Deep Learning Based Urban Analytics Platform: Applications to Traffic Flow Modeling and Prediction

Archit Parnami
Dept. of Computer Science
University of North Carolina - Charlotte
aparnami@uncc.edu

Prajval Bavi
Dept. of Computer Science
University of North Carolina - Charlotte
pbavi@uncc.edu

Dimitris Papanikolaou
Dept. of Software & Information Systems
University of North Carolina - Charlotte
dpapanik@uncc.edu

Srinivas Akella
Dept. of Computer Science
University of North Carolina - Charlotte
sakella@uncc.edu

Minwoo Lee
Dept. of Computer Science
University of North Carolina - Charlotte
minwoo.lee@uncc.edu

Siddharth Krishnan
Dept. of Computer Science
University of North Carolina - Charlotte
skrishnan@uncc.edu

ABSTRACT

Cities have become more networked and wired, thus generating unprecedented amounts of operational and behavioral data. For instance, a city's traffic is instrumented by online mapping services (Google maps) in conjunction with crowd-sourcing (Waze). In this work, we present an *Urban Analytics Platform (UAP)* that demonstrates a systematic framework to tackle data-driven problems that the modern urban settings present. UAP embodies an end-to-end system for ingesting multiple data sources and processing them via neural network models to tackle predictive urban computing problems, like traffic flow. Results show improved traffic flow prediction and how we can utilize the improved prediction to inform on the possible weather conditions for abnormal traffic patterns.

KEYWORDS

Data Fusion, Urban Data, Traffic, Weather, Tweet, Surrogate Data

ACM Reference Format:
Archit Parnami, Prajval Bavi, Dimitris Papanikolaou, Srinivas Akella, Minwoo Lee, and Siddharth Krishnan. 2018. Deep Learning Based Urban Analytics Platform: Applications to Traffic Flow Modeling and Prediction. In *Proceedings of Mining Urban Data Workshop (MUD3).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In today's data-rich world, datafication of critical infrastructures like transportation — road networks and public services — has become an integral aspect of the unprecedented urbanization that we are witnessing. Modeling and understanding traffic flow has always been a problem of interest for city planners. Web services like Google Maps, Waze, online weather monitoring, and surrogate sources like Twitter have become useful instruments to sense and predict traffic. Towards that end, we design an Urban Analytics Platform (UAP) (for the city of Charlotte) that will systematically ingest multi-modal urban data from traditional sources and surrogate sources to tackle urban problems in a data-driven fashion. In this work, we describe how UAP will embody predictive models to forecast and inform the city about the traffic flow and enable efficient operation of shared mobility systems.

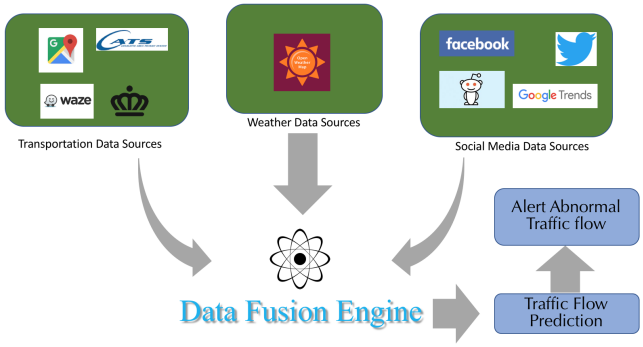

Figure 1: Data Fusion Engine. The UAP's data fusion engine aims to harness information from multiple data sources such as Google maps, weather forecasts, twitter and use it for applications like city planning, traffic prediction and detection.

The larger goal of UAP is to effectively and seamlessly integrate data from the city (community safety data, planning and zoning, transportation, neighborhoods & housing, and city government via http://clt-charlotte.opendata.arcgis.com/) with surrogate data sources like social media (Twitter, Facebook, Reddit), transportation (GoogleMaps, Waze) and weather data (OpenWeatherMap) as represented in Figure 1. The core component of the platform is a *data fusion engine*, which will combine the various data sources ingested into an application specific repository. For instance, in this work we demonstrate how UAP has utilized Waze to make traffic prediction and OpenWeatherMap to detect anomalies in traffic. UAP will further equip City of Charlotte with well-enriched urban datasets and "first principle" models that position us to answer

MUD3, Aug 2018, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

trans-disciplinary urban research questions using a data driven discovery and analytics approach.

Summary of Results

We utilize a stacked LSTM for making accurate traffic predictions. Training with selective data of weekdays and weekends separately, we further notice an improvement in prediction results. The trained LSTM prediction model is then used to detect abnormal traffic to correlate with the weather conditions. The results are presented in following sections.

2 RELATED WORK

Data fusion as defined by Waltz and Llinas [22] is, "A multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation and combination of data and information from single and multiple sources to achieve refined position and identify estimates, and complete and timely assessments of situations and threats and their significance." With so many sources of data, multi-sensor data fusion helps us to understand the environment and act accordingly. Baloch et al. [7] describe a way to gather more accurate and additional data using more than one sensor. They propose a layered approach for context acquisition and filtering using multiple subtasks.

Leveraging data fusion techniques, Faouzil et al. [10] describe the state-of-the-art practice of sensor data fusion to traffic data, and they reported that data fusion techniques indeed provide encouraging results. However, there are also challenges when there is a widespread use of data fusion in transportation field like obtaining data with accuracy to make application effective, dynamic and the need to process the data in real time. Bachmann et al. [6] investigate how multi-sensor data fusion of loop detectors, Bluetooth-enabled devices, vehicles equipped with GPS devices improved the estimation accuracy and concluded that accuracy depends on the technology, the number of probe vehicles and traffic state. He et al. [12] propose an optimization framework that improves traffic prediction using Twitter semantic data.

A major issue in getting traffic flow information in real time is that the majority of the links are not equipped with traffic sensor. Abadi et al. [4] used dynamic traffic simulator to generate flows in all links using the available traffic information, estimated demand, and historical data. Further, they used an autoregressive model to predict the traffic flows up to 30 min ahead using the real-time and estimated traffic data. With huge traffic data available, the problem of what data is relevant comes in to question. Polson et al. [18] have shown that data from the recent observations of the traffic conditions (i.e within last 40 min) are stronger predictors rather than historical values i.e measurements from 24 hours ago. Lv et al. [17] proposed the use of stacked autoencoder to learn the latent traffic flow features representation like nonlinear spatial and temporal correlation from traffic data and showed the performance of the proposed method to be superior to the other machine learning methods such as support vector machines, radial basis functions, and neural networks.

Recently, Long Short Term Memory (LSTM) [13, 19] have been used to predict various times series forecasting [5, 14, 16, 20, 21] and achieved the state-of-the-art results. For traffic congestion prediction, Chen et al. [8] have proposed an LSTM model. They proposed a classification technique to determine congestions and achieved improved accuracies. Our work extends their approach and utilizes LSTMs for making estimated traffic predictions. Moreover, we incorporate and utilize weather statistics in assessing traffic conditions. The proposed approach is expected to further broaden to inform with various surrogate data source.

3 DATA COLLECTION FRAMEWORK

3.1 Traffic Data

Our traffic data consists of the estimated time of travel from Point A to Point B using a car as a medium of transportation. The two simple ways in which we can get the estimated traffic are :

3.1.1 Google Maps. Using Google Map's Distance Matrix API [1], we can get the estimated travel duration and distance between a pair of points. Google's API calls are free only upto a limit of 2500 API calls per day.

3.1.2 Waze. Waze [3] is another navigation app which can provide the traffic information. Waze is however crowdsourced and does not requires an API key. For this project, we utilized web scraping to get this traffic information from Waze as represented in Figure 2. Since this data collection is done using web scrapping, there is only a limited number of requests that can be made per second to the Waze website. Through experimentation we determined that we can make at most two requests per second approximately from a unique IP. Following equations define the data collection parameters that determine the number of servers needed for data collection based on number of sources and destinations.

$$\text{Number of sources } = N$$

$$\text{Number of destinations } = M$$

$$\text{Total number of routes } = R = N * M$$

$$\text{Time taken to process a single request in seconds } = t$$

$$\text{Delay between each request in seconds } = d$$

$$\text{Limit on number of requests per second } = L = 1/(t + d)$$

$$\text{Total time in seconds to serve all requests } = T = R/L$$

$$\text{Data collection interval or period } = P$$

$$\text{Buffer time in seconds } = B$$

$$\text{Number of servers/Unique I.P needed } = S = \left\lfloor \frac{T}{P - B} \right\rfloor + 1$$

3.2 Weather Data

Weather data includes information such as temperature, pressure, humidity and current conditions for the day. OpenWeatherMap [2] is one such service that provides daily weather forecasts and current weather conditions. Therefore utilizing the OpenWeatherMap API, we get access to current and past weather conditions in the city of Charlotte.

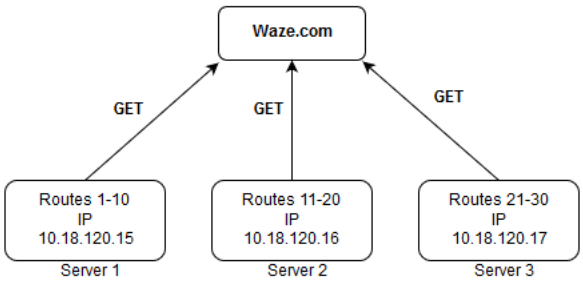

Figure 2: Web scraping. To obtain traffic information we periodically make GET requests to Waze website from different IP locations.

3.3 Interest Point Selection for Data Collection

Since we are constrained by the number of servers we can have for data collection, we need to carefully select our source and destination points for traffic data collection. The source and destination points should be such that they are evenly spread out throughout the city so that they represent all the traffic in the city. The city of Charlotte has 42 Fire stations which are evenly spread out throughout the city. These are represented by red in Figure 3. It also has 26 Light Rail Stations linearly distributed in the middle of the city. These are represented by blue dots. Collecting estimated travel durations between Fire Stations and Rail Stations periodically throughout the day will act as a proxy in help us modeling the inflow and outflow of traffic in the city.

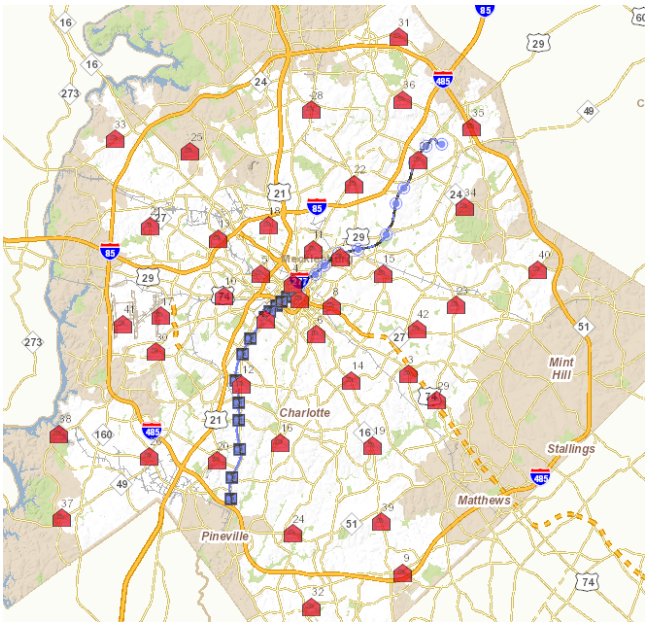

Figure 3: Source and destination points for traffic collection. The city of Charlotte has 42 Fire stations spread throughout the city (represented by red pentagons) which act as sources and has 26 rail stations (represented by blue dots) which act as destinations for our traffic data collection.

Using the equations listed in the previous section, we estimate the number of servers/unique I.P that will be required to collect data every 5 minutes using web scraping. The chosen parameters listed in Table 1.

Parameters	Values
N	42
M	26
R	$42 * 26 = 1092$
$t\,(experimental)$	0.5
$d\,(experimental)$	0.4
L	1/0.9
T	1092 *0.9 = 982.8
P	5*60=300
B	60
S	$\left\lfloor \frac{982.8}{300-60} \right\rfloor + 1 = 5$

Table 1: Data Collection Parameters

The buffer time B is the time in seconds to wait after making all the requests, this could be useful for making any read-write operations etc. and can be adjusted accordingly.

3.4 Sample Data

In the experiment, we are using traffic data (Table 2) that is collected from Waze API calls, between a given (fire station, rail station) pair over a period of 60 days.

Date	Clock	Fire Station	Rail Station	Time (s)	Distance (m)
3/7/18	0:00:00	40	I-485/South Blvd	2243	29885
3/7/18	0:00:02	40	Sharon Road West	2195	28668
3/7/18	0:00:04	40	Arrowood	2193	28269
3/7/18	0:00:05	40	Archdale	2061	26495
3/7/18	0:00:06	40	Tyvola	1976	25808
3/7/18	0:00:08	40	Woodlawn	2048	26957

Table 2: Traffic data.

At the same time, we also collect hourly weather information using OpenWeatherMap API. The data (Table 3) contains the current temperature, minimum and maximum temperature within an hour, humidity, wind speed and weather description (rain, light rain, thunderstorm, sky is clear).

Date	Time	Temp. (K)	Temp. Min	Temp. Max	Pressure	Humidity	Weather
3/7/18	0:00:00	278.64	277.1	280.15	1008	93	mist
3/7/18	1:00:00	278.14	276.15	279.15	1007	93	mist
3/7/18	2:00:00	278.06	276.15	279.15	1007	93	mist
3/7/18	3:00:00	278.31	277.15	279.15	1006	100	mist
3/7/18	4:00:00	278.65	277.15	280.15	1006	100	mist
3/7/18	5:00:00	278.73	277.15	280.15	1006	100	mist

Table 3: Weather data.

3.5 Traffic Analysis

In order to make a sensible prediction, we first analyze the traffic patterns on a selected route, throughout the week.

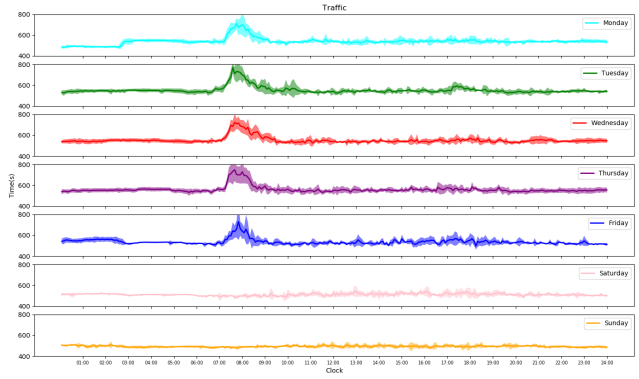

Figure 4: Average traffic from Monday to Sunday. x-axis represents the the (time of the day) and y-axis represents the estimated time of travel in seconds on this route.

As seen in Figure 4, the traffic flow pattern on weekdays are similar. There is a spike in traffic from 7:00 AM to 8:00 AM which can be attributed to people driving to work. The same spike is missing on Saturdays and Sundays reason it being a work holiday. There is also noticeably less traffic on a late Sunday night i.e the early hours of a Monday.

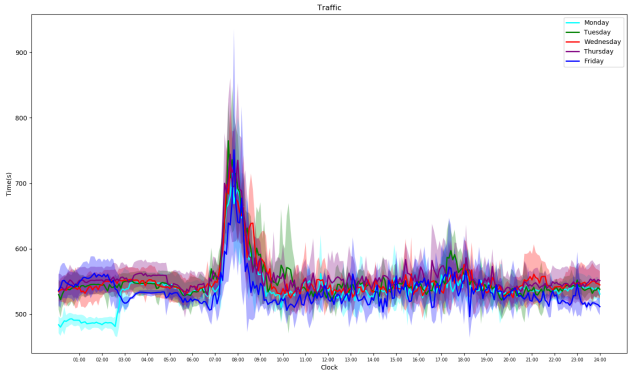

Figure 5: Weekday Pattern. Overlapping traffic patterns from Monday to Friday reveals weekday traffic pattern.

Traffic patterns from Monday to Friday when plotted (Figure 5) together reveal the general trend of traffic on weekdays.

Figure 6: Weekend Pattern

Saturdays and Sundays have similar traffic patterns except the intensity of traffic is lower on Sundays (Figure 6).

4 DEEP LEARNING-BASED TRAFFIC PREDICTION MODEL

In this section we discuss Long Short Term Memory (LSTM) networks, a particular type of Recurrent Neural Network (RNN) and how it can be used to model for traffic flow prediction.

4.1 Recurrent Neural Networks

Recurrent neural networks (RNN) [23] are similar to feedforward neural networks, except it also has connections pointing backwards. A simple RNN composed of one neuron receiving inputs, producing an output and sending that output back to itself. The output of the single RNN is

$$\mathbf{y}_{(t)} = \phi\left({\mathbf{x}_{(t)}}^T \cdot \mathbf{w}_x + {\mathbf{y}_{(t-1)}}^T \cdot \mathbf{w}_y + b\right).$$

At each time step t, this neuron receives the inputs x_t as well as its own output from the previous time step, y_{t-1}. Similarly, for a RNN layer at each time step t, every neuron in the layer receives both the input vector x_t and the output vector from the previous time step y_{t-1}.

Although RNN has potential to learn the contextual dependency and it is easy to compute its gradients, the range of input sequence is limited in practice due to the vanishing gradient problem. Various efforts were made to tackle this problem since 1990s. Among them, LSTM model, as a variant of standard RNN, was proposed by Hochreiter and Schmidhuber [13].

4.2 Long Short Term Memory (LSTM)

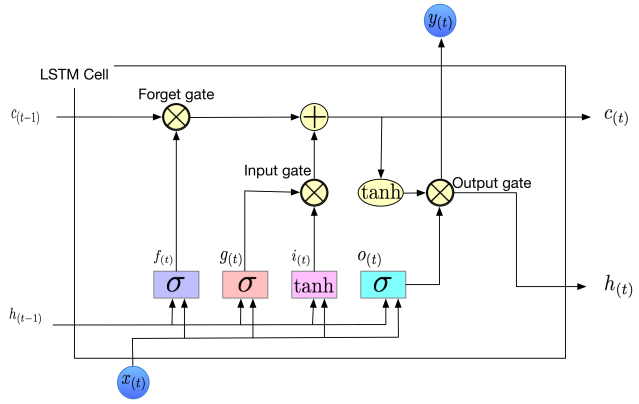

Figure 7: LSTM Cell

Figure 7 illustrates a LSTM [13, 19] architecture. This complicated cell has its state split in two vectors: $h_{(t)}$ and $c_{(t)}$, where $h_{(t)}$ is the short-term state and $c_{(t)}$ the long-term state. As the long-term state $c_{(t-1)}$ traverses the network, it first goes through a forget gate, dropping some memories, and then it adds some new memories via the addition operation which adds the memories that are selected by the input gate. The result $c_{(t)}$ is the output of the cell without any further transformation. Also, after the addition operation, the long-term state is copied and passed through a tanh function $h_{(t)}$ which is equal to the cell's output for this time step $y_{(t)}$. The main layer is the one that outputs $g_{(t)}$, which analyses the current inputs $x_{(t)}$ and the previous short term state $h_{(t-1)}$. The three other layers $f_{(t)}$, $i_{(t)}$ and $o_{(t)}$ are gate controllers, they use the logistic activation function, so their outputs range from 0 to 1. Their output is fed to the element-wise multiplication operations, so if they output 0s they close the gate, and if they output 1s, they open it. Hence, LSTM can learn to recognize an important input, store it in long-term state, learn to preserve for as long as it is needed and learn to extract it whenever it is needed. Below equations summarizes how the computation in the LSTM cell takes place.

$$\begin{aligned}
\mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
\mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)}),
\end{aligned}$$

where W_{hi}, W_{hf}, W_{ho}, W_{hg} are the weights of each of the four layers for their connections to the previous short-term state $h_{(t-1)}$, while W_{xi}, W_{xf}, W_{xo}, W_{xg} are the weight matrices of each of the four layers for connection to the input layer $x_{(t)}$. The b_i, b_f, b_o and b_g are the bias term for each of the four layer.

4.3 Prediction Model

While the original LSTM model is comprised of a single hidden LSTM layer followed by a standard feed forward output layer, the stacked LSTM has multiple hidden LSTM layers where each layer contains multiple memory cells. Studies [9, 11, 15] have shown that deep LSTM architectures with several hidden layers can build up progressively higher levels of representations of sequence data, and thus, work more effectively. The deep LSTM architectures are networks with several stacked LSTM hidden layers, in which the output of a LSTM hidden layer will be fed as the input into the subsequent LSTM hidden layer. We experimented with different number of LSTM layers stacked onto each other and empirically determined that a stacked LSTM with 4 layers works best for our data (Figure 8).

The code base and the data for this study will be made available on our GitHub repository (https://github.com/ArchitParnami/UAP) for reproducibility and further research.

5 TRAFFIC FLOW PREDICTION

The following subsections describe the traffic flow prediction model for a single route i.e., shortest route from Fire Station 20 to JW Clay Rail Station. This route connects the city of Charlotte from one end to another. The model can be generalized to predict traffic on any route. Furthermore we assume that the traffic data from Waze i.e., the estimated time of travel between two points obtained from Waze is the real representation of actual traffic.

5.1 Data Representation

Since we collect data every 5 minutes for a route, we have 288 time intervals in a 24 hour day. For a day i, the data x_i is

$$x_{\mathrm{i}} = [t_1, t_2, ..., t_{288}].$$

The training set has 60 days of data. Therefore the entire training data can be represented as:

$$X_{\text{train}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{60} \end{bmatrix}.$$

We try to predict the traffic for the 61st day (i.e., the subsequent day). Therefore the test set is the traffic for the next day,

$$X_{\text{test}} = \begin{bmatrix} x_{61} \end{bmatrix}.$$

5.2 Data Preprocessing

X_{train} is flattened into a single list with number of observation equal to 288×60 = 17,280, so

$$X_{\text{train}} = [t_1, t_2, ..., t_{17280}].$$

The training data is then normalized into a range of [0,1]:

$$\tilde{X}_{\text{train}} = Normalize(X_{\text{train}}).$$

Similarly, the test data is also normalized,

$$\tilde{X}_{\text{test}} = Normalize(X_{\text{test}}).$$

An LSTM model expects input in the form of a sequence. The length of the sequence, also know as the number of time-steps is

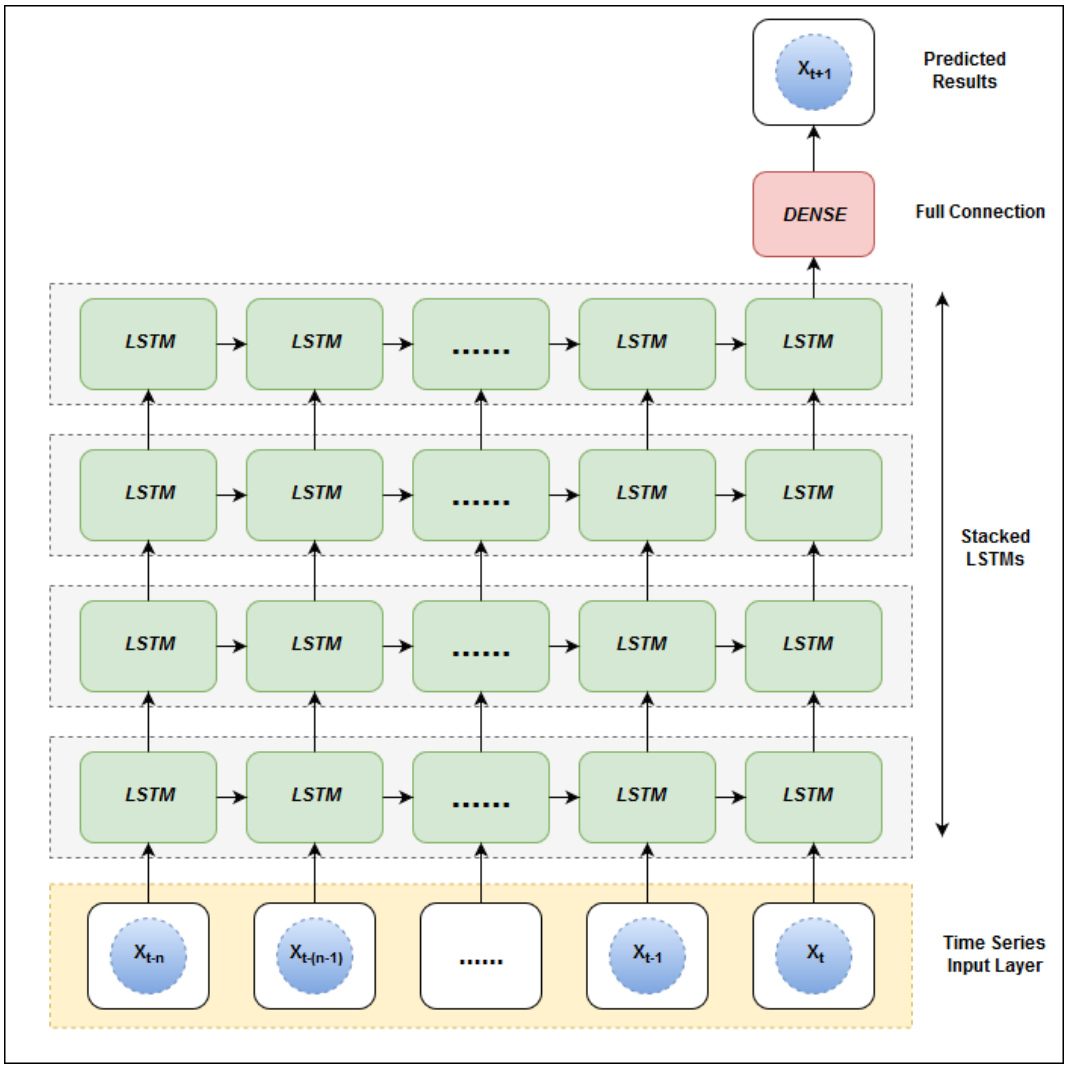

Figure 8: Architecture of the traffic prediction model. The first layer in this architecture represents the input layer which contains the traffic data from the current and previous time intervals. This data is then fed into a stacked LSTM model of four layers which is connected to a dense layer, which then outputs the estimated traffic for the next time interval.

a hyperparameter that can be set. For this problem the length of sequence is set to 12. That means we train the LSTM model on the past hour of traffic data continuously and try to predict the traffic in the next 5-minute interval. After this transformation the training observations looks like this:

$$x_1 = [t_1, t_2, ..., t_{12}], \qquad y_1 = [t_{13}],$$

$$x_2 = [t_{13}, t_{14}, ..., t_{24}], \quad y_2 = [t_{25}].$$

Finally the training set will have the form:

$$X_{\text{train}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad Y_{\text{train}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

The test data is also split into sequences of length 12. To predict the 1st time interval of the 61st day, we also add traffic information from the previous day, i.e., the last sequence of the 60th day to our test data.

5.3 Prediction Results

The red line in Figure 9 represents the real traffic where as the blue line represents the traffic predictions made by our LSTM model depicted in Figure 8. The y-axis is the ETA in seconds and x-axis is the time of the day.

While the stacked LSTM architecture fits very well for making traffic predictions, we observed that having a separate stacked LSTM model just for weekdays could improve the prediction results. This is based on our previous observations that the weekday traffic

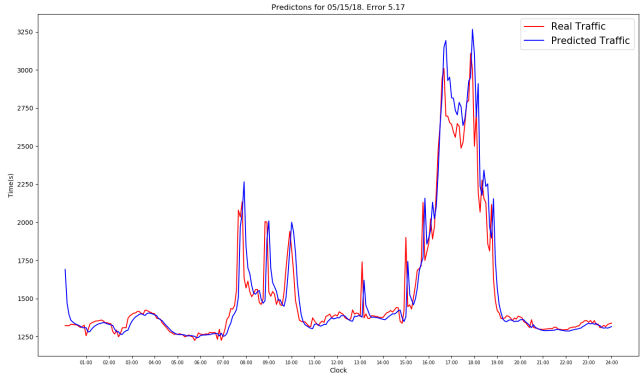

Figure 9: Real traffic (red) vs predicted traffic (blue) when trained on all the historic data (Mean Squared Error = 5.17).

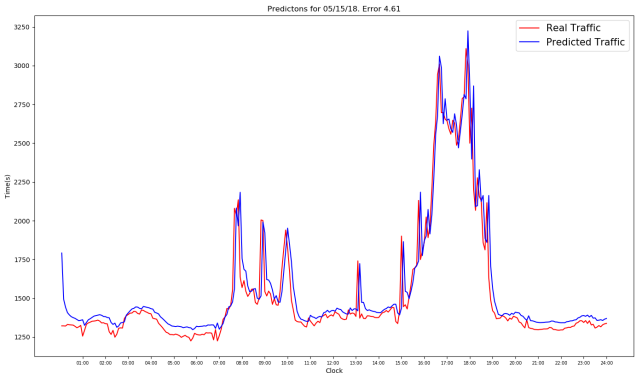

Figure 10: Real traffic vs predicted traffic when trained only on weekdays (Mean Squared Error = 4.61).

pattern differs from the weekend traffic pattern, so having a separate model would benefit (Figure 10).

Date	Weekday Average MSE	Weekday Average MAPE	All day Average MSE	All day Average MAPE
05/15/18	4.59	4.66	4.73	4.73
05/16/18	3.41	4.51	3.49	4.6
05/17/18	3.97	4.77	4.0	4.86
05/18/18	3.79	4.72	3.84	4.87
Average	**3.94**	**4.66**	**4.01**	**4.76**

Table 4: Mean squared error and mean absolute percentage error averaged over 10 training and test runs for weekday and all day prediction models.

From Table 4, we can conclude that the Weekday training model performed better than all days training model. Figure 12 displays the average residue for the Weekday model vs the All day model for a sample route. The green area represents the intervals where the average error for prediction was less when using the Weekday model whereas the red area captures the interval when average error for prediction was high when using the Weekday model. The figure reaffirms the efficacy of weekday training model.

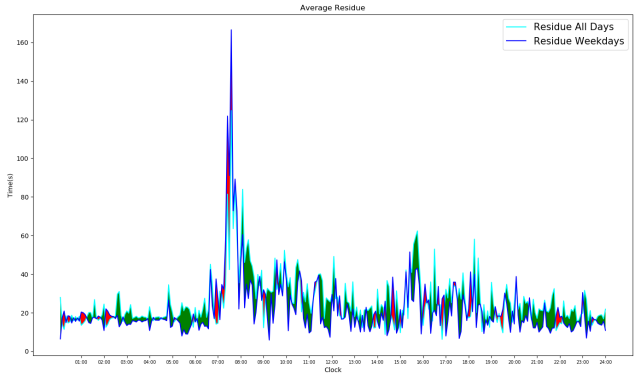

Figure 11: Training models' prediction error. Green regions indicate less prediction errors when model was trained only for weekdays, and red regions indicate higher prediction errors.

5.4 Comparing results with a Multilayer Perceptron model

The following results were obtained when we trained an artificial neural network with 3 hidden layers (288,16,8). The inputs to the model was the day of week(1-7) and the time interval (1-288) and the output was traffic (estimated time of travel).

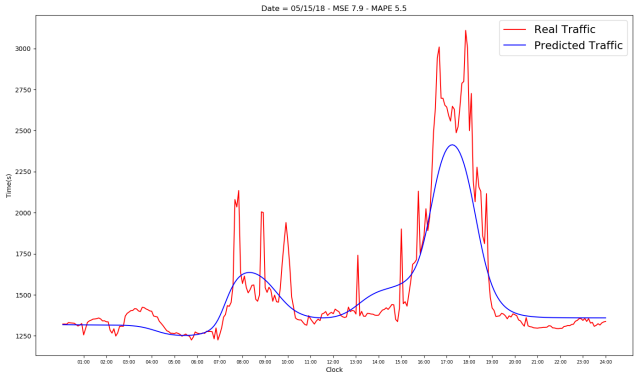

Figure 12: Real traffic vs Predicted traffic on 05/15/18 obtained using a multilayer perceptron model trained on all days.

Date	MSE	MAPE
05/15/18	7.9	5.5
05/16/18	3.7	5.46
05/17/18	4.92	6.22
05/18/18	7.35	6.19
Average	**5.97**	**5.84**

Table 5: Mean squared error and mean absolute percentage errors obtained using multilayer perceptron model.

It can be observed that while the multilayer perceptron model is able to learn the general traffic pattern, the LSTM model is able to better capture the traffic pattern more accurately with lower MSE and MAPE.

6 CORRELATING DEVIATIONS IN TRAFFIC PREDICTION WITH WEATHER CONDITIONS

Traffic conditions are dynamic in nature. There are many uncontrollable factors which can cause deviations in traffic patterns. For example a sudden thunderstorm, an accident or a game event. Therefore it is certain that an LSTM model can not make accurate predictions all the time considering that there are other factors in play which could cause sudden change in traffic conditions. Here we focus on one such factor i.e weather and try to correlate and see its influence on traffic conditions.

Figure 13 and Figure 14 shows the changing weather conditions at different time intervals. For example it rained around 1:00-2:00 AM and then there was thunderstorm followed by rain at 12:00 PM. The figures also display the real traffic and the LSTM predicted traffic. In certain regions, such as when there was a thunderstorm, the deviation or the error in prediction was observed to be more than normal i.e., when the sky was clear.

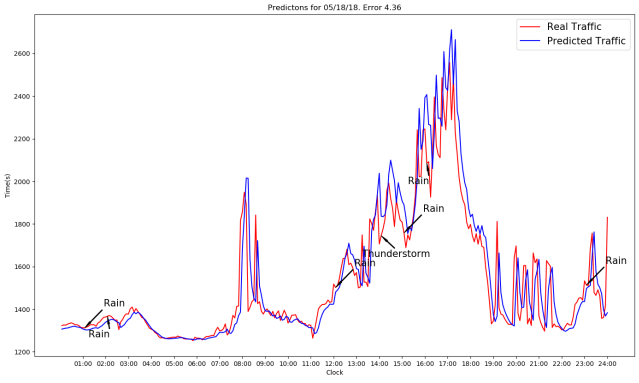

Figure 13: Peak deviations in traffic correlated with weather conditions. The arrows point at a traffic condition at a particular time and weather.

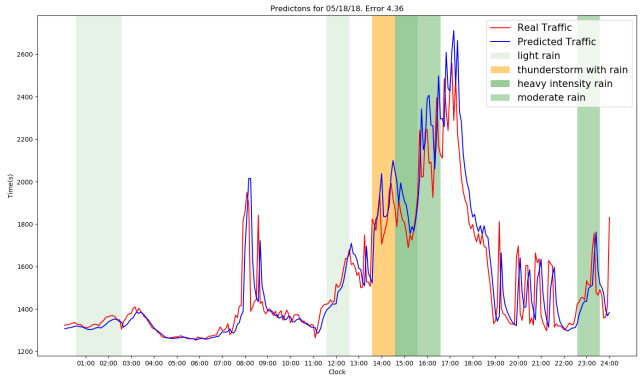

Figure 14: Deviation in traffic prediction with weather. The orange region highlight the interval of thunderstorm and dark green region highlight the interval of heavy rain. These two regions correspond to the highest deviation in traffic prediction.

Intuitively and experimentally from Figure 14 it can be said that traffic conditions are definitely affected by weather conditions. This impact of the anomalous weather on traffic can be measured using LSTM predictions as observed in this study.

7 CONCLUSION

In this paper, we have discussed a low-cost data acquisition and fusion model for traffic flow analysis. To get high quality traffic data, researchers [6, 10] must rely on government sources for getting data from hardware devices (i.e., loop detectors, GPS). In view of the large amount of data that was required for this project, paid services like Google maps are often not an option. Thus, acquiring high quality traffic data is one of the challenges in traffic prediction. However utilizing the power of webscraping on a crowdsourced website, i.e., Waze, we could gather estimated traffic information in the city. OpenWeatherMap API allows us to retrieve relevant information regarding possible weather conditions for unexpected traffic flows.

Our LSTM model has shown good performance for short term prediction; it however lacks the capability for making long-term predictions. Utilizing the data fusion engine, we aim to build a more sophisticated model that can not only make long term traffic predictions but could also inform us about the anomalous traffic behavior using diverse data sources such as social networks, news headlines, and accident logs. This will require development of an efficient anomaly detection technique that can automate the informing process.

REFERENCES

[1] [n. d.]. Google Maps Distance Matrix API. "https://developers.google.com/maps/documentation/distance-matrix/intro". Accessed:2018-05-29.
[2] [n. d.]. OpenWeatherMap. "https://openweathermap.org/". Accessed:2018-05-29.
[3] [n. d.]. Waze. "https://www.waze.com/". Accessed:2018-05-29.
[4] Afshin Abadi, Tooraj Rajabioun, and Petros A Ioannou. 2015. Traffic flow prediction for road transportation networks with limited traffic data. *IEEE transactions on intelligent transportation systems* 16, 2 (2015), 653–662.
[5] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. 3 (2013).
[6] Chris Bachmann, Baher Abdulhai, Matthew J Roorda, and Behzad Moshiri. 2013. A comparative assessment of multi-sensor data fusion techniques for freeway

traffic speed estimation using microsimulation modeling. *Transportation research part C: emerging technologies* 26 (2013), 33–48.

[7] Zartasha Baloch, Faisal Karim Shaikh, and Mukhtiar Ali Unar. 2018. A context-aware data fusion approach for health-IoT. *International Journal of Information Technology* 10, 3 (2018), 241–245.

[8] Yuan-yuan Chen, Yisheng Lv, Zhenjiang Li, and Fei-Yue Wang. 2016. Long short-term memory model for traffic congestion prediction with online open data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 132–137.

[9] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. 2018. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. *arXiv preprint arXiv:1801.02143* (2018).

[10] Nour-Eddin El Faouzi and Lawrence A Klein. 2016. Data fusion for ITS: techniques and research needs. *Transportation Research Procedia* 15 (2016), 495–512.

[11] Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57 (2016), 345–420.

[12] Jingrui He, Wei Shen, Phani Divakaruni, Laura Wynter, and Rick Lawrence. 2013. Improving Traffic Prediction with Tweet Semantics.. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. 1387–1393.

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

[14] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3128–3137.

[15] Xiangang Li and Xihong Wu. 2015. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 4520–4524.

[16] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. 2015. Learning to Diagnose with LSTM Recurrent Neural Networks. *CoRR* abs/1511.03677 (2015). arXiv:1511.03677 http://arxiv.org/abs/1511.03677

[17] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 865–873.

[18] Nicholas G Polson and Vadim O Sokolov. 2017. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies* 79 (2017), 1–17.

[19] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128* (2014).

[20] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[21] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 3156–3164.

[22] Edward Waltz, James Llinas, et al. 1990. *Multisensor data fusion*. Vol. 685. Artech house Boston.

[23] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).