Predicting Multi-step Citywide Passenger Demands Using Attention-based Neural Networks

Xian Zhou, Yanyan Shen*, Yanmin Zhu, Linpeng Huang
Department of Computer Science and Engineering
Shanghai Jiao Tong University
{zhouxian, shenyy, yzhu, lphuang}@sjtu.edu.cn

ABSTRACT

Predicting passenger pickup/dropoff demands based on historical mobility trips has been of great importance towards better vehicle distribution for the emerging mobility-on-demand (MOD) services. Prior works focused on predicting next-step passenger demands at selected locations or hotspots. However, we argue that multi-step citywide passenger demands encapsulate both time-varying demand trends and global statuses, and hence are more beneficial to avoiding demand-service mismatching and developing effective vehicle distribution/scheduling strategies.

In this paper, we propose an end-to-end deep neural network solution to the prediction task. We employ the encoder-decoder framework based on convolutional and ConvLSTM units to identify complex features that capture spatiotemporal influences and pickup-dropoff interactions on citywide passenger demands. A novel attention model is incorporated to emphasize the effects of latent citywide mobility regularities. We evaluate our proposed method using real-word mobility trips (taxis and bikes) and the experimental results show that our method achieves higher prediction accuracy than the adaptations of the state-of-the-art approaches.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**; • **Computing methodologies → Neural networks**; • **Applied computing → Forecasting**;

KEYWORDS

Demands Prediction; Convolutional Neural Network; LSTM; Attention Model

ACM Reference Format:
Xian Zhou, Yanyan Shen, Yanmin Zhu, Linpeng Huang. 2018. Predicting Multi-step Citywide Passenger Demands Using Attention-based Neural Networks. In *WSDM 2018: WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159682

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159682

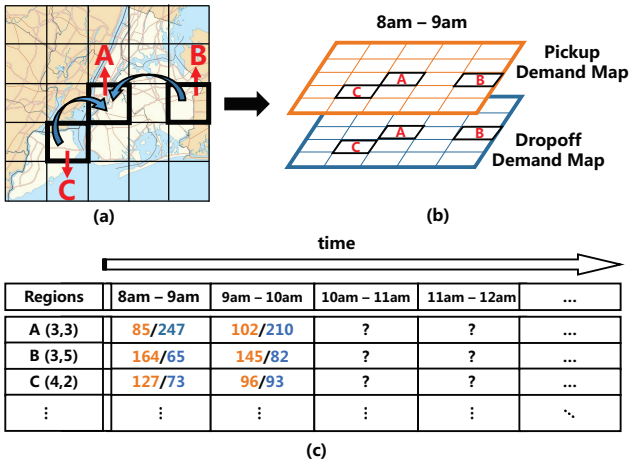


Figure 1: (a) Mobility trips; (b) pickup (yellow) and dropoff (blue) demand maps within a time interval; and (c) multi-step citywide passenger demands prediction problem.

1 INTRODUCTION

Recent years have witnessed the proliferation of various mobility-on-demand (MOD) services such as Uber, Didi and Mobike, which enable the evolution of passenger mobility demands towards more flexible and public vehicles, e.g., taxis, shared-use cars and bicycles. For instance, Uber doubled its gross bookings to $20 billion dollars in 2016 and now gives over one million rides a day [2]; Mobike, a bike sharing company, promoted its customers from 3 million to 6 million in the last three months of 2016 [1].

It is vital that the mobility-on-demand systems can be further developed to predict citywide passenger mobility demands (i.e., pickup and dropoff frequencies) with high accuracy, which assist greatly in customizing effective vehicle distribution and scheduling strategies to achieve demand-resource balance. It is worth mentioning that a mismatch between the vehicle distribution and passenger demand distribution may cause severe problems including traffic congestion, wasting of resources, and more importantly, driving down customer satisfaction.

Existing methods for passenger demands prediction have been focused on anticipating next-step passenger demands for a particular set of locations such as taxi stands, bike stations and hotspots [8, 9, 16, 19]. However, we argue that *multi-step citywide passenger demands prediction* (as illustrated in Figure 1) is considerably more

*Corresponding author.

beneficial to nowadays MOD services due to the following two reasons. First, multi-step passenger demands indicate the varying demand trend, which is useful to avoid impulsive vehicle scheduling responses in the presence of temporary demand fluctuation. In contrast, short-term passenger demands prediction results are typically shortsighted and more likely to cause unnecessary vehicle scheduling back and forth. Second, the large volume of vehicles are widespread in the whole city. Citywide passenger demands are expected to encapsulate global statuses and hence are more informative in terms of achieving better vehicles distribution. Intuitively, partial passenger demands in sub-areas are insufficient to generate an effective global solution to citywide vehicle distribution or scheduling.

In this paper, we study the problem of predicting multi-step citywide passenger mobility demands (i.e., pickup and dropoff) accurately. The key technical challenge of this problem is to deal with (1) *complex spatiotemporal influences on passenger demands*, and (2) *interactions between pickups and dropoffs*. Specifically, passenger demands in a region are typically correlated with the demands in its surrounding regions. Similarly, citywide passenger demand distribution over time is often changing near-continuously, and the demands in a region could be affected by its statuses in previous time intervals. It is also very likely that demand distributions in faraway time intervals exhibit periodicity. Furthermore, pickup demands and dropoff demands themselves are often correlated. For example, a large number of dropoffs in a region will increase nearby pickups in adjacent time intervals with high probability.

Inspired by the success of deep neural networks to model complex spatiotemporal features [26, 31, 32], we propose an end-to-end deep neural network solution to the multi-step citywide passenger demand prediction problem. We organize citywide pickup and dropoff demands at certain time period into a 3D *demand tensor* and take as input a sequence of demand tensors in previous time intervals. Our prediction model follows the general encoder-decoder framework [21–23, 26]. During the encoding phase, we extract spatial features from each tensor using convolutional units, where the features capture spatial influences and interaction between pickups and dropoffs effectively. We then utilize a neural network structure combining convolutional operations and long short-term memory to disclose complex spatiotemporal influences, which leads to a high-level representation for the input sequence. The decoder behaves reversely with respect to the encoder. It generates future demand tensors step by step following the encoded citywide passenger demands tendency.

More interestingly, we observe that *citywide passenger demands present regularity and there exist representative demand tensors indicating latent citywide mobility patterns*. For example, high pickup (resp. dropoff) demands occur in residential (resp. central business) areas during morning peak hours everyday. Failure to utilize these representative demand tensors may compromise the prediction performance.

We identify representative citywide demand tensors via clustering and introduce a novel attention model to incorporate these representatives into prediction. The attention mechanism [28] has been used to identify a form of *attention* that distills information in an image down to *salient parts* when generating its caption words iteratively. In our context, the salient parts correspond to the representative demand tensors that need to be concerned for next-step passenger demands prediction. Multilayer perceptrons are adopted to learn a weight vector that indicates the relative importance of each salient part in predicting next-step passenger demand, conditioned on previous demand tendency. Our attention model can be easily incorporated into the decoder, and leads to a substantial increase in prediction accuracy according to our experiments.

The main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to define the problem of predicting multi-step citywide passenger demands. We propose an end-to-end deep neural network solution to this prediction task. Our method employs an encoder-decoder framework based on convolutional and ConvLSTM units, which are able to capture complex spatiotemporal influences and pickup-dropoff interactions effectively.
- We identify representative citywide demands that reflect latent mobility regularities and introduce a novel attention model to exploit the effects of these representatives on citywide distributions. The attention model is easily incorporated into the decoder, to boost the prediction performance.
- We conduct extensive experiments to evaluate the performance of our proposed approach. We consider two kinds of real-world mobility trips, i.e., taxi and bike trips in New York. The results show that our method outperforms the adaptations of state-of-the-art prediction approaches by achieving 2.85%-79.69% lower root-mean-square error.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 provides problem definition. We describe the details of our prediction model in Section 4. The experimental results are presented in Section 5. We conclude this paper in Section 6.

2 RELATED WORK

2.1 Predicting Passenger Demands

Single-step demands prediction. Prior works on passenger demands prediction have been focused on anticipating next-step passenger demands for locations such as taxi stands [19], hotspots [8] and bike stations [9, 16]. To achieve high prediction accuracy, different prediction models have been proposed, e.g., a hierarchical prediction model [16] and a multi-similarity-based inference model, and a data stream ensemble framework [19] that combines weighted time-varying poisson model with the traditional ARMA model. Some works addressed the problem of predicting single-step citywide passenger demands. [29] proposed a time-location-sociality model to predict the distribution of passengers over all social functional regions, based on identifying three dimensional properties of city dynamics. A few works [24, 25] have employed ensemble models that combine spatial and temporal attributes to predict passenger demands for car-hailing services in a short time horizon. Most of these methods only considered spatiotemporal influences on passenger demands, regardless of other important factors such as latent citywide mobility regularities. Moreover, they focused on single-step prediction and utilized simple regression-based methods, where time-consuming feature engineering is required to identify effective spatial and temporal features.

Prediction with spatiotemporal data in other scenarios. As predicting citywide passenger demands can be viewed as a kind of spatiotemporal data prediction problem, prediction models for spatiotemporal data are also feasible to passenger demands prediction. An increasing number of works [12, 15, 31, 32] developed deep neural networks to predict spatiotemporal data. For example, [32] and [31] proposed deep learning based prediction models for spatiotemporal data by leveraging convolutional and residual networks [13]. Their approaches exploited geographically near/distant dependencies, temporal closeness, period and trend. However, these methods aim to predict short-term spatiotemporal data and the adaptations of these methods to multi-step prediction show inferior performance experimentally (see Section 5). Recently, a novel neural network structure ConvLSTM has been proposed in [26], to perform nowcasting for local regions over a relatively short period of time. Since ConvLSTM is able to capture spatiotemporal correlations effectively, we utilize ConvLSTM to our passenger demands prediction context. However, there are some key differences between precipitation nowcasting and passenger demands prediction as the latter is affected by pickup-dropoff interactions and latent citywide mobility regularities.

2.2 Attention Mechanism in Neural Networks

Attention mechanism has been successfully incorporated into neural networks to handle a variety of tasks such as visual object classification [18], vision question answering [27], image caption generation [28, 30], machine translation [5], speech recognition [6, 10], and so on. The basic idea is to process input representation and select relevant contents iteratively.

For visual object classification, a visual attention model proposed in [18] constructs a bandwidth-limited sensor to extract retina-like representation and an environment action to decide the deployment details of next sensor. They leveraged the recurrent model inspired by human perceptron, i.e., people pay attention to selective parts of visual space, and combine information from different fixations over time to build an internal representation of the scene [20]. For vision and textual question answering, Xiong et al. [27] introduced attention-based gated recurrent units (GRU) in dynamic memory network. They used an attention gate computed by input and question representations to extract contextual information, and the attention was focused on a subset of input facts. For machine translation [5], an attention extension was introduced to encode the input sentence into several vectors and adaptively choose a subset of these vectors while performing the translation. This method automatically searches for parts of a source sentence that are relevant to the next prediction word, and is reported to outperform conventional translation models, regardless of the sentence length. Similarly, for speech recognition, [6] proposed an end-to-end system which consists of a bi-directional RNN encoder and an attention-based recurrent sequence generator. Compared with vanilla encoder-decoder models, attentive encoder-decoder models inspect alignments between the generated words and those in the source sentence, and learn which part of input sentence the model should pay attention to when generating the next word.

Inspired by the advantages of the attention mechanism in sequence prediction [5, 28, 30], in this paper, we introduce an attention model to disclose the impact of representative historical citywide demands on next-step citywide demands. Our attention model is different from existing ones as our input is a sequence of 3D demand tensors rather than a sequence of words or a single picture.

3 PROBLEM DEFINITION

To predict citywide passenger demands, we divide the complete city area into pairwise disjoint regions (i.e., grids), in the similar way as in [31].

Definition 3.1 (Grid Map). We represent a complete city area as a rectangle where $p_l = (lon_l, lat_l)$ and $p_h = (lon_h, lat_h)$ denote the lowest and the highest points in the city along longitude and latitude coordinates, respectively. Given a length λ, we divide the rectangle into $n \times m$ regions along the two coordinates, where $n = \lceil \frac{lon_h - lon_l}{\lambda} \rceil$ and $m = \lceil \frac{lat_h - lat_l}{\lambda} \rceil$.

We denote by $G_\lambda = \{g_{ij} \mid i \in [1, n], j \in [1, m]\}$ the set of all regions, which is also referred to as a grid map. For any point $p = (lon_p, lat_p)$, we say p resides in region g_{ij}, denoted by $p \in g_{ij}$, iff $lon_p \in [lon_l + (i-1)\lambda, lon_l + i\lambda)$ and $lat_p \in [lat_l + (j-1)\lambda, lat_l + j\lambda)$.

We use length λ to control the granularity of regions. Intuitively, smaller value of λ leads to a larger number of regions as well as fine-grained passenger demands, and vice versa.

We consider a set of historical mobility trips occurred in a city. Each trip is represented by a tuple of $(p_{pick}, p_{drop}, t_{pick}, t_{drop})$, which means the trip starts from point p_{pick} at time t_{pick}, and ends in point p_{drop} at time t_{drop}. We then map all trips onto the grid map of the city. Given any time interval $[s_t, e_t)$, we can obtain *pickup/dropoff demand maps* that summarize citywide pickup/dropoff passenger demand frequencies during that time period.

Definition 3.2 (Pickup/Dropoff Demand Maps). Let $\mathcal{T}$ be a set of mobility trips over grid map G_λ. Given the t-th time interval $[s_t, e_t)$, we can compute the pickup demand map $P_t = \{\delta^p_{t,ij}\}_{g_{ij} \in G_\lambda}$ within the interval that satisfies:

$$\delta^p_{t,ij} = \{|T \in \mathcal{T} \mid | \, T.p_{pick} \in g_{ij} \wedge T.t_{pick} \in [s_t, e_t)\}.$$

Intuitively, $\delta^p_{t,ij}$ denotes the number of trips that start from region g_{ij} during the time interval $[s_t, e_t)$. Similarly, we can obtain a dropoff demand map $D_t = \{\delta^d_{t,ij}\}_{g_{ij} \in G_\lambda}$ within the t-th interval where:

$$\delta^d_{t,ij} = \{|T \in \mathcal{T} \mid | \, T.p_{drop} \in g_{ij} \wedge T.t_{drop} \in [s_t, e_t)\}.$$

We organize the pickup and dropoff demand maps for the t-th time interval as a 3D demand tensor $\mathcal{M}_t \in \mathbb{R}^{n \times m \times 2}$ over the grid map G_λ. We have $(\mathcal{M}_t)_{i,j,0} = \delta^p_{t,ij}$ and $(\mathcal{M}_t)_{i,j,1} = \delta^d_{t,ij}$.

We now formalize the problem of predicting multi-step citywide passenger demands as follows.

Definition 3.3 (Multi-step Citywide Demands Prediction). Consider a grid map G_λ. Given N demand tensors $\{\mathcal{M}_i \mid i = 1, \cdots, N\}$ over the previous N time intervals, we try to predict B demand tensors $\{\mathcal{M}_i \mid i = N+1, \cdots, N+B\}$ for the next B time intervals.

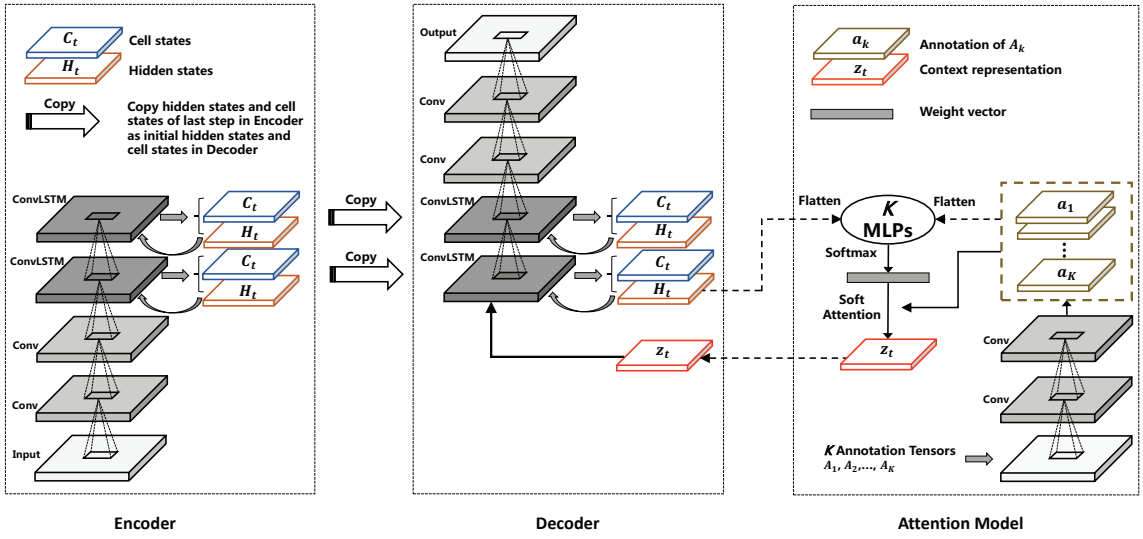


Figure 2: Overview of the Proposed Approach. The Encoder is composed of convolutional units and ConvLSTM units that encode the input sequence into fixed dimensional representations (hidden states and memory states); The Attention Model computes weights for salient parts (annotations) and identifies the attention information (context representation); The Decoder leverages the attention information (context representation) from the attention model and decodes the encoded representations from Encoder to generate future passenger demands.

4 ATTENTION BASED ENCODER-DECODER FRAMEWORK

4.1 Overview

Figure 2 provides an overview of our end-to-end deep neural network solution to the prediction task. We adopt the encoder-decoder framework which is widely used for generating target sentences [23, 28] and predicting future sequences [21, 26]. At a high-level, the idea is to first "encode" an input sequence of passenger demand tensors into fixed dimensional representations (i.e., sequence representations), and then "decode" the representations to produce the desired sequence of future demand tensors.

CNN+ConvLSTM based sequence encoding. The encoder consists of two major components: convolutional units in the lower layers and ConvLSTM units in the higher layers. Given a sequence of N demand tensors, we supply each tensor in the sequence to the stack of convolutional and ConvLSTM layers iteratively. The convolutional layers exploit the spatial influences over passenger demands and learn a latent representation I_t^e for each input demand tensor $\mathcal{M}_t$. The ConvLSTM layers absorb the sequence of latent representations $\{I_t^e\}_{t=1}^N$ produced by the last convolutional layer, and generate the sequence representations as the output. We leverage ConvLSTM, an extended version of FC-LSTM [11], to capture the temporal information flow in the sequence while retaining spatial correlations via convolutional operations. Specifically, we apply ConvLSTM to each I_t^e and the gating mechanism in ConvLSTM decides the amount of information from both I_t^e and LSTM memory state that will be propagated to enrich the final sequence representation. It is also worth mentioning that the convolutional structure in ConvLSTM is able to model the spatial information flow well. The final sequence representations are stored in the last states of ConvLSTM cells, which provide a guidance for the prediction during decoding phase.

Table 1: Notations

Symbol	Description
P_t/D_t	pickup/dropoff demand map at t-th step
$\mathcal{M}_t$	demand tensor comcatenating P_t and D_t at t-th step
N/B	input/output length
$I_{t,L}^e$	output of L-th CNN layer for t-th input $\mathcal{M}_t$ in encoder
i_t^e/i_t^d	input gate of encoder/decoder ConvLSTM
f_t^e/f_t^d	forget gate of encoder/decoder ConvLSTM
o_t^e/o_t^d	output gate of encoder/decoder ConvLSTM
C_t^e/C_t^d	cell state of encoder/decoder ConvLSTM for t-th input $\mathcal{M}_t$/output $\mathcal{M}_{N+t}$
H_t^e/H_t^d	hidden state of encoder/decoder ConvLSTM for t-th input $\mathcal{M}_t$/output $\mathcal{M}_{N+t}$
A_k	k-th annotation tensor
a_k	annotation of A_k
$\alpha_{t,k}^a$	weight of a_k for t-th output $\mathcal{M}_{N+t}$
z_t	context representation for t-th output $\mathcal{M}_{N+t}$ in decoder

Attention-enhanced sequence decoding. The decoder predicts a sequence of next B pickup/dropoff demand tensors based on the demand tendency learnt by the encoder. The structure of decoder exhibits a symmetrical form with respect to the encoder. That is, we construct ConvLSTM units in lower layers and deconvolutional units in higher layers to predict the next B demand tensors iteratively. For ConvLSTM units, we copy hidden states and cell states of last step in encoder as initial hidden states and cell states in decoder. In each iteration, ConvLSTM layers produce a high-level representation for the next-step demand tensor by updating the

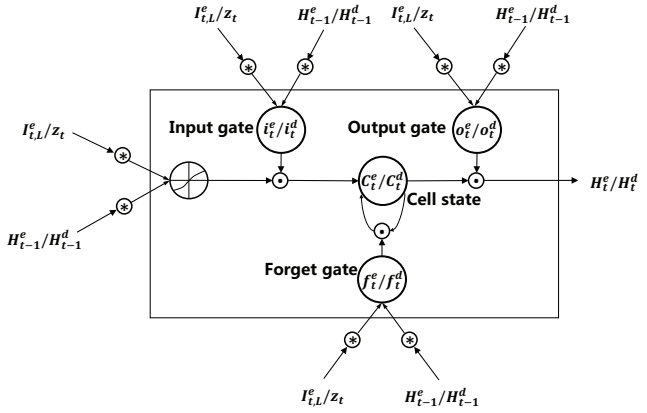


Figure 3: Structure of Encoder/Decoder ConvLSTM Cell

states accordingly. The deconvolutional layers then translate this high-level representation into the corresponding 3D demand tensor.

We find that passenger demand distributions offer certain spatiotemporal regularities, which may be caused by the latent citywide mobility patterns. For example, the demands around subway stations during weekday peak hours are always high, while those in midnight are quite low. To capture such regularities, we perform k-means clustering over historical demand tensors. The resultant K representative demand tensors are referred to as *annotation tensors*. In this paper, we leverage an attention model to incorporate these representatives into next-step demand prediction, which is a novel attempt.

Our attention model first learns high-level features, namely K *annotations*, from K annotation tensors via convolutional operation. Before decoding the next i-th demand tensor, we first put the K annotations and the current sequence representation (e.g., the encoder output if $i = 1$) into K multilayer perceptrons, and obtain a K-dimensional weight vector. The summation over weighted annotations composes a *context representation* that indicates to what degree each representative citywide demand tensor is supposed to be noticed in predicting the next-step demand tensor. This context representation is then used as an input to the decoder. With the help of attention model, we observe a substantial performance gain in predicting multi-step demand tensors (see the experiments in Section 5).

In what follows, we provide details of the encoder, attention model and the decoder, respectively. Table 1 summarizes the symbols and their meanings used throughout this paper.

4.2 Encoding Previous Passenger Demands

We construct the encoder structure based on a combination of convolutional and ConvLSTM layers. Consider a sequence $\{\mathcal{M}_1, \cdots, \mathcal{M}_N\}$ of N demand tensors as input. The convolutional layers extract spatial features from each input demand tensor. Specifically, we view each demand tensor $\mathcal{M}_t$ as a two-channel (i.e., pickups and dropoff) image and apply convolutional neural network to yield a 3D feature tensor $I^e_{t,L} = \underbrace{CNN \cdots CNN}_{L}(\mathcal{M}_t)$, where L is the number of CNN layers. The first two dimensions of $I^e_{t,L}$ correspond to spatial coordinates (i.e., rows and columns), and the third dimension contains the extracted features.

After convolutional layers, we obtain a sequence of 3D features $\{I^e_{t,L}\}^N_{t=1}$. We next encode the sequence by *encoder ConvLSTM*, a Recurrent Neural Network made of ConvLSTM [26] units. Figure 3 shows the structure of ConvLSTM unit. Each ConvLSTM unit has a memory cell with a state C^e_t at t-th step. In each step, the unit receives the past cell state C^e_{t-1} and two external inputs $I^e_{t,L}, H^e_{t-1}$, where $I^e_{t,L}$ is the feature tensor extracted by convolutional layers and H^e_{t-1} is the previous hidden state of the cell. Access or modification to the cell is controlled by its input gate i^e_t, forget gate f^e_t and output gate o^e_t. That is, the input gate i^e_t decides how much information from external input will be accumulated to the cell; the forget gate f^e_t controls to what degree the past cell state C^e_{t-1} will be forgotten; the output gate o_t determines how much information the latest cell state C^e_t will be emitted to the hidden state H^e_t. When there are multiple ConvLSTM layers, the output H^e_t of previous ConvLSTM layer is taken as input for the next ConvLSTM layer.

In ConvLSTM, the inputs, outputs, hidden states, cell states and all gates are 3D tensors. To capture spatial information, when a new input $I^e_{t,L}$ in the sequence comes, the ConvLSTM determines the future cell state C^e_t and whether to activate input/forget/output gates by using a convolutional operation (see circles containing "$*$" in Figure 3). After applying ConvLSTM recursively to the input sequence, we obtain the latest hidden state H^e_t and memory state C^e_t, which will be used for our decoder. Formally, the updating equations for ConvLSTM are provided below.

$$i^e_t = \sigma(W^e_{xi} * I^e_{t,L} + W^e_{hi} * H^e_{t-1} + b^e_i) \quad (1)$$

$$f^e_t = \sigma(W^e_{xf} * I^e_{t,L} + W^e_{hf} * H^e_{t-1} + b^e_f) \quad (2)$$

$$o^e_t = \sigma(W^e_{xo} * I^e_{t,L} + W^e_{ho} * H^e_{t-1} + b^e_o) \quad (3)$$

$$C^e_t = f^e_t \circ C^e_{t-1} + i^e_t \circ tanh(W^e_{xc} * I^e_{t,L} + W^e_{hc} * H^e_{t-1} + b^e_c) \quad (4)$$

$$H^e_t = o^e_t \circ tanh(C^e_t) \quad (5)$$

where '$*$' denotes the convolutional operator (represented as a circle containing "$*$" in Figure 3), '$\circ$' is the Hadamard product (represented as a circle containing"$\cdot$" in Figure 3) and $\sigma(\cdot)$ is sigmoid function. Different kernel matrices W and biases b are parameters to be learned.

4.3 Attention Model

Figure 4 illustrates the structure of our attention model which mainly performs two steps. First, we identify K representative citywide demand tensors $\{A_k\}^K_{k=1}$, referred to as *annotation tensors*, by clustering historical demand tensors using K-means++ [4]. Intuitively, each annotation tensor A_k is a 3D tensor where the value in $(A_k)_{ij}$ indicates the pickup and dropoff demands in region g_{ij} influenced by mobility regularities of the city. While clustering all the previous demand tensors (more than N) can be expensive, the clustering is only conducted once and the results can be reused for any predictions afterwards. We then use a convolutional neural network to extract spatial features from each A_k, and obtain K *annotations* $\{a_k\}^K_{k=1}$ corresponding to the k salient parts to be concerned. We develop a two-layer CNN to retain close correspondence between a_k and A_k. Note that the *annotations* produced by CNN are also 3D tensors.

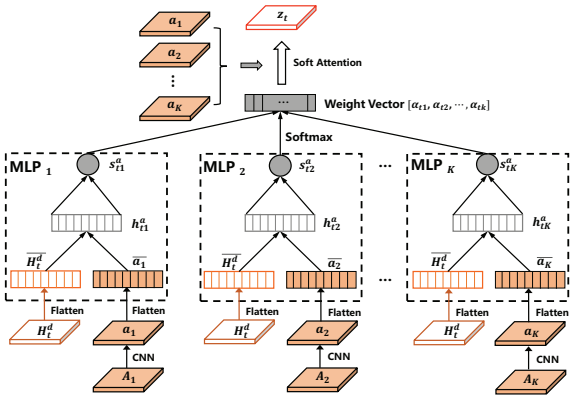


Figure 4: Structure of Attention Model

Second, we learn a weight vector $\langle \alpha_{t1}, \cdots, \alpha_{tk} \rangle$ for the K annotations, where the weight α_{tk} can be interpreted as the relative importance of a_k to the t-th output demand tensor. Each weight α_{tk} is computed by a multilayer preceptron (MLP) that takes flattened hidden state H^d_{t-1} of the previous demand tensor (denoted as $\overline{H^d_{t-1}}$) and flattened annotation a_k (denoted as $\overline{a_k}$) as input, followed by a softmax operation. Similar to the soft attention mechanism in [28], we then perform summation over weighted annotations to get a *context representation* z_t, which will be incorporated into the decoder for prediction.

The key equations to generate the context representation z_t for the t-th output demands prediction ($t \in [1, B]$) are summarized as follows.

$$h^a_{tk} = f(W^a_h \overline{H^d_{t-1}} + W_a \overline{a_k} + b^a_h), \quad \forall k \in [1, K] \tag{6}$$

$$s^a_{tk} = f(W^a_s h^a_{tk}), \quad \forall k \in [1, K] \tag{7}$$

$$\alpha_{tk} = \frac{exp(s^a_{tk})}{\sum_{k=1}^{K} exp(s^a_{tk})}, \quad \forall k \in [1, K] \tag{8}$$

$$z_t = \sum_{k=1}^{K} \alpha_{tk} a_k \tag{9}$$

where f is an activation function that could be *ReLU* or *tanh*. The context representation z_t is then fed into decoder.

4.4 Decoding Multi-step Passenger Demands

The structure of the decoder is symmetrical to that of the encoder. As shown in Figure 2, the decoder has ConvLSTM units in lower layers and deconvolutional units in higher layers. We initialize the memory states and hidden states of the *decoder ConvLSTM* by copying the last memory states and hidden states from the *encoder ConvLSTM*.

In the t-th step ($t \in [1, B]$) prediction, the external inputs to the decoder ConvLSTM are from two sources: the context representation z_t from the attention model (Equation 9) and the previous hidden state and memory state of the cell. The output is the updated hidden state H^d_t, which is a high-level representation for the next t-th step demand tensor. The key equations for the decoder ConvLSTM are the following.

$$i^d_t = \sigma(W^d_{zi} * z_t + W^d_{hi} * H^d_{t-1} + b^d_i) \tag{10}$$

$$f^d_t = \sigma(W^d_{zf} * z_t + W^d_{hf} * H^d_{t-1} + b^d_f) \tag{11}$$

$$o^d_t = \sigma(W^d_{zo} * z_t + W^d_{ho} * H^d_{t-1} + b^d_o) \tag{12}$$

$$C^d_t = f^d_t \circ C^d_{t-1} + i^d_t \circ tanh(W^d_{zc} * z_t + W^d_{hc} * H^d_{t-1} + b^d_c) \tag{13}$$

$$H^d_t = o^d_t \circ tanh(C^d_t) \tag{14}$$

where '$*$' denotes the convolutional operator and '$\circ$' denotes the Hadamard product.

Each high-level representation H^d_t will be put through deconvolutional units to compute the corresponding 3D demand tensor: $\mathcal{M}_{N+t} = \underbrace{DCNN \cdots DCNN}_{L}(H^d_t)$, where L is the number of DCNN layers, equal to the number of CNN layers in encoder.

5 EXPERIMENTS

In this section, we compare our method named **AttConvLSTM** with several state-of-the-art approaches to multi-step passenger demands prediction. We first describe the datasets used for the experiments and then list comparison methods. Finally, we present experimental results in detail.

5.1 Datasets

We conduct experiments using two real-world mobility datasets: taxi trips and bike trips in New York. The details of the two datasets are described as follows:

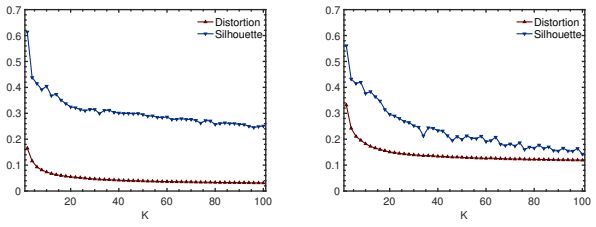
- **TaxiNYC**[1]: TaxiNYC consists of 1.19 billion taxicab trip records in New York from 1st January 2009 to 31st December 2015. On average, there are 170,235,925 trip records collected in each year. We use five-year records from 2009 to 2013 as training data, while the records in 2014 and 2015 are used as validation and test sets, respectively.
- **CitiBikeNYC**[2]: CitiBikeNYC contains 28,732,316 bike trips in New York from 1st July 2013 to 30th June 2016. Throughout these years, CitiBike has established over 600 stations and 10,000 bikes. An example bike trip contains: trip duration, starting and ending station IDs, start and end timestamps. We use the data from 1st July 2015 to 31st December 2015 as validation set, and the data from 1st January 2016 to 30th June 2016 as test set. The others are used as training data.

We preprocessed both datasets to generate pickup and dropoff maps as described in Section 3. By default, we set time interval to 1 hour. Regarding the relatively narrower coverage of CitiBikeNYC trips, we divided the area covered by citibikes into a 16×16 grid map, for which each grid is about $1km \times 1km$. For the larger TaxiNYC dataset, we used 64×64 grids to partition the whole area, where each grid covers $5km \times 5km$ area.

We find that the representative demand distributions exist when clustering historical demand tensors using K-means++ [4] as illustrated in Figure 5. As cluster number K increases, the distortion defined by the sum of the squared distances between each tensor and its closest centroid decreases to convergence, minimized by

[1] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
[2] https://www.citibikenyc.com/system-data

K-means clustering algorithm. The Silhouette value is a measure of how similar a tensor is to its own cluster compared to other clusters, where a high value indicates an appropriate match to its own cluster and poorly matched to neighboring clusters. From Figure 5, the clustering results for TaxiNYC and CitiBikeNYC justify the existence of representative demand distributions. Intuitively, an appropriate value of k with low distortion and high Silhouette strikes a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each tensor to its own cluster. Therefore, we set the cluster number K to 16 and 32 for CitiBikeNYC and TaxiNYC, respectively.



(a) TaxiNYC (b) CitiBikeNYC

Figure 5: Kmeans Results on TaxiNYC and CitiBikeNYC

5.2 Baseline Methods and Measurement

Baseline Methods. We compare our **AttConvLSTM** method with both basic and advanced methods as follows.

- **HA**: It predicts pickup and dropoff demands by averaging historical pickup and dropoff demand values for the same time interval. For example, pickup/dropoff demands at 9-10am on Monday are predicted as the average of historical pickup/dropoff demands at 9-10am on all past Mondays.
- **ARMA** [7]: Auto Regressive Moving Average (ARMA) is a widely used method for predicting future values over multiple time periods. We use this method to predict next-step passenger demands for each region.
- **VAR** [17]: Vector Auto-Regressive (VAR) is a multivariate model capable of capturing linear interdependencies among multiple time series. We implemented the prediction model by regarding the pickup/dropoff demands in different regions as variables.
- **ResNet** [31]: ResNet is a deep-learning-based approach to predicting traffic of flows based on unique properties of spatiotemporal data. It has achieved high accuracy in predicting next-step spatiotemperal data.
- **ConvLSTM**: ConvLSTM is our convolutional+ConvLSTM approach without attention model, which is used for evaluating the effectiveness of our attention model.

Experimental Settings. We now provide implementation details of **AttConvLSTM**. The encoder consists of 2-layer CNN and 2-layer ConvLSTM. The kernel size of the 2-layer CNN is set to 3×3 with a stride of 2 and the number of features is set to 8 for the lower layer and 16 for the higher layer. For the 2-layer ConvLSTM, the kernel size of all convolutional operations is 3×3 with a stride of 1, and both layers contain 64 hidden states for each grid. The decoder is composed of 2-layer ConvLSTM and 2-layer DCNN, with the same configuration as encoder. For the attention model, the number of nodes in the single hidden layer of MLP is set to 1024. The depth of AttConvLSTM makes different effects on the two datasets. Deeper model gives a better result on TaxiNYC while it predicts poorly on CitiBikeNYC, since the small grid map (i.e., 16×16) in CitiBikeNYC cannot fit deeper networks well. When training our model, we adopted mini-batch learning method with a batch size of 16 and used Adam [14] optimizer with a constant learning rate of 0.0002.

Table 2: Comparison of Different Methods

Methods	TaxiNYC	CitiBikeNYC
HA	38.675	10.095
ARMA	132.499	13.562
VAR	54.516	13.277
ResNet	43.974	10.584
ConvLSTM	34.784	7.984
AttConvLSTM	**26.912**	**7.756**

We implemented all the methods in Python and used TensorFlow [3] to implement ResNet, ConvLSTM and AttConvLSTM. For single-step prediction methods (i.e., ARMA, VAR and ResNet), we predict multi-step passenger demands by iteratively using previous prediction results as input for next-step prediction. For all methods, we used the validation set for parameter selection and test set for performance comparison. All experiments were conducted on a server with NVIDIA Tesla K80 accelerator.

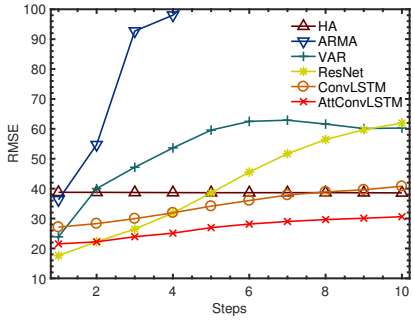
Measurement. We consider previous 10-step pickup/dropoff demands as input and predict citywide pickup/dropoff demands for the next 10 steps. We evaluate all approaches using Root Mean Square Error (RMSE) as follows.

$$RMSE = \sqrt{\frac{1}{Q}\sum_{t}\sum_{i}\sum_{j}(x_{ij}^{t} - \hat{x}_{ij}^{t})^2} \tag{15}$$

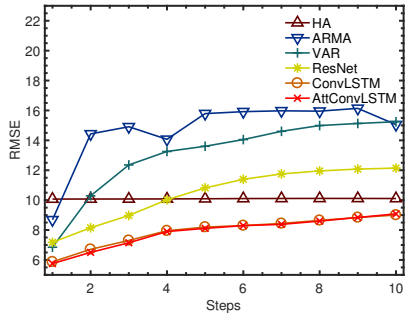
where $\hat{x}_{ij}^{t}$ and x_{ij}^{t} are respectively the predicted pickup/dropoff value and ground truth for region g_{ij} at t-th step; Q is the number of all the predicted values.

5.3 Results

5.3.1 Comparison of different methods. Table 2 shows the comparison results of six approaches over two datasets. As we can see, all the methods report smaller RMSE on CityBikeNYC than TaxiNYC. This is because the total number of taxi demands is about one order of magnitude larger than total bike demands. AttConvLSTM achieves the lowest RMSE values on both datasets. The incorporation of the attention model helps reduce RMSE by 22.63% on TaxiNYC and 2.36% on CityBikeNYC. This result confirms that attention model enhances decoder by exploiting intrinsic region-specific demand regularities. The larger reduction in RMSE over TaxiNYC is due to the fact that taxi demands are more easily affected by latent citywide mobility regularities than bike demands. Furthermore, both AttConvLSTM and ConvLSTM outperform ResNet, and AttConvLSTM achieves 38.80% and 26.35% lower RMSE on TaxiNYC and CitiBikeNYC datasets, respectively. ResNet focuses



(a) TaxiNYC



(b) CitiBikeNYC

Figure 6: Step-wise Prediction Results

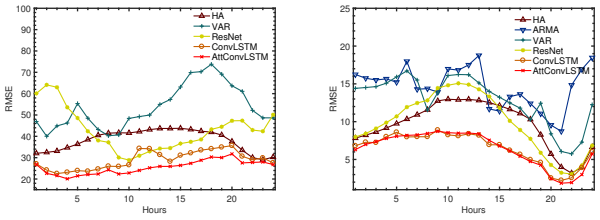
on next-step prediction and benefits from historical data in the same time period and previous time intervals. However, it performs poorly for multi-step prediction when passenger demands (e.g., taxi demands) in future time intervals vary a lot.

HA leverages all historical data and achieves good prediction results, but still reports 43.71% and 29.50% higher RMSE than AttConvLSTM over the two datasets. ARMA and VAR have the largest RMSE values on both datasets. As ARMA only explores temporal dependencies and VAR captures linear spatio interdependencies, the poor prediction performance indicates the limitations of using spatiotemporal features only. The parameters of ARMA and VAR are tuned for next-step prediction, which cause the inferior performance in multi-step predictions as discussed in next part.

5.3.2 Results on step-wise prediction. For further analysis, we present the step-wise prediction results over two datasets in Figure 6. AttConvLSTM and ConvLSTM are robust as the step number varies from 1 to 10, i.e., small increase in RMSE. It performs the best for all the 10 steps except step one on TaxiNYC, and outperforms all the other approaches across all the steps on CitiBikeNYC. ARMA, VAR and ResNet perform worse for later steps. This demonstrates that for these single-step approaches, low-quality predictions from previous steps significantly degrade the accuracy of later predictions. HA performs consistently over all the steps, thanks to the availability of a large amount of historical data. From Figure 6(a) and 6(b), we see that for the first-step prediction, HA performs the worst over two datasets because other baselines adopt spatiotemporal features to improve single-step prediction accuracy effectively.

Table 3: Results on Abnormal Cases

Methods	TaxiNYC	CitiBikeNYC
HA	954.889	28.787
ARMA	1872.487	45.238
VAR	1452.385	40.330
ResNet	1161.004	32.334
ConvLSTM	817.676	19.314
AttConvLSTM	**322.537**	**14.183**



(a) TaxiNYC (b) CitiBikeNYC

Figure 7: RMSE Values in Different Time Intervals

ResNet achieves the smallest RMSE value for step 1 on TaxiNYC, but performs worse than AttConvLSTM and ConvLSTM on CitiBikeNYC for all the steps. This may be caused by the irregular bike demands, i.e, the demands in the same hour but different days can vary greatly due to external conditions such as weather and temperature. It is worth mentioning that the inferior performance of AttConvLSTM for step 1 on TaxiNYC (compared to ResNet) is mainly caused by the different loss functions being used. The loss function used by AttConvLSTM considers prediction errors over multiple future steps while ResNet directly minimizes the prediction errors for the next step. In a nutshell, AttConvLSTM achieves higher prediction accuracy for later steps as well as over irregular mobility trips like Citibike.

5.3.3 Effects of different time intervals. In Figure 7, we present the performance of all methods over two datasets across 24 different time intervals, i.e., 0am-1am, 1am-2am, $\cdots$, 11pm-0am. That is, we try to predict demands for the next 10 intervals starting from the given time interval. To make the figures concise, we omit the results of ARMA since it gets the worst performance. AttConvLSTM achieves the best or comparable performance for all the 24 time intervals on both datasets. It performs consistently better than other approaches no matter whether the prediction starts from peak hours or off-peak hours. For TaxiNYC (in Figure 7(a)), all the methods have relatively higher RMSE values when predicting future demands from 3pm-7pm. The reason may be the uncertainty of taxi demands in the evening, when the demands can be easily influenced by people's arbitrary night activities. For CitiBikeNYC, all methods predict well for the intervals at night (8pm-11pm) but get high RMSE during daytime (5am-6pm). This is mainly because people ride bikes more often at daytime, which introduces more diversity in terms of demand distribution.

5.3.4 Analysis of abnormal cases. To further analysis on abnormal situations, we study the prediction of abnormal cases in test

data where the real demands deviate historical average greatly, which are the most difficult for vehicle distribution and scheduling. Specifically, we randomly select 1,011,941 and 379,338 cases from CitiBikeNYC and TaxiNYC, in which the relative error from historical average is more than 50%. The number of abnormal cases on TaxiNYC is smaller than CitiBikeNYC since bike demands are more likely to be affected by external conditions. Table 3 shows the results on those abnormal cases. We see that all the RMSE results are greatly worse than the average error in Table 2 and AttConvLSTM achieves 60.55% and 26.57% lower RMSE than second best method ConvLSTM on TaxiNYC and CitiBikeNYC respectively. Although there are less abnormal cases on taxi demands compared to bike demands, the bad prediction on large scale could incur terrible useless dispatch and scheduling. And the results in Table 3 shows that AttConvLSTM can effectively capture human movements that are significantly deviated from the historical average compared to other methods.

6 CONCLUSION

In this paper, we study the problem of predicting multi-step citywide passenger demands. The main technical challenge is to enforce the complex spatiotemporal influences, pickup-dropoff interactions and latent citywide demand regularities into future prediction. We solve the problem by introducing a novel attention-based encoder-decoder deep learning approach. We employ convolutional and ConvLSTM units in both encoder and decoder, and learn attention to emphasize the effects of representative citywide demand patterns on each-step prediction during the decoding phase. Our experiments on two real-world datasets (taxi and bike mobility trips) demonstrate that (1) our proposed approach outperforms the state-of-the-art prediction approaches; (2) the attention model can effectively improve the accuracy for multi-step prediction. In the future work, we will consider exogenous data (e.g., weather condition data) and fuse them to predict passenger demands more accurately.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their insightful and helpful comments, which improve the paper. This research is supported in part by 973 Program (no. 2014CB340303), NSFC (no. 61772341, 61472254, 61170238, 61602297 and 61472241), Singapore NRF (CREATE E2S2), and 863 Program (no. 2015AA015303). This work is also supported by the Program for Changjiang Young Scholars in University of China, and the Program for Shanghai Top Young Talents.

REFERENCES

[1] March 2, 2017. http://tech.qq.com/original/archives/a128.html. (March 2, 2017).
[2] Retrieved April 14, 2017. Uber financials 2016. (Retrieved April 14, 2017).
[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
[4] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.
[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[6] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*. IEEE, 4945–4949.
[7] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.
[8] Han-wen Chang, Yu-chin Tai, and Jane Yung-jen Hsu. 2009. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining* 5, 1 (2009), 3–18.
[9] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. 2016. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 841–852.
[10] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*. 577–585.
[11] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[12] Aditya Grover, Ashish Kapoor, and Eric Horvitz. 2015. A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 379–386.
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[14] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Benjamin Klein, Lior Wolf, and Yehuda Afek. 2015. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4840–4848.
[16] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 33.
[17] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.
[18] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*. 2204–2212.
[19] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.
[20] Ronald A Rensink. 2000. The dynamic representation of scenes. *Visual cognition* 7, 1-3 (2000), 17–42.
[21] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*. 843–852.
[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.
[23] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3156–3164.
[24] Dong Wang, Wei Cao, Jian Li, and Jieping Ye. 2017. DeepSD: Supply-Demand Prediction for Online Car-hailing Services using Deep Neural Networks. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 243–254.
[25] Hua Wei, Yuandong Wang, Tianyu Wo, Yaxiao Liu, and Jie Xu. 2016. ZEST: A Hybrid Model on Predicting Passenger Demand for Chauffeured Car Service. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2203–2208.
[26] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*. 802–810.
[27] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *arXiv* 1603 (2016).
[28] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.. In *ICML*, Vol. 14. 77–81.
[29] Qiuyuan Yang, Zhiqiang Gao, Xiangjie Kong, Azizur Rahim, Jinzhong Wang, and Feng Xia. 2015. Taxi Operation Optimization Based on Big Traffic Data. In *UIC-ATC-ScalCom*. IEEE, 127–134.
[30] Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. 2016. Review networks for caption generation. In *Advances in Neural Information Processing Systems*. 2361–2369.
[31] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[32] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *SIGSPATIAL*. ACM, 92.