A Sequence Learning Model with Recurrent Neural Networks for Taxi Demand Prediction

Jun Xu, Rouhollah Rahmatizadeh, Ladislau Bölöni and Damla Turgut
Department of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL
Email: {junxu,rrahmati,lboloni,turgut}@eecs.ucf.edu

***Abstract*—In this paper, we focus on an application of recurrent neural networks for learning a model that predicts taxi demand based on the requests in the past. A model that can learn time series data is necessary here since taxi requests in the future relate to the requests in the past. For instance, someone who requests a taxi to a movie theater, may also request a taxi to return home after few hours. We use Long Short Term Memory (LSTM), one of the best models for learning time series data. For training the network, we encode the historical taxi requests from the official New York City taxi trip dataset and add date, day of the week and time as impacting factors. Experimental results show that our approach outperforms the prediction heuristics based on feed-forward neural networks and naive statistic average.**

***Index Terms*— taxi demand prediction; time series regression; recurrent neural networks; mixture density networks; Internet of things.**

I. Introduction

Taxi plays an important role in public transportation system in cities. Taxi drivers and passengers prefer to find each other as soon as possible. In addition, drivers want to spend minimum fuel for finding the next passenger. In order to minimize the wait time for passengers and drivers, drivers need to evaluate the city areas in which the probability of a passenger requesting a taxi is higher. They usually gather some intuition by observing the density of requests in different areas of the city. For instance, on Saturday late nights more people request taxis from downtown close to night clubs compared to other nights of the week. However, this prediction can be improved by considering all the past and also real-time information from the entire city instead of only one driver's observations. If a model can predict the density of taxi demand, it can be available for all the drivers even for the ones who are not familiar with different areas of a city. We can also look a bit ahead and consider the future where self-driving taxis need to decide where to look for passengers without a human help.

This explains the motivation behind our work which is training a model that takes into account all the taxi trip information from the past to predict future taxi demands. Nowadays, many taxis are equipped with systems providing taxi trip information such as the pickup and drop-off GPS locations. Previous studies [1–4] have shown that such historical taxi trip data can provide us rich insights about how taxi demand varies from area to area and how the demand evolves during the course of a day and a week.

In this paper, we propose a taxi demand predictor that can predict taxi demand in any target area of the city in the next hours, days and weeks. For a given area, the past taxi demand data can be treated as a sequence. Fig. 1 shows taxi demands at two different places in New York City over a week. We observe that in a specific area of the city, the historical taxi demand shows a similar sequential pattern every week. Motivated by this periodical taxi demand pattern, we design a sequence learning model that learns the demand patterns from the sequential data. In our work, we only focus on designing such a predictive model. However, we can extend this work and design a centralized Internet of Things (IoT) application [5], [6] consists of a network of taxis aware of each other and cooperatively and efficiently respond to the taxi requests throughout the entire city.

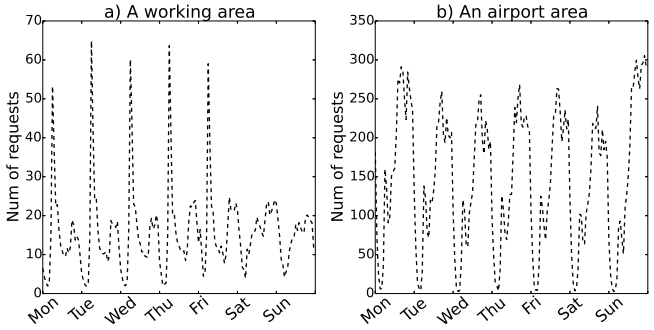

Fig. 1. Taxi demand patterns in two different areas.

We use Recurrent Neural Networks (RNNs) to predict taxi demand density in the future. RNNs can be trained to store all the relevant information in a sequence to predict particular outcomes in the future. They have been widely used in many applications such as unsegmented handwriting generation [7] and natural language processing [8]. They can process arbitrary-length sequences of inputs especially when the elements of the sequence are not independent, i.e., there exits some hidden relations among different elements of the sequence. Currently, the most commonly used type of RNN is Long Short Term Memory (LSTM) [9].

In this application, first we divide the whole city into small areas, then for each area, we encode all the past taxi demands into week-long sequences. Second, we feed the sequential data to the LSTM neural network and make the network

learn the taxi demand patterns in each area. We also include the date, day of the week and time information as impacting factors to the training model. Rather than forecasting a deterministic taxi demand number, we use mixture density networks (MDNs) [10], a stochastic model, that can predict the entire probability distribution of taxi demands in different areas. After training the model, the prediction can be made continuously by inputting new taxi demands in real-time. The model can remember the useful information and predict taxi demand densities of the future based on both the new input and the previously stored information. For example, taxi demand prediction for time-step $t+1$ will be made based on inputs at time-steps $[1, t-1]$ and the new input which is the current taxi demand on time-step t.

We evaluate the performance of the proposed network model with the New York taxi trip dataset [11], maintained by the NYC Taxi & Limousine Commission, containing taxi trips from January 2009 through June 2016. In this application, we use its most recent 3.5 years data: from January 2013 through June 2016, which contains over 600 million trips after data filtering. In the experiments, we divide the entire city into 6500 areas and let the proposed model predict for all areas at each time-step. We use 80% of the data for training and keep the remaining 20% for validation. Compared to baselines, the LSTM predictor can provide smaller prediction errors.

The remainder of this paper is organized as follows. Section II introduces related works on taxi demand prediction and sequence learning applications of LSTM. Section III shows how we encode the huge number of GPS records and a brief explanation of RNNs. 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 testing results. Lastly, in Section VI we conclude the paper.

II. Related Work

There are few previous research works conducted on taxi demand prediction. Zhang et al. [1] 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 will be predicted in each time-step and combined with the driver's location, the $top{-}k$ hot-spots are recommended. Zhao et al. [2] 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. [4] propose a framework consists 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. In addition, based on historical and real-time taxi data, dispatching center has been modeled in some studies. Zhang et al. [3] 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 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. [5] 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.

For the sequential learning application using RNNs, Graves [7] propose an online handwriting sequence generation with RNNs. 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. [12] propose to use RNNs to learn the sequential trajectories for a robot arm. Their goal is to make the robot perform complex manipulation tasks in real world such as pushing objects to a target area. Some other successful applications include language modeling [13], speech recognition [14] and visual recognition [15]. RNNs perform very well in modeling and reproducing patterns in sequential data.

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 prediction, most existing works either use a weighted average 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 both 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 existing 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 pattern learned by the LSTM in one area can be used in other areas. Additionally, our model predicts the entire probability distribution of taxi demand instead of deterministic-ally 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 describe 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

Table I shows some raw records of taxi pickups from the NYC taxi trip dataset [11]. By using GPS data encoding, we want to quickly convert 600 million raw GPS records into demand sequences in each area.

TABLE I
PIECES OF RAW TAXI PICKUP DATA

Pickup_datetime	Pickup_latitude	Pickup_longitude
2016-06-01 02:46:38	40.695178985595703	-73.930580139160156
2016-06-01 02:55:26	40.792552947998047	-73.946929931640625
2016-06-01 02:50:36	40.823955535888672	-73.944534301757813
2016-06-01 02:57:04	40.823871612548828	-73.95220947265625

We first need to divide the entire city into small areas. There are several ways for such division such as dividing based on zip code. However, the resulting areas by this type of division are too large. For instance, the area size of Brooklyn in New York City is $180km^2$, while there are 37 different zip codes in Brooklyn. This leads to an average of 4.86 km^2 for each zip code area. It is desired to predict taxi demand in small areas so that the drivers know exactly where to go. However, on the other side, learning to predict taxi demand in very small areas is difficult. So, we need to select an area size which is both easy to predict and sufficiently accurate for the drivers.

In this application, we use the Geohash library [16] which 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. An example of using Geohash library to encode different pairs of $(latitude, longitude)$ data:

$$g.encode(lat,\ long,\ precision = (1 - 12)) \tag{1}$$

$$\begin{cases} g.encode(40.69517898,\ -73.93058013,\ 6) \\ g.encode(40.69517898,\ -73.93058013,\ 7) \\ g.encode(40.6951789801,\ -73.9305801301,\ 7) \\ g.encode(40.6951789899,\ -73.9305801399,\ 7) \end{cases} \tag{2}$$

In Eq. 2, $(latitude, longitude)$ pairs encoded with precision 7 will be converted to $'dr5rt8m'$, and the one encoded with precision 6 will be identified by code $'dr5rt8'$. Table II shows the size of divided areas at different precision levels. In Geohash, each code represents a divided area. After encoding all the taxi trip GPS information, coordinates locate in the same area have the same geohash code - this can be seen from the last three pairs in Eq. 2. In addition, neighboring areas share the same code prefix.

In this way, we can quickly divide the entire city into areas with arbitrary size and have the taxi demand sequence in each area. In our experiment, we divide the entire city into around 6500 small areas with precision 7 and for each area, we count the number of taxi pickups at each time-step. We also test our model on precision 6 which divides the city into about 1000 areas. In this paper, we present precision 7 due to accuracy and closeness to the street block level.

TABLE II
GEOHASH PRECISION

Precision		Cell width		Cell height
1	$\leq$	5,000km	$\times$	5,000km
2	$\leq$	1,250km	$\times$	625km
3	$\leq$	156km	$\times$	156km
4	$\leq$	39.1km	$\times$	19.5km
5	$\leq$	4.89km	$\times$	4.89km
6	$\leq$	1.22km	$\times$	0.61km
7	$\leq$	153m	$\times$	153m
8	$\leq$	38.2m	$\times$	19.1m
9	$\leq$	4.77m	$\times$	4.77m
10	$\leq$	1.19m	$\times$	0.596m
11	$\leq$	149mm	$\times$	149mm
12	$\leq$	37.2mm	$\times$	18.6mm

B. Recurrent neural networks

The sequential taxi demand pattern leads us to the choice of a model that can handle time-series data. Recurrent neural networks (RNNs) are one of the high performing 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 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. Another way to think about RNNs is that they are neural networks with a "memory" which captures and stores relevant information seen as input. In theory, RNNs can store and later use this information in arbitrarily long sequences, but in practice they can look back only a limited number of time-steps. A typical RNN is given in Fig. 2.

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.

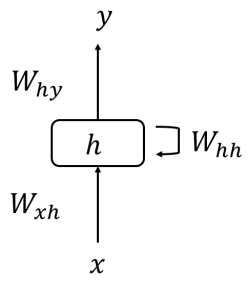

Fig. 2. The recurrent neural networks.

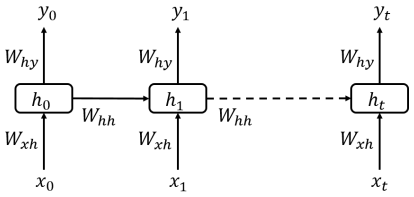

Fig. 3. The unrolled recurrent neural networks.

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.
- 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 memory mechanism. They were introduced by Hochreiter & Schmidhuber [9], and were refined and popularized by many people in the following years. LSTMs work tremendously well on a large variety of problems and are now widely used.

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 number, we use a stochastic model that can predict the entire probability distribution of taxi demand in different areas. We then use this probability distribution to decide the number of requests 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) [10] is to use the outputs of a neural network to parametrize 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 directly predicting the number of taxi requests in each area, the neural network outputs the parameters of the 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 encode the past taxi pickup information 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 $y_t = e_{t+1}$.

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 actual value from the dataset, to form the error signal. In our application, we predict y_t, the taxi demand in the next time-step. This prediction can be repeated in a loop to predict taxi demand for multiple time-steps. We use two LSTM layers in which each layer contains $1500 - 2000$ neurons based on the specific testing scenario.

As shown in Fig. 5, the output of LSTM layers would be mixture density parameters with the total number of

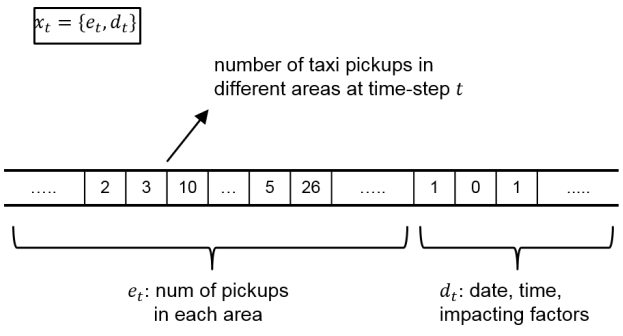

Fig. 4. The input data structure at one time-step.

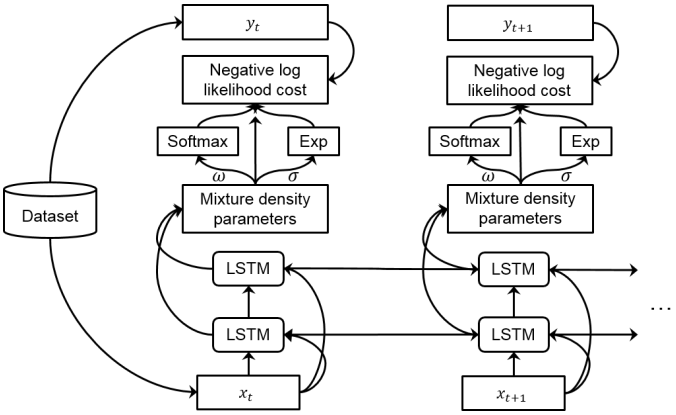

Fig. 5. The LSTM-MDN learning model unrolled through time-steps.

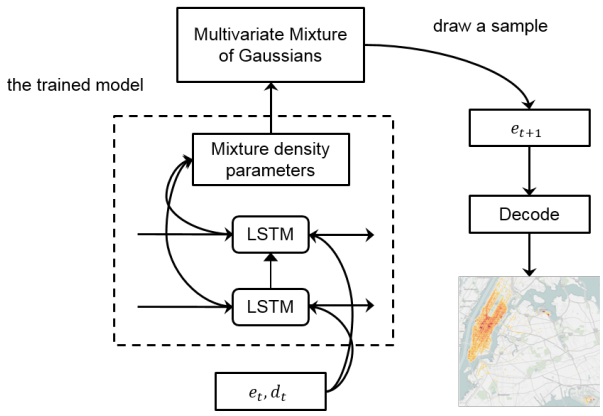

Fig. 6. The LSTM-MDN model performs prediction for time-step $t+1$.

$M \times (D+2)$ in which M is the number of Gaussian kernels, and D is the number of areas in the city. For each Gaussian kernel k, outputs from D number of neurons are combined for the calculation of the mean $\mu_k(x)$, one neuron for the variance $\sigma_k(x)$, 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 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:

$$p(y_t|x) = \sum_{k=1}^{M} w_k(x) g_k(y_t|x) \tag{3}$$

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 $x = \{x_1 \dots x_t\}$. The multivariate Gaussian kernel can be represented as:

$$g_k(y_t|x) = \frac{1}{(2\pi)^{D/2}\sigma_k(x)} \exp\left\{-\frac{\|y_t - \mu_k(x)\|^2}{2\sigma_k(x)^2}\right\} \tag{4}$$

where the vector $\mu_k(x)$ is the center of the 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 [17].

Finally, we can define the error function in terms of negative logarithm likelihood:

$$E_t = -ln\left\{\sum_{k=1}^{M} w_k(x) g_k(y_t|x)\right\} \tag{5}$$

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 parametrize a Gaussian mixture distribution. A sample can then be drawn based on this distribution and this sample would be the prediction of the next time-step taxi demand. Fig. 7 shows a density map of real and predicted taxi demands over the entire city.

V. Experimental Study

In this section, we evaluate our model on a dataset of taxi requests and see how well it can predict the requests in the future. In addition, we compare our model with two other baselines and show that it outperforms both.

A. Experimental setup

We evaluate the performance of the proposed network model with the New York City taxi trip dataset [11]. 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 contains a pickup time and the location information. In this application, we use its most recent 3.5 years data: from January 2013 through June 2016, which includes 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 [18] framework that is built on top of Theano [19]. We stop the training when the validation error does not change for 20 epochs.

Theoretically, LSTM can accept arbitrary sequence length. 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 $60mins$, the sequence length would be 24×7. If the time-step length is $20mins$, the sequence length would be $24 \times 3 \times 7$. For the $60mins$ 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, date, day of the week and time information. Table III includes the list of parameters in the experiments.

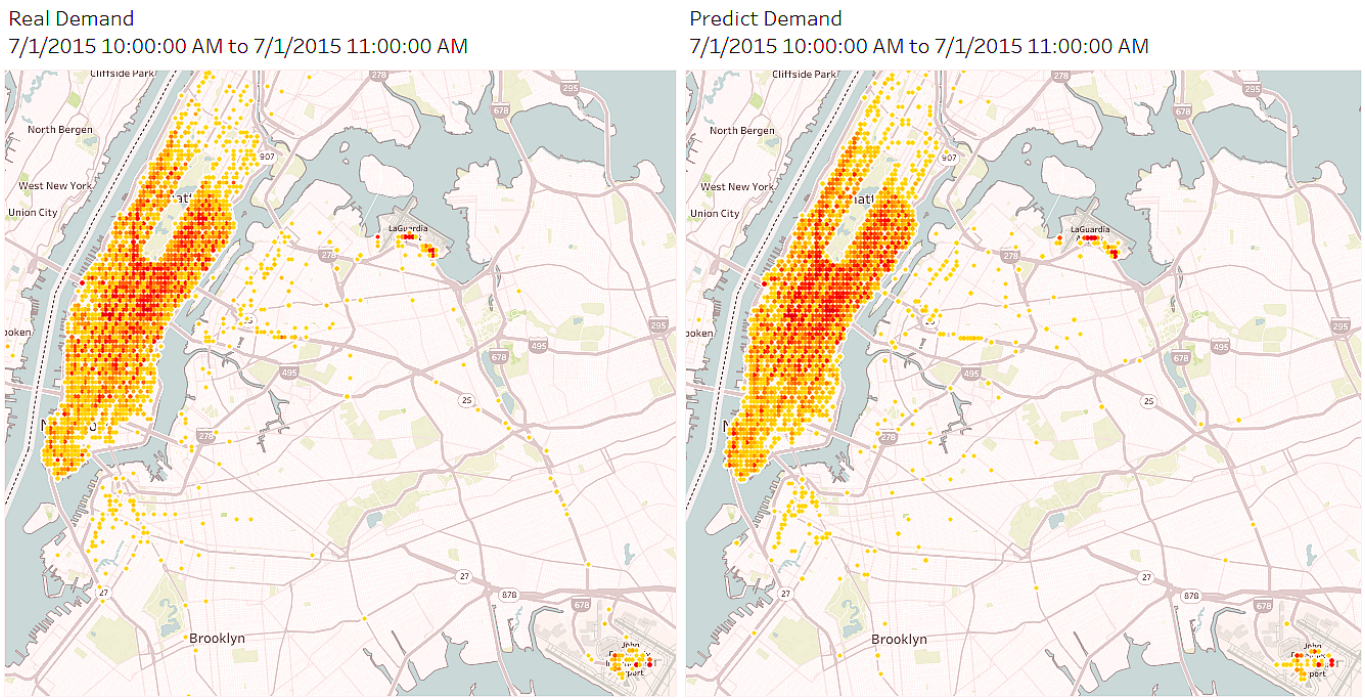

Fig. 7. The density map of real demand and the predicted demand. Red areas show high demand for taxis, yellow areas show lower demand, and there is no demand in other areas. The figure illustrates that the difference between the prediction and the real demand is very small.

TABLE III
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 (K)	6424
Number of features	6494
Number of hidden layers	2
Number of nodes in each hidden layer	1500-2000
Number of mixture Gaussian kernels	10

B. Performance metrics and baselines

To systematically examine the performance of our prediction approach, we include results with two error metrics: Symmetric Mean Absolute Percentage Error (sMAPE) and Root Mean Square Error (RMSE). $Y_{k,t}$ is the real taxi demand in area k at time-step t, while $\widehat{Y}_{k,t}$ is the predicted taxi demand. The sMAPE and RMSE in area k over time $[1, T]$ would be:

$$sMAPE_k = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_{k,t} - \widehat{Y}_{k,t}|}{Y_{k,t} + \widehat{Y}_{k,t} + c} \tag{6}$$

$$RMSE_k = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(Y_{k,t} - \widehat{Y}_{k,t}\right)^2} \tag{7}$$

The constant c in Eq. 6 is a small number ($c = 1$ in this application) to avoid division by zero when both $Y_{k,t}$ and $\widehat{Y}_{k,t}$ 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:

$$sMAPE_t = \frac{1}{K}\sum_{k=1}^{K}\frac{|Y_{k,t} - \widehat{Y}_{k,t}|}{Y_{k,t} + \widehat{Y}_{k,t} + c} \tag{8}$$

$$RMSE_t = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(Y_{k,t} - \widehat{Y}_{k,t}\right)^2} \tag{9}$$

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

To evaluate the performance of the proposed LSTM-MDN predictor, we compare its 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

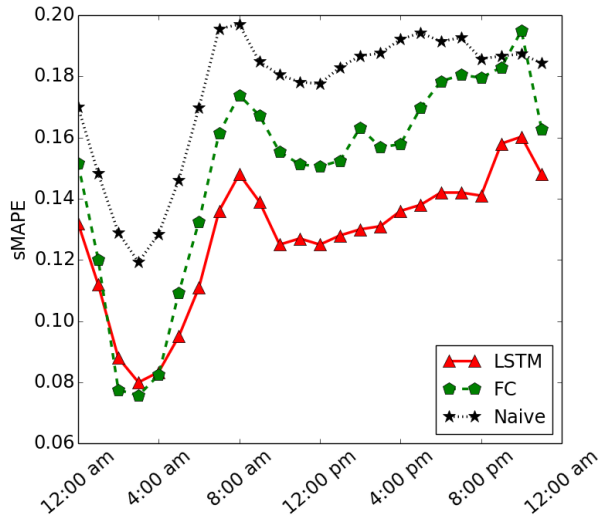

Fig. 8. Prediction performance of different approaches according to sMAPE.

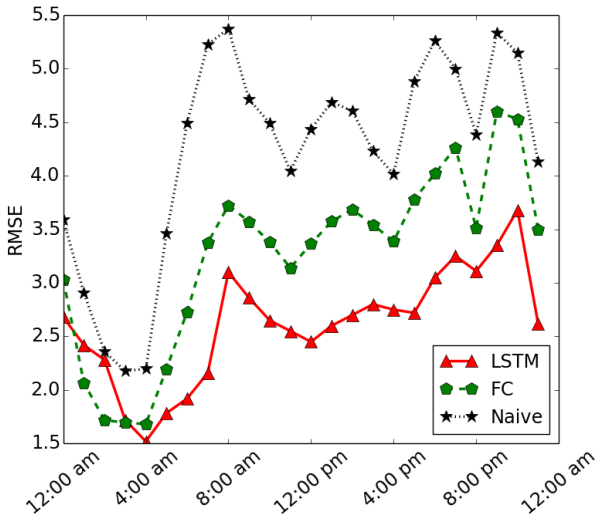

Fig. 9. Prediction performance of different approaches according to RMSE.

demand would be the average of demands at 10:00 am in the past 5 Mondays.

C. Performance results

First, we report sMAPE and RMSE errors over the entire city (all prediction areas). Second, we report the errors of some specific areas as time passes. For the three different predictors based on LSTM-MDN, fully connected feed-forward neural networks and naive statistic average, we respectively use *LSTM, FC* and *Naive* for short.

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. 8 and Eq. 9.

We report sMAPE and RMSE over the entire city in Fig. 8 and Fig. 9. 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. 8, sMAPE shows the mean percentage error, which gives us a way to calculate the prediction accuracy and observe that it is more than 80%. In Fig. 9, RMSE shows the mean deviation of the predicted demands from the real demands. In the real demands, we have $min = 0$, $max = 535$ and standard deviation $\sigma = 12.0$. The time-step length is 60 $mins$.

Fig. 10 reports the error bar of mean RMSE with the standard deviation over the entire city. We show this RMSE with different time-step lengths in the LSTM predictor. The time length of data we are using here is one week. Basically, smaller time-step length means smaller number of pickups in each step, which does affect the final RMSE. To avoid this, we sum the number of pickups every 60 $mins$. As we can see in Fig. 10, the model has the minimum RMSE at time-step

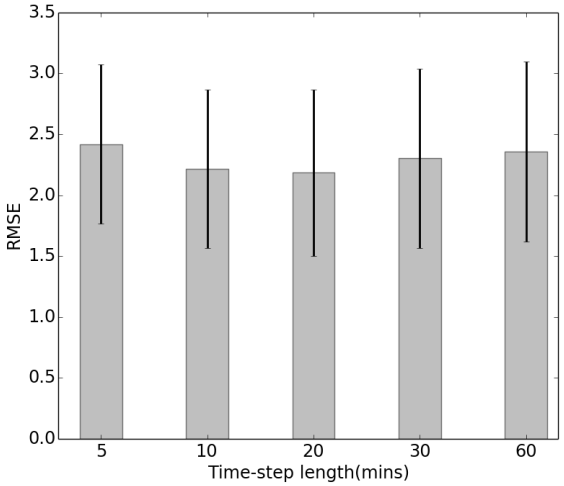

Fig. 10. RMSE of different time-step lengths. With the real number of pickups, $min = 0$, $max = 535$ and standard deviation $\sigma = 12.0$.

length either 10 or 20 $mins$. Overall, the RMSEs are very close under different time-step lengths.

2) Performance at specific areas: We compare the prediction performances between LSTM, FC and the Naive predictors in specific areas. First of all, we select two areas whose real demands in a week are shown in Fig. 11-a-1 and Fig. 11-b-1. The first one is a working area, with regular patterns on both weekdays and weekends. The other area is one of the most popular areas in NYC, in terms of taxi requests. The time-step length is 60 $mins$ in both areas. The bottom part of Fig. 11 shows the sMAPE performances in both areas. Each error bar includes the standard deviation on that day.

In Fig. 11-a-2, LSTM outperforms FC and Naive models in the regular working area because it is good at learning sequential information, even though the sequence length is as long as a week. FC sometimes results in larger errors than the

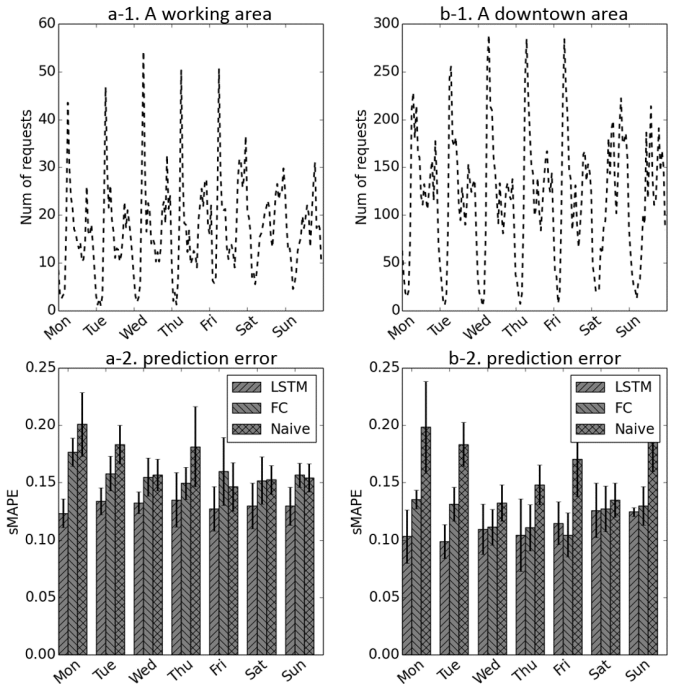

Fig. 11. Comparison in areas with different demand patterns.

Naive predictor due to the irregularities in sequence patterns. In Fig. 11-b-2, even though the request numbers are large in the downtown area, both LSTM and FC predictors show stable prediction performances.

Overall, the experimental results shows that LSTM outperforms 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 it and use this information to predict that after a couple of hours there would be almost the same number of requests in the concert area from the participants to return home. 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. This limitation causes more error in its prediction. 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.

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 sequential patterns from historical taxi requests, the proposed model can make taxi demand predictions for the future. We train our model on a dataset consists of 3.5 years taxi trips in New York City. Experimental results show that our predictor outperforms the prediction models based on fully connected feed-forward neural networks and naive statistic average. In addition, our model can continuously make real-time prediction of taxi demand for the entire city.

This work can be extended by adding more information to the inputs 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] F. Al-Turjman, M. Karakoc, and M. Gunay, "Path planning for mobile dcs in future cities," *Annals of Telecommunications*, vol. 72, no. 3-4, pp. 119–129, 2017.

[7] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[8] 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.

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

[10] C. M. Bishop, *Mixture density networks*. Aston University, 1994.

[11] 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

[12] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni, "Learning real manipulation tasks from virtual demonstrations using lstm," *arXiv preprint arXiv:1603.03833*, 2016.

[13] A. Karpathy, J. Johnson, and F.-F. Li, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[14] 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.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] G. Niemeyer. (2008) Tips & tricks about geohash. [Online]. Available: http://geohash.org/site/tips.html

[17] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. New York: Marcel Dekker, 1988, vol. 84.

[18] 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.

[19] The Theano Development, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.