A Taxi Dispatch System based on Prediction of Demand and Destination

Jun Xu, Rouhollah Rahmatizadeh, Ladislau Bölöni and Damla Turgut

***Abstract*—In this paper we describe an intelligent taxi dispatch system that has the goal of reducing the waiting time of the passengers and the idle driving distance of the taxis. The system relies on two separate models that predict the probability distributions of the taxi demand and destinations respectively. The models are learned from historical data and use a combination of long short term memory cells and mixture density networks. Using these predictors, taxi dispatch is formulated as a mixed integer programming problem. We validate the performance of the predictors and the overall system on a real world dataset of taxi trips in New York City.**

***Index Terms*—Taxi dispatch; demand prediction; destination prediction; distribution learning; mixture density network.**

I. Introduction

The emergence of ride-sharing systems had disrupted the public transportation model in many areas in the world, improved the travel experience and lowered the cost to the users [1]. However, the growing number of ride-sharing vehicles on the roads also increased fuel consumption, air pollution and traffic congestion [2].

Understanding and predicting passenger demand patterns promises to make taxi services and ridesharing systems more efficient. Studies such as [3–7] have shown that historical taxi trip data can provide rich insights about the temporal and spatial variation of the taxi demand, i.e. the points at which the taxi trips are *originated*. In addition, the prediction of the *destination* of the trips also plays an important role in a transportation system, because it helps to predict the spatial distribution of the vehicle fleet at some time in the future [8]. These problems will still remain if human drivers are replaced with self-driving cars because the passenger demand is not determined by the (possibly automated) drivers but by the human travellers.

Predicting traffic demand in space and time is a complex problem, where naïve approaches that predict expected demand as a single value have difficulty capturing the data patterns. Demand data is often multi-modal: it has multiple peaks during the day, varies with the days of the week and with special calendar and weather events. Predicting a demand that minimizes the mean squared error (MSE) is often uninformative in the case of a multi-modal distribution [9], [10], because the predicted value might fall between the modalities.

A solution to this problem is that instead of predicting a single real number, we estimate the (possibly multi-modal) probability distribution of the value. We propose to use a mixture of Gaussians to estimate this probability distribution and a mixture density network (MDN) [11] to find its parameters. Systems that require a single-point prediction can then sample from these distributions. The MSE-based approach can be regarded as the special case of using a single Gaussian kernel in the MDN and retaining only the mean of that kernel as the predictor.

Most existing studies focus on providing traffic demand and destination-pattern predictions. The vehicle dispatch system is also discussed in some of the previous works. However, there are not many successful examples of building an end-to-end intelligent vehicle dispatch system.

For the taxi demand prediction, we extend the approach [3], [10] that uses a long short term memory (LSTM) [12] recurrent neural network that takes as input features such as the date, time of day, day of the week and weather information. The output MDN predicts the distribution of the predicted load as a real valued numerical information.

We treat the destination prediction model as a one-input / multiple possible outputs problem and use a feed-forward neural network followed by an MDN. The output is a probability distribution over geographical information (the collection of Geohash cells of the city).

Using these predictors we describe a taxi dispatch system that balances the supply-demand ratio throughout the city. The system optimizes the taxi assignment and reallocation by solving a mixed integer programming problem with the goal of minimizing the average waiting time of the passengers and the idle driving distances of the taxis. The system takes into account the variable speed of the taxis which depend on the time of the day and differ on weekdays and weekends.

In our experiments, both the demand and destination prediction models were trained on the New York City taxi trip dataset [13] for the entire 2015 year. We evaluated the performance of the prediction models and the dispatch system with taxi data from February 2016 from the same dataset and found that they outperform both baselines using the historical means as a predictor and ablated versions of the proposed architecture.

The remainder of this paper is organized as follows. Section II surveys related work on taxi demand and destination prediction as well as intelligent transportation systems. Section III describes the proposed taxi demand and destination distribution learning models. Section IV presents the proposed taxi dispatch system. Section V describe experimental studies measuring the performance of the demand and destination prediction models and the dispatch system. Section VI concludes the paper.

II. Related Work

A. Taxi demand and destination prediction

In recent years, taxi demand prediction became the focus of several research efforts as one of the key components in improving the taxi dispatch performance as well increasing the sustainability of taxi companies. Moreira-Matias et al. [7] propose a prediction framework where the prediction is a weighted ensemble of outputs from three different models. The ensemble weights are updated based on the previous prediction performance of each model. The framework was shown to make real-time demand predictions for the 63 taxi stands in the city of Porto, Portugal. Zhao et al. [5] introduce the concept of maximum predictability based on the entropy of historical taxi demand. They prove that taxi demand is highly predictable and then propose three prediction algorithms to validate their maximum predictability theory. Miao et al. [14] propose a dispatch framework for balancing taxi demand and supply throughout a city. In their work, the future taxi demand is predicted by the mean value of repeated samples from historical demand.

Recent taxi demand prediction models take advantage of advances in deep neural networks. Yao et al. [15] use convolutional neural networks, a component popular in the image processing and computer vision community, for taxi demand prediction. Another direction is to use a recurrent neural network (RNN) such as Long short-term memory (LSTM). Given the sequential patterns in city taxi demand, the RNN showed an excellent performance in capturing the traffic patterns in our earlier work [10].

Taxi destination prediction is more complex than demand prediction because it needs to predict a distribution of a geographical location instead of a single number. One possible approach is to use the beginning of taxi trip's trajectories to predict its final destination. Brébisson et al. [16] investigate several neural network based models for predicting the final destination. They found the best performing model to be a multi-layer perceptron which, instead of directly predicting the destination position, predicts the final destination as a weighted sum of all destination cluster centers. The destination clusters are pre-generated while the weights for sum are learned by the neural network. Besse et al. [8] predict the destinations as a probability distribution. They first generate clusters of trajectories from historical data, and build Gaussian mixture models for each set of points in the trajectory cluster. When making a prediction, the approach first identifies the cluster with the highest similarity score and predicts the final destination as the mean of the destinations of the trajectories in the cluster.

Another class of scenarios cover cases when we do not know a prefix of the taxi trajectories. Alonso-Mora et al. [17] propose a dispatch system in which the destination estimation is sampled from a normalized distribution of past taxi trips destinations.

In this paper, we build an improved LSTM based taxi demand predictor. The motivation is to leverage the powerful sequential pattern learning ability of the RNN technique. For taxi destination predictions, we agree with Besse et al. [8] and Alfonso-Mora et al. [17] on the direction to formulate a distribution-based approach. However, different from them, our approach uses a mixture of 2D Gaussians to capture the likelihood of the destination. The motivation behind our approach is two-fold. We first would like to capture the distribution pattern of the passenger's destination. Secondly, we would like to consider passenger's stochastic behaviors on destinations. Thus, we propose a structure that utilizes a mixture density network (MDN) [11] on top of a neural network. The idea of MDN is to use the outputs of a neural network to parametrize a mixture distribution. It has been proved very useful in modeling patterns with stochastic behaviors [9], [18].

B. Modeling efficient taxi dispatching

Given the estimated future demand and destination, taxi dispatching can be formulated as an optimization problem. Zhang et al. [6] propose a real-time taxi dispatch application with the goal to balance the passenger demand and taxicab supply. Miao et al. [19] propose a dispatch model with the goal to reduce the average demand-supply ratio mismatch and the average total idle distance.

A more complex optimization problem is posed by systems where several customers share the vehicle over some portion of their trajectory. The word "ridesharing" is used to describe this technique, but it needs to be distinguished from the same general term applied to transportation network companies such as Uber, Lyft or DiDi[1]. Ridesharing improves the utilization of the vehicles, but in the general case lowers the utility for the customer whose trip can become longer and comfort lower. Some of the penalty for the customer can be reduced during dispatching by finding optimal customer pairings. Lin et al. [20] present a dispatch system for transportation hubs with steady passenger streams that takes into account virtual demand pools, passenger's walking time and the ridesharing mechanism. A match-making system is used for pairing trips and scheduling taxis. Chen et al. [2] propose a system for vehicle dispatch and ridesharing with the goal of balancing the supply-demand ratio while minimizing the idle mileage. Ridesharing is achieved by treating the taxis schedule as a mixed integer programming problem. Alonso-Mora et al. [17] propose a ridesharing framework that optimizes dynamic vehicle routing and request assignment. Future taxi demand is estimated by sampling from a probability distribution of historical taxi data. Similar studies based on ridesharing systems were conducted in [21], [22].

Our work focuses on learning and predicting taxi demand and destination distribution patterns. In contrast to existing work, we use a recurrent neural network to capture long term dependencies in the sequence of taxi demand patterns. For the destination prediction, our model predicts the entire probability distribution over all areas in the city instead of sampling from the destination frequencies of previously seen trips. Our approach gives a more realistic prediction as it takes into account the uncertainty while predicting. With the predicted results, we build a dispatch system intending to balance future

[1]In this sense of the word, UberPool is a ridesharing service, while UberX is not.

taxi supply and demand over the city while minimizing the passengers' average waiting time and the taxis' idle driving distances.

The starting point of this paper is our previous work [3], [10]. The novel contributions of this work are the introduction of the mixture density network based LSTM-MDN method that takes into account both past data dependency and data distributions, the use of a mathematical model that estimates realistic taxi travel time between the areas, and the consideration of the dispatch system performance trade-off.

III. Taxi Demand and Destination Prediction

A. Data representation

We train our system using a publicly available database of New York City taxi trips [13]. The training data includes approximately 170 million trips between 1/1/2015 and 12/31/2015. The dataset specifies for each pick-up and drop-off event a high resolution GPS location and a timestamp. Such a high spatial resolution is not useful for a dispatch system, where we are interested only in classifying whether a taxi is close enough to a location for a pickup without significant delay. To capture the desired level of accuracy, we divide the city into a grid using the Geohash library [23]. This technique encodes a $[latitude, longitude]$ pair into a geohash string where neighboring areas share the same string prefix. For instance, two neighboring areas in our map are geohash encoded as follows:

$$\begin{cases} g.encode(40.697,\ -73.931,\ 6) = dr5rt8 \\ g.encode(40.699,\ -73.931,\ 6) = dr5rt9 \end{cases} \quad (1)$$

The advantage of this encoding method is that in a sorted list, strings of neighboring areas will stay close together. In this work, we use a 6-character geohash, dividing the city into approximately 1000 grid cells of size 1200m $\times$ 600m.

B. Taxi demand prediction

To predict the taxi demand at an arbitrary time in the future, we start by discretizing the time of the day into a number of time slots $\{0, 1, .., t, ...\}$ where t is the t^{th} time-slot of a day. The length of the slot is a hyper-parameter. Next, we organize the taxi requests for each grid cell into a data sequence grouped by time slots. Preliminary data analysis shows that for each area of the city the historical taxi demand shows recognizable patterns influenced by the day of the week, time of the day and weather. Motivated by these observations, we design a sequence learning model that learns to predict future demand from the historical data. In addition to the past demand, the model will also take as input signals such as the date, day of the week, time-step in the day and weather [24].

Fig. 1 shows the input and output data structures. For time slot t, the input data x_t is a tuple of vectors $[f_t, e_t]$. f_t contains features such as the date, day of the week, time-step in the day and weather while e_t is a vector representing the number of pickups in each grid cell. Our scenario considers the map of New York, with its relatively regular grid-like streets. Thus, for this scenario we don't need to consider aspects such as reachability. This might need to be considered in other scenarios that are beyond the scope of this paper.

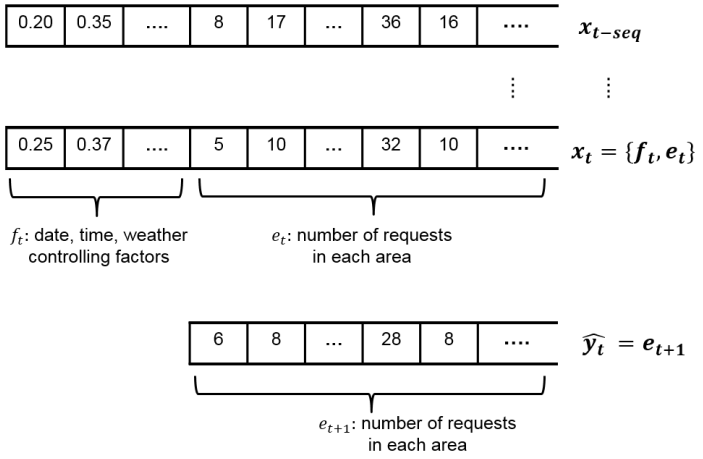


Fig. 1. The input and output data structures for the demand prediction model.

1) LSTM model: In a sequence learning model the next output depends not only on the current input, but also on the previous ones. To capture this relationship, the predictor needs to have a memory. In recent years, a popular model for such tasks is Long Short Term Memory, a type of recurrent neural network. LSTM has been widely used in many applications such as unsegmented handwriting generation [9], natural language processing [25] and robot control [26].

As shown in Fig. 2, the input data to the model at time-slot t is $[x_{t-seq}, ...x_{t-1}, x_t]$, where *seq* is the number of previous data points used to predict the data in the next time slot. *seq* is a hyper-parameter that is set large enough to enable the network to learn long-term dependencies. Given the input data at t, the network predicts the output $\hat{y}_t$, the number of requests in each area at the next time slot. A simple way to train the network using stochastic gradient descent is to minimize the mean squared error (MSE) between the predicted $\hat{y}_t$ and the ground-truth demand y_t.

2) LSTM-MDN model: Predicting the traffic demand is inevitably probabilistic. If the probability distribution is unimodal, a variation around a value, optimizing the MSE leads the network to predict the most likely value. However, if the

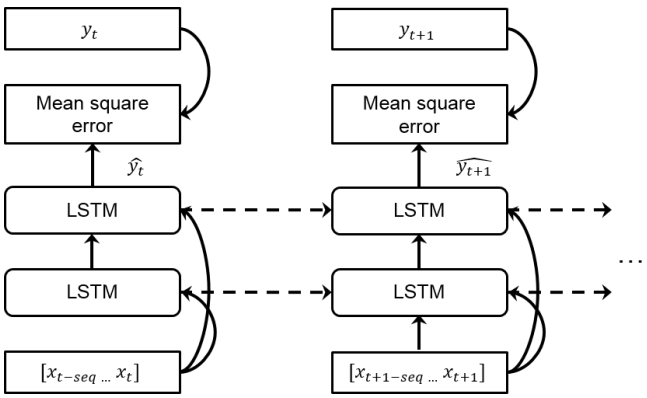


Fig. 2. LSTM-based demand sequence learning model.

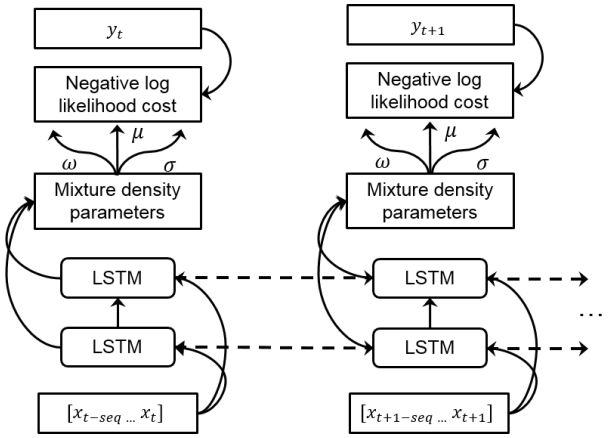


Fig. 3. LSTM-MDN based demand sequence learning model.

probability distribution is multi-modal, for instance, by having two independent peaks, optimizing the MSE would predict a value between those two peaks, which might be a value which is not likely to be the correct one. Preliminary data analysis on taxi data shows that indeed, taxi demand is multi-modal.

To account for this, we propose to use mixture density networks (MDNs) [11] to model stochastic behaviors. MDNs had been used to model multi-modal distributions in domains as varied as speed synthesis [27] and drawing sketches [28]. The outputs of an MDN parameterize a Gaussian mixture distribution. Using a sufficient number of Gaussian kernels, this system can approximate an arbitrary distribution. The MSE minimization approach can be regarded as the special case of the mixture Gaussian model with only one Gaussian kernel.

Our approach is to use the mixture of Gaussians controlled by the MDN to approximate the demand distribution in each grid cell. Fig. 3 shows the structure of the LSTM-MDN learning model. In contrast to Fig. 2, the output of LSTM layers would be mixture density parameters of size $M \times (N + 2)$ where M is the number of Gaussian kernels, and N is the number of grid cells. For each Gaussian kernel, we have N neurons for the means $\mu_k(x)$, one neuron for the variance $\sigma_k(x)$, and another neuron for the mixing coefficient $w_k(x)$. The vector w is normalized to satisfy the constraint $\sum_{k=1}^{M} w_k(x) = 1$. 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 slot $x = \{x_1 \dots x_t\}$. 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. Finally, we can define the error in terms of negative log-likelihood:

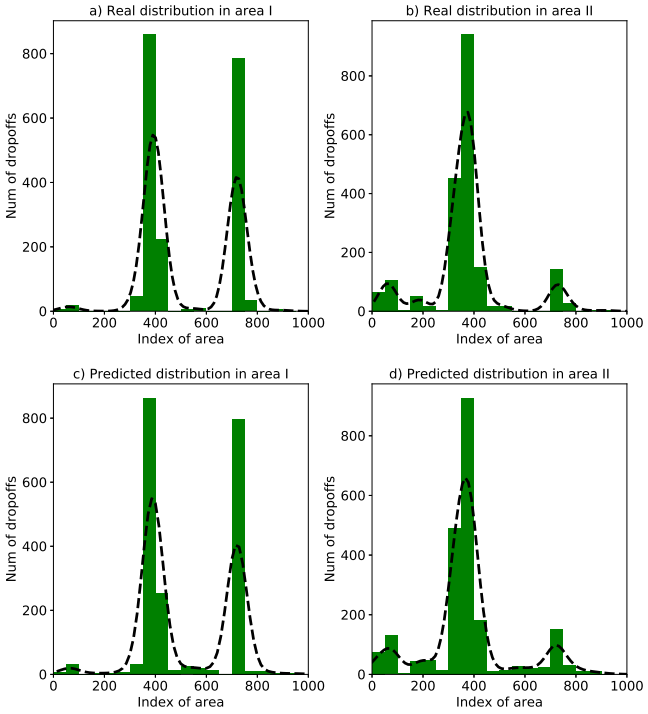


Fig. 4. Histograms of destination cells for taxi rides starting in two specific grid cells. Top: real values, bottom: predicted values.

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

To summarize, the LSTM-MDN model outputs the parameters of a Gaussian mixture distribution. To make a prediction of the taxi-demand in the next time-slot we can draw a sample $\widehat{e_{t+1}}$ from this distribution. This prediction can be repeated in a loop to predict taxi demand for multiple time-steps.

C. Taxi destination prediction

As we discussed in the introduction, we aim to predict not only the demand for taxis in a specific area in the future, but also the destination of those trips. As these trips will take place entirely in the future, we cannot rely on partial GPS traces of the trips to predict the destination. Our goal is to predict the probability distribution of likely destinations for a trip started in a specific grid cell.

To investigate the nature of destination distributions that appear in practice, let us consider the histogram of the destination cells from two specific starting cells (Fig. 4-top). For better visualization, in this figure we cluster neighboring cells sorted by geohash into different bins. The horizontal axis represents the area index while the vertical axis represents the number of dropoffs in the area. We notice that the distribution is far from uniform - certain locations are heavily favored as a destination. Furthermore, the distribution is multi-modal, with multiple independent peaks. The overall shape of the distribution makes it suitable to modeling using a mixture of Gaussians.

To build a distribution learning model, we first extract information for each trip from the historical taxi dataset, which

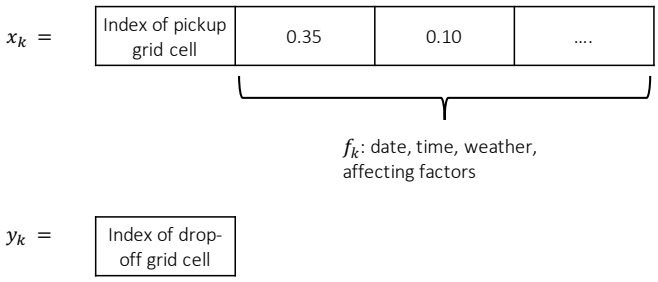


Fig. 5. The input and output data structure for destination prediction.

contains the time-stamp, pickup location, and dropoff location and encode it into the data structure shown in Fig. 5. One of the advantages of using geohashes is that neighboring grid cells will stay close when we sort them by the encoding string. Each trip is converted into a pair of data points $[x_k, y_k]$, where k represents the trip index in the dataset. x_k consists of the pickup area and the corresponding features such as the time slot in the day, the day of the week and the weather. y_k represents the destination area of the trip.

Fig. 6 shows the distribution learning model. The goal is to learn the parameters of a mixture of Gaussians for each area. The input data x_k is fed to a fully connected neural network. The expected output is a vector of distribution parameters with length $3 \times M$. M is the number of Gaussian kernels, which is a hyper-parameter. Each Gaussian kernel consist of 3 variables $[\omega, \mu, \sigma]$ where ω is the mixing coefficient, μ is the mean and σ is the standard deviation.

A suitable loss function is to minimize the log-likelihood of the distribution of the training data:

$$Cost = -\ln\left(\sum_{m=1}^{M} w_m(x)\phi_m\{y, \mu_m(x), \sigma_m(x)\}\right) \quad (5)$$

where $\phi_m\{y, \mu_m(x), \sigma_m(x)\}$ is the m^{th} Gaussian kernel. It can be represented as:

$$\phi_m\{y, \mu, \sigma\} = \frac{1}{2\pi\sigma_m(x)} \exp\left(-\frac{|y - \mu_m(x)|^2}{2\sigma_m^2(x)}\right) \quad (6)$$

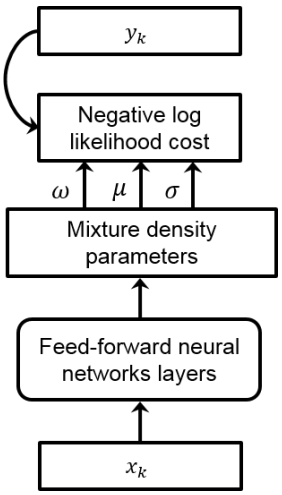


Fig. 6. Destination distribution learning model.

For each pair $[x_k, y_k]$ in the training dataset, we calculate the cost based on the predicted distribution versus the actual value, and minimize the sum of the costs. The resulting network, as illustrated in Fig. 4-bottom provides a close prediction of the real distribution.

IV. Dispatch System

In this section, we describe a taxi dispatch system that uses the predictive models trained in the previous section. In the scenario we are considering, customers send real time taxi requests to a centralized server, specifying the start and destination locations. The overall goal of the system is to balance the supply and demand, minimize the idle driving of the taxi and minimize the time customers are waiting for the taxi after making a request.

A. Estimating the travel time

In order to make efficient dispatch decisions, we need to know the time it takes to travel between two destinations (at the resolution of the grid). We introduce the driving distance matrix $Dist$ with $Dist[i][j]$ being the distance traveled by a taxi when going from grid cell i to grid cell j. For pairs for which we have historical data, we estimate this value as the average distance on these trips. However, as the number of values in $Dist$ scales quadratically with the number of cells, we found that not all cell pairs had historical trips between them. For such cell pairs, the distance can be estimated by considering the shortest trip composed of segments for which we have historical information. Algorithm 1 shows an iterative approach for generating the full distance matrix.

Another aspect that must be considered when dispatching is that the average speed of the taxis is influenced by the traffic, and thus varies with the time of the day and the day of the week. Following [17], [29] we estimate the speed of the taxi for each time period with the historical average for the given time separately for weekdays and weekend days (see Fig. 7).

Having the distance matrix and the estimated speed allows us to estimate the speed for any taxi trip.

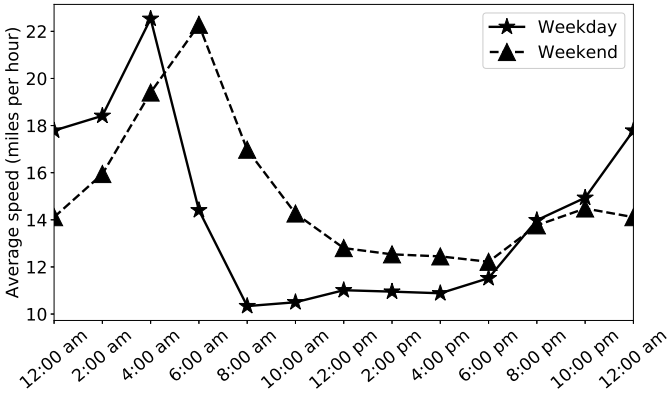


Fig. 7. The hourly variation of the average speed of the taxi for weekdays and weekend.

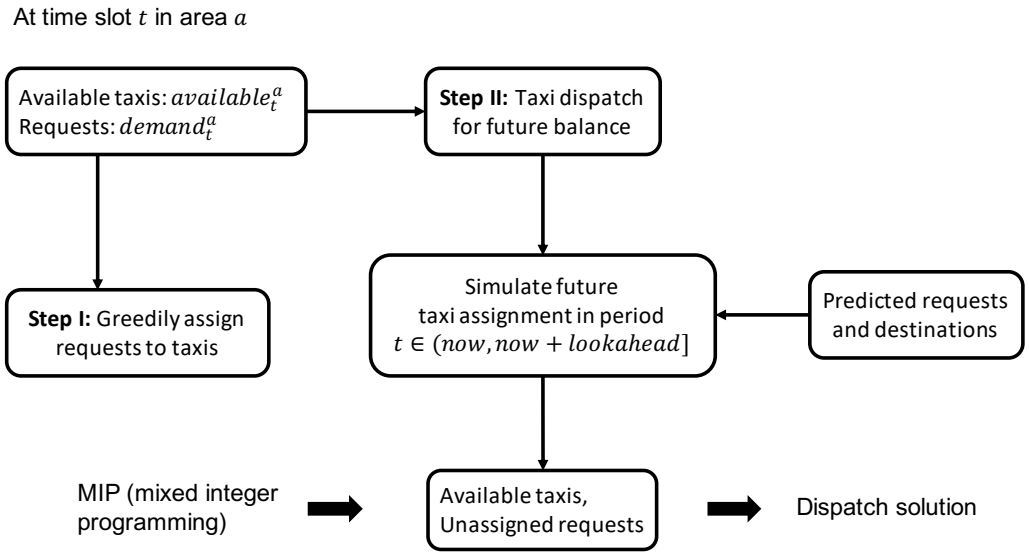


Fig. 8. The operation flow of calculating the future dispatch.

Algorithm 1: Distance matrix generation

```
all_cells, list of all cells
N = len(all_cells)
all_trips, grouped by trip start area
Distance matrix Dist[N][N]
/* initialize distance for cells pairs with records */
for (a_i, a_j) in all_cells do
    trips = get_trips(a_i, a_j, all_trips)
    Dist[i][j] = mean(trips.distance)
end
/* cell pairs without trip records */
while not converged do
    for (a_i, a_j) in all_cells do
        candidatepaths =
            bidirectionsearch(a_i, a_j, Dist)
        Dist[i][j] = min(candidatepaths)
    end
end
return Dist
```

B. Dispatch system

For the following discussion, we are going to use 1 minute as the length of our time slots. For any grid cell $a, a \in all\_cells$, we represent its taxi demand at time slot t as $demand_t^a$:

$$demand_t^a = new\_demand_t^a + unassigned_t^a \tag{7}$$

where $new\_demand_t^a$ represents the newly received taxi requests in area a at time step t, while $unassigned_t^a$ represents the unassigned taxi requests from previous time steps.

Similarly, the available number of taxis is represented as $available_t^a$ which consists of two parts: $idle_t^a$ represents the number of original idle taxis in area a at time step t while $arrival_t^a$ represents the number of estimated arriving non-occupied taxis to area a at time step t:

$$available_t^a = idle_t^a + arrival_t^a \tag{8}$$

The workflow of the dispatch system is shown in Fig. 8. At each time step t, the dispatch system executes two steps. In step I, for each grid cell a, we sort the requests $demand_t^a$ by receiving time then perform a greedy assignment with the taxis available in that cell $available_t^a$. As there is no guarantee that $available_t^a \geq demand_t^a$, it is possible that not all requests can be served in the given time slot. Step II has the goal of balancing the future supply and demand ratio. With the prediction models described in Section III, we first sample each area's future taxi requests and their destinations within a number of *lookahead* time steps. Then, we simulate the taxi assignment (step I) for each area in the next *lookahead* time steps. After this process, we obtain the remaining unassigned requests and available taxis in each area. A taxi dispatch process is conducted to balance the supply-demand curves in each area over the entire city. We optimize the dispatch strategy by solving a mixed integer programming problem whose objective is to minimize the total idle driving distances while serving all the incoming requests formulated as:

$$min \sum_{t=now}^{now+lookahead} \sum_{taxi=1}^{M} idle\_distance_{taxi}^t \tag{9}$$

subject to

$$\sum_{a=i}^{N} available^a + dispatch_t^a <= M \tag{10}$$

$$\sum_{t=now}^{now+lookahead} \sum_{a=i}^{N} available_t^a + dispatch_t^a >= demand_t^a \tag{11}$$

V. Experimental Study

A. Experimental setup

We used the New York City taxi trip dataset [13] to train and validate the performance of the proposed system. This dataset contains daily recorded taxi trips by more than 15000 taxis, including the yellow cabs, which operate mostly in Manhattan, and the green cabs, which operate mostly in the suburbs. The number of trips varies with the day; for instance, in one week in 2016, the recorded trips from Monday to Sunday were 374305, 395678, 408184, 432087, 453192, 480818 and 418237. We used the trips from the whole calendar year 2015 as training data, and validated the prediction models as well as the dispatch system with data from the first week of February 2016.

For the taxi demand prediction, we discretize the requests into time slots of 15 minutes for each area in 2015. The input data shape is $(365 \times 96, 96, 997 + 10)$ where 365×96 is the total number of time slots, 96 is the sequence length (one day or 24×4), 997 is the number of grid cells and 10 is the number of features. The features include the date, the time slot in the day, the day of the week and the weather.

For the destination prediction, the input is the trip information consisting of the starting grid cell and features such as the date, time step of the day, day of the week and weather type.

For the dispatch system, we initialize the taxi distribution based on the sum of historical requests in each cell. We used a time slot length of one minute for the prediction system. The experimental parameters are summarized in Table I.

Before proceeding with the experimental results, let us briefly discuss the computational cost of the proposed approach, which has a significant impact on the practical applicability. The computational cost most impactful for the practical application is the prediction, which is done as an inference on the neural network. This process takes less than 1 second, even on a computer without a GPU accelerator. The neural network does not need to be retrained unless a significant amount of new and different data has been accumulated. The training process takes approximately 12 hours on an NVidia GTX 1080 GPU, and it is done offline. Another component of the operational system is the mixed integer linear system used to find the dispatch solution. For the problem sizes we considered (with 3000 to 5000 taxis), this process took less than 10 seconds.

TABLE I
Experimental parameters

Grid cell size	$\leq 1.2km \times 0.61km$
Number of cells N	997
Dispatch system time slot	1 minutes
Number of layers for neural networks	2
Number of Gaussian kernels	1-24

B. Performance metrics

We measure the prediction performance of our approach using symmetric Mean Absolute Percentage Error (sMAPE), a widely used prediction error metric. sMAPE describes a percentile prediction error which, in our case, is defined as follows:

$$sMAPE_t = \frac{1}{N} \sum_{n=1}^{N} \frac{|Y_{n,t} - \widehat{Y}_{n,t}|}{Y_{n,t} + \widehat{Y}_{n,t} + c} \tag{12}$$

where t is the time slot in a day and N is the number of areas in the city. $Y_{n,t}$ represents the real taxi demand in grid cell a_n at time slot t while $\widehat{Y}_{n,t}$ is the predicted taxi demand. The constant c is a small number introduced to avoid division by zero when both $Y_{n,t}$ and $\widehat{Y}_{n,t}$ are 0 (we used $c = 1$) .

We define the classification accuracy for destination prediction as the fraction of correct predictions for all requests in a given time slot.

For the dispatch system, we show the performance in terms of the average time the passengers wait before being picked-up and the idle driving distance of the taxis, i.e. the distance they traverse without carrying a passenger.

C. Systems compared in the experimental study

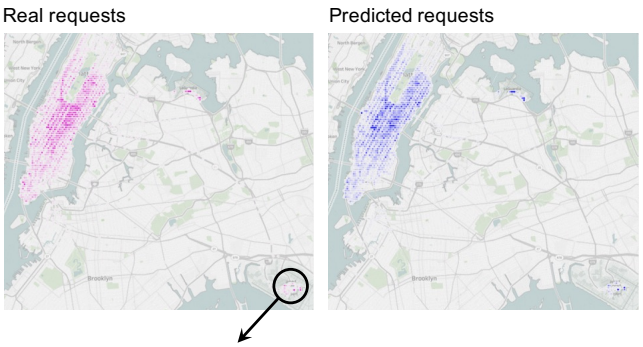
To investigate the performance of the proposed algorithms, we assembled several alternative systems by making choices for the demand prediction, destination prediction and the full dispatcher. Our strategy was to compare against both well chosen baselines that do not rely on deep learning systems, as well as ablated versions of our proposed architecture. We did not compare against clearly inferior baselines such as constant prediction or random allocation. The chosen models are listed below.

Demand predictors:

- *Sliding-Window Mean Demand (SWMD):* predict taxi demand as the mean of past demands in a sliding window over the same timeslot and day of the week. For example, predict the demand during $[10:00am, 10:00am + \Delta t]$ on Monday as the mean demand of the same time period on the previous 5 Mondays.
- *LSTM:* predict taxi demand using using an LSTM network and an MSE loss with the model described in Section III-B1.
- *LSTM-MDN:* predict the taxi demand using an LSTM network followed by an MDN loss as described in Section III-B2.

Destination prediction:

- *Sliding-window Destination Distribution (SWDD):* uses the frequency distribution in a sliding window to predict the distribution of destination. For example, the predicted destination distribution at Monday 10:00am from location n will be the frequency of all destinations from the same area, at the same time on the past five Mondays.
- *FN:* uses the traditional feed-forward neural network.
- *FN-MDN:* uses the feed-forward neural network followed by an MDN as described in Section III-C.
- *FN-MDN-neighbors:* modifies the FN-MDN model by accepting neighboring grid cells as correct prediction. This is justified by the fact that the grid cells are small (0.6 times 1.2 km) and thus a dispatcher can reasonably



Real taxi requests	52	56	47	36	32	....
Dispatch arrival/departure	+9	+18	+5	-4	-1	....
Requests to be fulfilled in system V	2	6	0	0	0	....
Requests to be fulfilled without dispatch	11	25	36	12	15	....

Fig. 9. The density map of real and predicted number of requests and an example of taxi dispatch flow from the JFK airport.

TABLE II
COMPARED DISPATCH SYSTEMS

Dispatch system	Demand prediction	Destination prediction
System I	SWMD	SWDD
System II	LSTM	FN
System III	LSTM	FN-MDN
System IV	LSTM-MDN	FN
System V	LSTM-MDN	FN-MDN

ignore the idle distance when traveling from the neighboring areas.

Dispatch systems: We compare the performance of five different dispatch systems based on various combinations of the demand and destination predictors as listed in Table II. The FN-MDN-neighbors approach is not included because from the point of view of the dispatch decisions is identical to FN-MDN. All dispatch systems use the dispatch strategy shown in Fig. 8.

Fig. 9 shows a density map of the real and predicted taxi requests. We find that there is a good match between the predicted and the real value. The figure also shows an example of the taxi dispatch flow from the John F. Kennedy airport, showing how the proposed system reduces the number of requests whose fulfilment is delayed.

D. Prediction Performance

Fig. 10-top compares the demand prediction performance of the different algorithms. We find that the sMAPE values vary with the time of the day, in general being the lowest during early morning hours, when the traffic is lower. For all days and all times of the day, the LSTM-MDN approach has the best prediction performance, followed by the LSTM and the SWMD approachess, with the sMAPE being almost twice for the SWMD approach compared with the LSTM-MDN. The difference between LSTM and LSTM-MDN are relatively minor when the traffic is low, such as at Monday 4:00am, but is much more pronounced at higher traffic hours, such as at 8:00am Monday.

Fig. 10-bottom shows the destination prediction accuracy (in these results, higher is better). We find that FN-MDN consistently outperforms SWDD, with the accuracy during daytime being about 75% versus 60%. The FN-MDN-neighbor graph is even higher, reaching an accuracy of around 85%. This shows that even when we cannot exactly predict the grid cell of the destination, we can still often predict the correct cell with an error of a single cell.

An interesting observation that can be made by comparing the top and bottom graphs in Fig. 10 is that during the early hours of the morning the demand prediction is consistently better, while the destination prediction is consistently worse. We hypothesize that the reason could be that in this time period the total number of requests is low and is mainly from areas such as airports and bars, but the corresponding destinations can be anywhere in the town and are not that predictable.

E. The impact of the Gaussian kernels

In the previous section, we have seen that the approaches using an MDN output outperformed alternative prediction models. In this section, we explore the impact of using different numbers of Gaussian kernels in the MDN. As we discussed in Section III, MDNs use the outputs of a neural network to parameterize a mixture distribution. Given a sufficient number of Gaussian kernels, theoretically we can model any data distributions. The MSE minimization approach can be regarded as similar to the special case of the mixture Gaussian model when $M = 1$ in Eq. 2 (but it is not exactly the same mathematical formula).

Fig. 11 shows the prediction performance for different number of Gaussian kernels in the demand prediction model over the course of the first week of February 2016. We find that the prediction performance improves up to $M = 8$, but then it starts to decrease as the number of kernels is increased further. The reason for this is that adding additional kernels, while increasing the expressiveness of the model, it also increases the number of parameters, which makes the model more prone to overfit on the training data and harder to train.

Fig. 12 shows the prediction performance for different number of Gaussian kernels in the destination prediction model over the course of the first week of February 2016. In contrast to the demand prediction model, we do not find any significant over-fitting; however, the performance appears to plateau at around 15 Gaussian kernels.

F. The Performance of the Dispatch System

As we discussed in the introduction of this section, we are comparing five different dispatch systems based on various combinations of the predictive models as listed in Table II. For each dispatch system, we show the performance in terms of the taxi passengers' average waiting time and taxi drivers' average idle driving distance.

The way the dispatcher can take into account the demand and destination predictions is by looking ahead. For instance,

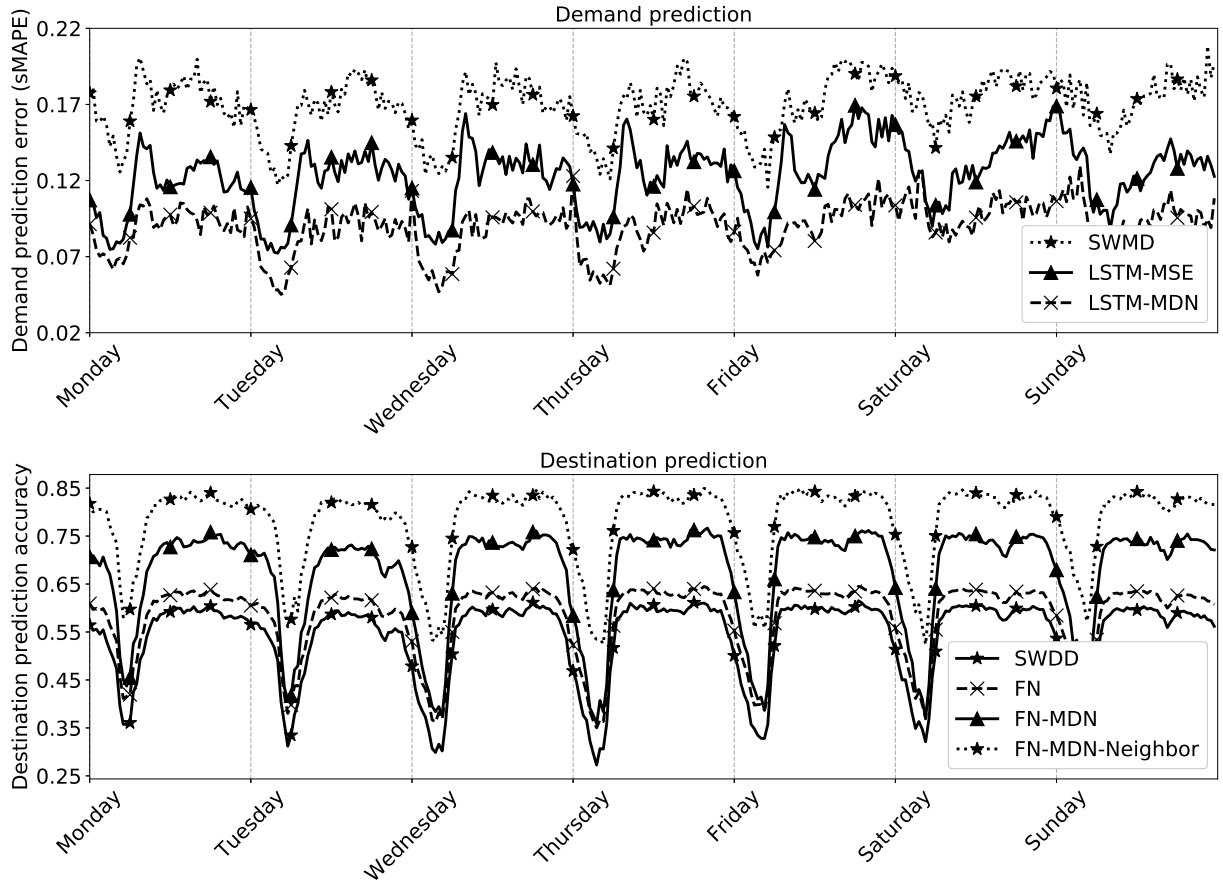


Fig. 10. Prediction performance in one week by different demand and destination prediction models.

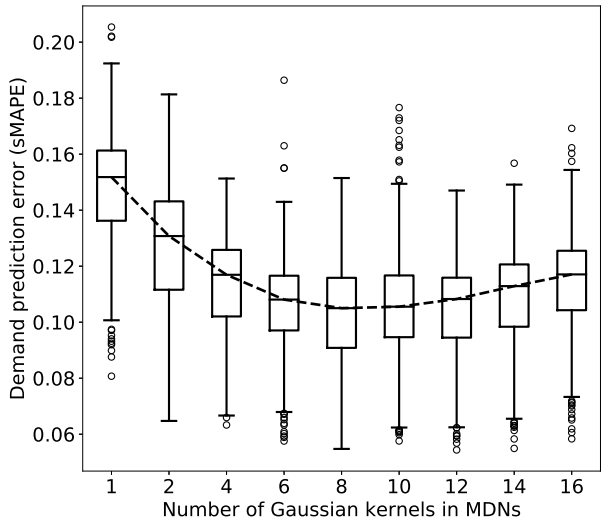


Fig. 11. The impact of the number of Gaussian kernels in the MDN for demand prediction using LSTM-MDN

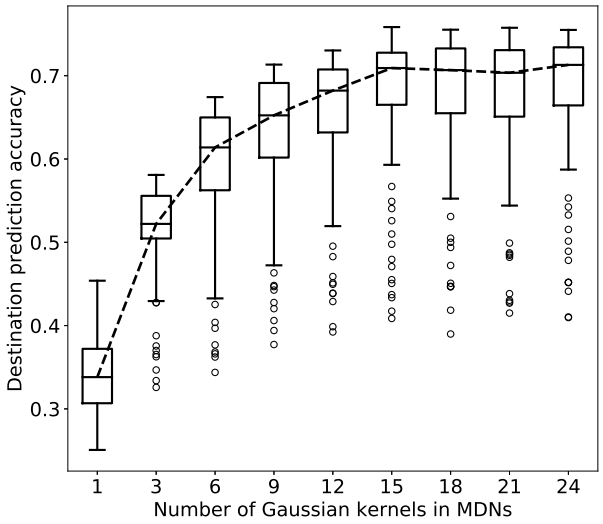


Fig. 12. The impact of the number of Gaussian kernels in MDNs for destination prediction using FN-MDN.

the dispatcher might preemptively send taxis to grid cells where high demand is expected. From this predicted demand, the dispatcher can subtract the number of taxis which are predicted to end up there anyhow, by having their predicted destinations in this point. Obviously, any benefit is contingent on the accuracy of the predictions. If too many predictions are wrong, the taxis will make many unnecessary idle trips to locations where too many customers were predicted, while

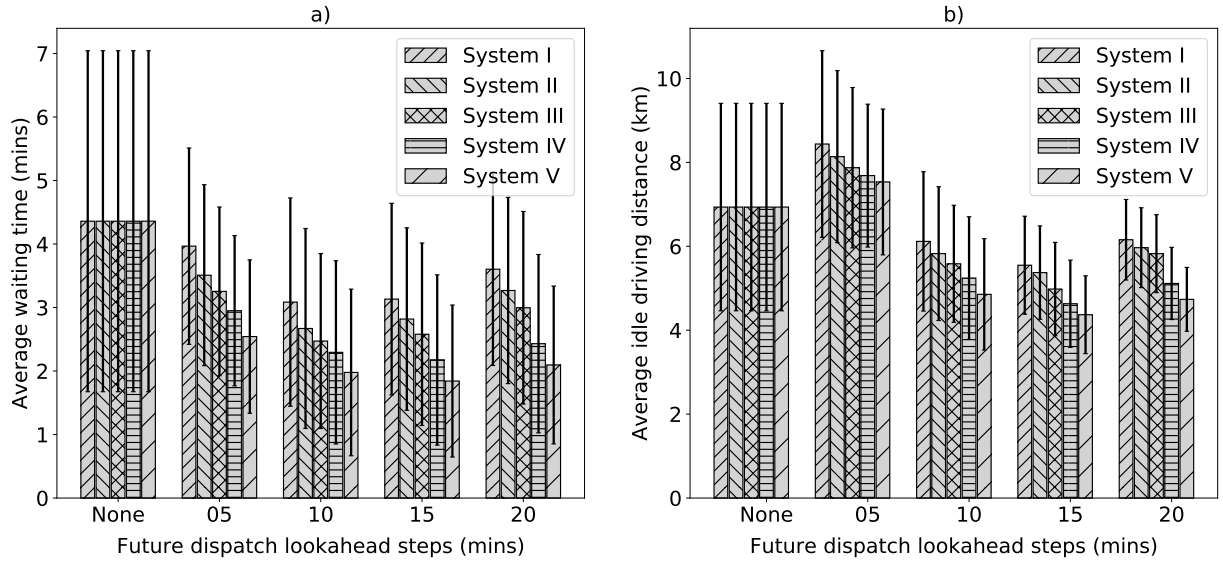


Fig. 13. Performance of passengers average waiting time, taxi average idle driving distance on different $lookahead$ time duration. Settings of each dispatch system: *System I: SWMD + SWDD*, *System II: LSTM + FN*, *System III: LSTM + FN-MDN*, *System IV: LSTM-MDN + FN* and *System V: LSTM-MDN + FN-MDN*.

Results obtained via using total number of 4000 taxis in the city.

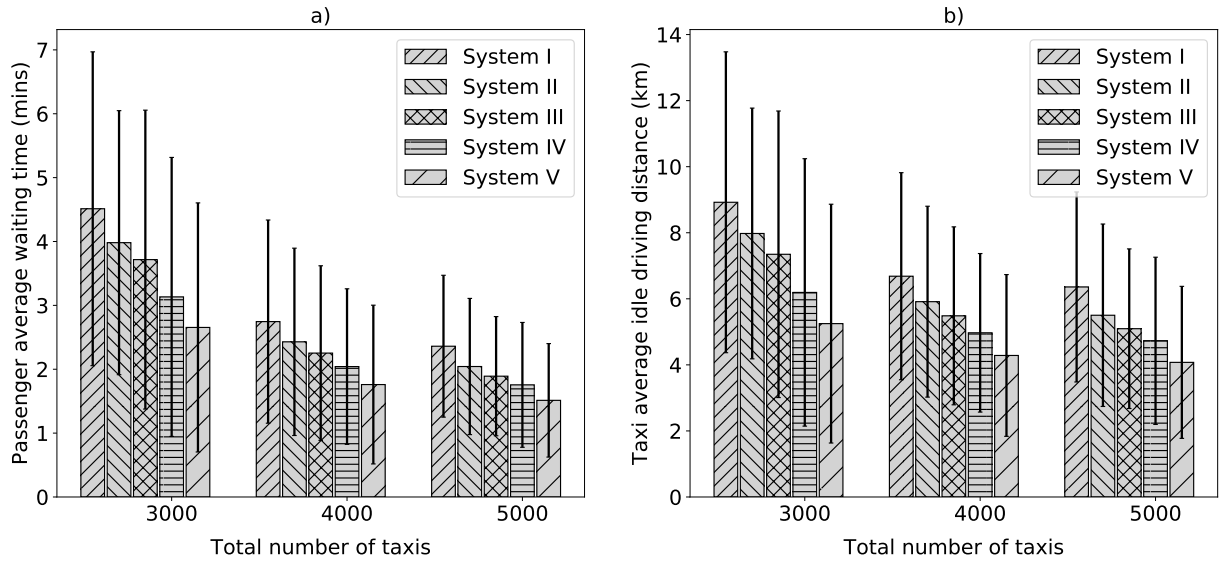


Fig. 14. The average waiting time of the passengers (left) and the average idle driving distance of the taxis (right) function of the number of taxis. Settings of each dispatch system: *System I: SWMD + SWDD*, *System II: LSTM + FN*, *System III: LSTM + FN-MDN*, *System IV: LSTM-MDN + FN* and *System V: LSTM-MDN + FN-MDN*. For all experiments we set $lookahead = 15$.

the customers who actually show up might have to wait longer as there are not enough nearby taxis available. The $lookahead$ parameter determines how far ahead in time will the dispatcher predict and take actions.

Fig. 13 shows the results and the standard deviation for the two performance metrics for different $lookahead$ time slot values. As expected, if there is no lookahead, all dispatchers have identical waiting time and idle driving values. The waiting time decreases with the lookahead up to around 10-15 minutes, after which it increases again, as the prediction errors become larger the farther in the future we make predictions. For a given lookahead, the performance waiting time decreases as we progress through Systems I, II, III, IV and V. This is expected because, as shown in the results in the previous section, the prediction components are getting better in this order. For the idle driving time, the pattern is different: moving from no lookahead to a 5 minute lookahead the idle driving distance actually increases, as taxis are required to make more pre-emptive trips. However, as the lookahead increases further to 10 and 15 minutes, the idle distance decreases as we find a more efficient dispatch that takes into account the destination. Starting with a lookahead of 20 minutes, the performance begins to weaken again with the higher prediction errors.

Fig. 14 shows experimental results and the standard de-

viation for the two performance metrics while varying the number of taxis in the city. As expected, raising the number of taxis from 3000 to 4000 reduces the waiting time of the passengers, with the differences being larger for the weaker predictor models. However, further raising the number of taxis to 5000 gives only a minor improvement in the average waiting time. A similar pattern applies to the idle driving distance - there is a more significant decrease from 3000 to 4000, and a minor decrease from 4000 to 5000. Another consideration here is that increasing the number of taxis, while does not add to the idle driving distance, it does add to the idle waiting time of the taxis and lowers the average utilization and thus the per-vehicle income.

The results of our dispatcher experiments can be summarized as follows. Using a better prediction engine improves the performance of the dispatcher for non-zero lookahead. An appropriately chosen lookahead in the dispatcher can improve the performance on the customer waiting time and the idle driving distance. Finally, the quality of the predictor in the dispatcher can achieve the same customer waiting time and idle driving distance with significantly lower number of taxis - for instance in Figure 14 the performance of System V with 3000 taxis is comparable to System I with 5000 taxis.

VI. Conclusion

In this paper, we used deep learning techniques to create two predictive models of taxi demand and destination in a city and built a dispatch system that takes advantage of these predictions. The optimal taxi assignment and reallocation strategy is obtained by formulating it as a mixed integer programming problem. We validated our work with real-world taxi trip data from New York City. The experimental results show that the proposed dispatch system can decrease the average waiting time of the passengers and the average idle driving distances of the taxis.

References

[1] W. Li, J. Cao, J. Guan, S. Zhou, G. Liang, W. K. Y. So, and M. Szczecinski, "A general framework for unmet demand prediction in on-demand transport services," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2018.

[2] X. Chen, F. Miao, G. J. Pappas, and V. Preciado, "Hierarchical data-driven vehicle dispatch and ride-sharing," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 4458–4463.

[3] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, "Taxi dispatch planning via demand and destination modeling," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, Oct 2018, pp. 377–384.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2470–2481, Aug 2018.

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

[10] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2572–2581, Aug 2018.

[11] C. M. Bishop, "Mixture density networks," Aston University, Tech. Rep., 1994.

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

[13] 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

[14] 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.

[15] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. of AAAI'18*, April 2018.

[16] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," *CoRR*, vol. abs/1508.00021, 2015.

[17] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in *Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 3583–3590.

[18] L. T. Loris Bazzani, Hugo Larochelle, "Recurrent mixture density network for spatiotemporal visual attention," *arXiv preprint arXiv:1603.08199v4*, 2017.

[19] F. Miao, S. Han, S. Lin, Q. Wang, J. A. Stankovic, A. Hendawi, D. Zhang, T. He, and G. J. Pappas, "Data-driven robust taxi dispatch under[-2pt] demand uncertainties," *IEEE Transactions on Control Systems Technology*, pp. 1–17, 2018.

[20] J. Lin, S. Sasidharan, S. Ma, and O. Wolfson, "A model of multimodal ridesharing and its analysis," in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, June 2016, pp. 164–173.

[21] D. Pelzer, J. Xiao, D. Zehe, M. H. Lees, A. C. Knoll, and H. Aydt, "A partition-based match making algorithm for dynamic ridesharing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2587–2598, Oct 2015.

[22] H. Zheng and J. Wu, "Online to offline business: Urban taxi dispatching with passenger-driver matching stability," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 816–825.

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

[24] National Oceanic and Atmospheric Administration. National oceanic and atmospheric administration (noaa) climate data online. [Online]. Available: https://www.ncdc.noaa.gov/cdo-web/

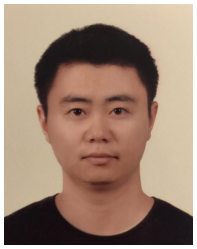
[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of NIPS'14*, December 2014, pp. 3104–3112.

[26] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni, "Learning real manipulation tasks from virtual demonstrations using LSTM and MDN," in *Proc. of AAAI Conf. on Artificial Intelligence (AAAI'18)*, Feb 2018.

[27] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3844–3848.

[28] D. Ha and D. Eck, "A neural representation of sketch drawings," *International Conference on Learning Representations*, 2018.

[29] X. Zhan, S. Hasan, S. V. Ukkusuri, and C. Kamga, "Urban link travel time estimation using large-scale taxi data with partial information," *Transportation Research Part C: Emerging Technologies*, vol. 33, pp. 37 – 49, 2013.



Jun Xu is currently a software engineer at Facebook. He has received his PhD degree in Computer Science at the University of Central Florida. 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 his BS degree in computer engineering from Sharif University of Technology, Tehran, Iran, and MS and Ph.D degrees in Computer Science at the University of Central Florida. His research interests include machine learning, robotics, and wireless sensor networks.

Ladislau Bölöni is a Professor 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 a Charles Millican Professor of Computer Science at University of Central Florida. She received her Ph.D. degree 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, smart cities, IoT-enabled healthcare and augmented reality, 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 the NCWIT 2021 Mentoring Award for Undergraduate Research (MAUR) Award, Women of Distinction Award by the UCF Faculty Excellence Center for Success of Women Faculty, UCF Research Incentive Award, and the University Excellence Award in Professional Service. Dr. Turgut serves on several editorial boards and program committees of prestigious ACM and IEEE journals and conferences. She is a senior member of IEEE, a member of ACM and the Upsilon Pi Epsilon honorary society.