PREDICTING THE NUMBER OF UBER PICKUPS BY DEEP LEARNING

Chao Wang, Ph.D. candidate
(Corresponding Author)
Department of Electrical and Computer Engineering, UC Riverside
1084 Columbia Avenue, Riverside, CA 92507
Tel: (951) 522-7381; Fax: (951) 781-5790
Email: cwang061@ucr.edu

Peng Hao, Ph.D.
Center for Environmental Technology and Research, UC Riverside
1084 Columbia Avenue, Riverside, CA 92507
Tel: (518) 364-3671; Fax: (951) 781-5790
Email: haop@cert.ucr.edu

Guoyuan Wu, Ph.D.
Center for Environmental Technology and Research, UC Riverside
1084 Columbia Avenue, Riverside, CA 92507
Tel: (951) 781-5630; Fax: (951) 781-5790
Email: gywu@cert.ucr.edu

Xuewei Qi, Ph.D.
Center for Environmental Technology and Research, UC Riverside
1084 Columbia Avenue, Riverside, CA 92507
Tel: (706) 206-5972; Fax: (951) 781-5790
Email: xqi001@ucr.edu

Matthew Barth, Ph.D.
Department of Electrical and Computer Engineering, UC Riverside
1084 Columbia Avenue, Riverside, CA 92507
Tel: (951) 781-5782; Fax: (951) 781-5790
Email: barth@ece.ucr.edu

Word Count: 4000 words + 7 figure(s) x 250 + 3 table(s) x 250 = 6500 words

Submission Date: August 1, 2017

ABSTRACT

On-demand, app-based ride services like Uber and Lyft have become an important part of today's transportation system with its flexibility and quick responsiveness. Compared with traditional taxicabs, Uber-like taxis have loggers to monitor and record trip information such as pickup location and trip distance, which can be a valuable data source for knowledge discovering. Nowadays, a Real-time prediction for ride service demand (always reflected by the number of pickups) is increasingly crucial for the purpose of improving the efficiency and sustainability of the urban transportation system. Newly aroused application topics like ride sharing and autonomous mobility dispatching are based on solid demand predictions. In this paper, we propose a deep learning based approach to make dynamic predictions for Uber pickups using historical data. A Long Short-Term Memory (LSTM) Networks model is developed to learn the long-term dependencies of the pickups over time. With the experimental comparison of time-varying Poisson model and regression tree model, the results demonstrate the superior performance of our proposed deep learning model.

Keywords: Prediction, demand, Uber, real-time ride service demand, data mining, machine learning.

INTRODUCTION AND MOTIVATION

Over the last two decades, the development of sensing and communication technology has brought the vehicles to a more connected and intelligent level. In particular, widely equipped positioning systems (GPS, GSM, etc.) offer us a new source of spatial-temporal data of running vehicles. Using data mining and machine learning techniques and discovering knowledge from this kind of data is becoming a very popular research topic. Taxicabs are one of the earliest types of vehicles equipped with positioning systems and collect trip data in a large scale. There are many available taxi data and many interesting studies. The first type of applications utilizing the taxi data is for traffic monitoring and prediction. Min and Wynter (*1*) developed a method that provides predictions of speed and volume over 5-min intervals for up to 1 h in advance of the traffic. This method takes into account the spatial characteristics of a road network in a way that reflects not only the distance but also the average speed on the links. Lv et al. (*2*) presented a deep learning method with stacked auto-encoders model to predict the traffic. This is the first work apply deep learning for traffic prediction and the authors showed the proposed method out performs many traditional machine learning methods. Another application of taxi data is planning. Chen et al. (*3*) proposed a method to plan the night bus route by looking at the taxi pickups at night. Spatial clustering and optimized routing are used in their method. Zhan et al. (*4*) researched the taxi data in New York City and tried to obtain the theoretical optimum of the taxi allocation by using optimal matching of the taxicabs and passengers. Their results showed a large gap between the current taxi service system and the theoretical maximum, indicating great room for improvement. Li et al. (*5*) proposed a strategy for taxi drivers to plan their manners to quickly find the next passenger(s) by learning the spatial temporal patterns of the historical pickups. Taxi data can also be used for monitoring the environment. Zheng et al. (*6*) proposed a framework to model the taxi trips and the air quality of Beijing City together so as to predict the air quality by observing the taxi information.

The number of taxi pickups itself reflects the demand of the ride service, which is also valuable to model. Moreira-Matias et al. (*7*) proposed a hybrid method to predict the number of taxi pickups at a certain taxi stop. Three models are utilized and combined to get the best guess. Besides traditional taxicabs, on-demand, app-based ride services like Uber and Lyft has become an important part of today's transportation system with its flexibility and quick responsiveness. While some data analysis and visualization work have been illustrated (*8, 9*), few studies are on-going to utilize the Uber-like taxi data. Uber-like taxis all have loggers to monitor and record trip information such as pickup location and trip distance, which can be a valuable data source for knowledge discovering. Compared with traditional taxis, Uber-like taxis are directed to the passengers and one request of ride service always correspond to a pickup. Therefore, Uber-like taxi pickups better reflect the true ride service demand. So it is more meaningful to predict the number of pickups of Uber-like taxis since a better understanding and forecasting of the travel demand helps to improve the efficiency and sustainability of the urban transportation system. And newly aroused applications like ride sharing and autonomous mobility dispatching are based on solid demand predictions. In this paper, we proposed a deep learning method to predict the number of Uber pickups with long short-term memory (LSTM) model. The LSTM model is compared with a statistic based baseline: time-varying Poisson model, and a basic machine learning model: regression tree model. The results show the proposed method outperforms the baselines and the improvement is significant.

MODELS

Consider the number of Uber pickups overtime in a certain region of interest is a discrete time series $\{X_0, X_1, ...X_t\}$. The problem is formulated as: given $\{X_0, X_1, ...X_{t-1}\}$, build a model to determine X_t. For this study, the time step of the series is one hour. Figure 1 shows a sample sequence of the number of Uber pickups over time at Midtown Center, Manhattan. We can clearly observe the periodic pattern of the pickup sequence. To solve the proposed prediction problem, three models are introduced in this section: time-varying Poisson model, regression tree model, and the long short-term memory (LSTM) model. The time-varying Poisson model and the regression tree model are introduced as baselines for comparison purpose. Our deep learning approach is based on the LSTM model.

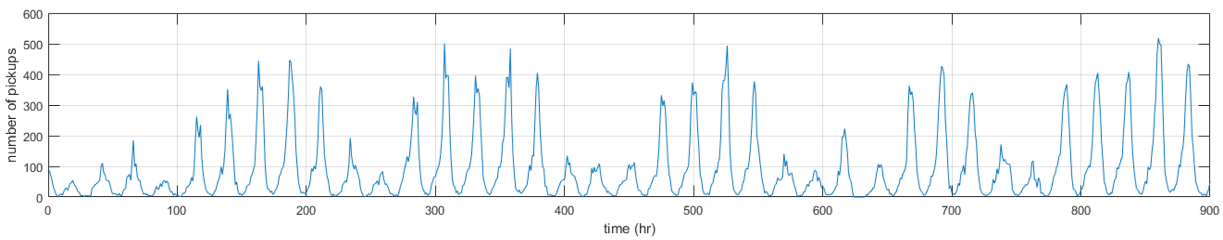


Figure 1.: Example sequence of Uber pickups.

Baseline Models

Time-Varying Poisson Model

This baseline model was proposed in (*7*). The pickup demand pattern reflects human activity and appear to be non-homogeneous. Assume the probability for n Uber pickups to emerge in a certain time slot follows a Poisson distribution:

$$y(n; \lambda) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{1}$$

where λ is the expectation of the number of pickups in this time slot. λ should be a time varying parameter ,i.e., $\lambda(t)$, since the average pickups varies with time. If we consider the pickup sequence is periodic and the period is one week, $\lambda(t)$ can be defined as Equation 2, where $\lambda(0)$ is the average pickups over a full week, $\delta_{d(t)}$ and $\eta d(t), h(t)$ are the correction factors for with respect to day of week and hour of day, respectively. Consider $lambda(t)$ as a discrete function (hourly aggregated). Validate Equation 2 with Equation 3 and 4.

$$\lambda(t) = \lambda(0)\delta_{d(t)}\eta_{d(t),h(t)} \tag{2}$$

$$\sum_{d=1}^{7} \delta(d) = 7 \tag{3}$$

$$\sum_{h=1}^{24} \eta(d,h) = 24 \forall d \tag{4}$$

Regression Tree Model

Decision tree learning is a widely used predictive modeling approach in statistics, data mining and machine learning. Regression trees are the kind of decision trees whose target variables can take continuous values. Regression tree based learning model builds a relationship between input variables and the targets by categorizing targets with boundary conditions of the inputs. A tree can be "learned" by splitting the source set into subsets based on an attribute value test (*10*). This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary splitting. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data. Figure 2 shows an example of regression tree construction (*11*). X_1 and X_2 are the two dimensions of the input feature. The output $Y \in \{R_1, R_2, \cdots, R_5\}$ is function of X_1, X_2, which can be express by the tree on the left. Every node of the tree corresponds to a split decision, and each leaf contains a subset of the data that satisfies the conditions. Therefore, to predict a new data point with input feature x_1, x_2, we can use Equation 5, where $f(x_1, x_2)$ is the regression tree model and h_m is the indicator of regions (For example, region R_1: $h_1 = I(x_1 \leq t_1)I(x_2 \leq t_1)$).

$$\hat{y}_t = f(x_1, x_2) = \sum_{m=1}^{M} \beta_m h_m(x_1, x_2) \tag{5}$$

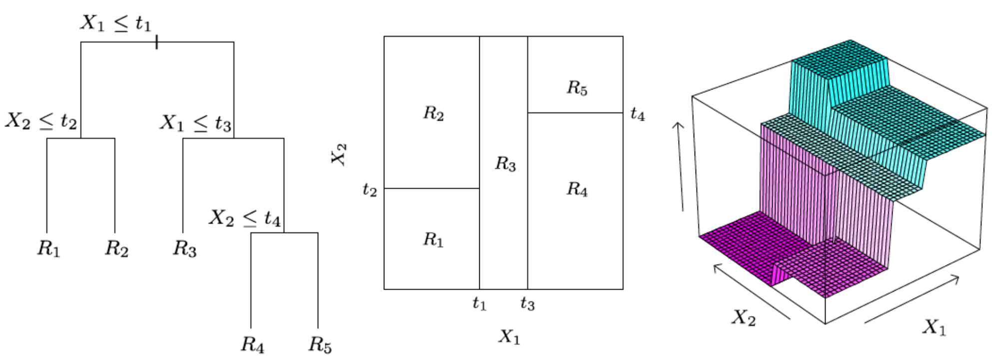


Figure 2.: Regression tree example (from book (*11*)).

Back to our number of pickups prediction problem. We can select proper features as the input and train a regression tree to predict the future number of pickups. From Figure 1, the pickups are showing weekly and daily periodic patterns. And the current hour pickups should be related to the previous hour's, considering the temporal correlation. Also, the pickups may be affected by the weather condition. Table 1 are the selected feature for the regression tree model.

LSTM Model

For our pickup demand prediction problem, we saw that the number of pickups are not only dependent on short-term data of near past but also present periodic patterns and tight dependencies on long-term histories. In other words, the correlation between two data point in the sequence

Table 1.: Selected features for the regression tree model

features	x_1	day of week
	x_2	hour of day
	x_3	number of pickups of current hour
	x_4	precipitation of the day
	x_5	temperature of the day
output	y	number of pickups of next hour

is not necessarily monumentally decreasing as time difference increases. To better present the long-term dependencies (repeating patterns), we calculated the autocorrelation of a sample pickup sequence. Autocorrelation is the correlation of the sequence with a delayed version of itself. Let X be the pickup sequence, X_s be the subsequence start at time s, and X_t be the subsequence start at t. Suppose the mean and variance of the two sequences are μ_s, μ_t, σ_s, σ_t respectively, then the autocorrelation between time s and t is:

$$R(s,t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s} \tag{6}$$

where E is the expected value operator. We took $s = 0$ and calculated the autocorrelation $A(t) = R(0,t)$. Figure 3 shows the result. We find that the number of pickups at a certain time of one day is highly correlated with the number of pickups of the same time on other days especially on the same day of other weeks.

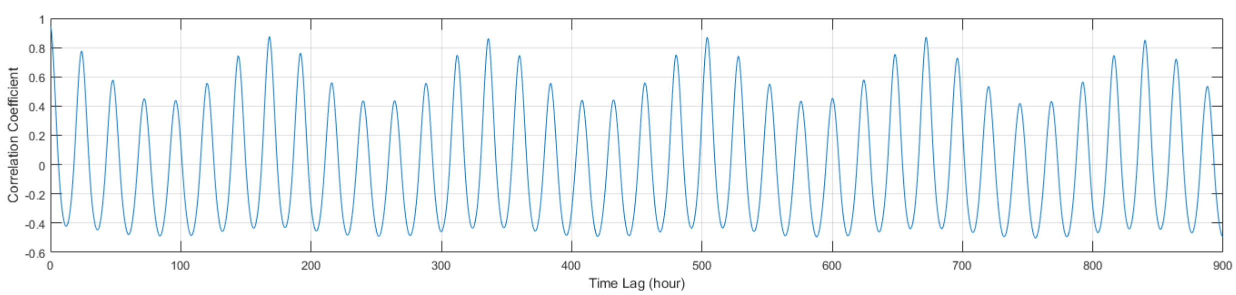


Figure 3.: Autocorrelation of the pickups sequence.

Recurrent neural networks (RNN) are able to capture the long-term dependencies reflected by Figure 3. They are artificial neural networks with a loop in them, allowing information to pass on to succeeding steps. Hence, to let the networks to "memorize" former inputs. Figure 4 presents an example RNN and its unfolding in time of the computation involved in its forward computation. x_t is the input at time step t, and s_t is the hidden state at time step t. s_t is the function of s_{t-1} and x_t: $s_t = f(Ws_{t-1} + Ux_t)$. o_t is the output at time step t. Despite the theoretical capability of memorizing previous information in the network, RNN still suffers to learn the connections between inputs as the time gap in between grows in practice. This shortcoming of RNN is further explained by pirastudiplomarbeit and Bengio et al. (*12*).

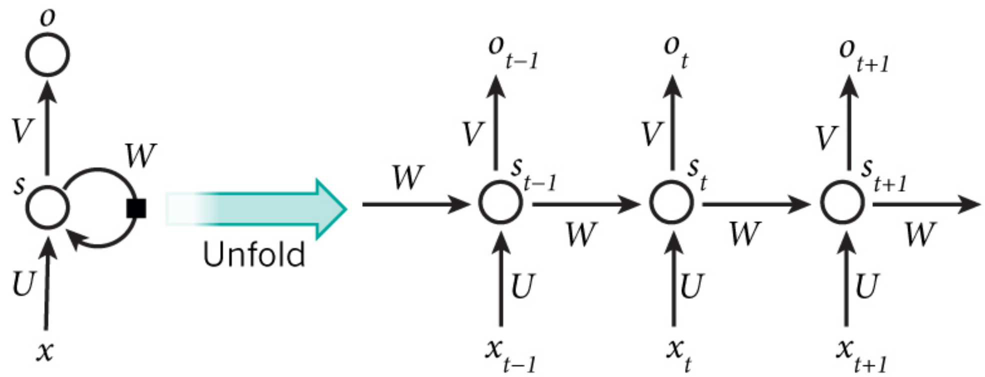


Figure 4.: Recurrent neural network and unfolding (by LeCun et al. (*13*)).

LSTM networks are a special kind of recurrent neural networks, which can learn long-term dependencies from sequences. LSTM was first proposed by Hochreiter and Schmidhuber (*14*) in 1997 and improved by Gers et al. (*15*) in 2000. LSTM is now very popular in many problems especially natural language text compression. LSTM is often implemented in "blocks" containing several units. This design is typical of deep neural networks and facilitates implementations with parallel hardware. In the equations below, each variable in lowercase italics represents a vector with a length equal to the number of LSTM units in the block. LSTM blocks control information flow via gates. Information can be stored in, written to, or read from a cell. The cell learns to make decisions about what to store, and when to allow reads, writes, and erasures, by opening or closing gates. The gates are implemented using the logistic function to compute a value between 0 and 1. Multiplication is applied with this value to partially allow or deny information to flow into or out of the memory. For example, an "input" gate controls the extent to which a new value flows into the memory. A "forget" gate controls the extent to which a value remains in memory. An "output" gate controls the extent to which the value in memory is used to compute the output activation of the block. An architecture of LSTM is shown in Figure 5.
The cell state updating equations are as follows, and Table 2 are the notations :

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{7}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{9}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ (W_c x_t + U_c h_{t-1} + b_c) \tag{10}$$
$$h_t = o_t \circ \sigma_h(c_t) \tag{11}$$

The weights in an LSTM block (W and U) are used to direct the operation of the gates. These weights occur between the values that feed into the block (including the input vector x_t and the output from the previous time at step h_{t-1}) and each of the gates. Thus, the LSTM block determines how to maintain its memory as a function of those values, and training its weights causes the block to learn the function that minimizes loss. To minimize LSTM's total error on a set of training sequences, iterative gradient descent such as backpropagation through time (BPTT) can be used

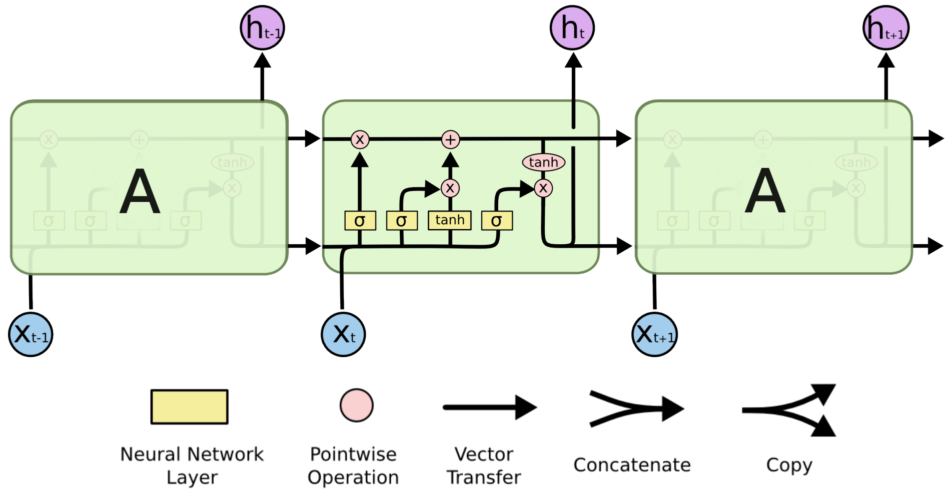


Figure 5.: LSTM architecture (by Olah (*16*)).

Table 2.: Notations of the LSTM equations

symbol	meaning
x_t	input vector
h_t	output vector
c_t	cell state vector
$W, U and b$	parameter matrices and vector
f_t	Forget gate vector. Weight of remembering old information.
i_t	Input gate vector. Weight of acquiring new information.
o_t	Output gate vector. Output candidate.
σ_g	sigmoid function
σ_c	hyperbolic tangent
σ_h	hyperbolic tangent

to change each weight in proportion to its derivative with respect to the error. A major problem with gradient descent for standard RNNs is that error gradients vanish exponentially quickly with the size of the time lag between important events. With LSTM blocks, however, when error values are back-propagated from the output, the error becomes trapped in the block's memory. This is referred to as an "error carousel" that continuously feeds error back to each of the gates until they become trained to cut off the value. Thus, regular backpropagation is effective at training an LSTM block to remember values for a long term.

EXPERIMENT

Data

The Uber pickup data used for this study is from Fivethirtyeight (*17*). This dataset contains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015. The data includes the pickup time and location (longitude and latitude for 2014 data, taxi-zone number for 2015 data) of each Uber trip. We used the first six months data in 2015 for this study. Figure 6 (a) visualizes the number of pickups in the 263 taxi-zones of New York City over the six months. It shows that Manhattan and the two airports (JFK and LaGuardia) have the densest Uber pickups. Figure 6 (b) shows the top 30% zones ranked by number of pickups. These taxi-zones contains 90% of the pickups. To evaluate the proposed deep learning method, we chose 3 representative zones for analyzing: zone 161 (Midtown City), which is the is the "Uber heaviest" zone and contains the most number of pickups; zone 132 (JFK Airport), which reflects the airport ride demand; and zone 45 (Chinatown), which represent a less busy (but not too few of pickups to be trivial) pattern.

Evaluation Metrics

To evaluate the accuracy of the prediction result, two well-known error measurements are used: the Symmetric Mean Absolute Percentage Error (SMAPE) and the Root Mean Square Error (RMSE). Consider $\hat{Y} = \{\hat{y_1}, \hat{y_2}, ...\hat{y_n}\}$ to be the Uber pickup predictions in time slot $[1,t]$, and $Y = \{y_1, y_2, ...y_n\}$ to be the true value of the pickup numbers. The RMSE and SMAPE are defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{12}$$

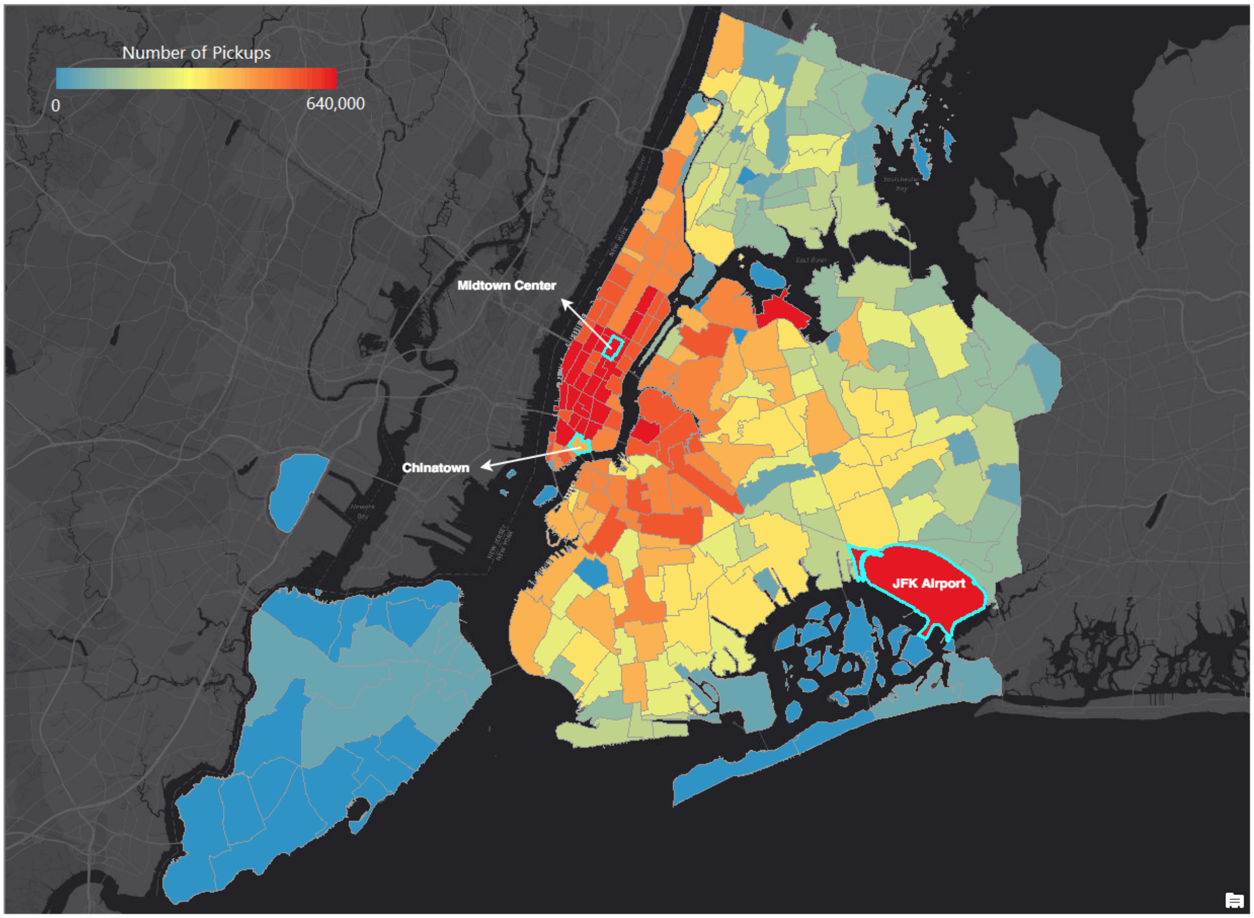
$$SMAPE = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \tag{13}$$

Result

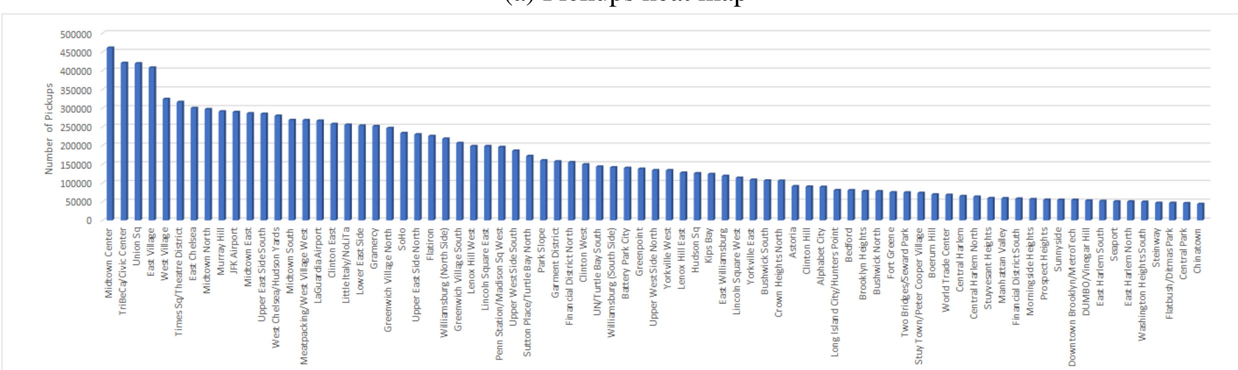
The prediction accuracy results for the three zones of interest are shown in Table 3. For all 3 representative taxi-zones, our proposed method outperforms the baseline ones and shows significant improvements. This means that the long-term dependencies captured by LSTM indeed have a positive influence on the prediction. Since the symmetric mean absolute percentage error (SMAPE) is more sensitive to smaller observations and the root means square error, the variation of the prediction error for different taxi-zones are reasonable. Figure 7 visualizes the prediction of the proposed method for the 3 zones of interest. Our deep learning model tracks the true value very well.

Table 3.: Performance comparison of time-varying Poisson, regression tree and LSRM

Zone of Interest	Performance	time-varying Poisson	regression tree	LSRM
Zone 161 (Midtown Center)	RMSE	33.5	28.4	22.3
	SMAPE	24.67%	23.30%	19.34%
Zone 132 (JFK Airport)	RMSE	31.8	29.9	26.7
	SMAPE	36.85%	34.36%	29.47%
Zone 45 (Chinatown)	RMSE	5.9	5.3	4.1
	SMAPE	38.05%	39.86%	32.38%



(a) Pickups heat map

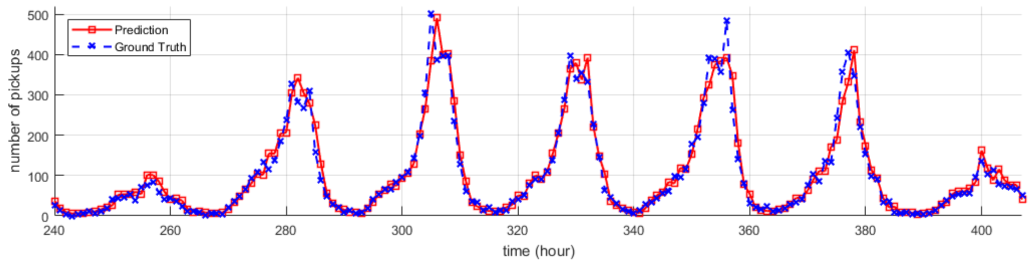


(b) Pickups for each census tract

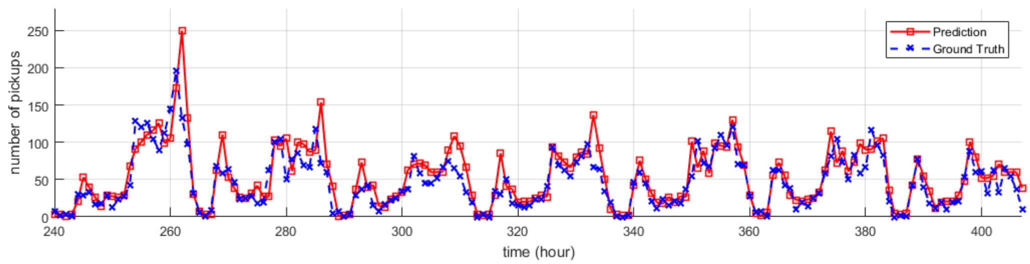
Figure 6.: Uber pickups from January to June, 2015.

CONCLUSION AND FUTURE WORK

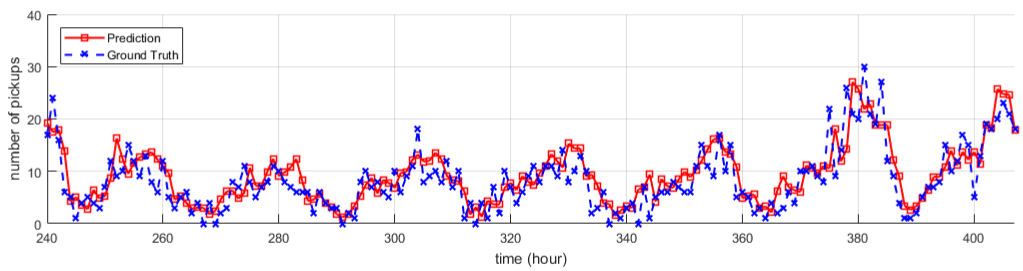
This paper has presented a deep learning based method to predict the number of Uber pickups. A long short-term memory (LSTM) model is proposed aiming to capture the long-term dependencies of the pickup sequence over time. Backpropagation through time (BPTT) is used to train our model. To evaluate the performance of the method, we applied it to predict the Uber pickups in 3 representative taxi-zones of New York City along with two baseline models: the time-varying



(a) Zone 161: Midtown Center



(b) Zone 132: JFK Airport



(c) Zone 45: Chinatown

Figure 7.: Prediction result by LSRM (10th week of 2015).

Poisson model and the regression tree model. The results show that the proposed method has significant improvement to the baselines. For future work, it would be interesting to compare the Uber pickup data with New York taxi pickup data. Fivethirtyeight (*8*) shows some statistic temporal and spatial difference of Uber and taxi pickups. More advanced data mining techniques like tensor mining may help to discover deeper knowledge that may help to improve the Uber and/or taxi service system. On the other hand, with an accurate prediction of the pickups, we are able to design a Uber dispatching system and guide the Uber cabs to the are with high possibilities of pick demand. The efficiency improvement of the dispatching system would be a good topic.

REFERENCES

[1] Min, W. and L. Wynter, Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, Vol. 19, No. 4, 2011, pp. 606–616.

[2] Lv, Y., Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 2, 2015, pp. 865–873.

[3] Chen, C., D. Zhang, Z.-H. Zhou, N. Li, T. Atmaca, and S. Li, B-Planner: Night bus route planning using large-scale taxi GPS traces. In *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, IEEE, 2013, pp. 225–233.

[4] Zhan, X., X. Qian, and S. V. Ukkusuri, A graph-based approach to measuring the efficiency of an urban taxi service system. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 9, 2016, pp. 2479–2489.

[5] Li, B., D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang, Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, IEEE, 2011, pp. 63–68.

[6] Zheng, Y., F. Liu, and H.-P. Hsieh, U-Air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2013, pp. 1436–1444.

[7] Moreira-Matias, L., J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 14, No. 3, 2013, pp. 1393–1402.

[8] Fivethirtyeight, *Uber Is Serving New York's Outer Boroughs More Than Taxis Are*. https://fivethirtyeight.com/features/uber-is-serving-new-yorks-outer-boroughs-more-than-taxis-are/, 2015, accessed: 2017-03-28.

[9] Fivethirtyeight, *Public Transit Should Be Uber's New Best Friend*. `https://fivethirtyeight.com/features/public-transit-should-be-ubers-new-best-friend/`, 2015, accessed: 2017-03-28.

[10] Wikipedia, *Decision tree learning*. `https://en.wikipedia.org/wiki/Decision_tree_learning`, 2017, accessed: 2017-04-30.

[11] Friedman, J., T. Hastie, and R. Tibshirani, *The elements of statistical learning*, Vol. 1. Springer series in statistics New York, 2001.

[12] Bengio, Y., P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, Vol. 5, No. 2, 1994, pp. 157–166.

[13] LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. *Nature*, Vol. 521, No. 7553, 2015, pp. 436–444.

[14] Hochreiter, S. and J. Schmidhuber, Long short-term memory. *Neural computation*, Vol. 9, No. 8, 1997, pp. 1735–1780.

[15] Gers, F. A., J. Schmidhuber, and F. Cummins, Learning to forget: Continual prediction with LSTM, 1999.

[16] Olah, C., *Understanding LSTM Networks*. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015, accessed: 2017-06-30.

[17] Fivethirtyeight, *uber-tlc-foil-response*. `https://tex.stackexchange.com/questions/3587/how-can-i-use-bibtex-to-cite-a-web-page`, 2015, accessed: 2017-03-28.