Real-time Prediction of Taxi Demand Using Recurrent Neural Networks

Jun Xu, Rouhollah Rahmatizadeh, Ladislau Bölöni and Damla Turgut

***Abstract*—Predicting taxi demand throughout a city can help to organize the taxi fleet and minimize the wait-time for passengers and drivers. In this paper, we propose a sequence learning model that can predict future taxi requests in each area of a city based on the recent demand and other relevant information. Remembering information from the past is critical here since taxi requests in the future are correlated with information about actions that happened in the past. For example, someone who requests a taxi to a shopping center, may also request a taxi to return home after few hours. We use one of the best sequence learning methods, Long Short Term Memory (LSTM) that has a gating mechanism to store the relevant information for future use. We evaluate our method on a dataset of taxi requests in New York City by dividing the city into small areas and predicting the demand in each area. We show that this approach outperforms other prediction methods such as feed-forward neural networks. In addition, we show how adding other relevant information such as weather, time, and drop-offs affects the results.**

***Index Terms*—Taxi demand prediction; time series forecasting; recurrent neural networks; mixture density networks.**

I. Introduction

TAXI drivers need to decide where to wait for passengers in order to pick up someone as soon as possible. Passengers also prefer to quickly find a taxi whenever they are ready for pickup. Effective taxi dispatching can help both drivers and passengers to minimize the wait-time to find each other.

Drivers do not have enough information about where passengers and other taxis are and intend to go. Therefore, a taxi center can organize the taxi fleet and efficiently distribute them according to the demand from the entire city [1], [2]. This taxi center is especially needed in the future where self-driving taxis need to decide where to wait and pick up passengers. To build such a taxi center, an intelligent system that can predict the future demand throughout the city is required.

Predicting taxi demand is challenging because it is correlated with many pieces of underlying information. One of the most relevant sources of information is historical taxi trips. Thanks to the Global Positioning System (GPS) technology, taxi trip information can be collected from GPS enabled taxis [3], [4]. Fig. 1 shows an example of taxi request patterns in two areas. Analyzing this data shows that there are repetitive patterns in the data that can help to predict the demand in a particular area at a specific time. Several previous studies have shown that it is possible to learn from past taxi data [5–8].

In this paper, we propose a real-time method for predicting taxi demands in different areas of a city. We divide a big city into smaller areas and aggregate the number of taxi requests in each area during a small time period (e.g. 20 minutes). In this way, past taxi data becomes a data sequence of the number of taxi requests in each area. Then, we train a Long Short Term Memory (LSTM) [9] recurrent neural network (RNN) with this sequential data. The network input is the current taxi demand and other relevant information while the output is the demand in the next time-step. The reason we use a LSTM recurrent neural network is that it can be trained to store all the relevant information in a sequence to predict particular outcomes in the future. In addition, taxi demand prediction is a time series forecasting problem in which an intelligent sequence analysis model is required. LSTMs are the state of the art sequence learning models that are widely used in many applications such as unsegmented handwriting generation [10] and natural language processing [11]. LSTM is capable of learning long-term dependencies by utilizing some gating mechanisms to store information. Therefore, it can for instance remember how many people have requested taxis to attend a concert and after a couple of hours use this information to predict that the same number of people will request taxis from the concert location to different areas.

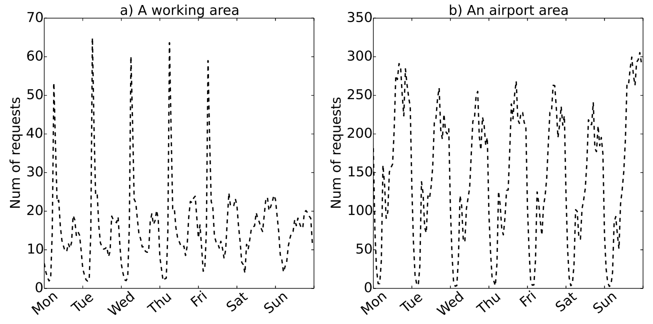


Fig. 1. Taxi demand pattern in two different areas of New York City.

However, predicting real-valued numbers is tricky because many times simply learning the average of the values in the dataset does not give a valid solution. It will also confuse LSTM in the next time-step since the network has not seen the average before. Therefore, we add Mixture Density Networks (MDN) [12] on top of LSTM. In this way, instead of direct predicting a demand value, we output a mixture distribution of the demand. A sample can be drawn from this probability distribution and be treated as the predicted taxi demand.

The remainder of this paper is organized as follows. Section II introduces related works on prediction applications using past taxi data and sequential learning applications of LSTMs. Section III shows how we encode the huge number of GPS records and a brief explanation of recurrent neural networks. Section IV describes the proposed sequence learning model, as well as the training and testing procedures. In Section V, we show the performance metrics and present the experiment results. Lastly, in Section VI we conclude the paper.

II. Related Work

A. Prediction applications using past taxi data

There are few previous research works conducted on taxi demand prediction. Zhang et al. [5] propose a passenger hot-spots recommendation system for taxi drivers. By analyzing the historical taxi data, they extract hot-spots in each time-step and assign a hotness score to each of them. This hotness score will be predicted in each time-step and combined with the driver's location, the $top-k$ hot-spots would be recommended. Zhao et al. [6] define a maximum predictability for the taxi demand at street blocks level. They show the real entropy of past taxi demand sequence which proves that taxi demand is highly predictable. They also implement three prediction algorithms to validate their maximum predictability theory. Moreira-Matias et al. [13] propose a framework consisting of three different prediction models. In each time-step, the predicted demand is a weighted ensemble of predictions from three models. The ensemble weights are updated with individual prediction performances of previous time-steps in a sliding-window. Their framework can make short term demand prediction for the 63 taxi stands in the city of Porto, Portugal. Davis et al. [14] use time-series modeling to forecast taxi travel demand in the city of Bengaluru, India. This information can be given to the drivers in a mobile application so that they know where the demand is higher.

In addition, based on historical and real-time taxi data, dispatching center has been modeled in some studies. Zhang et al. [7] propose a real-time taxi dispatching application. In their system, two kinds of passengers are defined to model real-time taxi demand: previously left-behind passengers and future arriving passengers. Both left-behind and arriving passengers can be simulated at the dispatch center based on the real-time GPS traces of each taxi. A demand inference model called Dmodel is proposed using hidden markov chain to model the state changes of both left-behind and arriving passengers. Miao et al. [8] propose a dispatching framework for balancing taxi supply in a city. Their goal is to match taxi demand and supply and minimize taxi idle driving distance. In their work, the next time-step taxi demand is calculated by the mean value of repeated samples from historical data.

More prediction applications using historical taxi information can be found on topics such as taxi sharing and destination prediction. Yuan et al. [15] present a recommender system for taxi drivers and people expecting to take a taxi. They do this recommendation by learning from taxi GPS traces and mobility pattern of passengers. Ma et al. [16] propose a taxi ride-sharing system that efficiently serves real-time requests sent by taxi users. Rong et al. [17] model the passenger seeking taxis as a Markov Decision Process (MDP) and propose a method to increase the revenue efficiency of taxi drivers. Azevedo et al. [18] look further in the future and investigate the problem of improving the mobility intelligence of self-driving vehicles through a large-scale data analysis. For a more extensive survey on different approaches to analyze and learn from taxi GPS traces, the reader is referred to a survey [19] that focuses on this topic.

B. Sequential learning applications with LSTMs

de Brébisson et al. [20] propose to use recurrent neural networks to predict taxi destination given the beginning of taxi trip GPS traces. However, in our work the network learns the past taxi demand pattern and continuously predicts when and where a new taxi trip will start.

There are many other applications that LSTMs have been widely used. Graves [10] proposed an online handwriting sequence generation with LSTMs. In this application, the data sequence consists of x and y pen coordinates and the points in the sequence when the pen is lifted off. His model can generate highly realistic handwriting. Rahmatizadeh et al. [21] propose to use LSTMs to learn the sequential trajectories for a robot arm. Their goal is to make the robot perform complex manipulation tasks in real world. Some other successful applications include language modeling [22], speech recognition [23] and visual recognition [24]. LSTMs perform very well in all these sequential learning applications.

Overall, aforementioned works on taxi demand prediction motivated us to rely on historical taxi trip information to predict future taxi demands. In terms of the taxi demand prediction, most works either use a weighted method on previous taxi demands or a time series fitting model to fit the demand sequence. The problem is that when the data sequence is very long, the performance is poor in these approaches. Furthermore, time series fitting model has to be trained separately for each area, hence, the patterns learned in one area can not be used in other areas.

One of the differences between our work and the previous works is that our model can capture long term dependencies in a sequence that happen very far away from each other. We train our network on sequences that are as long as a week and this can be easily extended to a month or a year if we have enough computational power to train the network. Another advantage of our model is that we predict all the areas of a city at once using a single model. With this formulation, the patterns learned by the LSTM in one area can be used in other areas. Additionally, our model predicts the entire probability distribution of taxi demands instead of deterministically predicting the number of requests for each area. This approach gives a more realistic prediction as it takes into account the uncertainty while predicting.

III. Mathematical Model

In this section, first we introduce how we convert the high resolution GPS data into the number of taxi requests in each small area of the city. Then, we briefly explain the mathematical formulation of recurrent neural networks.

A. GPS data encoding

In order to be able to accurately predict the demand, we divide the entire city into small areas. It is desired to predict taxi demand in small areas so that the drivers know exactly where to go. However, learning to predict taxi demand in very small areas is difficult. So, we need to choose an area size which is both easy to predict and sufficiently accurate for the drivers. In this paper, we use the Geohash library [25] that can divide a geographical area into smaller subareas with arbitrary precision. Geohash is a geocoding system that has a hierarchical spatial data structure which subdivides space into buckets of grid shape. The size of the grid is determined by the number of characters used in the geohash code.

In our experiment, we use taxi data from 1/1/2013 through 6/30/2016 at NYC [26], which includes around 600 millions taxi trips after data filtering. The dataset specifies for each drop-off and pick-up event the GPS location and the timestamp. In our experiments, we divide the entire New York City into around 6500 areas, with a Geohash precision 7. Each area size is smaller than $153m \times 153m$ under this precision. Then we count the number of taxi requests during every time-step length. In such a way, historical taxi data in each area becomes the number of requests data sequences. These data sequences are fed into the LSTM for sequential patterns learning.

B. Recurrent neural networks

The sequential nature of taxi demand data leads us to the choice of a model that can handle time series data. Recurrent neural networks (RNNs) are one of the most popular models that can process sequential data very well. The idea behind RNNs is to store relevant parts of the input and use this information while predicting the output in the future. Unlike feed-forward neural networks that only predict the output based on the current input, RNNs contain memory in which some important information from the past inputs can be stored. For instance, when we train RNNs on a language modeling task in which we generate a text one character at each time-step, it is better to store what characters the network has predicted in the previous time steps since the next character is dependent on the previous predictions.

RNNs are called recurrent because they perform the same computation on every element of a sequence, with the output being conditioned on the previous computations. A typical RNN structure is given in Fig. 2.

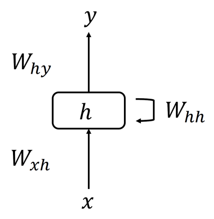


Fig. 2. A recurrent neural network

As we can see, the RNN processes input x, stores hidden state h and outputs y at each time-step t. A loop allows information to be passed over from one step to the next. All Ws are the shared weights among different time-steps. For training these weights, we unroll the network for finite number of time-steps as shown in Fig. 3.

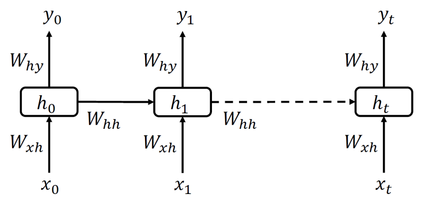


Fig. 3. An unrolled recurrent neural network.

When the network is unrolled, it is more clear why it is being used for sequence learning and how the information is being passed to the future. The computation at each time-step can be formulated as follows:

- x_t is the input at time-step t.
- h_t is the hidden state at time-step t. It is calculated based on the previous hidden state and the current input with the application of a non-linearity. In most RNN implementations, this non-linearity is a hyperbolic tangent: $h_t = tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$.
- y_t is the output at time step t. We can decide how it looks like according to the task. For example, in predicting next word in a sentence, the output y_t can be a probability distribution over a vocabulary.

All parameters W_{xh}, W_{hh} and W_{hy} are shared among each unrolled time-step. So the network is actually performing the same computation at each time-step, but with different inputs x_t. This greatly reduces the total number of parameters in the network and avoids over-fitting on smaller datasets. Hidden state h_t is the main feature of RNNs. It works as the network memory which captures useful information about what happened in all the previous time steps.

Currently, the most commonly used type of RNNs are Long Short Term Memory networks (LSTMs). LSTMs are a special kind of RNN, capable of learning long-term dependencies due to their gating mechanism. They were introduced by Hochreiter & Schmidhuber [9], and were refined and popularized by many people in the following years.

IV. Taxi Demand Prediction Model

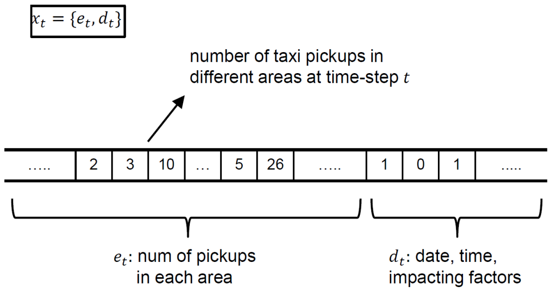
In this section, we discuss the sequence learning model. The number of taxi requests in each area depends on many underlying factors unavailable to our model. This will naturally cause uncertainty in the model. So instead of forecasting a deterministic taxi demand value, we use a stochastic model that can predict the entire probability distribution of taxi demands in all areas. We then use this probability distribution to decide taxi demand for each area.

A. Mixture density networks

The most successful application of neural networks has been achieved on classification tasks. When it comes to predicting real-valued data, the choice of network structure is very important. The idea of mixture density networks (MDNs) [12] is to use the outputs of a neural network to parameterize a mixture distribution. Unlike the model with mean squared error (MSE) cost which is deterministic, MDNs can model stochastic behaviors. They can be used in prediction applications in which an output may have multiple possible outcomes. In our application, rather than direct predicting the number of taxi requests, the neural network outputs the parameters of a mixture model. These parameters are the mean and variance of each Gaussian kernel and also the mixing coefficient of each kernel which shows how probable that kernel is. Given the parameters of the mixture distribution, we can draw a sample from it and use this sample as the final prediction.

B. LSTM-MDN sequence learning model

As described in Section III, we divide the entire city into small areas and convert the past taxi data into week-long data sequences. Fig. 4 shows the structure of the data sequence at one time-step. For each time-step t, the data sequence consists of two parts: e_t and d_t. e_t represents the number of pickups in each area and its length is the number of small areas in the entire city. d_t represents date, day of the week, hours, minutes and other impacting factors at time-step t. The input to the network at each time-step is $x_t = \{e_t, d_t\}$ and the network will try to predict $y_t = e_{t+1}$.

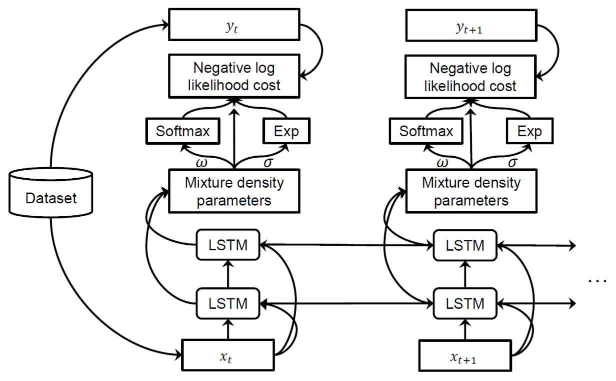


$x_t = \{e_t, d_t\}$
number of taxi pickups in different areas at time-step t
..... 2 3 10 ... 5 26 ..... 1 0 1 .....
e_t: num of pickups in each area
d_t: date, time, impacting factors

Fig. 4. The input data structure at one time-step.

The sequence learning model is created based on an LSTM recurrent neural network and the MDNs. Fig. 5 shows the structure of the unrolled LSTM-MDN learning model. The total unrolling length is a hyper-parameter that can be set according to testing scenario. LSTM can encode the useful information of the past in a single or multiple layers. The input to each layer is the output of the previous layer concatenated with the network input. Each LSTM layer predicts its output based on its current input and its internal state. The concatenation of outputs of all layers will be used to predict the output of the network which will be compared with y_t, the real demand value from the dataset, to form the error signal. We use two LSTM layers in which each layer contains $1200-1500$ neurons based on the specific testing scenario.

As shown in Fig. 5, the output of LSTM layers are mixture density parameters with the total number of $M \times (N+2)$ in which M is the number of Gaussian kernels, and N is the number of areas in the city. For each Gaussian kernel, we have N neurons for the means $\mu_k(x)$, one neuron for the variances $\sigma_k(x)$, and another neuron for the mixing coefficient $w_k(x)$. To satisfy the constraint $\sum_{k=1}^{M} w_k(x) = 1$, the corresponding neurons are passed through a softmax function. The softmax function is regularly used in multiclass classification methods to "squash" a vector of n arbitrary real values z into a set



y_t
Negative log likelihood cost
Softmax
Exp
ω
σ
Mixture density parameters
Dataset
LSTM
LSTM
x_t
y_{t+1}
Negative log likelihood cost
Softmax
Exp
ω
σ
Mixture density parameters
LSTM
LSTM
x_{t+1}
...

Fig. 5. The LSTM-MDN learning model unrolled through time-steps.

of values that add up to 1, and which can be interpreted as probabilities:

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}} \tag{1}$$

The neurons corresponding to the variances $\sigma_k(x)$ are passed through an exponential function and the neurons corresponding to the means $\mu_k(x)$ are used without any further changes. The probability density of the next output y_t can be modeled using a weighted sum of M Gaussian kernels:

$$p(y_t|x) = \sum_{k=1}^{M} w_k(x) g_k(y_t|x) \tag{2}$$

where $g_k(y_t|x)$ is the k^{th} multivariate Gaussian kernel. Note that both the mixing coefficient and the Gaussian kernels are conditioned on the complete history of the inputs till current time-step $x = \{x_1 \ldots x_t\}$. The multivariate Gaussian kernel can be represented as:

$$g_k(y_t|x) = \frac{1}{(2\pi)^{N/2}\sigma_k(x)} \exp\left\{-\frac{\|y_t - \mu_k(x)\|^2}{2\sigma_k(x)^2}\right\} \tag{3}$$

where the vector $\mu_k(x)$ is the center of k^{th} kernel. We do not calculate the full covariance matrices for each kernel, since this form of Gaussian mixture model is general enough to approximate any density function [27]. Finally, we can define the error function in terms of negative logarithm likelihood:

$$E_t = -ln\left\{\sum_{k=1}^{M} w_k(x) g_k(y_t|x)\right\} \tag{4}$$

After the model is trained, we can make a prediction for time-step $t+1$ by inputting taxi demand at time-step t. As we can see in Fig. 6, we use the output which is the mixture density parameters to parameterize a Gaussian mixture distribution. A sample can then be drawn from this distribution and this sample would be the prediction of the next time-step taxi demand, $\widehat{e_{t+1}}$. This prediction can be repeated in a loop to predict taxi demand for multiple time-steps.

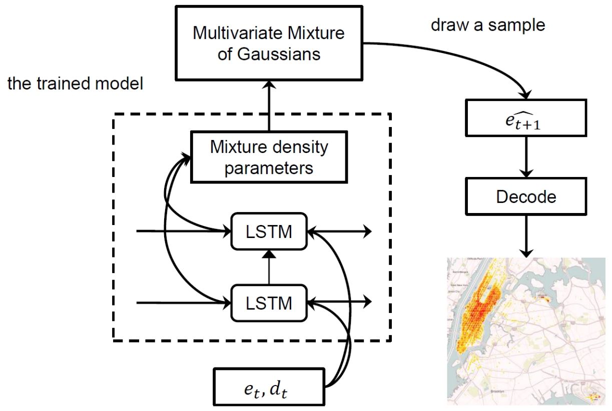


Fig. 6. The LSTM-MDN model perform one prediction for time $t+1$.

C. LSTM-MDN-Conditional model

In the LSTM-MDN model, the probability distribution of taxi demands in all areas are predicted at the same time-step. This means that prediction in each area is conditioned on all areas of all previous time-steps. However, the taxi demand in an area might not only be related to the past, but also to the taxi demands of other areas in current time-step. So instead of outputting a joint distribution for all areas in a single time-step, we ask the network to predict the conditional distribution of each area at a single time-step. This approach has been adopted in other works such as in Pixel RNNs [28] and Neural Autoregressive Distribution Estimator (NADE) [29]. Fig. 7 shows the idea of generating y_t^i conditioned on all the previously predicted demands left. Here y_t^i represents the predicted taxi demand in area i at time-step t.

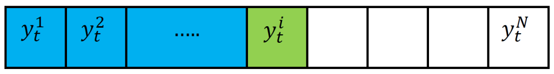


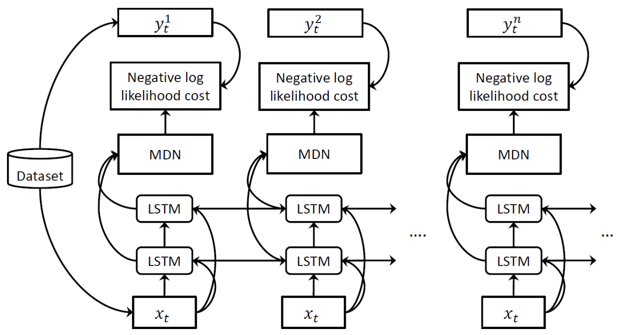
Fig. 7. Generate conditional distributions in a sequential way.

We call this approach LSTM-MDN-Conditional model. It has the same input x_t as the LSTM-MDN mode, but each x_t only leads to one area taxi demand output. Unlike predicting taxi demands for all areas in LSTM-MDN mode, this model sequentially predicts demand for each area in a conditional way. Fig. 8 shows the unrolled LSTM-MDN-Conditional model structure for one time-step prediction.

This LSTM-MDN-Conditional model not only learns demand patterns from past taxi data, but also takes into account current demands in other areas. Training such a model takes much longer time than the LSTM-MDN model because the LSTM needs to be unrolled for much longer time-steps. For a city with N areas, the model needs to be unrolled N times more compared to the LSTM-MDN model. Fig. 9 shows a density map of real and predicted taxi demands over the entire city.

V. Experimental Study

In this section, we evaluate the proposed LSTM-MDN and LSTM-MDN-Conditional models on a dataset of taxi requests



y_t^1
y_t^2
y_t^n
Negative log likelihood cost
Negative log likelihood cost
Negative log likelihood cost
Dataset
MDN
MDN
MDN
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
....
...
x_t
x_t
x_t

Fig. 8. The unrolled LSTM-MDN-Conditional model for one time-step prediction.

and see how well they can predict taxi demand in the future. In addition, we compare our models with two other baselines and show that they outperform both.

A. Experimental setup

We evaluate the performance of the proposed network models with the New York City taxi trip dataset [26]. There are two kinds of cabs in NYC: the yellow cabs, which operate mostly in Manhattan, and the green cabs, which operate mostly in the suburbs. The dataset contains taxi trips from January 2009 through June 2016 for both yellow and green cabs. Each taxi trip has a pickup time and location information. In this application, we use its most recent 3.5 years data: from January 2013 through June 2016, which contains over 600 million taxi trips after data filtering. We use 80% of the data for training and keep the remaining 20% for validation. The network model is implemented in Blocks [30] framework that is built on top of Theano [31]. We stop the training when the validation error does not change for 20 epochs. The training takes 2-4 hours on a GTX 1080 for each of the experiments. After the training, the time that takes for the network to predict the demand is less than a second. Note that the prediction time is more important than the training time. This is because the model can be trained once but once deployed it needs to predict the demand in a loop to provide this information in real-time.

Theoretically, LSTM can be trained using arbitrary sequence lengths. However, constrained by the computational power, we use every one week data as a sequence and cut it into time-steps with different lengths. For example, if the time-step length is $60 mins$, the sequence length would be 24×7. If the time-step length is $20 mins$, the sequence length would be $24 \times 3 \times 7$. For the $60 mins$ case, the encoded input data shape is $(182, 168, 6494)$ in which 182 is the total number of sequences in the dataset, i.e., number of weeks in the 3.5 years, 168 is the sequence length: 24×7, and 6494 is the number of features consisting of number of areas, impacting factors such as date, day of the week and other information. Table I includes the list of parameters in the experiments.

B. Performance metrics and baselines

To systematically examine the performance of our prediction approach, we include results with two widely used

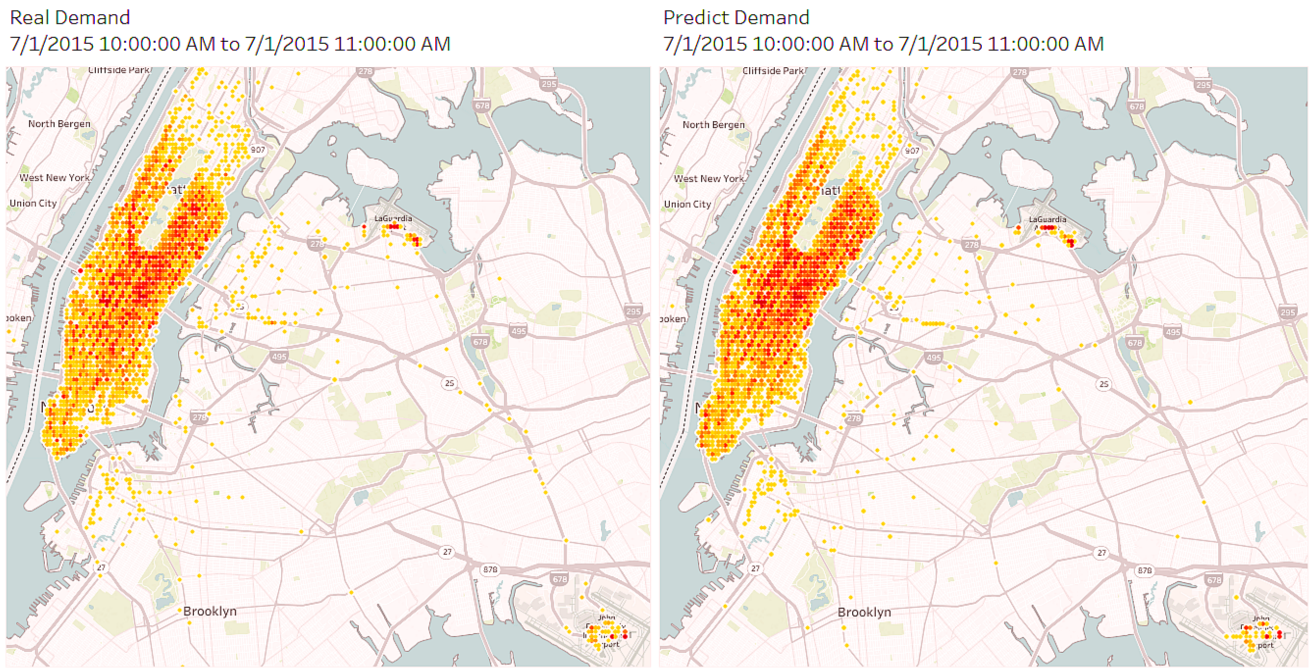


Fig. 9. The density map of real demand and the predicted demand. Red areas show high demand for taxis, yellow areas show lower demand, and there is no demand in other areas. The figure illustrates that the difference between the prediction and the real value is very small.

TABLE I
EXPERIMENTAL PARAMETERS

Area/grid size	$\leq 153m \times 153m$
Data of each sequence	1 week data
Time-step length	5/10/20/30/60 mins
Sequence length	2016/1008/504/336/168
Number of sequences	182
Number of areas (N)	6424
Number of hidden layers	1-2
Number of nodes in each hidden layer	1200-1500
Number of mixture Gaussian kernels	10-20

prediction error metrics: Symmetric Mean Absolute Percentage Error (sMAPE) [13], [32] and Root Mean Square Error (RMSE) [33], [34]. $Y_{i,t}$ is the real taxi demand in area i at time-step t, while $\widehat{Y}_{i,t}$ is the predicted taxi demand. The sMAPE and RMSE in area i over time $[1-T]$ would be:

$$sMAPE_i = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_{i,t}-\widehat{Y}_{i,t}|}{Y_{i,t}+\widehat{Y}_{i,t}+c} \tag{5}$$

$$RMSE_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(Y_{i,t}-\widehat{Y}_{i,t}\right)^2} \tag{6}$$

The constant c in Eq. 5 is a small number ($c = 1$ in this application) to avoid division by zero when both $Y_{i,t}$ and $\widehat{Y}_{i,t}$ are 0. Similarly, when evaluating the prediction performance over the entire city, the sMAPE and RMSE of all areas at time-step t would be:

$$sMAPE_t = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_{i,t}-\widehat{Y}_{i,t}|}{Y_{i,t}+\widehat{Y}_{i,t}+c} \tag{7}$$

$$RMSE_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Y_{i,t}-\widehat{Y}_{i,t}\right)^2} \tag{8}$$

Here N is the total number of areas in the city. From the statistic perspective, RMSE shows the difference of the predicted value from the real value while the sMAPE describes a percentile error.

To evaluate the performance of the proposed models, we compare the outcomes with prediction approaches based on another two strategies: the fully connected feed-forward neural networks and naive statistic average.

1) Fully connected feed-forward neural network predictor: Feed-forward neural networks are commonly used for classification and regression problems. Feed-forward neurons have connections from their input to their output. The main difference between feed-forward neural networks and recurrent neural networks is that in RNNs, the recurrent connection from the output to the input at next time-step makes the network capable of storing information. In this approach, the layers are fully connected which means that neurons between two adjacent layers are all connected together.

2) Naive statistic average predictor: This approach predicts based on the mean value of past demands in a sliding-window. For example, if it is 10:00 am on Monday, the predicted value would be the average of demands at 10:00 am in the past 5 Mondays. While we use the term "naive", even this approach requires the maintenance of long term, detailed statistics of both spatial and temporal distributions of pickups. It is very likely a good approximation of what taxi companies are currently deploying.

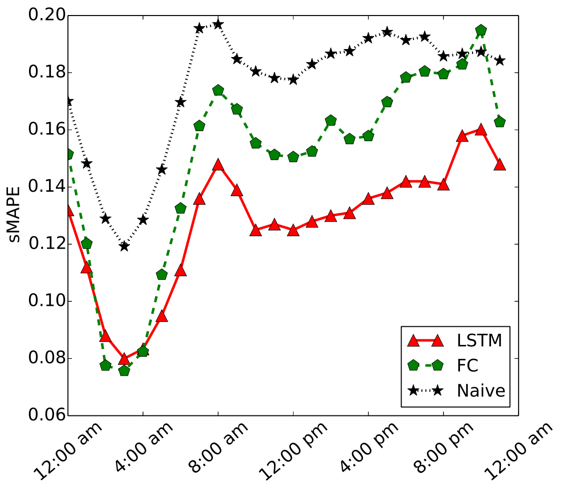


Fig. 10. Prediction performance of different approaches according to sMAPE.

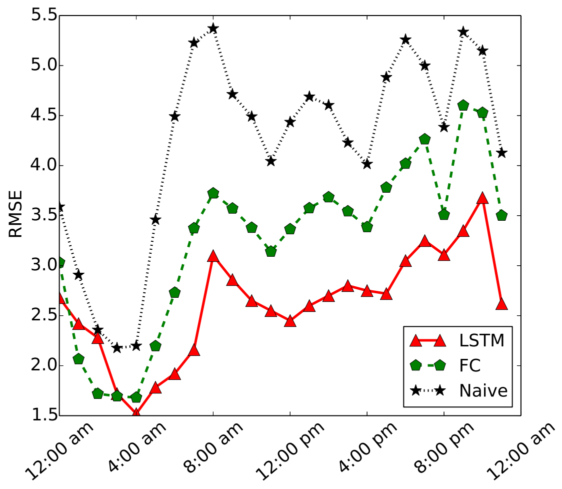


Fig. 11. Prediction performance of different approaches according to RMSE.

C. Performance results

First, we report prediction sMAPE and RMSE over the entire city (all prediction areas). Second, we show the prediction performance at some specific areas as time passes. Lastly, we analyze the importance of different impacting factors on prediction performance. For the four different predictors based on LSTM-MDN-Conditional, LSTM-MDN, fully connected feed-forward neural networks and naive statistic average, we respectively use *LSTM-C, LSTM, FC* and *Naive* for short. We do not include the LSTM-C predictor in the entire city performance comparison. We only evaluate the LSTM-C model on specific areas. This is because conditioning only neighboring areas on each other should be enough. Two areas that are very far from each other most probably will not affect each other. In addition, if we condition more areas on each other, LSTM will have a difficult time relating the events that happen many time-steps away from each other.

1) Performance over the entire city: To evaluate the prediction performance over the entire city which includes about 6500 areas, we compare the performance of the LSTM predictor, the FC predictor and the Naive predictor in terms of RMSE and sMAPE from Eq. 7 and Eq. 8.

We report sMAPE and RMSE over the entire city in Fig. 10 and Fig. 11. As we can see, though they are different prediction error metrics, they share some common patterns. For instance, both of them reach the minimum values at about 3am and peak at about 8am and 10pm. In both figures, LSTM shows better performance in prediction than the FC and Naive predictors. In Fig. 10, sMAPE shows the mean absolute percentage error, which gives us a way to calculate the prediction accuracy and observe that it is more than 80%. In Fig. 11, RMSE shows the root mean squared difference between the predicted demands and the real demands. Note that, to the real demands in different areas, we have $min = 0$, $max = 535$ and standard deviation $\sigma = 12.0$. The time-step length is $60mins$.

Fig. 12 reports the error bar of prediction RMSE over the entire city, with the standard deviation in one week. We show this RMSE with different time-step lengths in the LSTM predictor. Basically, smaller time-step length means smaller number of pickups in each time slot, which does affect the final RMSE. To avoid this, we sum up the predicted number of pickups every $60mins$. As we can see in Fig. 12, the model has the minimum RMSE at time-step length either 10 or 20 $mins$. Overall, the RMSEs are very close under different time-step lengths.

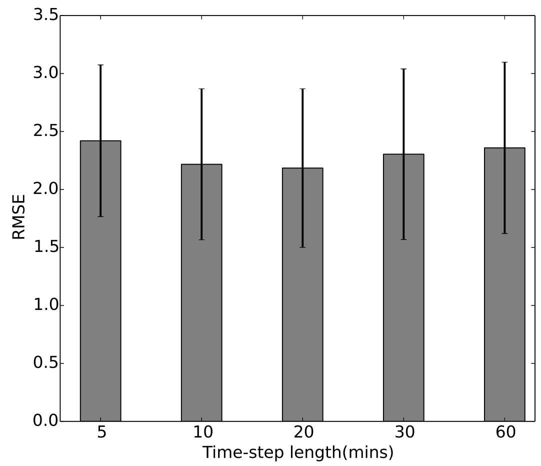


Fig. 12. RMSE of different time-step lengths. With the real number of pickups, $min = 0$, $max = 535$ and standard deviation $\sigma = 12.0$.

2) Performance at specific areas: We compare the prediction performances of LSTM-C, LSTM, FC and the Naive predictors in specific areas. First of all, we select two areas that their taxi demands in a week are shown in Fig. 13-a and Fig. 13-b. The reason we select these two areas is that both of them show regular demand patterns on both weekdays and weekends. The first area is a working area while the second area is one of the most popular areas (in terms of taxi requests) according to our analysis to all the past taxi data in NYC. The first area is close to the intersection of West 40rd Street and 9th Ave in downtown while the second area is close to the intersection of West 33rd Street and 7th Ave.

The time-step length used here is $60mins$ in both areas.

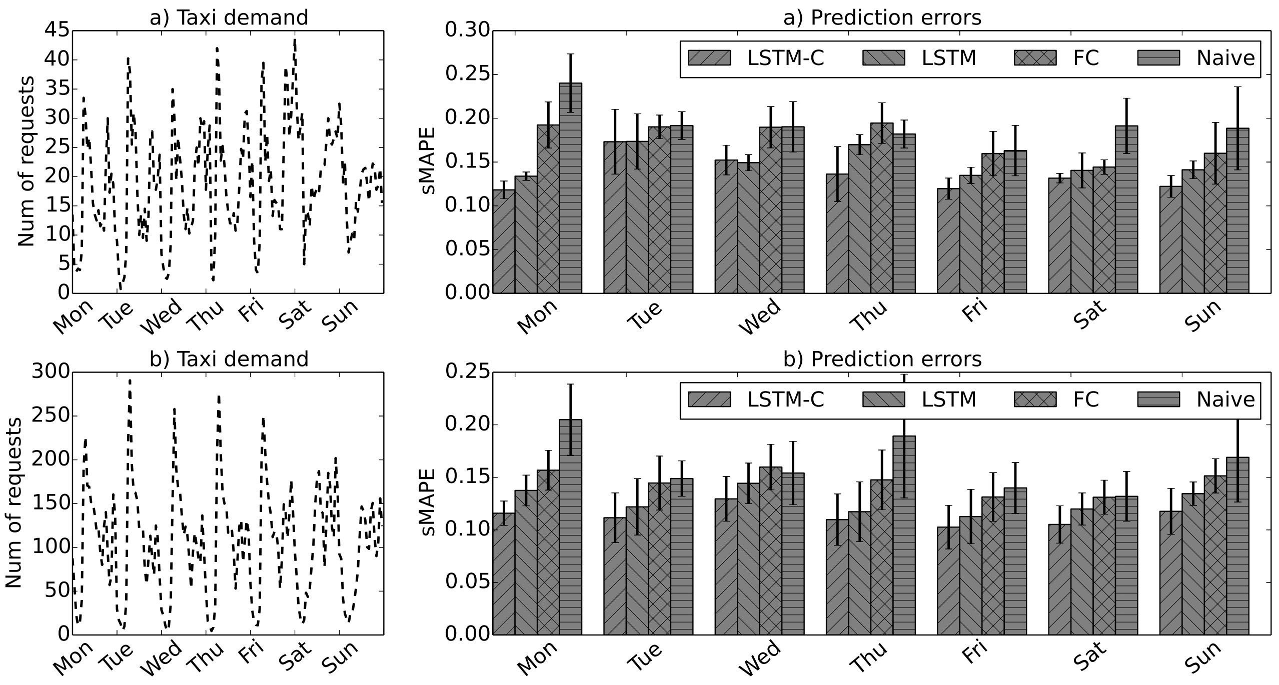


Fig. 13. Comparison in areas with different demand patterns.

The right part of Fig. 13 shows the prediction sMAPE of each day in both areas. We include continuous 5 weeks prediction results and show each standard deviation with an error bar.

As we can see from Fig. 13, in both areas, LSTM-C and LSTM models outperform FC and Naive models in most days. This proves LSTM is good at learning sequential information, even though the sequence length is as long as a week. LSTM-C can give more accurate prediction results than the LSTM, because it considers both past information and current conditional information of other areas. But it requires a lot of computational power to train such a conditional model. FC sometimes results in larger errors than the Naive predictor due to the irregularities in sequence patterns.

3) Importance of impacting factors: In this part, we report the importance of different impacting factors in our prediction models. The impacting factors in our model include *Date & Time, Day of week, Weather and Drop-offs*. *Date & Time* includes year, month, date and time of each time-step. *Day of week* represents which day of week that time-step is. *Weather* includes 4 different types: rain, snow, fog and thunder. We get the official weather information of NYC from National Oceanic and Atmospheric Administration (NOAA). It includes climate observations from three land-based stations in the city. We also include the number of *Drop-offs* as an impacting factor to see if there is any relation between the pickups and drop-offs in each area. To show the impacts of these inputs on prediction performance, we conduct two experiments. Both experiments are implemented on the LSTM-MDN model since we evaluate the prediction performance for the entire city.

In experiment I, we want to show the impact of each single factor on prediction performance. We design different models, which are shown in Table II, with each single factor as model input. The control group here is the model with only past taxi pickups as input. All models are expected to output the next time-step taxi demand in the city. Then we train each model and show the prediction performance in Fig. 14.

TABLE II
MODEL WITH DIFFERENT IMPACTING FACTORS: I

Model	Model input	Model output
Model A	Pickups	Next time-step demand
Model B	Date & Time	Next time-step demand
Model C	Day of week	Next time-step demand
Model D	Weather	Next time-step demand
Model E	Drop-offs	Next time-step demand

As it is shown in Fig. 14, pickup is the most valuable information in making future taxi demand prediction. In model B and C, since no past taxi trip information is provided, the models are trying to remember the mapping from the input to the real taxi demand at each time-step. In this case, LSTM works similar to the feed-forward neural networks. Model D has the worst performance due to two reasons: one reason is that it is hard for the model to make a taxi demand prediction only based on a weather information. Another reason is we only have climate information from three land-based observations, which is not a good descriptor of the whole city. In model E, we use number of drop-offs in each area as input. It is interesting to find out it can give a prediction performance close to model C, which means that the drop-offs pattern has a close relation to the pickups pattern. However, the drop-off information is not as useful as pickup information. This is because the network needs to remember the drop-off information to use them later. But, the pickup information is the most informative feature probably because it does not change much in one time-step (from input to the output).

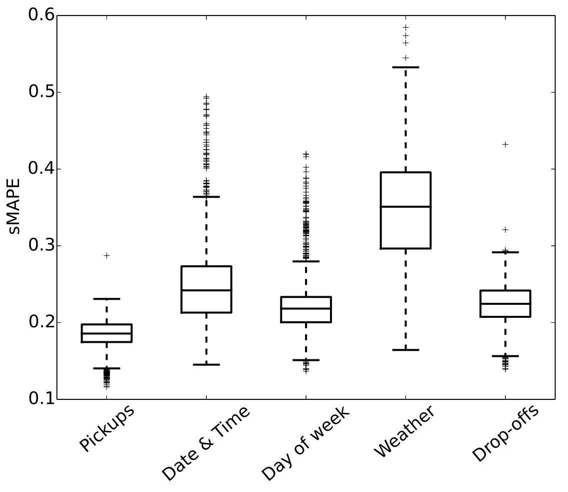


Fig. 14. Prediction performance on different single impacting factors.

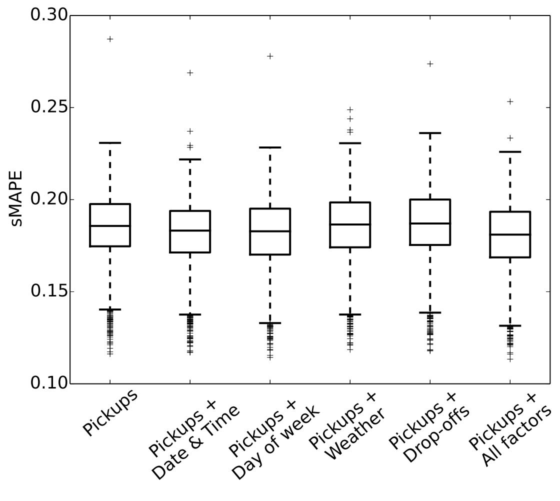


Fig. 15. Prediction performance on different impacting factors.

In experiment II, we add impacting factors to the number of pickups as model inputs to see if they really improve the prediction accuracy. Table III shows the input to each model. In the last group model F, we include pickups and all impacting factors as input.

TABLE III
MODEL WITH DIFFERENT IMPACTING FACTORS: II

Model	Model input	Model output
Model A	Pickups	Next time-step demand
Model B	Pickups + Date & Time	Next time-step demand
Model C	Pickups + Day of week	Next time-step demand
Model D	Pickups + Weather	Next time-step demand
Model E	Pickups + Drop-offs	Next time-step demand
Model F	Pickups + All factors	Next time-step demand

Fig. 15 shows each model's prediction performance. It can be seen that all the models have close performances. The reason is that pickup information is so informative that makes the effect of other factors very small. Model D and E have the worst performance compared with other models. In model F, we include all impacting factors. As we can see, the median prediction error is about 17%, which means a median accuracy of around 83% can be obtained.

Overall, the experimental results show that LSTM-C and LSTM outperform the other prediction approaches. This is because LSTM can see and process information in the previous time-steps. For instance, if a group of people request taxis to go to a concert, it will remember this information and use it to predict that after a couple of hours there would be almost the same number of requests in the concert area. The FC network can find the best mapping from the time and geographical information to the number of requests without having access to the demand in the previous time-steps. So this limitation causes larger errors in its prediction. The Naive approach is even more restricted since it has access to only a small history of the demand in one area unlike the FC which is trained on all the historical data of all areas.

To sum up, for better prediction, we need to use a model that is very powerful and properly conditions the output on all the available information. In addition, the best prediction performance is achieved when all the impacting factors considered in this work are available as input to the network.

VI. CONCLUSION

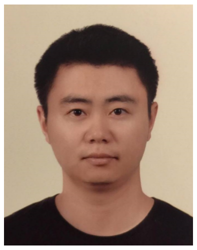
We propose a sequence learning model based on recurrent neural networks and mixture density networks to predict taxi demand in different areas of a city. By learning from historical taxi demand patterns, the proposed LSTM-MDN model can make taxi demand predictions for the entire city. Three and a half years taxi trips data of New York City is used to train our model. Experimental results show that the LSTM-MDN model can get a good accuracy of around 83% at the city level. We further extend the LSTM-MDN model to a conditional model in which each prediction is not only made based on past taxi information, but also conditioned on current demands in other areas. We show that this approach can further improve the prediction performance. In addition, we show that our models outperform two other prediction models based on fully connected feed-forward neural networks and naive statistic average.

This work can be extended by adding more information to the input of the network such as where businesses, shops, restaurants, etc. are located. In addition, we can organize the taxis in a city and distribute them in real-time according to the demand prediction by our model. This can help a lot in situations where in some areas there is large demand but the taxi drivers are competing with each other for having passengers in another area of the city. A central taxi dispatch system would be especially beneficial when in the future self-driving cars need to be organized automatically to respond to the taxi requests in a city. Such a system can save a lot of time from people who need taxis. In addition, it can save so much gas that is currently being spent by taxis to find passengers.

REFERENCES

[1] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie, "T-finder: A recommender system for finding passengers and vacant taxis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2390–2403, 2013.

[2] K. T. Seow, N. H. Dang, and D. H. Lee, “A collaborative multiagent taxi-dispatch system,” *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 607–616, 2010.
[3] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, “Quantifying the benefits of vehicle pooling with shareability networks,” *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13 290–13 294, 2014.
[4] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-scale transportation network congestion evolution prediction using deep learning theory,” *PloS one*, vol. 10, no. 3, p. e0119044, 2015.
[5] K. Zhang, Z. Feng, S. Chen, K. Huang, and G. Wang, “A framework for passengers demand prediction and recommendation,” in *Proc. of IEEE SCC’16*, June 2016, pp. 340–347.
[6] K. Zhao, D. Khryashchev, J. Freire, C. Silva, and H. Vo, “Predicting taxi demand at high spatial resolution: Approaching the limit of predictability,” in *Proc. of IEEE BigData’16*, December 2016, pp. 833–842.
[7] D. Zhang, T. He, S. Lin, S. Munir, and J. A. Stankovic, “Taxi-passenger-demand modeling based on big data from a roving sensor network,” *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2016.
[8] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas, “Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463–478, 2016.
[9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[10] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.
[11] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. of NIPS’14*, December 2014, pp. 3104–3112.
[12] C. M. Bishop, *Mixture density networks*. Aston University, 1994.
[13] L. Moreira-Matias, 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, pp. 1393–1402, 2013.
[14] N. Davis, G. Raina, and K. Jagannathan, “A multi-level clustering approach for forecasting taxi travel demand,” in *Proc. of IEEE ITSC’16*, December 2016, pp. 223–228.
[15] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, “Where to find my next passenger,” in *Proc. of ACM UbiComp’11*, September 2011, pp. 109–118.
[16] S. Ma, Y. Zheng, and O. Wolfson, “T-share: A large-scale dynamic taxi ridesharing service,” in *Proc. of IEEE ICDE’13*, April 2013, pp. 410–421.
[17] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu, “The rich and the poor: A markov decision process approach to optimizing taxi driver revenue efficiency,” in *Proc. of ACM CIKM’16*, October 2016, pp. 2329–2334.
[18] J. Azevedo, P. M. d’Orey, and M. Ferreira, “On the mobile intelligence of autonomous vehicles,” in *Proc. of IEEE NOMS’16*, April 2016, pp. 1169–1174.
[19] P. S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan, “From taxi gps traces to social and community dynamics: A survey,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, p. 17, 2013.
[20] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, “Artificial neural networks applied to taxi destination prediction,” *arXiv preprint arXiv:1508.00021*, 2015.
[21] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni, “Learning real manipulation tasks from virtual demonstrations using lstm,” *arXiv preprint arXiv:1603.03833*, 2016.
[22] A. Karpathy, J. Johnson, and F.-F. Li, “Visualizing and understanding recurrent networks,” *arXiv preprint arXiv:1506.02078*, 2015.
[23] A. Graves, A. rahman Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. of IEEE icassp’13*, May 2013, pp. 6645–6649.
[24] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
[25] G. Niemeyer. (2008) Tips & tricks about geohash. [Online]. Available: http://geohash.org/site/tips.html
[26] NYC Taxi Limousine Commission. Taxi and limousine commission (tlc) trip record data. [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
[27] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. New York: Marcel Dekker, 1988, vol. 84.
[28] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *Proc. of ICML’16*, June 2016, pp. 1747–1756.
[29] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” in *Proc. of AISTATS’11*, April 2011, pp. 29–37.
[30] B. Van Merrinboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski and Y. Bengio, “Blocks and fuel: Frameworks for deep learning,” *arXiv preprint arXiv:1506.00619*, 2015.
[31] The Theano Development, “Theano: A python framework for fast computation of mathematical expressions,” *arXiv preprint arXiv:1605.02688*, 2016.
[32] P. Lopez-Garcia, E. Onieva, E. Osaba, A. D. Masegosa, and A. Perallos, “A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 557–569, 2016.
[33] Y. Lv, 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, pp. 865–873, 2015.
[34] M. Yang, Y. Liu, and Z. You, “The reliability of travel time forecasting,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 162–171, 2010.



Jun Xu is currently working toward the PhD degree in Computer Science from the Department of Computer Science, University of Central Florida (UCF). He received his MS degree in Electrical Engineering from Beijing University of Posts and Telecommunications, China. His research interests include mobility model, agent path planning and machine learning.

Rouhollah Rahmatizadeh received the BS degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2012 and the MS degree in computer science from the University of Central Florida (UCF), Orlando, in 2014. He is currently pursuing the Ph.D. degree in computer science at UCF. His research interests include machine learning, robotics, and wireless sensor networks.

Ladislau Bölöni is a Professor at the Department of Computer Science at University of Central Florida (with a secondary joint appointment in the Dept. of Electrical and Computer Engineering). He received a PhD degree from the Computer Sciences Department of Purdue University in May 2000, an MSc degree from the Computer Sciences department of Purdue University in 1999 and BSc. Computer Engineering with Honors from the Technical University of Cluj-Napoca , Romania in 1993. He received a fellowship from the Computer and Automation Research Institute of the Hungarian Academy of Sciences for the 1994-95 academic year. He is a senior member of IEEE, member of the ACM, AAAI and the Upsilon Pi Epsilon honorary society. His research interests include cognitive science, autonomous agents, grid computing and wireless networking.

Damla Turgut is an Associate Professor at the Department of Computer Science of University of Central Florida. She received her BS, MS, and Ph.D. degrees from the Computer Science and Engineering Department of University of Texas at Arlington. Her research interests include wireless ad hoc, sensor, underwater and vehicular networks, cloud computing as well as considerations of privacy in the Internet of Things. She is also interested in applying big data techniques for improving STEM education for women and minorities. Her recent honors and awards include University Excellence Award in Professional Service in April 2017 and being featured in the UCF Women Making History series in March 2015. She was co-recipient of the Best Paper Award at the IEEE ICC 2013. Dr. Turgut serves as a member of the editorial board and of the technical program committee of ACM and IEEE journals and international conferences. She is a member of IEEE, ACM, and the Upsilon Pi Epsilon honorary society.