term IoT platform refers to an IoT infrastructure which constitutes of many devices connected to each other and to a back-end server via the Internet. An IoT platform enables deployment of an application that monitor, manage, and control connected devices. One main component of the IoT platform is the IoT devices which are connected to other IoT devices and applications to relay information via internet. These devices contain various sensors that collect the data and send it to the server, as well as a mechanism to perform various actions based on the command received by the server. These devices are deployed for a various range of application including remote surveillance, distributed sensing to collect data, and daily use devices that make our life smarter. The IoT infrastructure for distributed sensing where autonomous sensors are spatially distributed to monitor physical or environmental conditions is called a Wireless Sensor Network (WSN). The sensors are connected to a server via internet to send data and receive control information. One important application of WSN is Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. It is a network of moving vehicles which are providing each other the information such as traffic congestion, road condition, safety warnings etc. The vehicles can be connected to each other in an ad-hoc manner known as vehicular ad-hoc networks (VANETs) or they can be connected to an infrastructure (a controller) as well. A central controller helps in better management and control of the distributed vehicles in the network. It is further powered by Software Defined Networking (SDN) and Network Function Virtualization (NFV) which provides network programmability. The network is becoming more flexible to support many IoT devices, and also helps in their control and management. Further, the controller can be effectively implemented in a cloud platform.

The mobile phones are now becoming ubiquitous and are a good source of collecting data. Many works have proposed the use of cell phone sensors like GPS, accelerometer, WiFi as well as external sensors connected to the phone via Bluetooth. VTrack [20] uses mobile phone to collect GPS and WiFi sensor data and use it for road traffic delay estimation. Escort [7] is another application using mobile phone for human localization i.e. escorting a person to other person in a conference or hotel indoor environment.

The major challenges in implementation of such IoT platform is as follow-

1) **The sensors are battery hungry:** Most of the sensors are power hungry if remain active for long time. For instance, when GPS is on and active, it consumes around 150 mWatt of power in android phones [6].
2) **Internet Data Usage:** When mobile phones continuously send the sensor data to the server, they can exhaust the data plan. The 4G internet typically costs around $10 for 1Gb data usage. The mobile phones need to minimize the data sent over the network.

There are many papers that provide a solution to the above-mentioned challenges. Their technique revolves around the idea that the sensor data should be collected very sparsely. For instance, most of the sensors should remain inactive for a long time to save the power. Also, the data need to be sampled over time to reduce the total amount of data sent over the Internet.

We provide a solution to these challenges through our programmable IoT platform. We have a controller that collects the information about the health of a mobile phone. The health is decided based on the battery level, whether battery is in charging mode or not, the history of data usage *etc*. The controller has the information of about the health of all the phones in the system. It can then send the control messages to each of the phone to direct when to start/stop sensing, sampling frequency, and a list of active sensor *etc*.

An Android app has been developed by Soumie *et. al.* [future paper] for collecting the sensor data on the command of the controller. The controller logic has also been written and implemented by Shiva *et. al.* [future paper]. However, a

large scale implementation is very impractical for research purpose. It is hard to convince many users to download the app on their phone and run a long experimentation. An engineering approach is to simulate the whole setup on a computer for experiments and trial runs. There are many platforms that allow us to simulate the network [3]. For example ns-3, Netkit, Mininet, Marionnet, IMUNES, CORE, Cloonix *etc*.

In this paper, we used Mininet network emulator which is generally used in education and research as a learning and prototyping tool. It is an open-source Python based simulator that runs real software on the components of the network. It creates a virtual machine for every hosts and switches and provides a controller to manage the network. It provides support for the programmability of the host and controller software. It is traditionally built for wired networks, but can be extended to wireless network scenarios as well.

The Mininet code is provided by Michael Corso and Ben Cullaj [1]. The architecture is shown in Figure 1 and the key features are explained as follow

- There is a Controller which opens two connections with every client— a DataManager and a ClientManager.
- The ClientManager is for exchanging control messages like start/stop sensing, sensing rates, select a sensor etc.
- The DataManager actually collects the data, analyse and store it in a database (txt file for now).
- It also does round-trip-time calculation and network analysis.

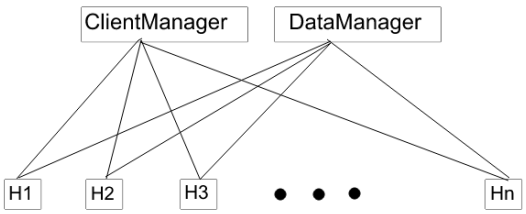

Fig. 1: Mininet architecture.

The second part of this work is based on an application of our IoT platform for road traffic inference using mobile devices.

Some apps like Sigalert [4] predicts the traffic map from the images obtained from the camera placed around the main segments of a city. But, this method is not scalable, requires large infrastructure cost and not very accurate (limited by the field of view of the camera and the image processing techniques). Waze [5], which was acquired by Google in 2013, is based on community crowd-sourcing, where a Waze user report about traffic, accidents, road condition *etc* to Waze. It also collects the location info from its users to predict the live traffic map. But it differs from our approach in the way that it does not have a full control of the mobile device and it is not optimized to solve the problem of battery and data usage.

We proposed a community driven traffic inference which is also optimized to use low battery power and low data usage to build a complete traffic map of a city.

The rest of the paper is organized as follow. We first discussed the related work in Section II and followed by our approach in Section III. We provide the data collection and refinement approach in Section IV and the results in Section V. Finally, we conclude our work in Section VI.

II. Related Work

The traffic inference problem has been studied for a long time. The data is generally given is a time series form of vehicle volume, average speed, or average travel time. [16] is one of the early paper that uses Kalman filtering to predict average vehicle volume on a road segment based on previous data and a flow model, and it corrects the traffic value based on actual measurement to predict future value. [22] extended this approach to a general stochastic traffic flow model and using Extended Kalman Filter. Another paper [18] suggests that the traffic is very different in different times of a day and utilizes a multivariate time-series state space models for predicting road traffic.

Other methods use machine learning techniques that utilize the spatiotemporal correlation from the past samples to predict future traffic. The early work includes non parametric approach to curve estimation from time-series data using k-nearest neighbour method [8]. A local linear regression with kernel method based traffic prediction is presented in [19]. [13] combined the above two methods and proposed a composite approach of traffic prediction. A support vector regression based approach is used in [23]. [11] is among the first few papers that utilizes the spatial correlation especially the distance from the predicting point into account. But it assumes a stationary spatial correlation. An improvement is suggested by [15] using not only the distance but also the average speed on the connecting links. It also uses other features into account such as weather conditions. [12] utilizes spatial correlation using Fuzzy clustering.

A local linear assumption to predicting road traffic fails when the traffic is highly non-linear. The non-linear approaches are studied under the umbrella of neural network techniques [14], [10]. [9] showed existing deterministic neural network models either suffer from low prediction accuracy or only work in a particular period. Traffic prediction using Hierarchical Bayesian Networks is given is [25].

Another approach involves clustering techniques [17] to group sites that exhibit similar traffic conditions. The motion patterns are learned by pairwise similarities of sites in a cluster. [24] first built the clusters based on similar traffic applied neural network based technique cluster by cluster by message passing to predict traffic in all clusters.

The most similar paper is [21] where the authors have used the mobile crowd sensing data for traffic prediction. It utilizes a Vehicle-to-Infrastructure (V2I) with cloud platform to process and manage the large amount of data collected by mobile phones and provide real-time, efficient and reliable services. However, this work does not consider the control of

mobile phones to provide commands (like start/stop sensing, sampling rates) based on battery level and data usage. Also, its focus is less on traffic prediction but more on applying the the graphical models in V2V communication to help reduce the accident rates.

III. Our Approach for Traffic Inference

Our approach to traffic inference is based on the system model that the mobile phones are placed in a moving vehicle and are controlled by a controller. The controller can send commands like start/stop sensing to individual devices based on their battery level and past data usage. In this scenario, we will have a highly sparse and sampled data from many disjoint segments in the road map. This puts a challenge in predicting traffic using spatio-temporal correlations in a selective way.

To solve the traffic inference problem in our setting, our initial approach is to compare some basic techniques for a benchmark. We have used Local linear regression and quadratic regression applied in both spatial and temporal domain to predict future traffic. We work on time-series data of speed samples a obtained across different road segments. Extensive simulations are done to compare these techniques under different settings which we explain in Section V. The Result shows good insights about traffic inference.

IV. Data Collection and Refinement

We used the MTA bus historical data [2] for our analysis. The data is very similar to the data being collected by our IoT platform. The buses are not very frequent and run on some specific paths, thus a good source of data based on our spatial and temporal sampling approach. The data set constitutes of the location, bus ID, bus number, direction, and location information at different times. The first step is to extract the useful data from this big dataset and refine it as per our need. The following steps are followed for data extraction and refinement-

- Segments are bounded by a rectangle. (Manually using Google Map).
- For each segment-
  1) Bus Location & Time is extracted.
  2) Speed is calculated from consecutive locations.
  3) Average is done for speed values in that segment.
  4) Linear Interpolation is done from past value to future value when data is not available.

One main challenge is the synchronization of a bus data with other bus data. We have the following problems-

1) Bus Arrival times- different and sparse
2) Frequency of arrival is different

So, for early simulations, we selected a specific bus M15 and identified two segments as shown in Figure 2.

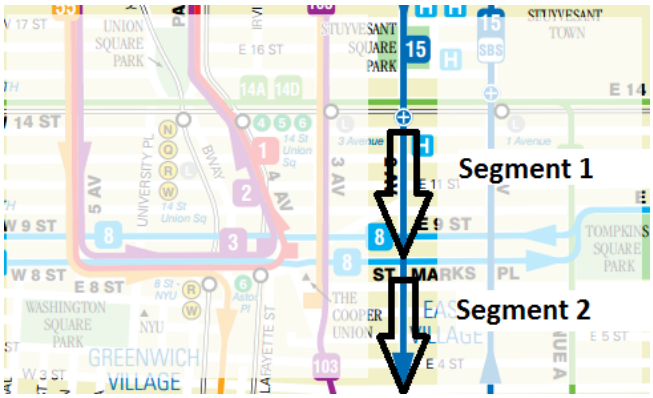

Fig. 2: The segment map of M15 bus on 2nd ave

We extracted the useful data and refine it to finally obtain a time-series sequence of average speed over two segments. We collected the data for four continuous week-days as shown in Figure 3. Since the bus frequency is every 20 minutes, the data is roughly sampled at the interval of 20 min. The zero values of speed indicates that the data is not available for that time instant. We have filled this speed with a value obtained by linear interpolation.

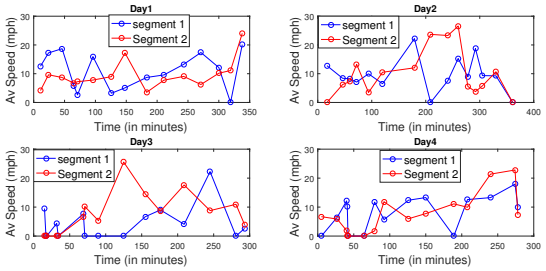

Fig. 3: Av Speed Data of two segments for 4 days (Monday through Thursday)

We also worked on another traffic data provide by Department of Transportation for traffic inference. This data is very rich and collected over a span of many years over some most congested segments of NYC. This data is available in MySQL format. The data extraction and refinement of this dataset is done as follow-

1) login to mysql database and identify a road segment number.
2) Extract the average speed sampled for every 1 min from the dataset for that segment.

By this process, we extracted time series data of five segments for our analysis. These segments are mostly around and including Brooklyn Bridge and Manhattan Bridge. The segments are shown in Figure 4 and the time-series data for Av. speed and total travel time in a segment is shown in Figure 5. A close inspection of the traffic data over one of the five segment shown in Figure 6 gives us a good picture of traffic variation. It shows that the traffic was centered around 15-20 mph speed for initial 20 minutes. The speeds gradually increased to 30 mph in next 15 minutes due to a

bulk of traffic left the road segment and it is less congested. After some fluctuations, it again become congested at 10-15 mph av. speed at around 80 minutes from start. Finally, the new traffic is again cleared at around 120 minutes and av. speed increased to 30-35 mph. With this insight, we now analyze the data using some machine learning models in the next section.

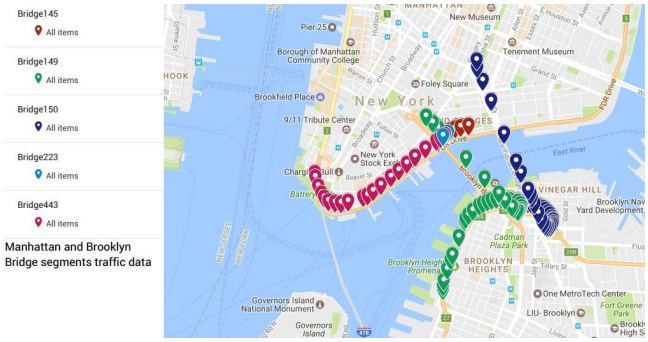

Fig. 4: Google Map showing Bridge segments in different colors.

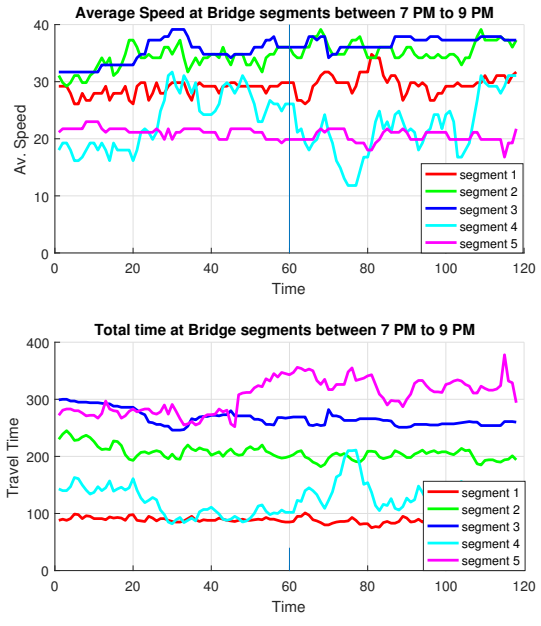

Fig. 5: Time-series data for bridge segments. Top figure shows Av. speed vs time and the bottom figure shown the Total travel time vs timestamp. We have one data sample available for any segment at a time

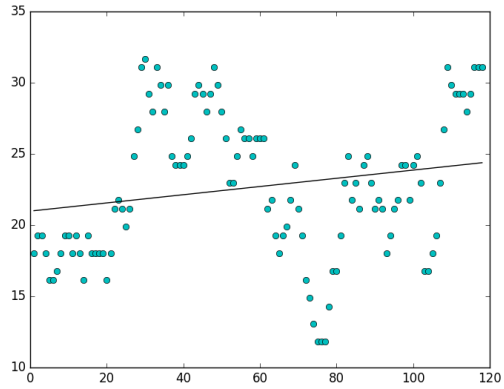

Fig. 6: Speed vs time scatter plot of one of the Bridge data. Please ignore the line.

V. Results

A. MTA Bus Data

We first applied linear regression on MTA bus data. Remember that the data is represented as time-series sequence of av. speed of two continuous segments sampled at every 20 minutes. Our approach is as follow. We used past few samples to predict the future samples. The performance criteria is root mean square error (RMSE). Especially, we vary the number of past samples on x axis and plot the value of RMSE for future 1, 2 or 3 samples (shown by red, green and blue curves in every graph) as shown in Figure 7. We have a total of 9 plots in a 3x3 grid. The three columns represents the following condition-

- Column 1: Predicting Segment 2 traffic using Segment 2 only.
- Column 2: Predicting Segment 2 traffic using Segment 1 only.
- Column 3: Predicting Segment 2 traffic using both Segment 1 and Segment 2.

The three rows have the following interpretation-

- Row 1: Training using Day 1 data only.
- Row 2: Training using Day 1 and Day 2 data combined.
- Row 3: Training using Day 1, Day 2 and Day 3 data combined (same time).

We also show the value of RMSE for different settings in table I. We conclude the following form these results-

1) The RMSE decreases when increasing past samples, means that the av. speed is not correlated to very old data for more than 20 minutes before.
2) Predicting future 1 sample gives lower RMSE than predicting future 3rd sample. This is obvious insight since predicting future 1 hour later traffic is very inaccurate.
3) It shows little improvement in RMSE when adding neighbour segment data in training.
4) Including data for 3 days saturates the value of RMSE to around 10mph. This may be due to the large amount

of data, the linear regression gives a roughly constant line.

5) Same analysis for Quadratic regression in Figure 8 shows poor results as compared to Linear regression.

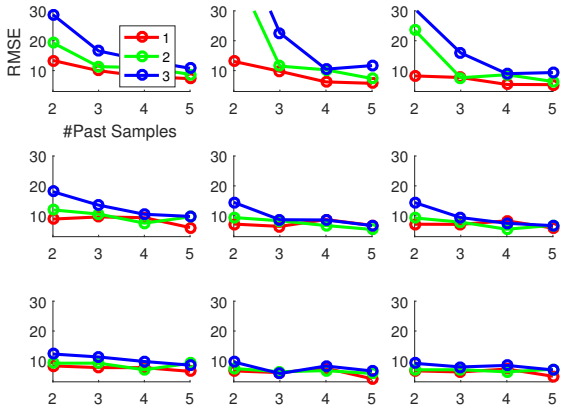

Fig. 7: Linear Regression on MTA bus data. The legend represents predicting future 1,2,3 samples under various conditions mentioned in text.

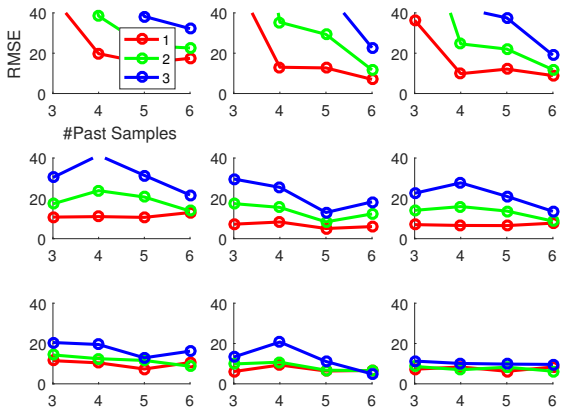

Fig. 8: Quadratic Regression on MTA bus data.

B. Bridge Data

The Linear regression and quadratic regression for Bridge data is shown in Figure 9 and 10 respectively. Here the three columns have the following interpretations. We are predicting Manhattan Bridge speeds based on-

- Column 1: Training using Man Bridge data
- Column 2: Training using Brooklyn Bridge data
- Column 3: Training using both Man and Brooklyn Bridge data

We interpret that varying the number of past samples has oscillating effect on RMSE, and the traffic of Brooklyn Bridge and Manhattan Bridge are not very related.

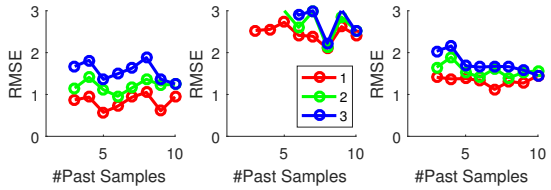

Fig. 9: Linear Regression on Bridge Data

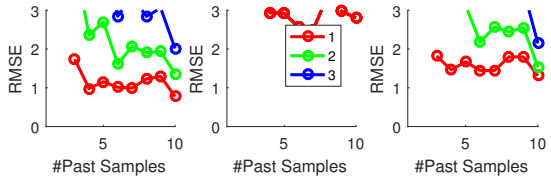

Fig. 10: Quadratic Regression on Bridge Data

C. Sampling of Training Data (Bridge Dataset)

Our final results shows the effect of sampling the data on prediction accuracy. The MTA bus data was already sampled at 20 minutes. So, we used the bridge data and varied the sampling rated from 1 min to 5 minutes as shown in Figure 11. The three columns have the same interpretation as the bridge data. We conclude that the increasing sampling actually increases the randomness in the prediction accuracy. Some of the sampled past traffic is exactly aligned with the linear regression result giving a very small error, but some points are becoming very off aligned providing a large prediction error.

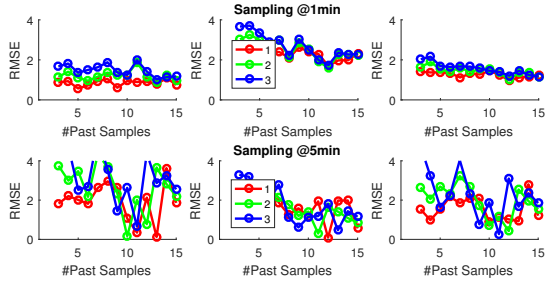

Fig. 11: Sampling at 1 minutes (first row) and sampling at 5 minutes (second row)

D. Logistic Regression on Bridge Dataset

We further analyzed Logistic regression, which is actually a classification problem. So, we quantized the speeds to many bins of size roughly 1 mile/hour. In our analysis, we used the data of last 5 speed samples to predict the next speed sample. This is done for future traffic prediction, which should be correlated with the past traffic. The error is defined as the percentage of misclassified points. In the Logistic regression framework, we have used, softmax activation function for classifying datapoints, negative log likelihood (NLL) function as cost function, and run the optimization using SGD for 1000 iterations with learning rate of 0.13. The results for different segments are summarized in Table

Day	Past Samples	Same Segment			Neighbour Only			Both		
		Future Samples								
		1	2	3	1	2	3	1	2	3
Day1 Only	2	13.39	19.3	28.72	13.08	44.5	58.2	8.218	23.54	30.7
	3	9.958	11.32	16.56	9.77	11.5	22.61	7.683	7.604	15.92
	4	8.032	10.99	13.39	6.215	10.25	10.48	5.362	8.62	8.963
	5	7.304	8.764	10.78	5.739	7.34	11.67	5.296	6.365	9.341

TABLE I: Linear Regression: RMSE values for different Past and future samples. The 3 cases are analyzed based on training done using the past samples of 1. same segment 2. Neighbour Segment 3. Both Segments.

II. We can see that the accuracy is below 40 % in all of the cases. We conclude that Logistic regression is not a good framework to analyze time-series data.

Segment#	**Train Loss**	**Test Loss**
1	63%	65%
2	84%	82%
3	73%	86%
4	84%	91%
5	65%	69%

TABLE II: Classification error on bridge data for 5 segments

Now, we analyze the effect of the size of training and testing dataset. We used different value of total datapoints (each consist of 5 dimensional example, 1D label). Every time 80% is used for training and the rest 20% for testing. The results summarized in Table III shows improvement in training accuracy with increasing dataset, but testing accuracy is fluctuating (may be due to smaller dataset). Here we have used the quantization bin size of 2 mph, which is another reason for high accuracy.

DataSize	20	40	60	80	100
Train Loss	19%	56%	37%	44%	30%
Test Loss	50%	25%	50%	38%	55%

TABLE III: Analysing the effect of varying dataset

VI. Conclusion and Future Work

We presented a Mininet testbed for implementing an IoT platform for many devices connected to a central controller. We had presented an idea of controlling and managing the IoT sensors using a controller and the IoT platform. We further demonstrated an application of our IoT platform in Road Traffic Inference. Our initial regression results in various settings showed a pattern in future speeds based on many different training scenarios. It can be further studied based on few techniques mention in Related Work section. In a nutshell, we can have the following work in future-

- Real-time traffic prediction on MTA Bus Data
- Use advanced ML Techniques:
  1) Generalized Linear Models. Non-linear models.
  2) Belief Propagation (Expectation Maximization)
  3) Bayesian Nets (Graphical Models)
  4) Arima Models (auto regressive time series model)
  5) Use Density instead of Speed.
- Network performance evaluation on Mininet.
- Data Collection using Mobile Phone (Android App-courtesy to Soumie)

References

[1] Michael corso and ben cullaj- mininet code. `https://github.com/shivariyer/NYU-Network-Iot-Project`. Accessed: 2017-05-19.

[2] Mta bus historical data. `http://web.mta.info/developers/MTA-Bus-Time-historical-data.html`. Accessed: 2017-05-19.

[3] Open source network simulators. `http://www.brianlinkletter.com/open-source-network-simulators/`. Accessed: 2017-05-19.

[4] Sigalert app. `https://www.sigalert.com/`. Accessed: 2017-05-19.

[5] Waze. `https://www.waze.com/livemap`. Accessed: 2017-05-19.

[6] Carroll, A., Heiser, G., et al. An analysis of power consumption in a smartphone. In *USENIX annual technical conference* (2010), vol. 14, Boston, MA, pp. 21–21.

[7] Constandache, I., Bao, X., Azizyan, M., and Choudhury, R. R. Did you see bob?: human localization using mobile phones. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking* (2010), ACM, pp. 149–160.

[8] Györfi, L., Härdle, W., Sarda, P., and Vieu, P. *Nonparametric curve estimation from time series*, vol. 60. Springer, 2013.

[9] Hall, J., and Mars, P. Limitations of artificial neural networks for traffic prediction in broadband networks. *IEE Proceedings-Communications 147*, 2 (2000), 114–118.

[10] Hodge, V. J., Krishnan, R., Jackson, T., Austin, J., and Polak, J. Short-term traffic prediction using a binary neural network. In *43rd Annual UTSG Conference* (2011), York.

[11] Kamarianakis, Y., and Prastacos, P. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 1857 (2003), 74–84.

[12] Kanoh, H., Furukawa, T., Tsukahara, S., Hara, K., Nishi, H., and Kurokawa, H. Short-term traffic prediction using fuzzy c-means and cellular automata in a wide-area road network. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE* (2005), IEEE, pp. 381–385.

[13] Kindzerske, M., and Ni, D. Composite nearest neighbor non-parametric regression to improve traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, 1993 (2007), 30–35.

[14] Ledoux, C. An urban traffic flow model integrating neural networks. *Transportation Research Part C: Emerging Technologies 5*, 5 (1997), 287–300.

[15] Min, W., and Wynter, L. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies 19*, 4 (2011), 606–616.

[16] Okutani, I., and Stephanedes, Y. J. Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological 18*, 1 (1984), 1–11.

[17] Papagiannakis, A., Bracher, M., and Jackson, N. Utilizing clustering techniques in estimating traffic data input for pavement design. *Journal of Transportation Engineering 132*, 11 (2006), 872–879.

[18] STATHOPOULOS, A., AND KARLAFTIS, M. G. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies 11*, 2 (2003), 121–135.

[19] SUN, H., LIU, H., XIAO, H., HE, R., AND RAN, B. Use of local linear regression model for short-term traffic forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 1836 (2003), 143–150.

[20] THIAGARAJAN, A., RAVINDRANATH, L., LACURTS, K., MADDEN, S., BALAKRISHNAN, H., TOLEDO, S., AND ERIKSSON, J. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (2009), ACM, pp. 85–98.

[21] WAN, J., LIU, J., SHAO, Z., VASILAKOS, A. V., IMRAN, M., AND ZHOU, K. Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors 16*, 1 (2016), 88.

[22] WANG, Y., AND PAPAGEORGIOU, M. Real-time freeway traffic state estimation based on extended kalman filter: a general approach. *Transportation Research Part B: Methodological 39*, 2 (2005), 141–167.

[23] WU, C.-H., HO, J.-M., AND LEE, D.-T. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems 5*, 4 (2004), 276–281.

[24] ZHANG, B., XING, K., CHENG, X., HUANG, L., AND BIE, R. Traffic clustering and online traffic prediction in vehicle networks: A social influence perspective. In *Infocom, 2012 Proceedings IEEE* (2012), IEEE, pp. 495–503.

[25] ZHOU, T., GAO, L., AND NI, D. Road traffic prediction by incorporating online information. In *Proceedings of the 23rd International Conference on World Wide Web* (2014), ACM, pp. 1235–1240.