Hexagon-Based Convolutional Neural Network for Supply-Demand Forecasting of Ride-Sourcing Services

Article *in* IEEE Transactions on Intelligent Transportation Systems · December 2018

DOI: 10.1109/TITS.2018.2882861

CITATIONS
14

READS
481

7 authors, including:

Jintao ke
The Hong Kong University of Science and Technology
13 PUBLICATIONS **286** CITATIONS

SEE PROFILE

Jieping Ye
Arizona State University
305 PUBLICATIONS **13,660** CITATIONS

SEE PROFILE

Hexagon-Based Convolutional Neural Network for Supply-Demand Forecasting of Ride-Sourcing Services

Jintao Ke, Hai Yang, Hongyu Zheng, Xiqun Chen, *Member, IEEE*, Yitian Jia, Pinghua Gong, and Jieping Ye, *Senior Member, IEEE*

***Abstract*— Ride-sourcing services are becoming an increasingly popular transportation mode in cities all over the world. With real-time information from both drivers and passengers, the ride-sourcing platform can reduce matching frictions and improve efficiencies by surge pricing, optimal vehicle-trip assignment, and proactive ridesplitting strategies. An important foundation of these strategies is the short-term supply-demand forecasting. In this paper, we tackle the problem of predicting the short-term supply-demand gap of ride-sourcing services. In contrast to the previous studies that partitioned a city area into numerous square lattices, we partition the city area into various regular hexagon lattices, which is motivated by the fact that hexagonal segmentation has an unambiguous neighborhood definition, smaller edge-to-area ratio, and isotropy. To capture the spatio-temporal characteristics in a hexagonal manner, we propose three hexagon-based convolutional neural networks (H-CNN), both the input and output of which are numerous local hexagon maps. Moreover, a hexagon-based ensemble mechanism is developed to enhance the prediction performance. Validated by a 3-week real-world ride-sourcing dataset in Guangzhou, China, the H-CNN models are found to significantly outperform the benchmark algorithms in terms of accuracy and robustness. Our approaches can be further extended to a broad range of spatio-temporal forecasting problems in the domain of shared mobility and urban computing.**

***Index Terms*— Short-term supply-demand forecasting, deep learning (DL), hexagon-based convolutional neural network (H-CNN), on-demand ride service, ride-sourcing service.**

Manuscript received December 26, 2017; revised August 22, 2018 and November 4, 2018; accepted November 14, 2018. This work was supported in part by the Hong Kong's Research Grants Council under Grant HKUST16222916, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR17E080002, in part by the National Natural Science Foundation of China under Grant 51508505, Grant 71771198, and Grant 51338008, in part by the Fundamental Research Funds for the Central Universities under Grant 2017QNA4025, and in part by the Key Research and Development Program of Zhejiang under Grant 2018C01007. The Associate Editor for this paper was N. Geroliminis. *(Corresponding author: Xiqun Chen.)*

J. Ke and H. Yang are with the Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: jke@connect.ust.hk; cehyang@ust.hk).

H. Zheng and X. Chen are with the College of Civil Engineering and Architecture, Zhejiang University, Hangzhou 310058, China (e-mail: zhenghongyu@zju.edu.cn; chenxiqun@zju.edu.cn).

Y. Jia, P. Gong, and J. Ye are with the Didi Research Institute, Didi Chuxing, Beijing 100085, China (e-mail: ytjia.zju@gmail.com; gongpinghua@didichuxing.com; yejieping@didichuxing.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2018.2882861

I. Introduction

With the boom of the mobile internet, the ride-sourcing service has been reshaping the traditional taxi market and greatly affecting commuters' travel mode choices. Compared to the traditional taxi service, the on-demand ride-sourcing service is generally more convenient, efficient, and comfortable. The reasons are mainly due to two aspects. Firstly, the number of taxis is bounded by the licenses issued, while the operating platform can design economic incentives to attract private car owners to register as full-time/part-time ride service providers, thus each private car owner can be a potential ride-sourcing driver. As a result, the car-hailing supply can be greatly improved, which forces the drivers to provide high-quality services and in turn attracts more travelers to select ride-sourcing services. Secondly, with the help of big data and emerging technologies, the ride-sourcing platforms (e.g., Didi, Uber, and Lyft) can reduce the matching friction by accurate supply-demand forecasting, optimal matching strategies, and efficient surge pricing. By accurately predicting the numbers of passengers and idle drivers for each zone in the following few minutes, the platform can dispatch/incentivize drivers to move from cool zones (where supply is greater than demand) to hot zones (where supply is less than demand) to reduce any zonal supply-demand disequilibrium.

Although there have been fruitful studies on predicting taxi/ride-sourcing passenger demand [1]–[3], only a few studies (e.g., [4]) considered both the demand and supply, and investigated the prediction of the supply-demand gap. Predicting the demand-supply gap is more challenging than merely predicting demand, due to the irregularities and strong local spatial correlations of the demand-supply gaps:

1) Irregularities: Passenger demand (requesting orders) has strong periodicity, since users' travel demand in each zone follows and approximately repeats a certain pattern every day. Even simple historical information can well estimate the passenger demand in the next time interval. However, the supply-demand gap (defined in Section III, Definition 1) is more mutable and irregular, since it is not only determined by the passenger demand but is also governed by vacant cars' movements that do not have strong and explicit day-to-day patterns in most cases. To support our arguments, we illustrate one-week trends of the demand and the supply-demand gap

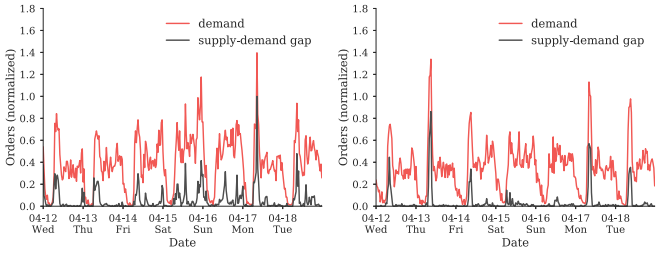


Fig. 1. One-week trends of demand and the supply-demand gap of two zones in Guangzhou, China. (a) Residential zone. (b) Business zone.

in one residential zone (zone 22 in Fig.1) and one business zone (zone 77 in Fig. 1) in Guangzhou, China, which are regular hexagons with a side length of around 660 meters over a time interval of 30 minutes. It can be clearly observed that both zones have regular passenger demand trends in each day. However, the variance of the supply-demand gap is drastic: the supply-demand gap in some time intervals in some days rises sharply to a certain level which is not commonly observed in the previous days.

2) Local Spatial Correlations: The supply-demand gap has stronger local spatial correlations than demand mainly due to the vacant cars' movements. The trip origins of commuters are usually fixed or within a small area (if passengers cannot obtain responses from ride-sourcing cars in a particular zone, they prefer to wait for a longer time, resend ride requests, or switch to other transportation smodes, but usually have a small probability to move to neighboring zones), thus the passenger demand in one zone has limited effects on its neighboring zones. However, vacant cars can move continuously and quickly from one zone to another; the requesting orders in one zone may be satisfied by the idle drivers from the neighboring zones.

The spatial correlations are commonly observed in a broad range of problems, including crowd flow prediction, travel time estimation, etc. The most common technique utilized for capturing spatial correlations is the convolutional neural network (CNN) [5], which is a deep learning approach with excellent performance in image recognition. Previous studies usually split the whole city into numerous squares by horizontal and vertical cut-off lines, and the labels/features of the squares were further placed in matrices. Then by feeding the historical matrices into convolutional layers, pooling layers, and activation layers of the CNN, one can predict the matrix in the next time interval. However, the simple square-like partition approach needs further assessment and discussion. Researchers (e.g., [6]) found that hexagons had several advantages over squares, including an unambiguous definition of the nearest neighborhood, smaller edge-to-area ratio, and more isotropic properties. The advantages of hexagons on region partition are further discussed in Section II.

Inspired by these advantages, many organizations started using hexagons as calculation units in their systems. For example, the dataset of the Environmental Mapping and Assessment Project (EMAP) identifies nationwide locations at 27-km intervals in the U.S. based on 12,600 regular hexagons [7]. Didi Chuxing (www.didichuxing.com/en), the largest ride-sourcing company in China, partitions each city area into numerous regular hexagons, based on which the supply-demand forecasting, hexagon-based subsidy, surge pricing, and dynamic dispatching are implemented. While a broad range of studies use square-like CNN in the domain of urban computing [8], [9], few studies have considered how to efficiently implement CNN in the hexagon-based system.

To incorporate spatio-temporal forecasting and hexagon-based systems, this paper proposes three versions of hexagon-based convolutional neural networks (H-CNN) and implements them in predicting the supply-demand gap of ride-sourcing services. In addition, we combine the local map-to-map prediction and a hexagon-based ensemble mechanism to strengthen the predictive performance. By well capturing the local spatial characteristics of both real-time and historical features (e.g., the number of requests, total vacant hours of the ride-sourcing cars, weather states, and traffic conditions), the proposed models significantly outperform benchmark algorithms, including XGBoost and multi-layer perception, based on a real-world ride-sourcing dataset. The methodological contributions of this paper are summarized as follows:

3) We Design Three H-CNNs That Are Compatible With the Standard Deep Learning Packages (e.g., TensorFlow): The back-propagation algorithm of deep learning is based on matrix transformation, which determines that the models should have square or tensor as their input/output. Thus, an intuitive way to design a hexagon-based CNN is to map hexagons to squares/tensors with few loss of the topology information. We thus build three coordinates for the hexagon system and generate three corresponding mapping functions, i.e., square-mapping, parity-mapping, and cube-mapping, which map the hexagons to squares/tensors. The mapping functions are embedded in the top layers of the architecture, followed by 2-D/3-D convolutional layers, batch-normalization layers, and activation layers.

4) We Devise a Framework Which Combines the Local Map-to-Map Prediction With Hexagon-Based Ensemble Mechanism: The framework provides valuable insights into spatio-temporal data mining in urban computing. Previous studies generally use features in a global city map as input and labels in a global city map as output. However, this kind of design requires long historical sequences of samples and faces the problem of feature dimensionality exploration, especially in the case where the city range is vast. Our framework can significantly enrich the sample size and reduce feature dimensionality (see Section V. D).

5) We Well Define the Problem, Extract Proper Features, and Verify the Proposed Framework With Real-World Data: A total of 384 features are extracted and divided into 47 groups by categories (supply, demand, matching, weather, time, etc.) and time intervals (last time interval, four-week average, weekday/weekend average, etc.). Verified with the real-world datasets provided by Didi, the three proposed H-CNNs integrated with hexagon-based ensemble mechanisms outperform the best benchmark algorithm by 6.5%, 8.0%, and 6.5%, respectively, in terms of RMSE and achieve outstanding

performance in predicting large values of the supply-demand gap.

II. Literature Review

In this section, we focus on the literature in several related fields, including the spatio-temporal forecasting approaches in transportation and urban computing, the advantages of hexagon-based spatial partitions, and the hexagon lattices/grids utilized in various fields.

A. Spatio-Temporal Forecasting Problems in Transportation/Urban Computing

The emergence of ride-sourcing services is having great impact on commuters' travel mode choice behavior and creating new occupation opportunities for private car owners. Many interesting and challenging issues, such as surge pricing and labor supply [10], ride-sharing behavior [11], and dynamic vehicle assignment [12], have attracted researchers' attention. Understanding the short-term supply and demand conditions is one of the most fundamental and important issues, based on which proactive dynamic pricing schemes, dynamic trip-vehicle assignments, and optimal carpooling algorithms can be enhanced. Wang *et al.* [4] proposed a tailored deep residual neural network, named DeepSD, for predicting the supply-demand gap of ride-sourcing services in Hangzhou, China. However, they only modeled temporal characteristics and neglected the spatial connectivity. Ke *et al.* [13] forecast the short-term passenger demand of ride-sourcing services with a fusion Conv-LSTM (FCL-Net), which modeled the spatio-temporal features with a combination of CNN and long short-term memory (LSTM). However, the supply information was not taken into account.

There are numerous spatio-temporal forecasting problems which are similar to and insightful for our problem. Typical problems are traffic flow/speed prediction [14], [15], passenger flow prediction [16], network congestion evolution [17], high-speed rail demand prediction [18], car-following behavior analysis [19], etc. To model the temporal characteristics, both traditional time-series models, such as the ARIMA family [20], and prevailing deep learning approaches [21]–[24] have been successfully implemented. It is reported that researchers have been moving fast from classical statistical models to neural networks in recent years, which might be attributed to the boom in accessible big data and deep learning techniques [25]. Besides, to characterize the spatial features, one of the most efficient and effective approaches is CNN. Due to its born superiority on modeling spatial features, CNN is especially suitable for predictions with strong local connections, such as city-wide inflow and outflow prediction [8]. Moreover, recent years have also witnessed the attempts [9], [13] to incorporate CNN and LSTM/RNN in an end-to-end deep learning architecture for better capturing the spatio-temporal characteristics.

B. The Advantages and Implementations of Hexagons

There are only three regular polygons which can completely tessellate a plane without any overlapping or space: hexagons, squares, and triangles [26]. Compared to squares and triangles, hexagons have the following three main advantages:

1) An Unambiguous Definition of the Nearest Neighborhood: each regular hexagon has six symmetrically equivalent side-connected hexagons [27]. On the other hand, each square has two kinds of adjacent neighbors: orthogonal neighbors which are side-connected and diagonal hexagons which are corner-connected. A simple, precise, and symmetric definition of the nearest neighbors can better characterize the connectivity in the hierarchical network topology. Especially, in our problem, the movements among the zones of the vehicles may influence the spatio-temporal distribution of the supply-demand gap. Based on hexagon-based region partition, the distance between one zone and its six neighboring zones are equivalent, thus movements from this zone to its neighbors can be more accurately characterized.

2) Smaller Edge-to-Area Ratio: hexagons are closer to circles than squares. With unit area, the perimeter of a hexagon (i.e., 3.722) is shorter than that of a square (i.e., 4), which can reduce bias produced by edge effects. The edges of hexagons are smoother, which can potentially better capture the inflow/outflow characteristics between adjacent zones in the domain of transportation/urban computing. In spatio-temporal supply-demand gap prediction, the inflow/outflow of vehicles between every two adjacent zones has strong impacts on the supply-demand gap distribution. Hence, with a smaller edge-to-area ratio, the hexagon-based region partition can better capture the inflow/outflow of the zones.

3) More Isotropic: some studies (such as in simulation) require the distance definition to be based on grids (grid-based distance) instead of the actual straight-line distance [28]. Hexagons have more consistent and stable ratios of grid distance to straight-line distance than squares. Since features in various neighboring hexagons are fed into the proposed neural network simultaneously in our problem, a good measurement of the distance (using grid distance to infer actual straight-line distance) can with no doubt better characterize complex spatial structures.

In field implementation, triangles are rarely used since their edge-to-area ratio is too large. Although square tessellation is the most widely used polygon due to the easy implementation in the orthogonal coordinates, hexagons are also applied in a broad range of fields, including ecology, geography, computer vision, etc. In ecology, some researchers used hexagons instead of squares in field surveys of vegetation coverage and multi-species plant competition experiments [29], and in studying animals' interactions with plants [30]. Birch *et al.* [6] stated that hexagon grids had advantages on both the measurement accuracy and visual effects. In computer vision, hexagons have been used for image processing for over 40 years. Many pieces of evidence indicate that hexagonal lattices have higher sampling efficiencies and superior semantic representation [31]. Motivated by these advantages, a broad range of hexagon-based applications are implemented, including the edge detection, feature extraction, and surface area estimation [32], etc.

C. integrating Irregular Lattices With Neural Networks

An early study in the last decade [33] tried to combine hexagon lattices with the cellular neural network (CeNN), which is a parallel computing paradigm widely used for image processing [34]. However, CeNN is different from the prevailing back-propagation based neural networks, such as the CNN and RNN, since the communication of CeNN is only restricted to the neighboring units. In image processing, CNN can be well adapted to deep structures, has more accurate predictions, and is more widely used compared to CeNN [35]. Moreover, CNN is broadly embedded in most of the high-efficient open-source software libraries, such as TensorFlow, PyTorch, and Caffe, while CeNN is rarely implemented. One of the limitations of CNNs is that the inputs and hidden geometric structures of CNNs are restricted to matrices or tensors.

A few recent studies made attempts to establish new building modules to enhance the representative ability of CNNs. Jifeng *et al.* [36] proposed deformable convolutional networks, in which two novel modules, i.e., deformable convolution and deformable POI pooling, were added to traditional CNNs. In the proposed networks, the convolution kernels were augmented with non-regular offsets, which enabled the receptive fields to be adaptive. However, the inputs of the deformable convolutional networks are still square-like images, thus are not well adaptive to our problem in which the inputs are hexagon maps. Sun *et al.* [37] designed several "quasi-hexagonal" kernels to enhance the receptive field of CNNs for better feature representations. Through extensive experiments, the authors showed that the proposed method integrated with ¡°quasi-hexagonal¡± kernels outperformed the state-of-the-art methods in CIFAR-10/100 and ILSVRC-2012 datasets. However, the "quasi-hexagonal" kernels were used to extract features from square-like images instead of hexagon lattices. Hoogeboom *et al.* [38] investigated how to implement convolution over hexagon lattices, by following the existing standard routines on CNN frameworks. They proposed four candidate coordinate systems, i.e., axial, cube, double width, and offsets, to transform hexagon lattices to matrices or tensors. Through experiments on CIFAR-10 and AID datasets, they showed that CNNs established over hexagon lattices achieved higher predictive accuracy than CNNs established over square lattices, given a fixed parameter budget. Bruna *et al.* [39] proposed graph CNNs, which were generalized versions of the traditional CNNs, with the help of the graph Laplacian spectrum and hierarchical clustering. In contrast to traditional CNNs, the graph CNNs operated directly on graphs and enable end-to-end learning on graphs with the arbitrary shape and size. Extensive studies on the graph CNNs were conducted by Henaff *et al.* [40], Duvenaud *et al.* [41], etc.

Our paper is different from the previous studies in the following aspects. Firstly, we provide explicit definitions for the coordinate transformations and discuss the topology losses and dimensionality of these transformations. The predictive performances of H-CNNs integrated with different coordinate transformations are also empirically tested and compared. Secondly, the proposed H-CNNs have a strong application background and tackle the real obstacles in the domain of urban computing or transportation. In reality, it is hard for companies or institutes to implement CNNs in spatio-temporal prediction problems on a hexagon-based system.

III. Preliminary

In this section, we present explicit definitions for the supply-demand gap, region/time partition, supply-demand gap prediction problem, and features utilized in this paper.

Firstly, we look at the rigorous definition of the supply-demand gap. Passenger demand can be easily defined as the number of requesting orders starting from one specific zone in one specific time interval, while the number of vacant cars (supply) changes continuously and spatially. Thus it is hard to define how many vacant cars there will be within one specific zone in one specific time interval as the supply. Therefore, to find a suitable variable to coordinate supply and demand and to reflect their relationship is challenging and important. In this paper, we define the supply-demand gap in Definition 1.

Definition 1 (Supply-Demand Gap): the supply-demand gap is represented by the total number of unsatisfied requesting orders in each hexagon and each time interval. One unsatisfied requesting order is counted if the request is not dispatched to any ride-sourcing driver. The supply-demand gap of hexagon i during time interval t of day d is denoted as $G_i^{(d,t)}$, where $d \in D$ (the set of date), $t \in T$ (the set of time intervals in a day), $i \in L$ (the set of partitioned zones), and $G_i^{(d,t)} \in [0, +\infty)$.

This definition has two main benefits. Firstly, the definition is explicit, and the number of unsatisfied requesting orders can be flexibly obtained from the platform and aggregated in zones and time intervals. Secondly, predicting the spatial distribution of unsatisfied requesting orders in the next few minutes closely matches the requirement of the platform, which can further dispatch/incentivize vacant cars to the zones with a large number of unsatisfied orders.

Next, an explicit definition of the region-time partition implemented in this paper is given as follows:

Definition 2 (Region-Time Partition): The investigated area is partitioned into various regular hexagons based on the hexagon discrete global grid system [31]. Each day is uniformly partitioned into several intervals. By calculating the summation of the attributes of orders, drivers, weather conditions, etc., the spatio-temporal features can be obtained.

A. Feature Categories

By considering the characteristics and data sources of the features, we divide them into 7 main categories, i.e., supply-related, demand-related, matching-related, weather-related, time-related, location-related, and traffic condition-related features as follows:

1) *Supply-related features* include the number of vacant drivers which have passed the hexagon in one specific time interval, the total waiting minutes of the vacant drivers in each hexagon during each time interval, the vacant drivers' average cruising distance and time, etc. These features can be easily calculated from the

trajectory data of vacant ride-sourcing cars. All the supply-related features in each spatio-temporal state are placed in vector $\mathbf{S}_i^{(d,t)}$, where the meaning and domain of d, t, i are defined in Definition 1.

2) *Demand-related features* are extracted from the order record table and consist of the number of requesting orders, the number of passengers who open the software (they may determine not to request an order), the estimated fare and time, etc. The vector of these features is denoted as $\mathbf{D}_i^{(d,t)}$.
3) *Matching-related features*, also obtained from the order record table, reveal the matching efficiency and matching frictions between supply and demand. They include the satisfied requesting orders (successfully matched), unsatisfied requesting orders (not matched), the answer rate, average answer duration (the average time between passenger requested time and matched time), etc. The vector of matching-related features is defined as $\mathbf{M}_i^{(d,t)}$.
4) *Traffic condition-related features* are comprised of the number of inflow/outflow occupied ride-sourcing cars (which reflects how many cars drop-off/pickup commuters in one hexagon during one specific time slot), the average inflow/outflow speed and distance, etc. Traffic condition-related features are encoded in vector $\mathbf{TC}_i^{(d,t)}$.
5) *Location-related features* simply include the latitude and longitude of one hexagon and are represented by vector $\mathbf{L}_i^{(d,t)}$.
6) *Weather-related features* contain weather states, precipitation, wind speed, and population. The weather states (snowy, sunny, rainy, cloudy, and foggy) are encoded with one-hot representation. The weather-related features are grouped in vector $\mathbf{W}_i^{(d,t)}$.
7) *Time-related features* include the time-of-day and day-of-week. In this paper, there are 48 time-of-day slots, which are encoded by a 6-dimensional binary vector ($2^6 = 64 > 48$). Day-of-week is encoded with one-hot representation. The time-related features are jointly presented by vector $\mathbf{T}_i^{(d,t)}$.

B. Time Stamps of Features

Our goal is to predict $G_i^{(d,t)}$, thus all the features before (d, t) can be used for training. However, training the models with the whole historical time sequence of features is unaffordable and unnecessary. Zhang *et al.* [8] found that the inflow/outflow of a crowd was affected by both the short-term and long-term historical crowd flows. The short-term crowd flow captured the tendency, while the long-term crowd flow characterized the periodicity. In this paper, we extract features from both short-term and long-term time slots.

Firstly, we extract four kinds of long-term features: the last-four-week mean, weekdays/weekends mean, features on the day of the last week, and features on the last day. For each kind, the features of both time intervals t and $t-1$ are selected. To illustrate the design and notations, we take the supply-related feature group as an example. Suppose we are predicting $G_i^{(d,t)}$, then we extract:

1) The last-four-week (LFW) means supply at time interval t, that is, $\mathbf{S}_{i,LFW(0)}^{(d,t)} = \text{mean}(\mathbf{S}_i^{(\tau,t)}|_{\tau\in\{d-28,\cdots,d-1\}})$, and at time interval t-1, we have $\mathbf{S}_{i,LFW(1)}^{(d,t)} = \text{mean}(\mathbf{S}_i^{(\tau,t-1)}|_{\tau\in\{d-28,\cdots,d-1\}})$,.
2) Day-of-last-week (DOLW) supply at time interval t, that is $\mathbf{S}_{i,DOLW(0)}^{(d,t)} = \mathbf{S}_i^{(d-7,t)}$, and at time interval $t-1$, we have $\mathbf{S}_{i,DOLW(1)}^{(d,t)} = \mathbf{S}_i^{(d-7,t-1)}$.
3) The last-day (LD) supply at time interval t is $S_{i,LD(0)}^{(d,t)} = S_i^{(d-1,t)}$, and at time interval $t-1$, we have $S_{i,LD(1)}^{(d,t)} = S_i^{(d-1,t-1)}$.
4) The weekday/weekend (W) supply at time interval t is $\mathbf{S}_{i,W(0)}^{(d,t)} = \text{mean}(\mathbf{S}_i^{(\tau,t)}|_{\tau\in P(d)})$, and at time interval $t-1$, we have $\mathbf{S}_{i,W(1)}^{(d,t)} = \text{mean}(\mathbf{S}_i^{(\tau,t-1)}|_{\tau\in P(d)})$, where set $P(d)$ includes all the historical weekdays/weekends when d is a weekday/weekend.

   Secondly, we extract three kinds of short-term features, including features in time intervals $t-1$ and $t-2$ of the current day, and the predicted features in time interval t. Suppose we are predicting $G_i^{(d,t)}$, then we have:
5) Supply at time interval $t-1$, $S_i^{(d,t-1)}$; supply at time interval $t-2$, $S_i^{(d,t-2)}$; predicted supply at time interval t, $S_{i,predict}^{(d,t)} = S_i^{(d,t-1)} + S_{i,LFW(0)}^{(d,t)} - S_{i,LFW(1)}^{(d,t)}$.

IV. H-CNN Framework

In this section, we first present an explicit definition of the local hexagon map and then design three mapping functions which transform local hexagon maps (represented as a vector) into matrices or tensors. The transformed matrices and tensors are further connected with multiple 2-D or 3-D convolutional layers, batch-normalization layers, and activation functions in the proposed deep learning architecture. To strengthen the predictive performance, we design a hexagon-based ensemble mechanism where the supply-demand gap of each hexagon is estimated by the mean of the predicted values of all overlapped local maps.

A. Local Hexagon Map

As a local hexagon map cannot be directly encoded into a matrix like a local square map, we use vector $\mathbf{H}_{(i_0)}^n$ to represent a local hexagon map centralized at hexagon i_0, where n denotes the sequence index of neighboring layers. For example, each hexagon has 6 one-circle surrounding hexagons, and 12 two-circle surrounding hexagons, then the one-circle surrounding local map (including the hexagon itself and the one-circle surrounding hexagons) can be presented as vector $H_{i_0}^1 = (i_0, i_1, \cdots, i_6)$, and the two-circle surrounding local map (including the hexagon itself, the one-circle surrounding and the two-circle surrounding hexagons) can be represented as vector $H_{i_0}^2 = (i_0, i_1, \cdots, i_{18})$. Fig. 2(a) illustrates the relative positions of the hexagons in a two-circle surrounding local map. The index of the hexagons of the nth circle always starts from the left side (270°) of the central hexagon, thus the vector $\mathbf{H}_{(i_0)}^n$ can explicitly represent the relative positions of the hexagons of the n-circle surrounding local map centralized at i_0.

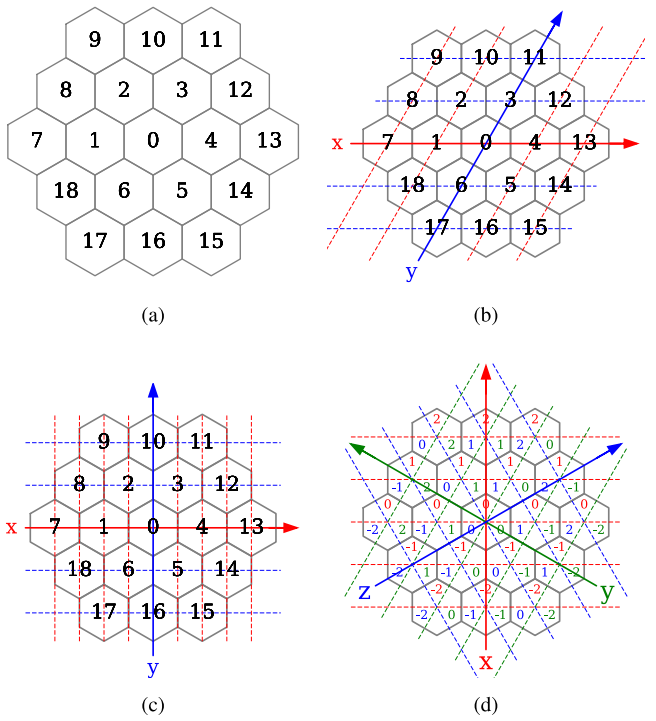


Fig. 2. The indexes and coordinates of local hexagon maps. (a) Encoding of a 2-circle surrounding local map. (b) Square coordinates. (c) Parity coordinates. (d) Cube coordinates.

B. Three Hexagon-Based Mapping Functions

By building three different coordinates, namely square-coordinates, parity-coordinates and cube-coordinates, each vector of the local hexagon map can be transformed into matrices or tensors with different dimensions. Firstly, as shown in Fig. 2(b), by rotating the vertical axis by 30°, the square-coordinates can be obtained, and each hexagon can be placed in one intersection of the horizontal axis and the sloping vertical axis. Secondly, as illustrated in Fig. 2(c), by building parity-like coordinates (for a fixed vertical axis, the neighbor hexagon is placed on the horizontal axis with either odd or even numbers), each hexagon can be placed in one intersection, although half of the intersections are not filled with any hexagons. Thirdly, as demonstrated by Fig. 2(d), three axes with 120° relative angles are established and thus each hexagon can be placed in 3-D coordinates, namely cube-coordinates.

We then denote $\mathbf{M}_{i_0}^{s,n}, \mathbf{M}_{i_0}^{p,n}, \mathbf{M}_{i_0}^{c,n}$ as the transformed matrices of the n-circle surrounding local hexagon map centralized at i_0, i.e., $\mathbf{M}_{i_0}^{n}$. For illustrative purposes, we present the transformed matrices of one-circle surrounding local hexagon map, i.e., $\mathbf{M}_{i_0}^{s,1}, \mathbf{M}_{i_0}^{p,1}, \mathbf{M}_{i_0}^{c,1}$ in Eqs. (1-3).

$$\mathbf{M}_{i_0}^{s,1} = \begin{bmatrix} i_2 & i_3 & \times \\ i_1 & i_0 & i_4 \\ \times & i_6 & i_5 \end{bmatrix} \tag{1}$$

$$\mathbf{M}_{i_0}^{p,1} = \begin{bmatrix} \times & i_2 & \times & i_3 & \times \\ i_1 & \times & i_0 & \times & i_4 \\ \times & i_6 & \times & i_5 & \times \end{bmatrix} \tag{2}$$

$$\mathbf{M}_{i_0}^{c,1} = \left[\begin{bmatrix} \times & i_6 & \times \\ i_5 & \times & \times \\ \times & \times & \times \end{bmatrix}, \begin{bmatrix} \times & \times & i_4 \\ \times & i_0 & \times \\ i_1 & \times & \times \end{bmatrix}, \begin{bmatrix} \times & \times & \times \\ \times & \times & i_3 \\ \times & i_2 & \times \end{bmatrix} \right] \tag{3}$$

The three proposed coordinates lead to different topology losses and require different dimensionality (i.e. space for data storage). One of the advantages of hexagon lattices is that the distance between the centers of any two adjacent hexagon lattices is equivalent. Motivated by this fact, we evaluate the topology loss with the distance among adjacent hexagons in a local hexagon map. Let $d(i_m, i_n)$ denote the distance between the centers of two adjacent hexagons i_m and i_n in a coordinate system (e.g. square, parity, or cube). Then the average distance between the centers of each pair of adjacent hexagons in one local hexagon map equals $d = \frac{1}{2N}\sum_{i_m}\sum_{i_n \in A(i_m)} d(i_m, i_n)$, where $A(i_m)$ means the set of adjacent hexagons of hexagon i_m, and N refers to the number of edges connecting each pair of adjacent hexagons. Note that each edge connecting two adjacent hexagons are counted twice in $\sum_{i_m}\sum_{i_n \in A(i_m)} d(i_m, i_n)$, thus the denominator is $2N$ instead of N. Next, we define the topology ratio $\delta = d/min(d(i_m, i_n))$, where $min(d(i_m, i_n))$ refers to the minimum $d(i_m, i_n)$ in the local hexagon map (for square coordinate, the minimum is 1, for parity and cube coordinate, the minimum is $\sqrt{2}$). Clearly, the topology ratio is not less than 1, and the smaller the topology ratio, the smaller topology loss the transformation from local hexagon maps to matrices/tensors. Particularly, when the topology ratio equals 1, all the distances are equivalent and thus the coordinate system does not lead to any topology losses after the transformation.

Calculating the topology ratio δ is equivalent to calculating $\sum_{i_n \in A(i_m)} d(i_m, i_n)$ for an arbitrary hexagon i, due to the fact that all the hexagons are structurally equivalent (with neighboring hexagons in the same relative locations based on the coordinates). Without loss of generality, we use hexagon i_0 in Eqs. (1-3) for an illustration. Clearly, hexagon i_0 has six neighboring hexagons i_1, i_2, i_3, i_4, i_5, i_6. In the square coordinate, the distances from hexagon i_0 to hexagons i_1, i_2, i_3, i_4, i_5, i_6 are 1, $\sqrt{2}$, 1, 1, $\sqrt{2}$, 1, respectively, with a mean of $(\sqrt{2}+2)/3$, and a minimum of 1. In the parity coordinate, distances from hexagon i_0 to hexagons i_1, i_2, i_3, i_4, i_5, i_6 are 2, $\sqrt{2}$, $\sqrt{2}$, 2, $\sqrt{2}$, $\sqrt{2}$ respectively, with a mean of $(2\sqrt{2}+2)/3$ and a minimum of $\sqrt{2}$. In the cube coordinate, distances from hexagon i_0 to hexagons i_1, i_2, i_3, i_4, i_5, i_6 are all $\sqrt{2}$. Evidently, the topology ratios δ in the square, parity, and cube coordinate are $(\sqrt{2}+2)/3$, $(\sqrt{2}+2)/3$, and 1, respectively. It indicates that the transformation with the cube coordinate does not lead to topology loss, in other words, does not change the relative locations or distances among all hexagons. However, both the square and parity coordinates result in topology loss to some extent. Meanwhile, we can easily find that the dimensionality of the squares/tensors transformed by the square, parity, and cube coordinates are $(2k+1)¡Á(2k+1)$, $(4k+1)¡Á(2k+1)$ and $(2k+1)^3$, respectively, in a k-circle surrounding local hexagon map.

In summary, in terms of the topology loss, square H-CNN = H-CNN > Cube H-CNN; in terms of feature dimensionality, square H-CNN < Parity H-CNN < Cube H-CNN. Clearly, the Cube H-CNN has the smallest topology

loss and the highest feature dimensionality. The small topology loss indicates a more explicit description of the spatial correlations among the hexagons (which is beneficial to the predictive performance), while a high feature dimensionality may lead to over-fitting issues or make the training process unstable (which is harmful to the predictive performance).

Suppose the L-length feature vector of hexagon i is $S_i^{(d,t)}$, then the features of the n-circle surrounding local map centralized in hexagon i_0 can be represented by an $N \times L$ feature matrix $\mathbf{X}_i^{(d,t)} = (S_{i_0}^{(d,t)}, \cdots, S_{i_N}^{(d,t)})^{\mathrm{T}}$, where N is the length of the n-circle surrounding local hexagon map vector. We then define three mapping functions $f_{s-\mathrm{vmap}}, f_{p-\mathrm{vmap}}, f_{c-\mathrm{vmap}}$, where $f_{s-\mathrm{vmap}} : \mathbf{X}_i^{(d,t)} \rightarrow \boldsymbol{\mathcal{X}}_{s,i}^{(d,t)}$, $f_{p-\mathrm{vmap}} : \mathbf{X}_i^{(d,t)} \rightarrow \boldsymbol{\mathcal{X}}_{p,i}^{(d,t)}$, $f_{c-\mathrm{vmap}} : \mathbf{X}_i^{(d,t)} \rightarrow \boldsymbol{\mathcal{X}}_{c,i}^{(d,t)}$. These functions conduct transformations from $\mathbf{H}_{i_0}^n$ to $\mathbf{M}_{i_0}^{s,n}, \mathbf{M}_{i_0}^{p,n}, \mathbf{M}_{i_0}^{c,n}$ for each column (feature) of $\mathbf{X}_i^{(d,t)}$, thus the dimensionality of the obtained tensors $\boldsymbol{\mathcal{X}}_{s,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{p,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{c,i}^{(d,t)}$ are $M_1^{s,n} \times M_2^{s,n} \times L$, $M_1^{p,n} \times M_2^{p,n} \times L$ and $M_1^{c,n} \times M_2^{c,n} \times M_3^{c,n} \times L$, respectively. $M_1^{s,n}, M_2^{s,n}$ are the lengths of the first and second dimensions of $\mathbf{M}_{i_0}^{s,n}$, and $M_1^{p,n}, M_2^{p,n}$ are the lengths of the first and second dimensions of $\mathbf{M}_{i_0}^{p,n}$, while $M_1^{c,n}, M_2^{c,n}, M_3^{c,n}$ denote the first, second, and third dimensions of $\mathbf{M}_{i_0}^{c,n}$. The labels of each local map, $\mathbf{G}_i^{(d,t)} = (G_{i_0}^{(d,t)}, \cdots, G_{i_N}^{(d,t)})^{\mathrm{T}}$ with dimensionality of $N \times 1$ can also be transformed by these three mapping functions to tensors $\boldsymbol{\mathcal{G}}_{s,i}^{(d,t)}$ with dimensions of $M_1^{s,n} \times M_2^{s,n} \times 1$, $\boldsymbol{\mathcal{G}}_{p,i}^{(d,t)}$ with dimensions of $M_1^{p,n} \times M_2^{p,n} \times 1$, and $\boldsymbol{\mathcal{G}}_{c,i}^{(d,t)}$ with dimensions of $M_1^{c,n} \times M_2^{c,n} \times M_3^{c,n} \times 1$, respectively.

C. Training Process of 2-D/3-D H-CNN

The convolutional neural network is one of the most powerful deep learning algorithms for image recognition, since it shows great ability in capturing the hierarchical spatial structure information. In most tasks, such as face recognition, CNN has tensors with two spatial dimensions (three for 3-D CNN) and one channel dimension as the input and a single neuron as the output. In such cases, the CNN architecture usually consists of the convolutional layer, pooling layer, batch-normalization layer, fully-connected layer, and activations. The convolutional layer consists of several learnable filters which are convolved across the width and height of the input tensors, and the dot products between the filers and the input tensors are computed and passed to the next layer. The pooling layers are usually inserted between two convolutional layers and are used to progressively reduce the spatial dimensionality, which helps reduce the parameter quantities and avoid over-fitting. The batch-normalization layer is utilized to reduce the internal covariate shift of neural networks and avoid over-fitting by normalizing the layer inputs for each training mini-batch. The full-connected layers are used for flattening the 3-D tensor (4-D tensor for 3-D CNN) to 1-D vector which is further passed through the activations to be the output neuron. Recently, some researchers eliminated the full-connected layers in the CNN architectures and designed map-to-map CNNs. These map-to-map CNNs are suitable for the problems where the inputs and outputs have the same spatial dimensions, such as precipitation forecasting [9] and city-wide crowd flow prediction [8].

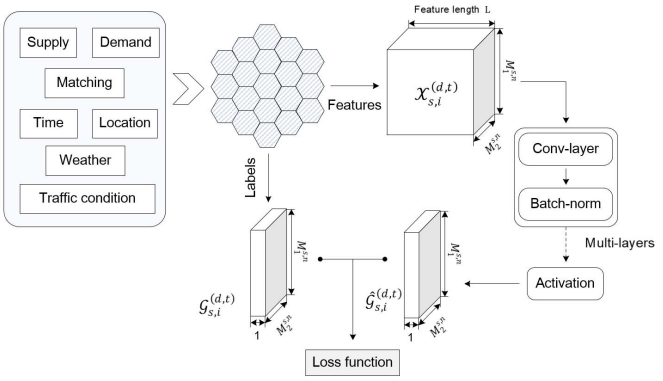


Fig. 3. The training process for H-CNN.

Our model is an extended version of the previous 2-D/3-D map-to-map CNN. Fig. 3 illustrates the framework of the proposed H-CNN architecture. The first layer of H-CNN is one of the mapping function (square-mapping, parity-mapping, or cube-mapping) which converts the input feature matrix and label vector to 3-D tensors (4-D for cube-mapping), i.e., $\boldsymbol{\mathcal{X}}_{s,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{p,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{c,i}^{(d,t)}$. These tensors are connected with several groups of 2-D convolutional layers (3-D for cube mapping) and batch-normalization layers. The number of channels of the first convolutional layer is equal to the number of features of the input tensor (L), and the number of channels in the following convolutional layers decrease in proportion ($L/2, L/4, \cdots$). The last convolutional layer is set to have only one channel, which makes the output tensor have a consistent shape with the label tensors, i.e., $\boldsymbol{\mathcal{G}}_{s,i}^{(d,t)}, \boldsymbol{\mathcal{G}}_{p,i}^{(d,t)}, \boldsymbol{\mathcal{G}}_{c,i}^{(d,t)}$. With f_{conv} referring to the hierarchical CNN architecture, the estimated labels can be represented by

$$\widehat{\boldsymbol{\mathcal{G}}}_{s,i}^{(d,t)} = f_{\mathrm{conv}}(\mathbf{W}\boldsymbol{\mathcal{X}}_{s,i}^{(d,t)} + \mathbf{b}) \tag{4}$$

where $\widehat{\boldsymbol{\mathcal{G}}}_{s,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{s,i}^{(d,t)}$, the estimated labels and input feature tensors based on square-mapping, can be replaced with $\widehat{\boldsymbol{\mathcal{G}}}_{p,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{p,i}^{(d,t)}$ (parity-mapping) and $\widehat{\boldsymbol{\mathcal{G}}}_{c,i}^{(d,t)}, \boldsymbol{\mathcal{X}}_{c,i}^{(d,t)}$ (cube-mapping), respectively. $\mathbf{W}$ and $\mathbf{b}$ are the weighted parameters and intercepts.

An L2-norm regularized objective function is used for training the H-CNN. As an illustration, we employ the regularized objective function of the Squared H-CNN in Eq. (5):

$$\min_{\mathbf{w},\mathbf{b}} \left\| \widehat{\boldsymbol{\mathcal{G}}}_{s,i}^{(d,t)} - \boldsymbol{\mathcal{G}}_{s,i}^{(d,t)} \right\|_2^2 + \alpha \left\| \mathbf{W} \right\|_2^2 \tag{5}$$

The first term of Eq. (5) minimizes the squared errors between the estimated labels and real labels, while the second term is an L2-norm regularization term which reduces the model complexity and avoids over-fitting. $\mathbf{W}$ refers to all weighted parameters, while α is for balancing the trade-off between bias and variance. Note that $\widehat{\boldsymbol{\mathcal{G}}}_{s,i}^{(d,t)}, \boldsymbol{\mathcal{G}}_{s,i}^{(d,t)}$ are replaced with $\widehat{\boldsymbol{\mathcal{G}}}_{p,i}^{(d,t)}, \boldsymbol{\mathcal{G}}_{p,i}^{(d,t)}$ in Parity H-CNN and with $\widehat{\boldsymbol{\mathcal{G}}}_{c,i}^{(d,t)}, \boldsymbol{\mathcal{G}}_{c,i}^{(d,t)}$ in Cube H-CNN, respectively.

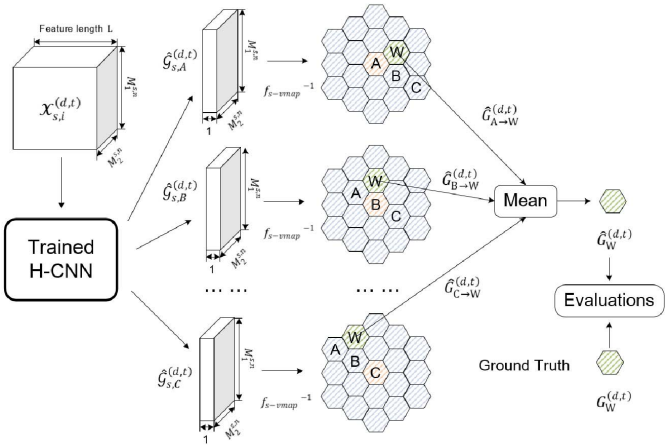


Fig. 4. Prediction process of the Square H-CNN.

D. Prediction: Hexagon-Based Ensemble Mechanism

With the calibrated parameters $\mathbf{W}$ and $\mathbf{b}$, the predicted Square H-CNN tensor centralized at hexagon i, i.e., $\widehat{\mathcal{G}}_{s,i}^{(d,t)}$ can be obtained from Eq. (4). Note that each hexagon has such a predicted tensor. With the inverse function of $f_{s-\text{vmap}}$, we can further obtain the predicted local map $\widehat{\mathbf{G}}_i^{(d,t)}$ from $\widehat{\mathcal{G}}_{s,i}^{(d,t)}$, as shown in Eq. (6).

$$\widehat{\mathbf{G}}_i^{(d,t)} = f_{s-\text{vmap}}^{-1}\left(\widehat{\mathcal{G}}_{s,i}^{(d,t)}\right) \tag{6}$$

where $\widehat{\mathbf{G}}_i^{(d,t)} = (G_{i_0\to i_0}^{(d,t)}, \cdots, G_{i_0\to i_N}^{(d,t)})^{\mathrm{T}}$ is an N-length vector with the same size as the real labels in the local hexagon map centralized at i, i.e., $\mathbf{G}_i^{(d,t)}$. The term $\widehat{G}_{i\to j}^{(d,t)}$ refers to the supply-demand gap of hexagon j predicted by the local hexagon map centralized at i.

A natural way to estimate the supply-demand gap value of hexagon i is to directly use $\widehat{G}_{i\to j}^{(d,t)}$. However, to strengthen the predictive performance, we design the hexagon-based ensemble mechanism on top of the local map-to-map prediction. We estimate the final forecasted supply-demand gap in hexagon j by estimating the mean of all possible $\widehat{G}_{i\to j}^{(d,t)}$ with overlapped i, given by

$$\widehat{G}_j^{(d,t)} = \text{mean}\left(\left\{\widehat{G}_{i\to j}^{(d,t)} | \text{for all overlapped } i\right\}\right) \tag{7}$$

To illustrate this design in detail, we use Fig. 4 for demonstration. Fed with the real-time features, the trained H-CNN outputs numerous H-CNN label tensors, which are further transformed into local hexagon maps, with the hexagons A, B, C, W, etc. The overlapped local hexagon maps have various predictions for hexagon W, we then predict the supply-demand gap of hexagon W with the mean value of these overlapped prediction values.

The hexagon-based ensemble mechanism is motivated by the widely used ensemble algorithms, such as Random Forest [42], AdaBoost [43], and XGBoost [44], in the domain of machine learning, which combine the predictions of several relatively weak learners to improve the accuracy and robustness of prediction. The motivation is that the combination of many weak learners is able to alleviate the randomness of a single weak learner which may lead to high variance and over-fitting issues. There are two main families of ensemble methods: 1) averaging approaches which build various independent weak learners and then average their predicted results; 2) boosting methods which sequentially generate weak learners and add them to the final strong learner (the weak learners are weighted according to their predictive performance). For example, Random Forest [42] randomly constructs a multitude of decision trees in the training process and the final output is found by the mean of the predictions of all individual trees with equaled weights.

In Section V, we will justify the necessity of implementing the hexagon-based ensemble mechanism via a sensitivity analysis, where the predictive performances of three H-CNNs are compared in the case with or without the hexagon-based ensemble mechanism. The experiment results show that the predictive performances of three H-CNNs are significantly improved after this mechanism being applied.

V. Experiments and Discussion

In this section, we present our experimental results on the real-world data provided by Didi. We first describe the dataset in Section V.A and then provide the technical settings of the proposed models in Section V.B. Then, Section V.C compares the predictive performances measured by RMSE, MAE, and MAPE of the H-CNNs and benchmark algorithms. Finally, we discuss the necessity to implement hexagon-based ensemble mechanism in H-CNNs.

A. Data Description

The dataset, provided by the Bigdata Research Lab of Didi, is comprised of the order table, the trajectory data of vacant ride-sourcing cars, the weather data, and traffic condition data. The order table contains 4,143,787 orders collected from the downtown area of Guangzhou, China, from March 29 to April 18, 2017 (three weeks). The investigated area, with longitude between $113.25°E$ and $113.40°E$ and latitudes between $23.05°N$ and $23.15°N$, is partitioned into numerous hexagons with side lengths of around 660 m based on the Didi hexagon system, which is an encrypted form of the hexagon global discrete grid system. We only investigate the hexagons with daily requesting orders larger than 100 in this paper, and the 130 eligible hexagons are shown in Fig. 5(a). Each day is partitioned into 48 time intervals (each with 30 min). With these rules, the orders are aggregated by time and hexagon based on the starting points of the orders. The order table contains the demand-related features, matching-related features, and the label, i.e., the supply-demand gap itself. Fig. 5(b) illustrates the frequency of the supply-demand gap, which approximately follows a power-law shape probabilistic density function.

Fig. 5(c) illustrates the sum of the demand/supply-demand gaps of all hexagons in different time intervals, where it can be seen that the demand has a regular two-peak day-to-day pattern while the peak of the supply-demand gap occurs irregularly in the AM or PM peak. Fig. 5(d) shows the spectrogram

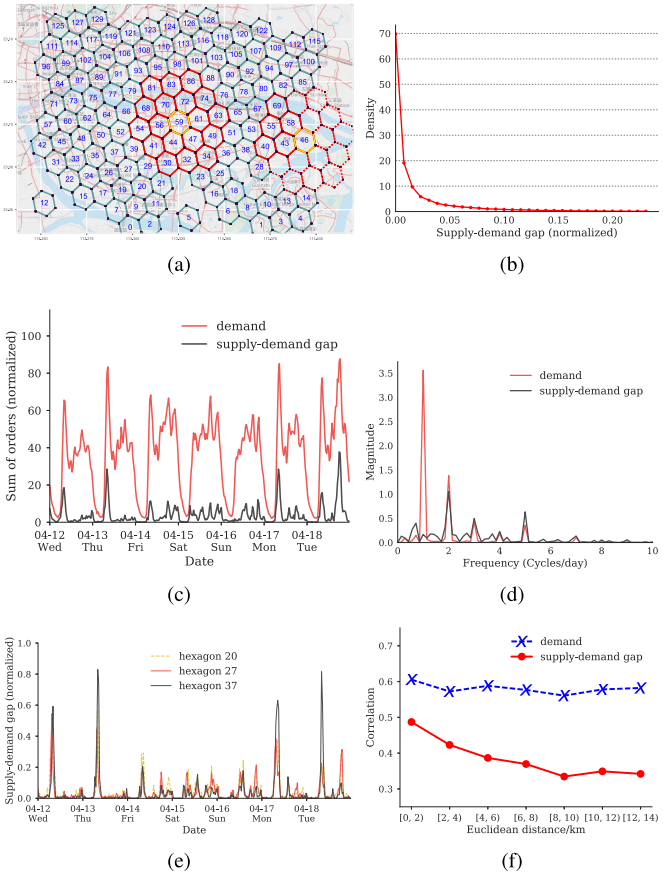


Fig. 5. Empirical data features. (a) Investigated area. (b) Supply-demand gap distribution. (c) Sum of all hexagons. (d) Spectrogram. (e) Supply-demand gap in three nearby hexagons. (f) Mean of correlation.

of the demand and supply-demand gap, where the horizontal axis refers to the frequency and the vertical axis denotes the amplitude. The demand has a larger main amplitude (one cycle per day) and its amplitudes are more centralized, implying that it has stronger periodicity than the supply-demand gap. By visualizing the supply-demand gap of the neighboring hexagons (e.g., hexagons 20, 27, and 37 in Fig. 5(a)), we find that the trend of the demand-supply gap is similar in the neighboring hexagons (see Fig. 5(e)). Furthermore, pair-to-pair correlations of the demand and demand-supply gap time-series of all the hexagons are calculated and then averaged in various groups of Euclidean distances from 0 to 14 km with steps of 2 km (see Fig. 5(f)). It is observed that the demand has the higher correlations because the demand time series in different hexagons have similar periodicity and day-to-day patterns. While the correlations of demand in different ranges of distance are similar, the correlations of the supply-demand gap drop rapidly with the increase in the Euclidean distance. It proves that the supply-demand gap in one hexagon has stronger connections with nearby hexagons than faraway hexagons. Therefore, utilizing the local spatial information for predicting the supply-demand gap is intuitive and necessary.

The other three datasets are also obtained during the same period in the same area and aggregated in the same way as the ride order table. The trajectory data of vacant ride-sourcing cars provide supply-related features, such as the number of vacant drivers passing through one hexagon during a particular time interval, while the weather data provide 30-min aggregated weather states, precipitation, etc. The traffic condition dataset reports the traffic condition-related features, such as the inflow/outflow averaged speed of a certain hexagon during a time interval.

For each hexagon, we generate a two-circle surrounding hexagon local map by selecting two-circle surrounding hexagons (e.g., the red hexagons around hexagon 59 in Fig. 5(a)). For the hexagons on the border where there are no nearby hexagons in some directions (e.g., hexagon 46 in Fig. 5(a)), we create virtual hexagons whose labels and features are filled with 0.

The features and labels are split into a training set and a test set. The training set contains two-weeks of data between March 29 and April 11, 2017, while the one-week test set is from April 12 to April 18, 2017. Remembering that the hexagon local map (sample) is constructed for each hexagon (130 hexagons in total) and the time slot is 30 min, then we can obtain $130 \times 48 \times 14 = 87360$ training samples and $130 \times 48 \times 7 = 43680$ test samples. Note that spatio-temporal forecasting problems [8], [13] commonly split the time-series dataset into a training set followed by a test set, rather than n-splits cross-validation which is widely used in other machine learning problems. The reason is that the time-series data are temporally correlated, and one can only use the past features for predicting the future labels in order to avoid utilizing future information. Hence, the time of the test set is required to be later than that of the training set. Due to the data limitation (only three-week data was accessible in this study), here we only use the first two-week of data for training and the last-week data for testing. Four metrics, i.e., RMSE, MAE, MAPE, and the Pearson Correlation, are utilized to evaluate the predictive performance of the proposed models and benchmark algorithms.

B. Models' Setting

In the architecture of the Square H-CNN and Parity H-CNN, we use four stacked 2-D convolutional layers and each one is inserted with a batch-normalization layer. For Cube H-CNN, 3-D convolutional layers are used. The channel dimensions of the first, second, and third, and fourth convolutional layers are $L, L/2, L/4$, and 1. We select the rectified linear function (Relu) as the activation function for each convolutional layer (including the output convolutional layer). The optimization algorithm is the Adaptive Moment Estimation (Adam), which is an efficient and robust mini-batch gradient descent approach. The batch size is set to be 64 and the training epochs are 30. Both the features and labels (supply-demand gap) are scaled by max-min normalization to $[0, 1]$.

The experiment platform is a server with 20 GPU cores (Intel(R) Xeon(R)) CPU E5-26030 v4 @ 2.20GHz), 251 GB RAM, and one GPU (NVIDIA UNIX x86-64 Kernel Module 375.66). The operating system of the server is CentOS Linux release 7.2.1511 (Core). The proposed models and baselines

are implemented with Python 2.7.5 with TensorFlow, scikit-learn, and XGBoost [44].

C. Model Comparisons

This subsection describes the predictive performance of the test set and the training/test time of the three kinds of H-CNNs, recalling that we use a hexagon-based ensemble mechanism to strengthen the predictive performance of H-CNNs. In this part, we compare RMSE of the three H-CNNs with and without the hexagon-based ensemble mechanism. H-CNNs without the hexagon-based ensemble mechanism directly predict the supply-demand gap in (d, t, j) with the features of hexagon j's two-circle surrounding local hexagon map $\mathbf{H}_i^2$. In other words, $\widehat{G}_i^{(d,t)} = \widehat{G}_{j\to j}^{(d,t)}$ is implemented instead of Eq. (7).

Since the supply-demand gap prediction is essentially a regression problem, we compare our models with several classical or state-of-the-art benchmark regressors. To ensure fairness, we make sure that the benchmark algorithms use the same feature information as our models. Specifically, for predicting the supply-demand gap in (d, t, i), each benchmark algorithm uses features obtained from both hexagon i and its two-circle surrounding hexagons $\mathbf{H}_i^n$ as its input. The introductions and fine-tuned settings of the benchmark algorithms are shown as follows:

1) LASSO (Least Absolute Shrinkage and Selection Operator) [45]: LASSO adds an L1-norm regularization term to the linear regression, which penalizes the absolute size of the regression coefficients. The parameter α which measures the trade-off between empirical errors and mode complexity is tuned from 0.5 to 5 with a step of 0.5.

2) RF (Random Forest) [42]: RF is a classical ensemble algorithm which builds up various decision trees, each of which is trained with a group of bootstrapped samples. The final output of RF is estimated by the mean of the outputs of all the trees. The ratio of the maximum number of features for training an individual tree to the number of features is tuned from 0.2 to 0.8 with a step of 0.1. Number of trees is tuned from 20 to 200 with a step of 20.

3) MLP (Multi-Layer Perception) [46]: MLP is also called the artificial neural network, and is a basic neural network trained with the back-propagation algorithm. The MLP is tuned with the number of hidden layers (from 1 to 5 with a step of 1), the number of hidden neurons (from 256 to 1024 with a step of 128), and learning rate (0.0001, 0.001, and 0.01). The Relu activation is used for all hidden units while the model is trained with Adam optimizer.

4) XGB (XGBoost) [44]: XGB is a scalable, portable and distributed library with the gradient boosting decision tree (GBDT) as its main algorithm. XGB has gained high popularity and attention since its release as it shows great performance on a broad range of machine learning competitions, such as Kaggle (www.kaggle.com). The maximum depth of a tree is tuned from 3 to 8 with a step of 1, the step size shrinkage is tuned from 0.1 to 0.9 with a step of 0.1, while the subsampling ratio is tuned from 0.1 to 0.9 with a step of 0.1.

For the three H-CNNs, we tune two hyper-parameters: the number of convolutional layers (from 1 to 5 with steps of 1) and the learning rate (from 0.0002 to 0.001 with steps of 0.0002). To assess the consistency of the performance of the models, each model is trained and tested for 10 runs with the best parameters, and then the means of the RMSE, MAE, and Pearson Correlation can be calculated, as shown in Table I. The means of the training and test times are also recorded. It can be observed that the Square H-CNN, Parity H-CNN and Cube H-CNN combined with ensemble mechanism outperform the best benchmark algorithm (XGB) by 6.5%, 8.0%, and 6.5% respectively, in terms of RMSE. T-tests are also undertaken to compare the RMSE of the models, and the results (t-statistics and p-values) are displayed in Table II. Note that a negative t-statistic (such as -77.0 when comparing Square H-CNN to MLP) with a p-value (0.00 in this example) less than 0.05 indicates that the RMSE of one model (Square H-CNN in this example) is significantly smaller than that of another model (MLP in this example) at the significant level of 0.05. Clearly, the results show that the proposed H-CNN models significantly outperform all the benchmark algorithms. The possible reason is that the H-CNNs well capture the spatial-temporal characteristics of the features. Many previous studies in the domain of spatio-temporal prediction have realized the importance of spatio-temporal characteristics and hence designed special neural network structures to capture them [1], [4], [8]. Especially, the family of convolutional neural networks has shown outstanding performance on well characterizing the spatio-temporal features in a broad range of implementations, such as crowd flow prediction [8], precipitation nowcasting [9], passenger demand prediction [13], etc.

TABLE I
COMPARISON OF THE MODELS' PERFORMANCE

Model	RMSE ($\times 10^{-2}$)	MAE ($\times 10^{-2}$)	Pearson Corr(%)	Training time(min)	Test time(min)
LASSO	3.98	1.95	53.2	0.17	0.01
RF	3.67	1.87	35.9	1.33	0.01
MLP	3.60	1.64	60.0	0.48	0.01
XGB	3.23	1.42	64.1	2.21	0.02
Square H-CNN	3.20	1.38	67.7	3.13	0.02
Parity H-CNN	3.09	1.33	68.6	3.14	0.02
Cube H-CNN	3.09	1.34	68.3	21.18	0.14
Square H-CNN + Ensemble	3.02	1.29	70.2	3.13	0.02
Parity H-CNN + Ensemble	2.97	1.30	70.5	3.14	0.02
Cube H-CNN + Ensemble	3.02	1.30	69.3	21.18	0.14

Further, Fig. 6 provides the detailed trends of MAPE under different thresholds of the models. The horizontal axis refers the supply-demand gap threshold, and the vertical axis means the MAPE of samples with a real supply-demand gap greater than the threshold. Clearly, the three H-CNNs outperform the benchmarks in conditions with either high or low supply-demand gaps. The Parity H-CNN is the best model which not only has the lowest RMSE, but also reports low MAPE in samples with large supply-demand gaps (see Fig. 6). It indicates that the Parity H-CNN has a better prediction accuracy for the samples with larger supply-demand gaps. Precise predictions on the hexagons with large supply-demand gaps are

TABLE II
T-STATISTICS (P-VALUES) FOR COMPARING RMSE OF THE MODELS

Model	XGB	MLP	LASSO	RF	Square H-CNN	Parity H-CNN	Cube H-CNN	Square H-CNN + Ensemble	Parity H-CNN + Ensemble	Cube H-CNN + Ensemble
XGB	-	-	-	-	2.2 (0.04)	9.5 (0.00)	12.9 (0.00)	41.3 (0.00)	22.5 (0.00)	21.8 (0.00)
MLP	-	-	-	-	77.0 (0.00)	46.1 (0.00)	56.4 (0.00)	120.5 (0.00)	60.5 (0.00)	66.3 (0.00)
LASSO	-	-	-	-	152.3 (0.00)	82.9 (0.00)	100.2 (0.00)	200.3 (0.00)	98.8 (0.00)	111.1 (0.00)
RF	-	-	-	-	91.3 (0.00)	53.1 (0.00)	64.8 (0.00)	135.7 (0.00)	67.8 (0.00)	74.8 (0.00)
Square H-CNN	-2.2 (0.00)	-77.0 (0.00)	-152.3 (0.00)	-91.3 (0.00)	-	7.6 (0.00)	10.1 (0.00)	26.7 (0.00)	19.1 (0.00)	17.6 (0.00)
Parity H-CNN	-9.5 (0.00)	-46.1 (0.00)	-82.9 (0.00)	-53.1 (0.00)	-7.6 (0.00)	-	1.0 (0.33)	8.6 (0.00)	8.7 (0.00)	6.4 (0.00)
Cube H-CNN	-12.9 (0.00)	-56.4 (0.00)	-100.2 (0.00)	-64.8 (0.00)	-10.1 (0.00)	-1.0 (0.33)	-	8.5 (0.00)	8.4 (0.00)	6.0 (0.00)
Square H-CNN + Ensemble	-41.3 (0.00)	-120.5 (0.00)	-200.3 (0.00)	-135.7 (0.00)	-26.7 (0.00)	-8.6 (0.00)	-8.5 (0.00)	-	2.4 (0.03)	-1.2 (0.23)
Parity H-CNN + Ensemble	-22.5 (0.00)	-60.5 (0.00)	-98.8 (0.00)	-67.8 (0.00)	-19.1 (0.00)	-8.7 (0.00)	-8.4 (0.00)	-2.4 (0.03)	-	-3.0 (0.01)
Cube H-CNN + Ensemble	-21.8 (0.00)	-66.3 (0.00)	-111.1 (0.00)	-74.8 (0.00)	-17.6 (0.00)	-6.4 (0.00)	-6.0 (0.00)	1.2 (0.23)	3.0 (0.01)	-

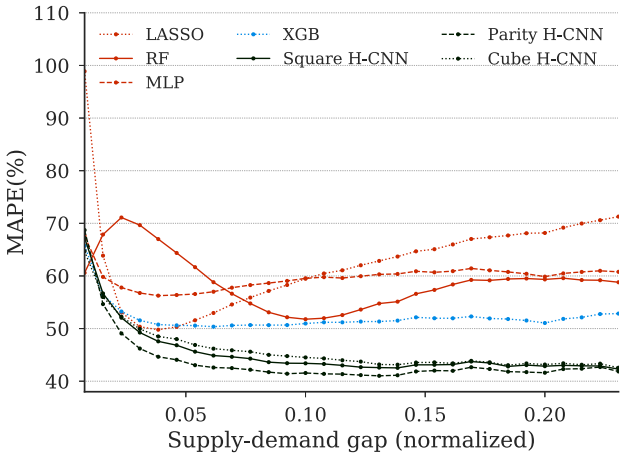


Fig. 6. MAPE of models under thresholds.

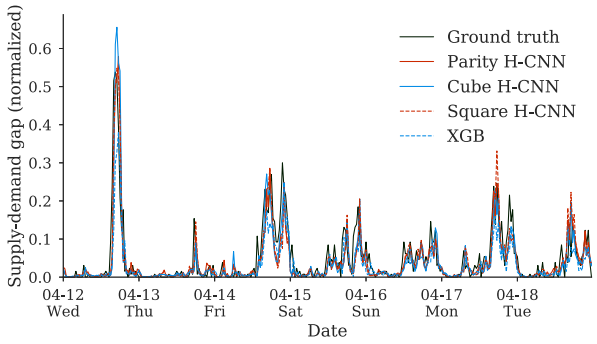


Fig. 7. Predicted supply-demand gap and ground truth in one hexagon.

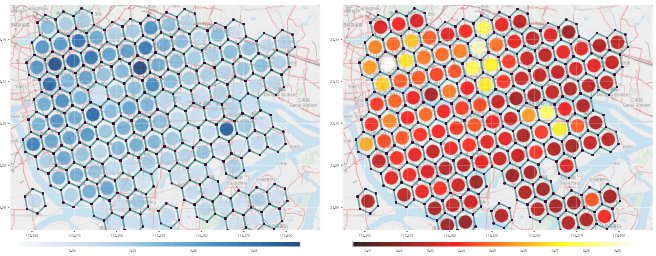


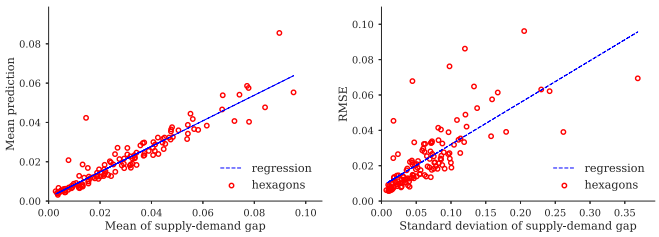
Fig. 8. Spatial distributions. (a) Mean of supply-demand gap. (b) Mean of RMSE.

major concerns of the platform, which can dispatch/incentivize vacant drivers to these hexagons.

Fig. 7 illustrates a one-week trend of the predicted supply-demand gap (by three H-CNNs and XGB) and the ground truth in one selected hexagon. It can be observed that the models are comparable when predicting low supply-demand gaps, but H-CNNs has a better prediction for the high supply-demand gap and captures its sudden increase more powerfully. Fig. 8 demonstrates the spatial distributions of the mean supply-demand gap (Fig. 8(a)) and the RMSE of the predicted Parity H-CNN (Fig. 8(b)). It is not surprising that the zones with larger mean supply-demand gap have larger predicted errors. Fig. 9(a) presents the relation between the mean predicted values in each hexagon and its mean ground truth, indicating that the model can well estimate the one-week mean (in the test set) of the supply-demand gap of each hexagon. Fig. 9(b) illustrates the relationship between RMSE in each hexagon and the standard deviation of its supply-demand gap, showing that the forecasting errors (RMSE) are largely proportional to the variance (standard deviation) of the supply-demand gap.

D. Effects of Hexagon-Based Ensemble Mechanism

We further discuss the effects of implementing the hexagon-based ensemble mechanism on top of the H-CNNs. It is clearly shown in Table I that Square H-CNN, Parity H-CNN, and Cube H-CNN with the hexagon-based ensemble mechanism outperform their variants without the hexagon-based ensemble mechanism by 5.7%, 4.0%, and 2.3% in terms of RMSE,



(a) (b)

Fig. 9. Hexagons' mean prediction, ground truth, and RMSE. (a) Prediction v.s. ground truth. (b) RMSE v.s. standard deviation.

respectively. The results in Table II further verify that the hexagon-based ensemble mechanism significantly improves the predictive performance of the three H-CNNs. These results are unsurprising, since using single prediction is commonly random and unstable, while aggregating the prediction of multiple predictors usually improves the prediction accuracy and robustness. Similar phenomena can be widely observed in ensemble algorithms, such as GBDT and RF, which perform better and are more robust than traditional decision trees.

Furthermore, our design for the combination of the local hexagon map and hexagon-based ensemble mechanism provides more samples and reduces feature dimensionality than the commonly used global map design. For example, for a 2-week training set with hundreds of hexagons/squares, the global map design needs a CNN with huge dimensionality (130 hexagons) while the number of samples is only 7 (days) $\times$ 2 (weeks) $\times$ 48 (30-min time interval in each day) $=$ 672 (suppose the time interval to be 30 min). However, in our design, the dimensionality is restricted to one local hexagon map (with 19 hexagons), while the sample size is improved to 130 (local maps/hexagons) $\times$ 7 (days) $\times$ 2 (weeks) $\times$ 48 (30-min time interval in each day) $=$ 87360 samples.

VI. Conclusions

In this paper, we describe a hexagon-based framework for predicting the supply-demand gap of ride-sourcing services. Three H-CNNs (Square, Parity, and Cube H-CNN) are proposed to explore the spatio-temporal characteristics and dependencies of the supply/demand on top of the hexagon-based region partitions. Furthermore, a hexagon-based ensemble mechanism is developed to further enhance the predictive performance of H-CNNs. Validated by real-world data provided by Didi, we find that our models significantly outperform the best benchmark algorithm and, specifically, Parity H-CNN with an ensemble mechanism achieves the best predictive performance. It is also reported that the hexagon-based ensemble mechanism significantly improves the predictive accuracy of the three H-CNNs.

This study makes the first initial attempt to combine hexagon region partition with spatio-temporal deep learning approaches. The proposed hexagon-based CNNs are easy to be implemented and are compatible with the state-of-the-art deep learning framework/library, such as TensorFlow. As the hexagon-based partition is superior to the simple squared partition in some aspects and has increasingly attracted attention from various institutes and industry sectors, the proposed methods could find more applications in the domain of urban computing and transportation, such as city-wide traffic flow prediction, human mobility analysis, commuters' OD matrix estimation, etc.

This paper has several limitations. Firstly, although the experiments on the ride-sourcing dataset show that the H-CNNs significantly outperform the benchmark algorithms, no rigorous theoretical guarantee can be provided to prove the superiority of our models, due to the complex nature of convolutional neural networks. This remains to be solved in future studies. Secondly, we analyze the advantages and disadvantages of three H-CNNs from the aspects of topology loss and dimensionality and show that the Parity H-CNN achieves the best performance in the experiments. However, due to the limitation of the available data resources, the three-week ride-sourcing datasets in this paper are not adequate to theoretically or empirically evaluate and compare the three H-CNNs during more time periods in other cities. Therefore, more extensive experiments should be conducted to implement and verify the proposed frameworks on other datasets, for other spatio-temporal forecasting problems, such as traffic flow prediction, traffic emission forecasting, etc., in the future studies.

References

[1] Y. Tong *et al.*, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1653–1662.

[2] D. Zhang, T. He, S. Lin, S. Munir, and J. A. Stankovic, "Taxi-passenger-demand modeling based on big data from a roving sensor network," *IEEE Trans. Big Data*, vol. 3, no. 3, pp. 362–374, Sep. 2017.

[3] 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, Sep. 2013.

[4] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 243–254.

[5] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, vol. 1681. Berlin, Germany: Springer-Verlag, 1999, pp. 319–345.

[6] C. P. D. Birch, S. P. Oom, and J. A. Beecham, "Rectangular and hexagonal grids used for observation, experiment and simulation in ecology," *Ecological Model.*, vol. 206, nos. 3–4, pp. 347–359, Aug. 2007.

[7] S. S. Hale *et al.*, "Managing scientific data: The EMAP approach," *Environ. Monit. Assessment*, vol. 51, nos. 1–2, pp. 429–440, Jun. 1998.

[8] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. AAAI*, Feb. 2017, pp. 1655–1661.

[9] S. Xingjian, 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. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 802–810.

[10] L. Zha, Y. Yin, and H. Yang, "Economic analysis of ride-sourcing markets," *Transp. Res. C, Emerg. Technol.*, vol. 71, pp. 249–266, Oct. 2016.

[11] X. M. Chen, M. Zahiri, and S. Zhang, "Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach," *Transp. Res. C, Emerg. Technol.*, vol. 76, pp. 51–70, Mar. 2017.

[12] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 2151–2159.

[13] J. Ke, H. Zheng, H. Yang, and X. M. Chen, "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach," *Transp. Res. C, Emerg. Technol.*, vol. 85, pp. 591–608, Dec. 2017.

[14] 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.

[15] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 871–882, Jun. 2013.

[16] M. Ni, Q. He, and J. Gao, "Forecasting the subway passenger flow under event occurrences with social media," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1623–1632, Jun. 2017.

[17] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS ONE*, vol. 10, no. 3, p. e0119044, Mar. 2015.

[18] X. Jiang, L. Zhang, and X. M. Chen, "Short-term forecasting of high-speed rail demand: A hybrid approach combining ensemble empirical mode decomposition and gray support vector machine with real-world applications in China," *Transp. Res. C, Emerg. Technol.*, vol. 44, pp. 110–127, Jul. 2014.

[19] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, "Capturing car-following behaviors by deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 910–920, Mar. 2018.

[20] B. Ghosh, B. Basu, and M. O'Mahony, "Multivariate short-term traffic flow forecasting using time-series analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 246–254, Jun. 2009.

[21] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[22] 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, Apr. 2015.

[23] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 644–654, Jun. 2012.

[24] 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.

[25] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 3–19, Jun. 2014.

[26] D. B. Carr, A. R. Olsen, and D. White, "Hexagon mosaic maps for display of univariate and bivariate geographical data," *Cartogr. Geograph. Inf. Syst.*, vol. 19, no. 4, pp. 228–236, Jan. 1992.

[27] C. P. D. Birch, N. Vuichard, and B. R. Werkman, "Modelling the effects of patch size on vegetation dynamics: Bracken [Pteridium aquilinum (L.) Kuhn] under grazing," *Ann. Botany*, vol. 85, no. 2, pp. 63–76, Apr. 2000.

[28] E. Pachepsky, J. W. Crawford, J. L. Bown, and G. Squire, "Towards a general theory of biodiversity," *Nature*, vol. 410, no. 6831, pp. 923–926, Apr. 2001.

[29] J. L. Harper, *Population Biology of Plants*. London, U.K.: Academic, 1977.

[30] R. V. Cartar and L. A. Real, "Habitat structure and animal movement: The behaviour of bumble bees in uniform and random spatial resource distributions," *Oecologia*, vol. 112, no. 3, pp. 430–434, 1997.

[31] K. Sahr, "User documentation for discrete global grid generation software," Southern Oregon Univ., Ashland, OR, USA, Tech. Rep. Dggrid version 3.1b, 2011. [Online]. Available: http://webpages.sou.edu/~sahrk/sqspc/pubs/dggriddoc31.pdf

[32] X. He, Q. Wu, W. Jia, and T. Hintz, "Edge detection on hexagonal structure," *J. Algorithms Comput. Technol.*, vol. 2, no. 1, pp. 61–78, Mar. 2008.

[33] C.-H. Huang and C.-T. Lin, "Bio-inspired computer fovea model based on hexagonal-type cellular neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 35–47, Jan. 2007.

[34] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. CAS-35, no. 10, pp. 1273–1290, Oct. 1988.

[35] A. Horváth, M. Hillmer, Q. Lou, X. S. Hu, and M. Niemier, "Cellular neural network friendly convolutional neural networks—CNNs with CNNs," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 145–150.

[36] D. Jifeng *et al.*, "Deformable convolutional networks," in *Proc. ICCV*, Oct. 2017, pp. 764–773.

[37] Z. Sun, M. Ozay, and T. Okatani, "Design of kernels in convolutional neural networks for image classification," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 51–66.

[38] E. Hoogeboom, J. W. T. Peters, T. S. Cohen, and M. Welling. (Mar. 2018). "Hexaconv." [Online]. Available: https://arxiv.org/abs/1803.02108

[39] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. (Dec. 2013). "Spectral networks and locally connected networks on graphs." [Online]. Available: https://arxiv.org/abs/1312.6203

[40] M. Henaff, J. Bruna, and Y. LeCun. (Jun. 2015). "Deep convolutional networks on graph-structured data." [Online]. Available: https://arxiv.org/abs/1506.05163

[41] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. NIPS*, 2015, pp. 2224–2232.

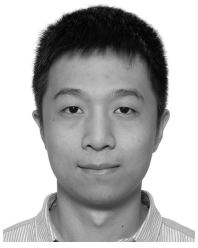
[42] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[43] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[44] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

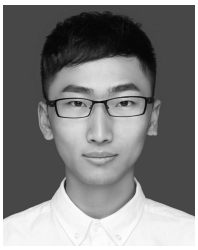
[45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B (Methodological)*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.



Jintao Ke received the B.S. degree in civil engineering from Zhejiang University. He is currently pursuing the Ph.D. degree with the Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology. His research interests include spatio-temporal traffic forecasting, economical modeling, and behavioral studies for innovative shared mobility.

Hai Yang is currently a Chair Professor at The Hong Kong University of Science and Technology. He is internationally known as an active scholar in the field of transportation, with more than 240 papers published in SCI/SSCI indexed journals and an H-index citation rate of 51. Most of his publications have appeared in leading international journals, such as *Transportation Research and Transportation Science*. He is currently the Editor-in-Chief of *Transportation Research Part B: Methodological*, a top journal in the field of transportation.



Hongyu Zheng received the B.E. degree in civil engineering from Zhejiang University, Hangzhou, China, in 2016, where he is currently pursuing the master's degree with the Institute of Intelligent Transportation Systems, College of Civil Engineering and Architecture. His research interests include on-demand ride services, travel behavior modeling, travel demand forecasting, and machine learning.

Xiqun (Michael) Chen received the B.E. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2008 and 2012, respectively. He is currently a "Hundred Talents Program" Professor and a Principal Investigator with the Institute of Intelligent Transportation Systems, College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, China. He has published one book and three book chapters, over 60 peer-reviewed journal papers, and over 50 conference papers. His research interests include intelligent transportation systems, shared mobility, traffic flow theory, and simulation-based optimization. He was a recipient of the 2013 IEEE Intelligent Transportation Systems Society Best Ph.D. Dissertation Award for his thesis entitled "Stochastic Evolutions of Dynamic Traffic Flow: Modeling and Application".

Yitian Jia received the B.S. degree in electrical and electronics engineering and the master's degree in computer software engineering from Zhejiang University, Hangzhou, China, in 2012 and 2014, respectively. His was a Senior Algorithm Engineer with the Didi Research Institute, Didi Chuxing, and contributed to this work. He is currently a Machine Learning Engineer with Tubi TV.

Pinghua Gong received the B.S. degree from the Department of Automation, Xi'an Jiaotong University, Xi'an, China, in 2008, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2013. His is currently a Principal Scientist at Didi Chuxing, Beijing. He has published more than 20 papers in top-tier journals and conferences, including JMLR, NIPS, ICML, KDD, IJCAI, and AAAI. His research interests include machine learning and data mining.

Jieping Ye received the Ph.D. degree in computer science from the University of Minnesota, Twin Cities, MN, USA, in 2005. He is the Head of Didi AI Labs, the VP of Didi Chuxing, and a Didi Fellow. He is also an Associate Professor with the University of Michigan, Ann Arbor, MI, USA. His research interests include big data, machine learning, and data mining with applications in transportation and biomedicine. He was a recipient of the NSF CAREER Award in 2010. His papers have been selected for the Outstanding Student Paper at ICML in 2004, the KDD Best Research Paper Runner Up in 2013, and the KDD Best Student Paper Award in 2014. He has served as a Senior Program Committee/Area Chair/Program Committee Vice Chair of many conferences, including NIPS, ICML, KDD, IJCAI, ICDM, and SDM. He serves as an Associate Editor for *Data Mining and Knowledge Discovery*, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.