Deep Multi-Scale Convolutional LSTM Network for Travel Demand and Origin-Destination Predictions

Kai-Fung Chu, Albert Y.S. Lam, and Victor O.K. Li

***Abstract*—Advancements in sensing and Internet of Things (IoT) technologies generate huge amount of data. Mobility on Demand (MoD) service benefits from the availability of big data in intelligent transportation system. Given the future travel demand or origin-destination (OD) flows prediction, service providers can pre-allocate unoccupied vehicles to the customers' origins of service to reduce waiting time. Traditional approaches on future travel demand and OD flows predictions rely on statistical or machine learning methods. Inspired by deep learning techniques for image and video processing, through regarding localized travel demands as image pixels, a novel deep learning model called Multi-Scale Convolutional Long Short-Term Memory network (MultiConvLSTM) is developed in this work. Rather than using traditional OD matrix which may lead to loss of geographical information, we propose a new data structure, called OD Tensor to represent OD flows, and a manipulation method, called OD Tensor Permutation and Matricization, is introduced to handle the high dimensionality features of OD Tensor. MultiConvLSTM considers both temporal and spatial correlations to predict the future travel demand and OD flows. Experiments on real-world New York taxi data of around 400 million records are performed. Our results show that MultiConvLSTM achieves the highest accuracy in both one-step and multiple-step predictions and it outperforms the existing methods for travel demand and OD flow predictions.**

***Index Terms*—travel demand prediction, origin-destination prediction, deep learning, multi-scale convolutional LSTM network, origin-destination tensor.**

I. Introduction

NOWADAYS, we inevitably need to travel to different places for various activities and thus an efficient intelligent transportation system is important and valuable in a smart city. Different from the public transport with established routes and schedules, vehicles for hiring, like taxis, provide private transportation services that best fit the passengers' routes and schedules. Traditionally, taxi drivers need to roam the streets to look for passengers by chance in urban city and passengers also have to wait on the streets for taxis by chance. In recent years, a more traveler-centric transportation system – Mobility on Demand (MoD), has emerged, such as Uber [2] and Lyft [3], which allows passengers to actively submit travel requests to specify their pickup locations. By incorporating autonomous driving into MoD, a new form of transportation – Autonomous MoD (AMoD), which utilizes autonomous vehicles as the transportation service carriers with more service options, has been proposed [4], [5]. However, the common shortcoming of all these transportation systems is that, after submitting their travel requests, passengers have to wait until the assigned vehicles arrive at the dedicated pickup locations. The waiting time varies with the distance between the original location of the assigned vehicle and the pickup point. If the vehicle supply and service demand are imbalanced, there may be no vehicles available nearby and a faraway vehicle may need to be assigned, resulting in even longer waiting time for the passenger. The survey done at San Francisco [6] shows that most MoD services require less than 10 minutes waiting time while taxi calling services need 10–20 minutes. The situation becomes much worse during rush hours or in congested areas. If vehicles can be pre-allocated to those areas with high travel demand, the waiting time will be much reduced and the service quality will then be improved. Moreover, unoccupied vehicles on the road can be reduced, thus reducing energy consumption and air pollution from vehicle emission. Hence, the location and time information of travel demand in the city is valuable for developing a re-balancing strategy.

Advancements in sensing and Internet of Things (IoT) technologies enable analysis of our physical world with huge amount of data. These technologies have been widely used in transportation system, such as global positioning system (GPS) for vehicle localization and navigation [7], vehicular communication of road map data for autonomous driving [8], traffic flow estimation by inductive loop or camera to count and classify vehicles for traffic management [9], etc. With big data, the traditional technology-driven transportation system is being transformed into data-driven intelligent transportation system [10]. Many vehicles are also equipped with various sensors and communication units to collect information. This facilitates many new applications, such as autonomous intersection management [11], dynamic lane reversal [12], and vehicular energy network [13]. The availability of traffic data can also be used to construct static Origin-Destination (OD) flows [14]. However, future dynamic travel demand and OD flows are generally unknown. Unlike weather prediction which can be more easily modeled due to physical laws, predicting travel demand and OD flows is more challenging since there are no clear rules available for guiding the complicated human movements. Hence, we need an accurate prediction method for predicting both the travel demand and OD flows.

In this paper, we investigate the incorporation of both temporal and spatial correlations from historical transportation and related data to predict the future travel demand and OD flows. To do this, we "visualize" the travel demands in the city at a particular moment as an image, where each small area is considered as a pixel and its travel demand as the cor-

A preliminary version of this paper was presented in [1].

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, and also with the Shenzhen Institute of Research and Innovation, The University of Hong Kong, Hong Kong (e-mail: kfchu@eee.hku.hk; ayslam@eee.hku.hk; vli@eee.hku.hk).

responding pixel value. For OD flows, we propose a new data structure, called OD Tensor, that stores the OD flows without losing any geographical information. We also introduce a method called permutation and matricization to manipulate OD Tensor to reduce the dimensionality. Inspired by deep learning technique for image and video processing, a model called Multi-Scale Convolutional Long Short-Term Memory network (MultiConvLSTM) is proposed for travel demand and OD flow predictions. MultiConvLSTM is able to handle either historical demand data or matricized OD Tensor as the input data. It performs convolution between the input and weights, and stores the extracted features as in LSTM. Historical data of different spatial resolutions and metadata, such as time and weather information, can be used in MultiConvLSTM to further enhance the prediction accuracy. Experiments are conducted to evaluate the proposed deep learning model with six and a half years (Jan. 2009 – Jun. 2015) of real world taxi data, which contain around 400 million records for a selected region of Manhattan, New York. Experimental results show that the proposed methods outperform the existing methods. The main contributions of this paper can be summarized as follows:

1. A new deep learning model, called Multi-Scale Convolution LSTM (MultiConvLSTM) Network, is developed and applied for predicting travel demand and OD flows.
2. We propose a new data structure, called OD Tensor, to represent OD flows that can preserve most of the geographical information of the OD flows.
3. A manipulation method of OD Tensor, called OD Tensor Permutation and Matricization, is introduced to handle the high-dimension features of OD Tensor.
4. Extensive experiments on real-world New York taxi data of around 400 million records are performed. Our results show that MultiConvLSTM achieves the highest accuracy in both one-step and multiple-step predictions and it outperforms the existing methods for travel demand and OD flow predictions.

The rest of this paper is organized as follows. Section II reviews the existing methods for travel demand and OD flows predictions. Section III defines the prediction problems and presents our proposed data structure and data processing methods for the data handling. We present our proposed deep learning model in Section IV and experiments on real-world trip data is given in Section V. Finally, Section VI concludes this paper.

II. Related Work

Future travel information is important to transportation service providers for improving their quality of service. Many researchers have paid effort on the related traffic data prediction problems. Two types of traffic data are important: travel demand and OD flows. The former is defined as the total number of travel requests in specific pickup location and time. Transportation service providers can balance the supply and demand by considering future travel demand. The latter represents the traffic flows between origins and destinations. Travel strategies and operational efficiency can be optimized with predicted OD flows. While various prediction methods have been specifically proposed to predict travel demand and OD flows individually, deep learning has been generally applied to solve many traffic problems. In this section, we review the existing prediction methods for travel demand and OD flows. Deep learning approaches for solving various traffic problems are also discussed.

A. Travel Demand Prediction

Travel demand is valuable for reducing the service waiting time. Various prediction methods have been proposed to predict travel demand using historical travel demand. Autoregressive integrated moving average (ARIMA) [15] is well-known for addressing the short-term time-series prediction problem and it was used to predict the travel demand in [16]. [17] presented a time-series prediction technique based on time-varying Poisson model and ARIMA with improved accuracy. [18] modeled the demand data using time-series analysis and further improved the accuracy by the multi-level clustering technique. [19] analyzed the maximum predictability of taxi demand with the entropy of taxi demand sequence by considering both randomness and the temporal correlation. Three different prediction algorithms, namely, the probability-based Markov predictor, the sequence-based Lempel-Ziv-Welch predictor, and machine learning based Neural Network predictor, were developed to validate the maximum predictability theory [19]. In [20], historical taxi trajectory data were analyzed to identify hotspots in different time slots and regions, and a hotness score could be determined by an exponentially weighted moving average. The top k hot-spots would be recommended to a taxi driver based on the hotness score and the driver's location.

B. OD Flow Prediction

OD flows are commonly represented in the form of OD matrix. The row and column indexes stand for origin and destination, respectively, and each entry quantifies the travels of the corresponding origin-destination pair. OD flows can be estimated through static and dynamic predictions. Many works estimated the static OD matrix with other data sources, such as vehicular traffic size [14], [21], [22], communication data [23], [24], [25] and GPS [26], [27], [28], [29]. For dynamic OD flow prediction, temporal correlation of historical data was considered and modeling methods, such as least-square modeling [30] and Kalman-filter approach [31], were applied to predict the future OD flows. Stochastic approximation algorithms, such as Simultaneous Perturbation Stochastic Approximation (SPSA) [32] and cluster-wise SPSA [33], were also shown to be able to find good solutions based on gradient approximation. However, predicting the dynamic OD flows for specific time and location is not trivial due to the high dimensionality of OD flows. Hence, dimension reduction and OD matrix approximation based on Principal Component Analysis were developed to address the dimensionality issue [34]. [35] proposed an algorithm based on Nonnegative Matrix Factorization and Autoregressive to decompose the OD matrix. However, all these methods based on OD matrix may suffer from the loss of geographical information. An OD flow representation without geographical information loss is desired for prediction. In [36], OD is modeled by a four-order tensor which consists of four attributes: origin, destination, vehicle type, and time. Through tensor decomposition, the time factor matrix is extracted and used to predict the future OD flows. With the additional vehicle type and time information, the prediction accuracy is improved when compared to ordinary OD matrix. However, tensor decomposition may still result in data loss. There are no existing efficient prediction algorithms which can process the high dimensional tensor data without data loss.

C. Deep Learning for Transportation

Due to its recent success in applications to various disciplines [37], [38], [39], [40], [41], deep learning [42], [43] has been applied to address many transportation problems. For example, [44] and [45] used a deep neural network to predict traffic flow with high accuracy. [46] applied the Restricted Boltzmann Machine (RBM) and Recurrent Neural Network (RNN) to predict traffic congestion evolution. Lane detection [47] and autonomous driving [48], [49] were also facilitated by deep learning. A novel deep learning model called Diffusion Convolutional Recurrent Neural Network was proposed to predict traffic speed based on spatial and temporal correlations [50]. In particular, [51] and [52] used RNN to predict taxi demand and [53] developed a multi-pattern data fusion model that divided the dataset into different clusters based on the data pattern and used a deep belief network for each cluster for bus passenger flow forecasting. However, the spatial correlation may be overlooked if RNN is solely used to handle the sequential nature of taxi demand. Thus we need a model that considers both the temporal and spatial correlations of the historical data to further improve the future travel demand and OD flows prediction. In the preliminary version of this work [1], we proposed MultiConvLSTM, which explores the temporal and spatial correlations of historical travel demand data to predict the future travel demand. A parallel work [54] employs a similar concept but with different architecture to predict travel demand. However, having only travel demand predicted may be insufficient for many services. OD flows are also important in an intelligent transportation system. To the best of our knowledge, there is no deep learning approach that predicts OD flows using temporal and spatial correlations of historical data. A deep learning approach which simultaneously employs temporal and spatial correlations to predict both travel demand and OD flows is important for intelligent transportation system.

III. Problem Formulation

In this section, we define two traffic data prediction problems, i.e., travel demand prediction and OD flows prediction, and present the corresponding data pre-processing methods.

A. Travel Demand Prediction

1) Problem definition: We aim to predict the travel demand for a specific region $\mathcal{A}$ and time period $\mathcal{T}$, which are equally divided into $m \times n$ grid cells and T time intervals, respectively. Each interval has a duration of τ_g, typically in the order of minutes. The travel demand θ_{ij}^t is defined as the total number of travel requests originated at the grid cell $(i,j) \in \mathcal{A}$ in the time interval $t \in \mathcal{T}$. While historical travel demands are the major source of information for the prediction, some other metadata, such as the day, time, and weather, may also have correlations with travel demand. For example, higher demand is expected during the rush hours in working days. Thus they may optionally be considered to further improve the prediction. The objective is to predict the demand of each grid cell $(i,j) \in \mathcal{A}$ for the future $\tau \geq 1$ intervals. Suppose we are in the T_cth interval. We predict the travel demands, $\hat{\Theta} = \{\theta_{ij}^{T_c+1}, \ldots, \theta_{ij}^{T_c+\tau}, \forall (i,j) \in \mathcal{A}\}$, based on the historical travel demand $\Theta = \{\theta_{ij}^t, \forall (i,j) \in \mathcal{A}, t \in 1, 2, \ldots, T_c\}$ and other optional metadata.

2) Pre-processing: We model each travel demand data record $r \in \mathcal{R}$ by the 3-tuple $\langle t^r, o^r, d^r \rangle$, where $t^r \in \mathcal{T}$ is the time of submitting the request r. $o^r \in \mathcal{A}$ and $d^r \in \mathcal{A}$ are the origin and destination of travel request r, respectively. To facilitate the deep learning process, we group r's based on each interval, resulting in a set of matrices $\Theta = \{(\theta_{ij}^t), t \in \mathcal{T}\}$. To do this, we set θ_{ij}^t to be the total number of r which occurs in t with the same origin (i,j), i.e., $\theta_{ij}^t = |\{r | o^r \in (i,j), t^r \in t\}|$. For grid cell (i,j) and time interval t with no travel requests, we have $\theta_{ij}^t = 0$.

B. OD Flow Prediction

1) Problem definition: An OD flow captures the number of travels for specific origin-destination pairs in a particular time period. An OD matrix is commonly used to represent OD flows. Consider a region $\mathcal{A}$ equally divided into $m \times n$ grid cells. The OD flows originated at time interval $t \in \mathcal{T}$ are represented by $\mathcal{M}^t = (a_{od}^t) \in \mathbb{R}^{N \times N}$, where $N = m \times n$ is the total number of grid cells and a_{od}^t denotes the number of flows from origin $o \in \mathcal{A}$ departed in $t \in \mathcal{T}$ to destination $d \in \mathcal{A}$. Suppose we are in interval T_c. Our objective is to predict the OD flows $\{\hat{\mathcal{M}}^{T_c+1}, \ldots, \hat{\mathcal{M}}^{T_c+\tau}\}$, for the future $\tau \geq 1$ intervals, based on the historical OD flows $\{\mathcal{M}^1, \ldots, \mathcal{M}^{T_c}\}$ and other optional metadata.

Using OD matrix as the data structure to represent OD flows may result in degradations of prediction accuracy. For $m, n > 1$, the grid cells are naturally arranged as a two-dimensional array retaining the actual geographical information. Each grid cell generally has certain spatial correlation with its neighboring grid cells since neighborhoods are likely to be of similar land-use and people around may share similar travel behaviors. To construct an OD matrix, the two-dimensional array $(m \times n)$ is first vectorized into a one-dimensional vector $(N \times 1)$ and then we form the corresponding $N \times N$ OD matrix. However, such vectorization of two-dimensional array leads to loss of geographical information in the resultant OD matrix. As will be discussed in Section IV, our deep learning model includes the features of Convolutional Neural Network (CNN), which is inspired by human visual system and is commonly applied to analyze visual imagery. The model can achieve higher accuracy if the input image has stronger visual pattern. We can see that OD matrix may degrade the prediction due to the absence of visualization of geographical information. To overcome the shortcomings of the OD matrix, we propose a new data structure, called OD Tensor, to represent OD flows. We will also process the OD Tensor using permutation and matricization to retain the visualization of spatial correlation for higher prediction accuracy. The matricized OD Tensor can be applied to our deep learning model for prediction.

2) OD Tensor: A tensor is a multidimensional array, whose order is the dimensionality. For instance, a matrix is a second-order tensor since two indexes are used to represent a matrix. An OD Tensor is a general data structure that can represent OD flows without geographical information loss.[1] Let $\mathcal{O} \subseteq \mathcal{A}$ and $\mathcal{D} \subseteq \mathcal{A}$ be the sets of origin and destination, respectively. The OD Tensor at time interval t is given as $\Omega^t = (\omega^t_{(i^o,j^o),(i^d,j^d)}) \in \mathbb{R}^{m\times n\times m\times n}$, where $(i^o, j^o) \in \mathcal{O}$ and $(i^d, j^d) \in \mathcal{D}$. $\omega^t_{(i^o,j^o),(i^d,j^d)}$ denotes the total number of flows from (i^o, j^o) departed in t to (i^d, j^d). Since OD Tensor can retain spatial relationship of OD flows with its four indexes representing the geographical locations of both the origin and destination, predicting OD flows from OD Tensor is a better option. However, OD Tensor is hard to process by most existing deep learning models due to its high dimensionality. Hence, we apply permutation and matricization on OD Tensor to reduce its order while avoiding the loss of geographical information.

3) Permutation and matricization: The permutation and matricization process can be considered as a function $f : \mathbb{R}^{m\times n\times m\times n} \rightarrow \mathbb{R}^{m^2\times n^2}$. Recall that the indexes of an OD Tensor are arranged as $(i^o, j^o), (i^d, j^d)$. In this process, for an input tensor $\Omega^t = (\omega^t_{(i^o,j^o),(i^d,j^d)})$, we first pair up the indexes of the corresponding origins and destinations, i.e., permuting the structure $(i^o, j^o), (i^d, j^d)$ into $(i^o, i^d), (j^o, j^d)$ such that we get the permuted OD Tensor $\tilde{\Omega}^t = (\tilde{\omega}^t_{(i^o,i^d),(j^o,j^d)}) \in \mathbb{R}^{m\times m\times n\times n}$. Then we matricize the permuted OD Tensor by combining (i^o, i^d) and (j^o, j^d) into k^i and k^j, respectively, where $k^i = i^o + i^d \times m$ and $k^j = j^o + j^d \times n$. The resultant matricized OD Tensor at time interval t is given as $\bar{\Omega}^t = (\bar{\omega}^t_{k^i,k^j}) \in \mathbb{R}^{m^2\times n^2}$. The process of converting Ω^t to ($\tilde{\Omega}^t$ and then to) $\bar{\Omega}^t$ can reduce the order of OD Tensor from 4 to 2, leading to an $m^2\times n^2$ matrix. Although the number of elements in $\bar{\Omega}^t$ is the same as that in the original OD matrix $\mathcal{M}_t$ which has $mn \times mn$ elements, the geographical information in OD Tensor is reserved. To see this, we examine the simple example given in Fig. 1, which represents a region in terms of 2×3 $(m\times n)$ grid cells with the arrows indicating three flows to cell 2 from cells 1, 3 and 5, respectively. The OD flows of this region form a tensor $\Omega^t = \left(\omega^t_{(i^o,j^o),(i^d,j^d)}\right) \in \mathbb{R}^{(2\times 3\times 2\times 3)}$. The tensor $\omega^t_{(i^o,j^o),(i^d,j^d)}$ denotes the total number of OD flows from grid cell (i^o, j^o) to (i^d, j^d). The tensor of the example is:

$$\omega^t_{(i^o,j^o),(i^d,j^d)} = \begin{cases} 1, & \text{if } (i^o, j^o), (i^d, j^d) = (1, 0), (0, 1), \\ 1, & \text{if } (i^o, j^o), (i^d, j^d) = (1, 1), (0, 1), \\ 1, & \text{if } (i^o, j^o), (i^d, j^d) = (1, 2), (0, 1), \\ 0, & \text{otherwise.} \end{cases}$$

Then we perform permutation and matricization on the tensor. For permutation, $\Omega^t = \left(\omega^t_{(i^o,j^o),(i^d,j^d)}\right) \in \mathbb{R}^{(2\times 3\times 2\times 3)}$ is transformed to $\tilde{\Omega}^t = \left(\tilde{\omega}^t_{(i^o,i^d),(j^o,j^d)}\right) \in \mathbb{R}^{(2\times 2\times 3\times 3)}$ by permuting the structure $(i^o, j^o), (i^d, j^d)$ to $(i^o, i^d), (j^o, j^d)$. The resulting permuted tensor is:

$$\omega^t_{(i^o,i^d),(j^o,j^d)} = \begin{cases} 1, & \text{if } (i^o, i^d), (j^o, j^d) = (1, 0), (0, 1), \\ 1, & \text{if } (i^o, i^d), (j^o, j^d) = (1, 0), (1, 1), \\ 1, & \text{if } (i^o, i^d), (j^o, j^d) = (1, 0), (2, 1), \\ 0, & \text{otherwise.} \end{cases}$$

For matricization, $\tilde{\Omega}^t = \left(\tilde{\omega}^t_{(i^o,i^d),(j^o,j^d)}\right) \in \mathbb{R}^{(2\times 2\times 3\times 3)}$ is matricized to $\bar{\Omega}^t = \left(\bar{\omega}^t_{k^i,k^j}\right) \in \mathbb{R}^{(2^2\times 3^2)}$ by combining (i^o, i^d) and (j^o, j^d) into k^i and k^j, respectively, where $k^i = i^o + i^d$ and $k^j = j^o + j^d$. The matricized OD Tensor is:

$$\omega^t_{(k^i,k^j)} = \begin{cases} 1, & \text{if } (k^i, k^j) = (1, 3), \\ 1, & \text{if } (k^i, k^j) = (1, 4), \\ 1, & \text{if } (k^i, k^j) = (1, 5), \\ 0, & \text{otherwise.} \end{cases}$$

Fig. 3 gives the output of the permutation and matricization process, $\bar{\Omega}^t = \left(\bar{\omega}^t_{k^i,k^j}\right) \in \mathbb{R}^{(2^2\times 3^2)}$, which is a 4×9 $(m^2 \times n^2)$ matrix. For comparison, Fig. 2 gives the ordinary OD matrix $\mathcal{M}^t$ corresponding to the example in Fig. 1. We can see that $\mathcal{M}^t$ cannot illustrate the geographical movement of the three flows while $\bar{\Omega}^t$ can. At the macroscopic level, Fig. 3 is a 2×3 matrix of blocks, each of which also contains 2×3 elements. The number of blocks is equal to the number of grid cell, in which each block represents the corresponding grid cell. Sum of each block indicates the total number of flows with destination at the corresponding grid cell. For example, sum of all elements in block 2, i.e., $\bar{\omega}^t_{0,3}+\bar{\omega}^t_{1,3}+\bar{\omega}^t_{0,4}+\bar{\omega}^t_{1,4}+\bar{\omega}^t_{0,5}+\bar{\omega}^t_{1,5} = 0+1+0+1+0+1 = 3$, indicates three flows to grid cell 2 (c.f. Fig. 1). That is, if the destinations of the three flows are grid cell 2, the sum of block 2 will be three. At the microscopic level, within each block, each element represents the corresponding grid cell and its value indicates the number of flows originated from the corresponding grid cell. For instance, $\bar{\omega}^t_{1,3}$ shows that there is one flow from grid cells 1 to 2 (c.f. Fig. 1). The sum of $\bar{\omega}^t_{1,0}$, $\bar{\omega}^t_{3,0}$, $\bar{\omega}^t_{1,3}$, $\bar{\omega}^t_{3,3}$, $\bar{\omega}^t_{1,6}$ and $\bar{\omega}^t_{3,6}$ also indicates the total number of flows originated from grid cell 1. That is, if the origin of the three flows are the bottom three grid cells, the ones appearing in block 2 will be at the bottom three entries of block 2. The patterns of both origin and destination of the OD flows are preserved in the matricized OD Tensor. From this, we can visually infer the OD flows from cells 1, 3 and 5 to cell 2 shown in Fig. 1 accordingly. But the three 1's in the OD matrix cannot visually provide this geographical information if

[1] We only consider land transportation in this work and thus it is sufficient to divide $\mathcal{A}$ into two-dimensional grid cells. But OD Tensor is extensible to regions with dimensions higher than two. For instance, we need higher order OD Tensor for aircrafts that fly in three-dimensional space.

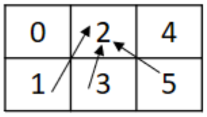


Fig. 1. Region with 6 grid cells.

o \ d	0	1	2	3	4	5
0						
1			1			
2						
3			1			
4						
5			1			

Fig. 2. Example of OD matrix of the region.

k^i \ k^j	0	1	2	3	4	5	6	7	8
0		~~block 0~~			~~block 2~~			~~block 4~~	
1				1	1	1			
2		~~block 1~~			~~block 3~~			~~block 5~~	
3									

Fig. 3. Example of matricized OD Tensor of the region.

the grid cell arrangement is not given. Hence, the permuted and matricized OD Tensor $\bar{\Omega}^t$ can retain geographical information and visualize the OD flows. It can be utilized in our proposed deep learning model directly.

4) Pre-processing: Similar to travel demand prediction, we need to convert the travel records $\mathcal{R}$ into OD Tensor $\bar{\Omega}$ for further processing. Recall that each $r \in \mathcal{R}$ is modeled by the 3-tuple $\langle t^r, o^r, d^r \rangle$, where $t^r \in \mathcal{T}$ is the departure time of r. $o^r \in \mathcal{A}$ and $d^r \in \mathcal{A}$ are the origin and destination of travel record r, respectively. To transform $\mathcal{R}$ to $\bar{\Omega}_t$, we group r's based on each interval, resulting in a set of OD Tensors $\{\Omega^t, t \in \mathcal{T}\}$, where $\Omega^t = (\omega^t_{(i^o,j^o),(i^d,j^d)})$. We set $\omega^t_{(i^o,j^o),(i^d,j^d)}$ to be total number of r which happened in t with the same origin and destination, i.e., $\omega^t_{(i^o,j^o),(i^d,j^d)} = |\{r | o^r \in (i^o, j^o), d^r \in (i^d, j^d), t^r \in t\}|$. For example, consider three travel records, namely, ⟨1:03pm, cell 1, cell 2⟩, ⟨1:07pm, cell 1, cell 3⟩, and ⟨1:14pm, cell 1, cell 4⟩. If the length of a time-step is equal to 10 minutes, then we have $\omega^{\text{1:00pm}}_{(1,0),(0,1)} = 1$, $\omega^{\text{1:00pm}}_{(1,0),(1,1)} = 1$, $\omega^{\text{1:10pm}}_{(1,0),(0,2)} = 1$. For grid cell pair $((i^o, j^o), (i^d, j^d))$ and time interval t with no traffic flows, we have $\omega^t_{(i^o,j^o),(i^d,j^d)} = 0$. Ω^t is premuted to $\tilde{\Omega}^t$, which is then matricized to $\bar{\Omega}^t$ following the methods stated in Section III-B3.

IV. Multi-Scale Convolutional LSTM Network

We incorporate both the temporal and spatial correlations of historical data to predict the future travel demand and OD flows. The basic unit used in our model is ConvLSTM [55], which performs convolution between the input and weights, and stores the extracted feature as in LSTM. However, the spatial correlation captured in ConvLSTM is of short range. Increasing the kernel size may improve the accuracy but requires higher computational power. Moreover, since the output resolution is the same as that of the input, the pooling layer is not required in ConvLSTM. The resolution of data remains the same throughout the network. However, since the inclusion of the pooling layer can improve the performance of CNN, incorporation of different resolution data within the network may further improve the prediction accuracy. Therefore, we consider a multi-scale network and propose MultiConvLSTM for the travel demand and OD predictions. In this section, we first review two related popular neural network architectures, namely, RNN and CNN, and then discuss our proposed deep learning model.

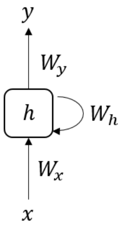


Fig. 4. RNN architecture.

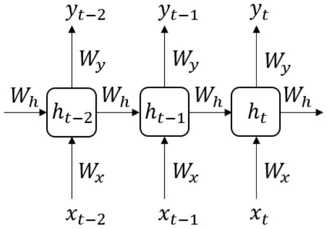


Fig. 5. Unfolded RNN architecture.

A. Recurrent Neural Network

Predicting travel demand and OD flows is similar to sequence prediction and thus RNN may be a good model to handle time-series data in travel demand prediction. It makes use of a series of historical data as input and it stores information extracted from previous inputs in its memory for later prediction. This is achieved by recurrently computing all elements of the input sequence to output the results. A loop is used to pass the information from one time-step to the next. Fig. 4 shows a typical structure of RNN with input x, hidden state h, output y, and the corresponding weights W_x, W_h, W_y. All these input, hidden state, output and weights are usually scalars or vectors in RNN. If input data are given in the matrix form, they will be flattened to vectors before being applied to the model. In practice, the infinite loop can be unfolded to a finite number of input time-steps as shown in Fig. 5. The hidden state h_t at time t can be considered as the memory of the network, computed as follows:

$$h_t = f(W_h h_{t-1} + W_x x_t), \tag{1}$$

where f is an activation function, such as $tanh$ or $ReLU$.

A popular kind of RNN is Long Short-Term Memory (LSTM) [56], which is capable of learning long-term dependencies. The main difference is the inner structure of the repeating cell. Recall that inside the cell of standard RNN, there is a single $tanh$ or $ReLU$ layer only. However, in LSTM, a memory cell state, c_t, is added and there are four layers interacting together as shown in Fig. 6. A "Forget gate layer", f_t, is used to decide which information from the previous cell should be forgotten. An "Input gate layer", i_t, is for deciding which part of the information from the current input and hidden state should be stored. We use a $tanh$ layer to

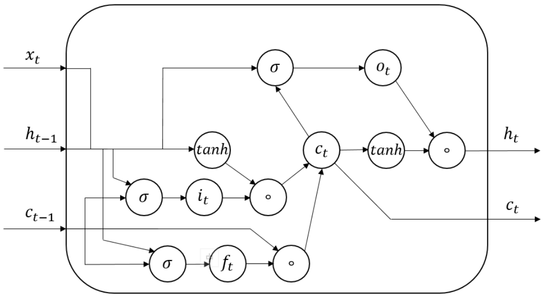


Fig. 6. LSTM architecture.

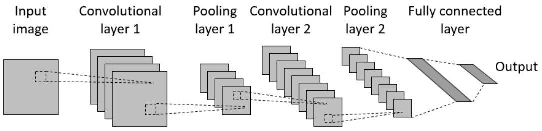


Fig. 7. CNN architecture.

extract the information from h_{t-1} and x_t, and combined with i_t, f_t and c_{t-1} to get the new cell state c_t. An "Output gate layer", o_t, acts as a filter to extract the information from the cell state to produce the output. More specifically, f_t, i_t, c_t, o_t, and h_t are updated as follows:

$$f_t = \sigma(W_{fc} \circ c_{t-1} + W_{fh}h_{t-1} + W_{fx}x_t + b_f), \tag{2}$$

$$i_t = \sigma(W_{ic} \circ c_{t-1} + W_{ih}h_{t-1} + W_{ix}x_t + b_i), \tag{3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c), \tag{4}$$

$$o_t = \sigma(W_{oc} \circ c_t + W_{oh}h_{t-1} + W_{ox}x_t + b_o), \tag{5}$$

$$h_t = o_t \circ tanh(c_t), \tag{6}$$

where $\circ$ denotes the Hadamard product.

B. Convolutional Neural Network

Since neighboring locations may share similar travel demand patterns, adjacent grid cells may have certain spatial correlations. Handling the travel demand with RNN-LSTM alone may overlook this spatial information. One popular architecture to capture such high dimensional spatial information is CNN, which has been applied with great success to many challenging computer vision problems, such as object detection and recognition. CNN consists of a number of convolutional and pooling layers followed by a fully connected layer at the end of the network. A convolutional layer is used to extract high-dimensional features. It has k kernels of size n-by-n as filters, each of which computes the dot product between the weights of the kernel and a small part of the input until it slices through the whole image. A pooling layer performs subsampling of the image, where the max operation is a typical pooling method that selects the maximum value for each m-by-m pixel. The fully connected layer computes the output based on the extracted features from the convolutional and pooling layers. Fig. 7 shows a typical CNN for image classification.

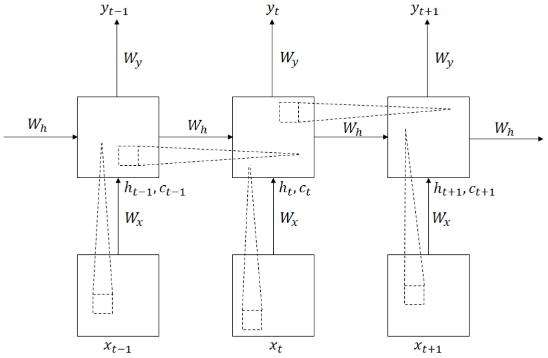


Fig. 8. ConvLSTM architecture.

C. Convolutional LSTM Network

Convolutional LSTM Network (ConvLSTM) [55] contains hidden state and cell state units and preserves the ability of LSTM to store the extracted information. In LSTM, the input and hidden states are multiplied by the weights. However, in ConvLSTM, instead of multiplying input and hidden states by the weights directly, we perform convolution with the weights as well. Fig. 8 shows the architecture of ConvLSTM and the key equations are as follows:

$$f_t = \sigma(W_{fc} \circ c_{t-1} + W_{fh} * h_{t-1} + W_{fx} * x_t + b_f), \tag{7}$$

$$i_t = \sigma(W_{ic} \circ c_{t-1} + W_{ih} * h_{t-1} + W_{ix} * x_t + b_i), \tag{8}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{ch} * h_{t-1} + W_{cx} * x_t + b_c), \tag{9}$$

$$o_t = \sigma(W_{oc} \circ c_t + W_{oh} * h_{t-1} + W_{ox} * x_t + b_o), \tag{10}$$

$$h_t = o_t \circ tanh(c_t), \tag{11}$$

where $\circ$ and $*$ denote the Hadamard product and the convolution operator, respectively. In this way, the spatial correlation of data can also be considered in the prediction. Note that ConvLSTM can be reduced to RNN-LSTM by setting the kernel size to 1×1.

D. Multi-Scale Network

The pooling layer is important for CNN to extract the features from the input as different features can be extracted from adjacent grid cells in different resolutions using a fixed size of kernel. However, the pooling function is missing in ConvLSTM since the output of ConvLSTM has the same resolution as the input and the data flow in ConvLSTM is similar to that in LSTM. Therefore, without the pooling layer, the convolution in ConvLSTM can only account for short-range spatial correlation and only data of the same resolution can be considered. If lower-resolution data is inputted to the model, long-range spatial correlation can only be considered by the kernel with same size. To address this issue, we convert the model into a multi-scale network by inputting lower-resolution versions of the same set of data [57]. A pixel in lower resolution data is combined with several nearby pixels of higher resolution data. Thus, the pixels in high-resolution data are condensed resulting in fewer pixels for the low-resolution data. Hence, with the same kernel size, more pixels can be considered, which works like a pooling layer. Let λ_s be the resolution of the input data, for $s = 1, 2, \dots, S$, where s and S are the resolution index and the highest resolution index, respectively. Let $X^{\lambda_{s-1}}$ and $Y^{\lambda_{s-1}}$ be the lower resolution data sets of X^{λ_s} and Y^{λ_s}, respectively, where $X^{\lambda_s} = \{x_1^{\lambda_s}, \dots, x_{T-1}^{\lambda_s}, x_T^{\lambda_s}\}$ is the historical data and $Y^{\lambda_s} = \{y_{T+1}^{\lambda_s}\}$ is the future value. We aim to predict future demand, Y^{λ_S}, with input data in different resolutions, $\{X^{\lambda_1}, X^{\lambda_2}, \dots, X^{\lambda_S}\}$. The input data are prepared by duplicating the data in origin resolution, X^{λ_S}, and scaling down to different resolutions. The input data in different resolutions, $\{X^{\lambda_1}, X^{\lambda_2}, \dots, X^{\lambda_S}\}$, are then input to the corresponding layers of the model. The prediction of Y^{λ_S} is computed by:

$$\widehat{Y}^{\lambda_S} = ConvLSTM(X^{\lambda_S}, up(\widehat{Y}^{\lambda_{S-1}})), \tag{12}$$

where $ConvLSTM(\cdot)$ and $up(\cdot)$ are the ConvLSTM network and upsampling operations, respectively.

E. Multi-Scale Convolutional LSTM

We develop our deep learning model MultiConvLSTM for travel demand and OD predictions by integrating ConvLSTM with the multi-scale network and Fig. 9 shows its architecture. The basic unit in the model is ConvLSTM, and there are multiple ConvLSTMs for different resolutions. They form a hierarchical structure such that the output of lower-resolution ConvLSTM is fed to the input of the next level for higher resolution. Metadata such as time and weather information can also be input and a fully connected neural network (FCNN) is used to convert the metadata into predicted output denoted by ψ^{λ_S}, of the same dimension as the predicted demand, concatenated with the input data. The function of FCNN is to convert the potentially useful metadata, such as time and weather information, into an $m \times n$ matrix ψ^{λ_S}. It is in the same format as $x_t^{\lambda_s}$, where $x_t^{\lambda_S}$ is the historical data of travel demand or matricized OD Tensor in λ_S resolution. ψ^{λ_S} is purposely designed to share the same format with $x_t^{\lambda_S}$ such the it can be inputted to the final layer of MultiConvLSTM. Such metadata may contain some hidden information that correlates with travel demand and OD flows. For example, traffic will be high during rush hours as people go to work in the morning and return home in the evening during weekdays. They may be less willing to go out if the weather is bad.

In order to overcome the degradation of such a deep network, we employ the technique from deep residual learning [58], in which the average of ψ^{λ_S} and $up(\hat{Y}^{\lambda_{S-1}})$ (i.e. $\frac{\psi^{\lambda_S}}{2} + \frac{up(\hat{Y}^{\lambda_{S-1}})}{2}$) are added to the output. ψ^{λ_S} and $up(\hat{Y}^{\lambda_{S-1}})$ are expected to have similar values as the ground truth during the training process. This allows the ConvLSTM in the last layer to model the residual, which is easier to be optimized in the training process, instead of the original predicted future demand. Although MultiConvLSTM can only predict one future time-step in each execution, subsequent time-steps can be predicted by using historical data and the recently predicted demand as the input.

V. Experiments

We evaluate the proposed MultiConvLSTM deep learning model using the New York taxi dataset [59], which contains the pickup and dropoff times, and pickup and dropoff locations of taxi services recorded in New York from January 2009 to June 2015. Every record in the dataset represents a taxi service with origin, destination, pickup, and dropoff time. The time and location information is precise to the nearest second and six decimal places of latitude and longitude format, respectively. Due to the extremely high resolution of the New York dataset, our general model cannot process these raw data directly. Therefore, we pre-process the raw data into the format of travel demand or OD Tensor for the use of our proposed model which is in lower resolution according to the Section III-A2 and III-B4, respectively. There are around 400 million taxi traveling records. The duration of one time-step is set to 10 minutes. We extract 80% of the data as training data while the rest are for testing. The experiments are divided into two parts for travel demand and OD flows predictions.

Three commonly used performance metrics are used in our study: root mean square error (RMSE), mean absolute error (MAE) and symmetric mean absolute percentage error (SMAPE), which are computed as follows:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2} \tag{13}$$

$$MAE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}|y - \hat{y}| \tag{14}$$

$$SMAPE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y_i}|}{y_i + \hat{y_i} + c}, \tag{15}$$

where y_i and $\hat{y}_i$ are the ground truth and predicted values, respectively, and c is a positive constant to avoid division by zero as $y_i + \hat{y}_i$ in the denominator may equal to zero. A common practice is to set $c = 1$, as indicated in [60]. Hence, we follow this practice in our simulation. All the models were developed using Tensorflow [61] and Python. The experiments are run on a GPU machine with GeForce GTX 1080 Ti. The default values of different parameters are summarized in Table I. The hyperparameters are selected by trial-and-error. We have tested several commonly used hyperparameter values and selected the one with the best performance. For example, we test the performance with kernel size of 1×1, 3×3 and 5×5 and determine 3×3 as the best kernel size for the models.

A. Travel Demand Prediction

Fig. 10 shows the region in Manhattan selected for the study. It is divided into 28×20 grid cells, each of which has an area of 220 × 170 m^2. The time-steps of MultiConvLSTM is 12. The kernel size is 3×3. Three levels of resolutions are used, including 28×20, 14×10 and 7×5. The following models are also used to compare with MultiConvLSTM to evaluate the performance:

1. Autoregressive Integrated Moving Average (ARIMA) [15]: a well known statistical analysis method for time series data.
2. FCNN [62]: two hidden layers with 256 hidden units in each layer.

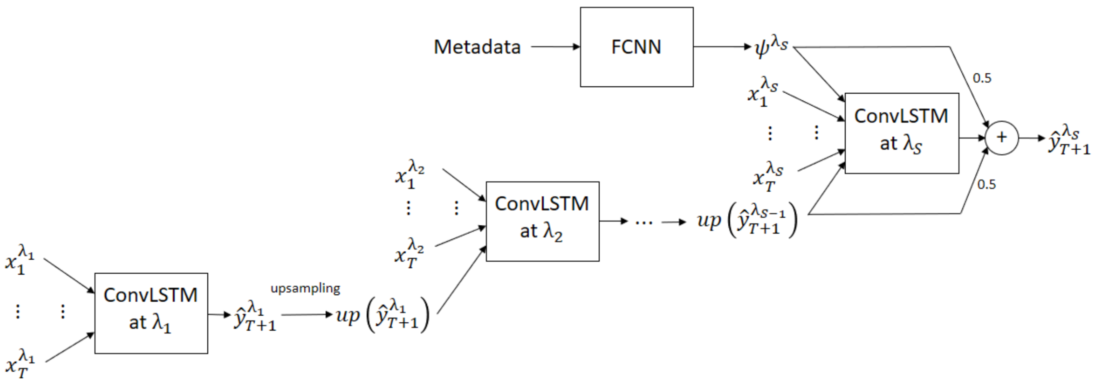


Fig. 9. MultiConvLSTM architecture.

TABLE I
DEFAULT VALUE OF DIFFERENT PARAMETERS IN THE EXPERIMENTS.

Parameters	Definition	Value
$m \times n$	Grid cells arrangement	28 × 20
N	Number of grid cells	560
τ_g	Length of a time-step	10 mins
τ	Number of time-steps to be predicted	12
S	Highest resolution index of MultiConvLSTM	3
T	Number of input historical time-steps	12
c	Positive constant in SMAPE	1
	Kernel size	3 × 3
	Learning rate of Adam	0.001
	Batch size	50

3. CNN [37]: Four convolutional layers with filters size of 16, 32, 16, 1. The kernel size is 3 × 3.
4. RNN-LSTM [56]: 12 time-step input and the input data of each time-step composed of demand data and metadata.
5. ConvLSTM [55]: 12 time-steps input and 3×3 kernel size.

The activation function used in the models is *RELU* except the one in between the fully connected layers which is *sigmoid*. Mean square error (MSE) is used as the loss function for all the models except ARIMA for training:

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y - \hat{y})^2, \tag{16}$$

where N is the total number of grid cells. The training algorithm is Adam optimization [63] with 0.001 learning rate. The batch size is set to 50 and the number of training steps is set to 100,000 where one training step (also called iteration) represents a parameters update process. We consider time information metadata, represented as a 5-tuple vector ⟨month, day, hour, minute, day of week⟩. All the experimental results in the following sections are the average value of predictions using 10,000 testing data.

1) One-step prediction: Table II shows the RMSE, MAE and SMAPE results of different methods for predicting one future time-step, i.e., $\tau = 1$. ARIMA provides the baseline performance in time-series analysis. Although FCNN and CNN are powerful for classification in general, their performances are not satisfactory for travel demand prediction. We can see that MultiConvLSTM achieves the lowest RMSE (1.5405), MAE (0.9255) and SAMPE (0.1663) among all the methods. ConvLSTM and RNN-LSTM have similar performance. We may calculate the prediction accuracy by (1 – SMAPE) from the results shown in Tables II. The accuracy of our proposed model MultiConvLSTM is 1 - 0.1663 = 83.37%, which outperforms that of RNN-LSTM (1 - 0.1766 = 82.34%) by 1.03% as shown in Table II. As discussed in Section IV-C, ConvLSTM can be reduced to RNN-LSTM by setting the kernel size to 1 × 1. The kernel size is set as 3 × 3 of ConvLSTM in this experiment which only accounts for short range dependence and thus, the results computed by RNN-LSTM and ConvLSTM are similar. The performance is further improved with MultiConvLSTM since it can account for long-range dependence even with the same kernel size of 3 × 3.

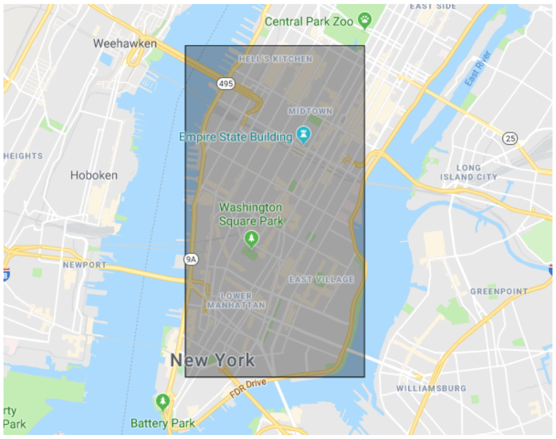


Fig. 10. Selected region in Manhattan.

2) Multi-step prediction: The length of a time-step in our experiment is 10 minutes which may not be appropriate for some applications. Demand in more distant future may also be more useful for certain applications. Subsequent time-steps can be predicted by using historical data and the recently predicted demand as the input. Figs. 11(a), 11(b) and 11(c) show the multi-step prediction errors, RMSE, MAE and SMAPE of different models, respectively, in which 12 future time-steps

TABLE II
TRAVEL DEMAND PREDICTION ERRORS OF DIFFERENT PREDICTION MODELS FOR $\tau = 1$.

	RMSE	MAE	SMAPE
FCNN	1.9609	1.1337	0.2140
CNN	1.8561	1.1252	0.1949
ARIMA	1.7671	1.0711	0.1840
RNN-LSTM	1.6243	0.9682	0.1766
ConvLSTM	1.6485	0.9825	0.1776
MultiConvLSTM	**1.5405**	**0.9255**	**0.1663**

TABLE III
AVERAGE TRAVEL DEMAND PREDICTION ERROR FOR 12 STEPS PREDICTION OF DIFFERENT PREDICTION MODELS

	RMSE	MAE	SMAPE
FCNN	2.6893	1.6066	0.3002
CNN	2.2625	1.3061	0.2134
ARIMA	2.0774	1.2168	0.2050
RNN-LSTM	2.3295	1.4450	0.2700
ConvLSTM	1.7645	1.0328	0.1828
MulitConvLSTM	**1.6683**	**0.9764**	**0.1720**

TABLE IV
TRAVEL DEMAND PREDICTION ERRORS WITH AND WITHOUT TIME INFORMATION FOR $\tau = 1$.

Model	With time info?	RMSE	MAE	SMAPE
RNN-LSTM	No	1.7439	1.0224	0.1911
	Yes	**1.6243**	**0.9682**	**0.1766**
ConvLSTM	No	1.6616	0.9980	0.1811
	Yes	**1.6485**	**0.9825**	**0.1776**
MulitConvLSTM	No	1.6107	0.9723	0.1789
	Yes	**1.5405**	**0.9255**	**0.1663**

(a total of 2 hours) are considered for all methods. Table III shows the average RMSE, MAR and SMAPE for 12 time-steps of different models. In general, the errors of all models increase with more time-steps due to error accumulation. Our proposed model, MultiConvLSTM, achieves the lowest RMSE, MAE and SMAPE in prediction for all time-steps. While RNN-LSTM is the second best in one-step prediction, the error increases and accumulates dramatically with more time-steps. It is because RNN-LSTM focuses on temporal prediction only. Even when metadata are concatenated with demand data as the input, the prediction still mainly depends on the historical data. When noise and error are present in the historical data, RNN-LSTM cannot identify and filter out the noise. As a result, the error accumulates in multi-step prediction. On the other hand, the errors of MultiConvLSTM and ConvLSTM increase sightly in multi-step prediction. They can extract both temporal and spatial features from historical data and metadata for prediction and account for the demand in time and space. Even though noise and error appear in historical data, their effect can be suppressed.

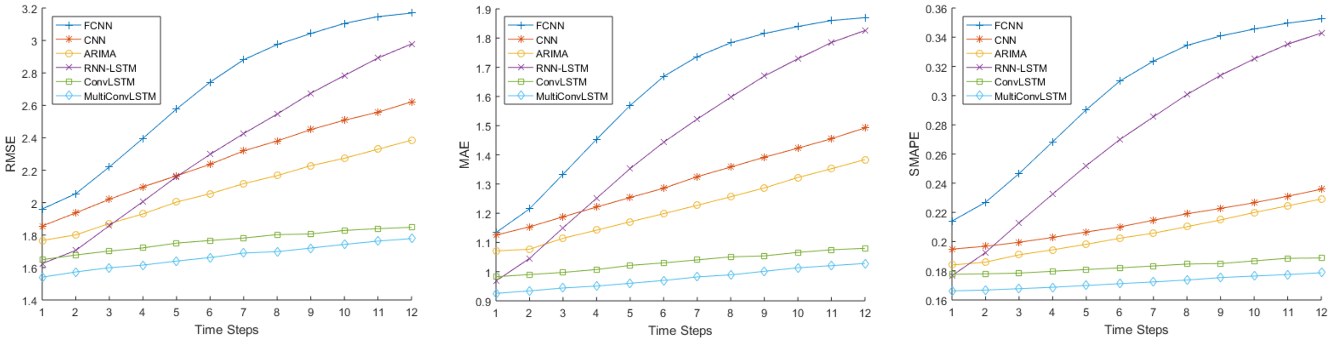
3) Time of Day: The prediction error varies with different times of day. We investigate the relationship between prediction error and the average number of travel requests. The average prediction error of specific time and average number of travel requests are shown in Fig. 12. As shown in the figure, the error is proportional to the average number of travel requests. The number of travel requests illustrate human mobility, where two travel demand valleys and one peak appeared in 4-5am and 4pm, and 7pm, respectively. The higher the travel demand, the higher the prediction error.

4) Day of Week: Similar to the experiment on Time of Day, prediction error and the average number of travel requests vary with different days of a week. Fig. 13 shows the prediction error and the average number of travel requests by day of week. As expected, the number of travel requests is higher on Friday and Saturday while it is the lowest on Monday. The trend of prediction error and number of travel requests is consistent. In other words, as the number of travel requests increases, the performance of the model degrades. The conclusion may not be related to the noise of the data, the model architecture, or the processing techniques. We suggest it is due to the sparsity of the dataset. When we reduce the resolution of the dataset temporally and spatially, the matrices contain many zeroes and small numbers for travel demand. Given such sparse matrices as the training set for the deep learning model, we expect a small bias toward zero for the prediction. Hence, the lower the number of travel requests, the lower the prediction error.

5) Effect of time information: The time information is used as the metadata incorporated in the deep learning models in Sections V-A1 to V-A4. As discussed, it is converted to an $m \times n$ matrix ψ^{λ_S}, which share the same format as of $x_t^{\lambda_S}$. Such metadata may contain some hidden information that correlates with travel demand and OD flows. For example, traffic will be high during rush hours as people go to work in the morning and return home in the evening during weekdays. They may be less willing to go out if the weather is bad. We investigate the effect of incorporating the time information as input to the deep learning models for travel demand prediction. Table IV shows the travel demand prediction errors of different deep learning models with and without the metadata. In general, the travel demand prediction accuracy without the metadata are lower than that with the metadata. For simplicity, weather data is not in this work. However, [64] showed that adding weather to traffic flow prediction can improve the prediction accuracy. Hence, we can conclude that including relevant data to the deep learning model can improve the travel demand prediction accuracy.

B. OD Flow Prediction

The region selected in Section V-A is also considered for OD flow prediction. However, due to the high dimensionality of OD flows, the region is re-divided into 8×4 grid cells instead, resulting in an OD matrix and matricized OD Tensor with 1024 elements. Other hyper parameters and training methods of MultiConvLSTM follow those utilized in Section V-A. In addition, the-state-of-the-art non-deep learning method called Nonnegative Matrix Factorization AutoRegressive (NMF-AR) [35] is reproduced for comparison. We follow the parameters setting given in [35] where the number of basic



(a) RMSE (b) MAE (c) SMAPE

Fig. 11. Multi-step error of travel demand prediction.

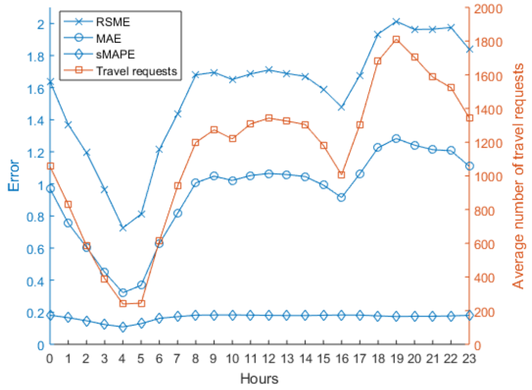


Fig. 12. Average travel demand prediction error by time of day.

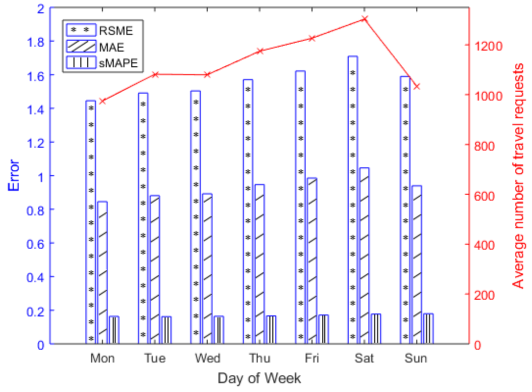


Fig. 13. Average travel demand prediction error of by day of week.

patterns and the order of autoregressive model are equal to 6 and 2, respectively.

TABLE V

OD flows prediction errors of different prediction models for $\tau = 1$.

	RMSE	MAE	SMAPE
ARIMA	1.7796	0.9620	0.2686
FCNN	1.2809	0.7168	0.2079
CNN	1.2171	0.7524	0.2037
NMF-AR	1.2319	0.6910	0.1867
RNN-LSTM	1.1208	0.6470	0.1873
ConvLSTM	1.0876	0.6367	0.1826
MultiConvLSTM (OD matrix)	1.0229	0.6242	0.1807
MultiConvLSTM (OD Tensor)	**0.9899**	**0.6015**	**0.1723**

1) One step prediction: Table V shows the RMSE, MAE and SMAPE results of different methods for predicting one future time-step, i.e., $\tau = 1$. Among all the compared methods, MultiConvLSTM with matricized OD Tensor input achieves the best prediction accuracy in terms of all three performance metrics (RMSE: 0.9899; MAE: 0.6015; SMAPE: 0.1723). Even without the permutation and matricization in pre-processing, prediction based on traditional data structure (OD matrix) with MultiConvLSTM can still outperform all other methods including the-state-of-the-art NMF-AR. We may calculate the prediction accuracy by (1 – SMAPE) from the results shown in Table V. The accuracy is 1 - 0.1723 = 82.77% using MultiConvLSTM with OD tensor method as shown in Table V. This is better than those by ConvLSTM (1-0.1826 = 81.74%) and MultiConvLSTM with OD matrix method (1 - 0.1807 = 81.93%) by 1.03% and 0.84%, respectively. In addition, the training times of the machine learning methods are provided in Table VI. The time required to convert the OD flows to tensor form is basically negligible as it only takes around thirty milliseconds for each time-step. However, in practice training is usually done before implementation in real systems. The length of training time will not affect the actual prediction of the well trained model. Therefore, there is almost no significant difference in computational cost between the compared machine learning methods when the model is ready for prediction.

2) Multi-step prediction: Similar to travel demand prediction, we predict subsequent time-steps by using historical data and the recently predicted OD flows as the input. Figs. 14(a), 14(b), and 14(c) show the multi-step prediction errors, RMSE, MAR and SMAPE, of different models, respectively, in which 12 future time-steps (a toal of 2 hours) are considered for all methods. We can see that MultiConvLSTM performs the best among all compared methods for all time-steps with respect to RMSE, MAE, and SMAPE. With the additional permutation and matricization to pre-process the OD Tensor,

TABLE VI
TRAINING TIME OF DIFFERENT PREDICTION MODEL.

	Training time (s)
FCNN	2.0×10^5
CNN	1.7×10^5
RNN-LSTM	2.0×10^5
ConvLSTM	2.3×10^5
MultiConvLSTM (OD matrix)	2.5×10^5
MultiConvLSTM (OD Tensor)	2.5×10^5

TABLE VII
AVERAGE OD FLOWS PREDICTION ERROR FOR 12 STEPS PREDICTION OF DIFFERENT PREDICTION MODELS.

	RMSE	MAE	SMAPE
ARIMA	1.8004	0.9710	0.2711
FCNN	1.6564	0.9062	0.2557
CNN	1.3445	0.8051	0.2132
NMF-AR	1.2444	0.6981	0.1905
RNN-LSTM	1.3154	0.7883	0.2244
ConvLSTM	1.1131	0.6552	0.1869
MultiConvLSTM (OD matrix)	1.0500	0.6373	0.1835
MultiConvLSTM (OD Tensor)	**1.0157**	**0.6202**	**0.1768**

the accuracy is further improved. We can also see that NMF-AR has a valley shape and reaches its minimum point among its 12 predicted time-steps at about 3 to 5 future time-steps while other prediction methods incur increasing errors with increasing time steps. The average prediction error for 12 steps is shown in Table VII. Although errors are generally accumulated with more time-steps, the increment in error of our methods is still smaller than other methods.

3) *Time of Day:* We analyze the variation in prediction error with different times of day with respect to OD flows. The average number of travel requests and prediction error vary during the day as shown in Fig. 15. In particular, there are two valleys at 4-5am and 4pm, and a peak at 7pm, which aligned with travel demand discussed in Section V-A3. The prediction error is also proportional to the number of travel demand. The higher the number of travel requests, the higher the prediction error.

4) *Day of Week:* Similarly, the relationship between the number of travel requests and prediction error are illustrated in Fig. 16. The prediction error is proportional to the number of travel requests. The higher the number of travel requests, the higher the prediction error. Similar to travel demand prediction, we suggest it is due to the sparsity of the dataset. When we reduce the resolution of the dataset temporally and spatially, the matrices contain many zeroes and small numbers for OD flows. Given such sparse matrices as the training set for the deep learning model, we expect a small bias toward zero for the prediction. Hence we can conclude that the prediction error is proportional to the number of travel requests.

5) *Effect of time information:* The time information is used as the metadata that incorporated in the deep learning models in Section V-B1 to V-B4. We investigate the effect of incorporating the time information as input to the deep learning models for OD flows prediction. Table VIII shows the OD flow prediction error of different deep learning model with and without metadata. The OD flow prediction accuracy of deep learning models without the metadata are lower than that with the metadata. Hence, we can conclude that incorporating some relevant data to the deep learning model can improve the OD flow prediction accuracy.

TABLE VIII
OD FLOW PREDICTION ERRORS WITH AND WITHOUT TIME INFORMATION FOR $\tau = 1$.

Model	With time info?	RMSE	MAE	SMAPE
RNN-LSTM	No	1.2044	0.6781	0.1936
	Yes	**1.1208**	**0.6470**	**0.1873**
ConvLSTM	No	1.1815	0.6738	0.1933
	Yes	**1.0876**	**0.6367**	**0.1826**
MulitConvLSTM	No	1.0524	0.6434	0.1852
	Yes	**0.9899**	**0.6015**	**0.1723**

VI. CONCLUSION

In most existing on-demand transportation services, passengers have to wait until the assigned vehicles arrive at the passengers' pick up points. In order to reduce the unnecessary waiting time, we propose a new deep learning model MultiConvLSTM to accurately predict the future travel demand and OD flows, such that unoccupied vehicles can be pre-allocated to potential customers' origin of service. MultiConvLSTM is designed based on the multi-scale network and ConvLSTM, and it can extract both temporal and spatial features from historical data and metadata to predict future demand and OD flows. The travel demand and OD flow prediction problems are presented and pre-processing methods for both the travel demand and OD flows are discussed. To preserve geographical information, we propose a novel data structure called OD Tensor to store the OD flows and apply permutation and matricization to OD Tensor for dimension reduction. Experiments on real-world New York transportation dataset with around 400 million records show that MultiConvLSTM outperforms the existing prediction methods in both one-step and multiple-step predictions. The OD Tensor permutation and matricization method can help the deep learning model achieve higher prediction accuracy when compared to the traditional OD matrix representation.

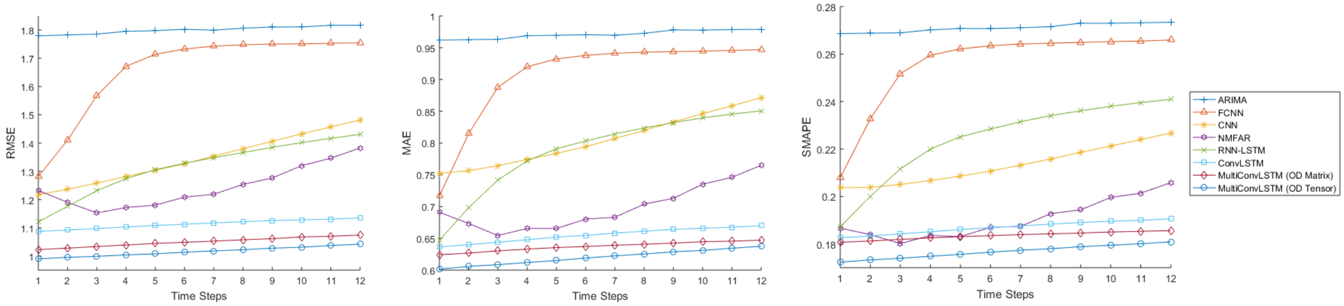
In the future, additional data analyzing methods can be applied to the OD Tensor to study the OD flows. Besides, a re-balancing strategy is worth investigation to further reduce the waiting time. Optimizing the operational efficiency by alternating the travel strategies with the consideration of predicted OD flows is another potential application.

REFERENCES

[1] K. F. Chu, A. Y. S. Lam, and V. O. K. Li, "Travel demand prediction using deep multi-scale convolutional LSTM network," in *Proc. 21st IEEE Int. Conf. on Intell. Transp. Syst.*, Nov 2018, pp. 1402–1407.

[2] Uber. (2018) Uber. [Online]. Available: https://www.uber.com/

[3] Lyft. (2018) Lyft. [Online]. Available: https://www.lyft.com/



(a) RMSE

(b) MAE

(c) SMAPE

Fig. 14. Multi-step error of OD flows prediction.

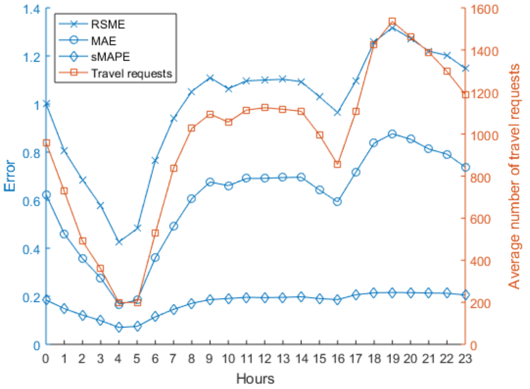


Fig. 15. Average OD flows prediction error by time of day.

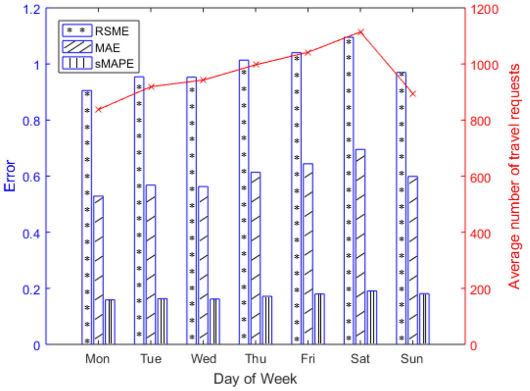


Fig. 16. Average OD flows prediction error by day of week.

[4] A. Y. S. Lam, “Combinatorial auction-based pricing for multi-tenant autonomous vehicle public transportation system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 859–869, March 2016.

[5] A. Y. S. Lam, Y. W. Leung, and X. Chu, “Autonomous-vehicle public transportation system: Scheduling and admission control,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1210–1226, May 2016.

[6] L. Rayle, D. Dai, N. Chan, R. Cervero, and S. Shaheen, “Just a better taxi? a survey-based comparison of taxis, transit, and ridesourcing services in san francisco,” *Transport Policy*, vol. 45, pp. 168 – 178, 2016.

[7] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity imu/gps navigation loop for autonomous land vehicle applications,” *IEEE Trans. Robot. Autom.*, vol. 15, no. 3, pp. 572–578, Jun 1999.

[8] K. F. Chu, E. R. Magsino, I. W.-H. Ho, and C.-K. Chau, “Index coding of point cloud-based road map data for autonomous driving,” in *Proc. 2017 IEEE 85th Veh. Technol. Conf.*, June 2017, pp. 1–7.

[9] S. S. M. Ali, B. George, L. Vanajakshi, and J. Venkatraman, “A multiple inductive loop vehicle detection system for heterogeneous and lane-less traffic,” *IEEE Trans. Instrum. Meas.*, vol. 61, no. 5, pp. 1353–1360, May 2012.

[10] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu, and C. Chen, “Data-driven intelligent transportation systems: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec 2011.

[11] K. Dresner and P. Stone, “A multiagent approach to autonomous intersection management,” *J. of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[12] K. F. Chu, A. Y. S. Lam, and V. O. K. Li, “Dynamic lane reversal routing and scheduling for connected autonomous vehicles,” in *Proc. 3rd IEEE Annu. Int. Smart Cities Conf.*, Sept 2017, pp. 1–6.

[13] A. Y. S. Lam, K. C. Leung, and V. O. K. Li, “Vehicular energy network,” *IEEE Trans. Transport. Electrific.*, vol. 3, no. 2, pp. 392–404, June 2017.

[14] T. Abrahamsson, “Estimation of origin-destination matrices using traffic counts-a literature survey,” 1998.

[15] G. E. P. Box and D. A. Pierce, “Distribution of residual autocorrelations in autoregressive-integrated moving average time series models,” *J. of the American Statistical Assoc.*, vol. 65, no. 332, pp. 1509–1526, 1970.

[16] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang, “Prediction of urban human mobility using large-scale taxi traces and its applications,” *Frontiers of Comput. Sci.*, vol. 6, no. 1, pp. 111–121, Feb 2012.

[17] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, “Predicting taxi-passenger demand using streaming data,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sept 2013.

[18] N. Davis, G. Raina, and K. Jagannathan, “A multi-level clustering approach for forecasting taxi travel demand,” in *Proc. IEEE 19th Int. Conf. on Intell. Transp. Syst.*, Nov 2016, pp. 223–228.

[19] 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. IEEE Int. Conf. on Big Data*, Dec 2016, pp. 833–842.

[20] K. Zhang, Z. Feng, S. Chen, K. Huang, and G. Wang, “A framework for passengers demand prediction and recommendation,” in *Proc. IEEE Int. Conf. on Services Computing*, June 2016, pp. 340–347.

[21] L. G. Willumsen, “Estimation of an od matrix from traffic counts–a review,” 1978.

[22] H. Yang and J. Zhou, “Optimal traffic counting locations for origin–destination matrix estimation,” *Transp. Res. Part B: Methodological*, vol. 32, no. 2, pp. 109–126, 1998.

[23] M. A. Munizaga and C. Palma, “Estimation of a disaggregate multimodal public transport origin–destination matrix from passive smartcard data from santiago, chile,” *Transp. Res. Part C: Emerging Technologies*, vol. 24, pp. 9–18, 2012.

[24] J. Barceló, L. Montero, L. Marqués, and C. Carmona, “Travel time forecasting and dynamic origin-destination estimation for freeways based on bluetooth traffic monitoring,” *Transp. Res. Rec.: J. of the Transp. Res. Board*, no. 2175, pp. 19–27, 2010.

[25] F. Calabrese, G. D. Lorenzo, L. Liu, and C. Ratti, “Estimating origin-destination flows using mobile phone location data,” *IEEE Pervasive Computing*, vol. 10, no. 4, pp. 36–44, April 2011.

[26] J. Farzin, “Constructing an automated bus origin-destination matrix using farecard and global positioning system data in sao paulo, brazil,” *Transp. Res. Rec.: J. of the Transp. Res. Board*, no. 2072, pp. 30–37, 2008.

[27] J. Tang, F. Liu, Y. Wang, and H. Wang, “Uncovering urban human mobility from large scale taxi gps data,” *Physica A: Statistical Mechanics and its Appl.*, vol. 438, pp. 140–153, 2015.

[28] Y. Yue, Y. Zhuang, Q. Li, and Q. Mao, “Mining time-dependent attractive areas and movement patterns from taxi trajectory data,” in *Proc. 2009 17th Int. Conf. on Geoinformatics*, Aug 2009, pp. 1–6.

[29] Z. Deng and M. Ji, “Spatiotemporal structure of taxi services in shanghai: Using exploratory spatial data analysis,” in *Proc. 2011 19th Int. Conf. on Geoinformatics*, June 2011, pp. 1–5.

[30] M. Bierlaire and F. Crittin, “An efficient algorithm for real-time estimation and prediction of dynamic od tables,” *Operations Research*, vol. 52, no. 1, pp. 116–127, 2004.

[31] J. Barceló Bugeda, L. Montero Mercadé, M. Bullejos, O. Serch, and C. Carmona, “A kalman filter approach for the estimation of time dependent od matrices exploiting bluetooth traffic data collection,” in *Proc. of Transp. Res. Board 91st Annu. Meeting*, 2012, pp. 1–16.

[32] E. Cipriani, M. Florian, M. Mahut, and M. Nigro, “A gradient approximation approach for adjusting temporal origin–destination matrices,” *Transp. Res. Part C: Emerging Technologies*, vol. 19, no. 2, pp. 270 – 282, 2011.

[33] A. Tympakianaki, H. N. Koutsopoulos, and E. Jenelius, “c-spsa: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin–destination matrix estimation,” *Transp. Res. Part C: Emerging Technologies*, vol. 55, pp. 231 – 245, 2015.

[34] T. Djukic, G. Flötteröd, H. van Lint, and S. Hoogendoorn, “Efficient real time od matrix estimation based on principal component analysis,” in *Proc. 15th Int. IEEE Conf. on Intell. Transp. Syst.*, Sept 2012, pp. 115–121.

[35] X. Li, J. Kurths, C. Gao, J. Zhang, Z. Wang, and Z. Zhang, “A hybrid algorithm for estimating origin-destination flows,” *IEEE Access*, vol. 6, pp. 677–687, 2018.

[36] J. Ren and Q. Xie, “Efficient od trip matrix prediction based on tensor decomposition,” in *Proc. 2017 18th IEEE Int. Conf. on Mobile Data Management*, May 2017, pp. 180–185.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. Advances in Neural Inform. Process. Syst.* Curran Associates, Inc., 2012, pp. 1097–1105.

[38] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[39] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] D. Silver *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[41] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, “Short-term residential load forecasting based on LSTM recurrent neural network,” *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan 2019.

[42] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[43] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[44] W. Huang, G. Song, H. Hong, and K. Xie, “Deep architecture for traffic flow prediction: Deep belief networks with multitask learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct 2014.

[45] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, “Traffic flow prediction with big data: A deep learning approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, April 2015.

[46] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-scale transportation network congestion evolution prediction using deep learning theory,” *PLOS ONE*, vol. 10, no. 3, pp. 1–17, 03 2015.

[47] J. Li, X. Mei, D. Prokhorov, and D. Tao, “Deep neural network for structural prediction and lane detection in traffic scene,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 690–703, March 2017.

[48] M. Bojarski *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[49] ——, “Explaining how a deep neural network trained with end-to-end learning steers a car,” *arXiv preprint arXiv:1704.07911*, 2017.

[50] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *Proc. Int. Conf. on Learning Representations*, 2018.

[51] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, “Real-time prediction of taxi demand using recurrent neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2572–2581, Aug 2018.

[52] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, “Time-series extreme event forecasting with neural networks at uber,” in *Proc. Int. Conf. on Machine Learning Time Series Workshop*, 2017.

[53] Y. Bai, Z. Sun, B. Zeng, J. Deng, and C. Li, “A multi-pattern deep fusion model for short-term bus passenger flow forecasting,” *Applied Soft Computing*, vol. 58, Sept 2017.

[54] H. Yao *et al.*, “Deep multi-view spatial-temporal network for taxi demand prediction,” in *Proc. of AAAI Conf. on Artificial Intelligence*, 2018.

[55] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *Proc. Advances in Neural Inform. Process. Syst.*, 2015, pp. 802–810.

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

[57] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” *arXiv preprint arXiv:1511.05440*, 2015.

[58] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. 2016 IEEE Conf. on Comput. Vision and Pattern Recognition*, June 2016, pp. 770–778.

[59] New York City. (2017, Oct.) New york city open data. [Online]. Available: https://opendata.cityofnewyork.us/

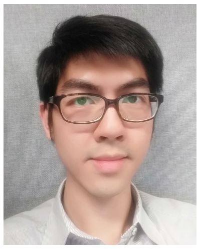
[60] E. T. Jaynes, *Probability theory: the logic of science*. Cambridge university press, 2003.

[61] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[62] R. J. Schalkoff, *Artificial neural networks*. McGraw-Hill New York, 1997, vol. 1.

[63] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[64] A. Koesdwiady, R. Soua, and F. Karray, “Improving traffic flow prediction with weather information in connected cars: A deep learning approach,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9508–9517, Dec 2016.



Kai-Fung Chu (S'17) received the B.Eng. (First Class Honors) and M.Sc. degrees both in electronic and information engineering from The Hong Kong Polytechnic University, Hong Kong, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong. He was a Project Engineer from 2013 to 2016. His research interests include deep learning and reinforcement learning, artificial intelligence, distributed control, optimization, and intelligent transportation systems.

Albert Y.S. Lam (S'03–M'10–SM'16) received the BEng degree (First Class Honors) in Information Engineering and the PhD degree in Electrical and Electronic Engineering from the University of Hong Kong (HKU), Hong Kong, in 2005 and 2010, respectively. He was a postdoctoral scholar at the Department of Electrical Engineering and Computer Sciences of University of California, Berkeley, CA, USA, in 2010-–12. Now he is the Chief Scientist and the Chief Technology Officer at Fano Labs, and an adjunct assistant professor at the Department of Electrical and Electronic Engineering of HKU. He is a Croucher research fellow. His research interests include optimization theory and algorithms, artificial intelligence, smart grid, and smart city.

Victor O.K. Li (S'80--M'81--F'92) received SB, SM, EE and ScD degrees in Electrical Engineering and Computer Science from MIT. Prof. Li is Chair of Information Engineering and Cheng Yu-Tung Professor in Sustainable Development at the Department of Electrical & Electronic Engineering (EEE) at the University of Hong Kong. He is the Director of the HKU-Cambridge Clean Energy and Environment Research Platform, and of the HKU-Cambridge AI to Advance Well-being and Society Research Platform, which are interdisciplinary collaborations with Cambridge University. He was the Head of EEE, Assoc. Dean (Research) of Engineering and Managing Director of Versitech Ltd. He serves on the board of Sunevision Holdings Ltd., listed on the Hong Kong Stock Exchange and co-founded Fano Labs Ltd., an artificial intelligence (AI) company with his PhD student. Previously, he was Professor of Electrical Engineering at the University of Southern California (USC), Los Angeles, California, USA, and Director of the USC Communication Sciences Institute. He served as Visiting Professor at the Department of Computer Science and Technology at the University of Cambridge from April to August 2019. His research interests include big data, AI, optimization techniques, and interdisciplinary clean energy and environment studies. In Jan 2018, he was awarded a USD 6.3M RGC Theme-based Research Project to develop deep learning techniques for personalized and smart air pollution monitoring and health management. Sought by government, industry, and academic organizations, he has lectured and consulted extensively internationally. He has received numerous awards, including the PRC Ministry of Education Changjiang Chair Professorship at Tsinghua University, the UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the Croucher Foundation Senior Research Fellowship, and the Order of the Bronze Bauhinia Star, Government of the HKSAR. He is a Fellow of the Hong Kong Academy of Engineering Sciences, the IEEE, the IAE, and the HKIE.