IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 25, 2019, accepted February 17, 2019, date of publication February 25, 2019, date of current version March 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2901289

Learn, Assign, and Search: Real-Time Estimation of Dynamic Origin-Destination Flows Using Machine Learning Algorithms

JISHUN OU[1], JIAWEI LU[2], JINGXIN XIA[1], CHENGCHUAN AN[1], AND ZHENBO LU[1]
[1]Intelligent Transportation System Research Center, Southeast University, Nanjing 211189, China
[2]School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ 85281, USA

Corresponding author: Jingxin Xia (xiajingxin@seu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 71871055, and in part by the Key Research and Development Program of Jiangsu, China, under Grant BE2017027.

ABSTRACT A common way to estimate dynamic origin–destination (O-D) flows is to establish and solve a bilevel optimization model. Though numerous efforts have been devoted to effectively and efficiently solving the model, challenges still exist because of the interdependence of jointly solving the upper level O-D estimation and lower level traffic assignment problems and the nonconvexity of the model. This paper presents an alternative framework for estimating dynamic O-D flows using machine learning algorithms. The framework consists of three major modules: a learner that learns the dynamic mapping patterns describing the relationship between prior O-D flows and observed link flows, an assigner that assigns a given O-D matrix to different links based on the learner, and a searcher that iteratively searches the optimal O-D solution using the assigner. A convolutional neural network is designed as the learner and trained as the assigner. Next, the algorithms to estimate a regular O-D matrix and real-time O-D flows are separately developed by using the assigner and two designed genetic algorithms built as the searcher. The framework was evaluated with a realistic network in the downtown area of Kunshan, China. The experimental studies show that the framework can achieve satisfactory estimation performances in real time. Meanwhile, it takes raw flow ranges as the prior inputs, making it robust in the case of lacking an accurate target O-D matrix.

INDEX TERMS Dynamic O-D estimation, bilevel optimization, convolutional neural network, genetic algorithm.

I. INTRODUCTION

Real-time estimation of dynamic origin-destination (O-D) flows plays an important role in proactive traffic control and dynamic route guidance in intelligent transportation systems (ITSs). For a general network, e.g. an open urban network with limited number of links, a typical way for acquiring dynamic O-D flows is to estimate them indirectly from observed link flows [1]–[3]. To achieve this goal, two major issues need to be properly tackled. The first lies in how to formulate and solve the problem of dynamic O-D estimation (DODE) with a well-designed optimization model in an effort to minimize the difference between the estimated O-D matrix and the target O-D matrix. The second concerns on developing a dynamic traffic assignment (DTA) model to establish the dynamic mappings between the estimated O-D flows and the observed link flows. The two issues need to be solved endogenously, meaning the sought O-D matrix should not be determined by using an independent DTA process. The underlying reason is that there is an inconsistency between the O-D matrix estimated from the observed link flows and the link flows obtained by assigning the O-D matrix to the network when using the same set of route choice proportions [4], [5]. In a realistic congested network, the inconsistency would become more apparent.

An effective way to avoid the inconsistency is to combine the DODE and DTA problems into one solution framework. Formally, the framework can be formulated as a bilevel optimization model, where the upper-level is to solve a DODE problem using a constrained objective function

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Awais Javed.

(e.g. generalized least squares) or a state transition equation (e.g. Kalman filter), and the lower-level is to solve a DTA problem satisfying a dynamic user equilibrium (DUE) constraint. Because evaluating the upper-level objective function requires solutions of the lower-level problem, it is generally difficult to solve the bilevel optimization model [6]. More specifically, the intrinsic nonlinearity of the lower-level DTA problem can easily cause the nonconvexity of the bilevel optimization problem, and consequently, a global optimum might be hard to find [5]. For real-time application scenarios, the solving complexity of the problem would be further increased because of the high computational efficiency requirements.

To find a satisfactory DUE solution to the DTA problem, it usually needs to establish an iterative solving process since ''each traveler's best route choice depends on congestion levels throughout the journey, which in turn depend on the route choices and progress through the network of other travelers who depart earlier, at the same time or later'' [7]. Moreover, a DTA process commonly requires an elegant design and a careful calibration, making the DTA a quite challenging task in transportation network modeling. For the DODE problem, a key role of a DTA model is to establish the dynamic mappings between the estimated time-varying O-D flows and the assigned link flows. By iteratively using the established mappings to assign O-D flows to different links, the DTA model gradually converges to a DUE solution. Because the establishment of the dynamic mappings involves an iterative and intractable assignment modeling process, a question is raised: can we learn and capture the dynamic mapping patterns for real-time dynamic O-D estimation with a satisfactory accuracy but a low computational cost?

With the advances of computational power of modern computers, artificial intelligent techniques, in particular machine learning algorithms, achieve breakthroughs during the past decade. The power of these algorithms lies in their ability to effectively learn complex and hierarchical features and patterns from available data. A recent study conducted by Wu *et al.* [8] developed a deep learning network model to estimate hierarchical travel demand, offering a promising exploration direction for using machine learning algorithms to solve travel demand estimation problems. In their study, the machine learning algorithms were tailored to estimate travel demand for transportation planning purpose. In this paper, we develop a framework which introduces machine learning algorithms into real-time dynamic vehicle flow O-D estimation for online traffic management purpose. The framework consists of three major modules: a learner, an assigner and a searcher. A convolutional neural network (CNN) is designed as the learner to capture the patterns that characterize the dynamic mappings between the estimated O-D flows and the assigned link flows. Then, the methods to estimate a regular O-D matrix and real-time O-D flows are sequentially developed by using the trained CNN as the assigner and two developed Genetic Algorithms (GAs) as the searcher.

The contributions of this study are highlighted on the following aspects:

(1) A learning-assigning-searching framework to solve the real-time DODE problem is proposed. Compared with a classical solution framework with an iterative and intractable traffic assignment modeling process, the proposed framework develops a learning-assigning strategy to implement a one-time assignment, significantly speeding up the solving process of the DODE problem.

(2) A CNN-based model to learn the patterns that characterize the dynamic mappings between the estimated time-varying O-D flows and the assigned link flows is developed. In the model, the time-varying O-D flows are encoded as a tensor and the assigned link flows are encoded as a vector, while a series of CNN components are tailored to extract intrinsic and hierarchical features and patterns from a prior O-D data set.

(3) Two GA-based algorithms are separately presented for offline O-D pattern extraction and real-time dynamic O-D estimation. Unlike the O-D estimation methods that heavily rely on an accurate target O-D matrix, the presented algorithms take raw flow ranges as the inputs, making them more robust in the case of lacking an accurate target O-D matrix.

II. LITERATURE REVIEW

For a general network, two issues are usually required to estimate dynamic O-D flows, i.e. the upper-level dynamic O-D estimation problem and the lower-level DTA problem, though some single-level based approaches have been proposed [9]–[12].

To deal with the first issue, three categories of methods have been proposed. The first formulates the O-D estimation problem using constrained statistical optimization functions, e.g. entropy maximization, generalized least squares, and Bayesian inference. Willumsen [13] first extended the entropy maximization model to handle the dynamic traffic counts for a general network. Cascetta *et al.* [1] presented two types of generalized least squares estimators to infer O-D demand in a dynamic context. One is a simultaneous estimator that estimates the entire set of dynamic O-D flows in one step using link traffic counts from all time intervals, while the second is a sequential estimator that derives the O-D flows using both previous O-D estimates and the current and previous traffic counts. Due to its ability to incorporate prior information in a natural manner, the Bayesian inference method was proposed by Hazelton [14] to estimate time-varying O-D matrices from link count data collected on a daily basis. As described, a significant advantage of the Bayesian approach is that, given the Bayesian posterior distribution for model parameters, the confidence and prediction intervals can be constructed to provide guidance to the range of estimated O-D flows. Note that many of the statistical functions mentioned above are equivalent under certain assumptions [15].

The second category of the methods aims to build a state space model to iteratively estimate dynamic O-D flows.

Typical methods include the Kalman filtering and its variants. Okutani [16] developed a standard linear Kalman filter model to obtain optimal estimates of dynamic O-D flows. Although the model has proven to be potential for online applications, the use of the autoregressive transition equation makes it hard to capture complex structural information on trip making [17]. To overcome this shortcoming, Ashok and Ben-Akiva [17], [18] improved the transition equation by replacing O-D flows with O-D flow deviations. The use of the deviations better supports the required normality assumptions, but requires the knowledge of historical O-D flows [19]. Moreover, the stationary assumption in the autoregressive model could not be satisfied. In view of this, Zhou and Mahmassani [20] presented a structural state space model to incorporate regular demand information, structural deviations and random fluctuations. Similarly, Lu *et al.* [21] developed a Kalman filter model for dynamic O-D estimation using multi-source sensor data. Experiments show the estimation performance can be effectively improved.

The third category of the methods uses heuristic search algorithms since the DODE problem is usually formulated as a nonconvex optimization problem. Yin [6] proposed a GA-based (GAB) approach to solving bilevel programming problems, and concluded the approach is more efficient than the sensitivity-analysis-based approach presented by Yang [5] in a simpler manner. Subsequently, Kim *et al.* [22] proposed a GAB approach for the cases with significant differences between the target and true O-D matrices. Later, a series of work has been explored with respect to using heuristic search algorithms, e.g. Baek *et al.* [23], Stathopoulos and Tsekeris [24], Yun and Park [25], Kattan and Abdulhai [26], Park and Zhu [27], and Huang *et al.* [28].

To solve the lower-level DTA problem, the analytical and simulation-based DTA models have been presented, most of which are subject to a user equilibrium constrain. Earlier analytical studies [29]–[31] simply used link performance functions to determine path travel costs, and suffer from the limitation to represent the dynamics and complexity of real-world traffic flow systems [12]. Therefore, simulation-based DTA models have been proposed. Zhou and Mahmassani [2] employed the DYNASMART-P simulation program [32] to obtain the link flow and link-to-link flow proportions for DODE. Huang *et al.* [28] applied the microscopic simulation tool, TRANSIMS, to estimate dynamic O-D flows with hourly traffic counts. Toledo and Kolechkina [33] proposed a scheme for off-line DODE problem, where the mesoscopic traffic simulation model, Mezzo, was adopted to conduct network loading. Other simulation models employed include VISSIM [34], DynaMIT [35], PARAMICS [36], etc.

In a bilevel solution framework, the lower-level DTA model is to establish the dynamic mappings between the estimated time-varying O-D flows and the assigned link flows. Generally, the establishment of the dynamic mappings involves an iterative and intractable assignment modeling process. A potential idea is that if we can learn and capture the patterns of the dynamic mappings, the iterative process could be thus avoided to speed up the solving process of the DODE problem. For a given road network, though the mappings vary time-to-time, they show time-of-day and day-of-week patterns because of regular travel behaviors and traffic conditions in the network. In the following sections, we try to present a solution framework for the real-time DODE problem using machine learning algorithms which can learn and capture the patterns.

III. METHODOLOGY

A. PROBLEM STATEMENT AND DEFINITIONS

In general, the DODE problem that uses observed link flows to indirectly infer dynamic O-D flows can be formulated as the following optimization problem:

$$\min_{d_t, v_t} \left[F_1\left(d_t, \bar{d}_t\right) + F_2\left(v_t, \bar{v}_t\right) \right], \tag{1}$$

subject to

$$v_t = M\left(d_{t-\delta}, \ldots, d_{t-1}, d_t\right), \tag{2}$$

where

t is the time interval that is usually defined as 15-min for online traffic management and control, as adopted in this study,

d_t is the estimated O-D matrix (represented as vector) in time interval t whose element, d_t^i $(1 \le i \le n)$, is the trips between the i^{th} of n O-D pairs in the network,

$\bar{d}_t$ is the target O-D matrix in time interval t obtained by traffic surveys or a prior estimator,

v_t is the vector of estimated link flows in time interval t whose element, v_t^j $(1 \le j \le m)$, is the flow on the j^{th} of m links in the network,

$\bar{v}_t$ is the vector of observed link flows in time interval t,

$F_1\left(d_t, \bar{d}_t\right)$ represents the function of generalized distance measurement or errors between d_t and $\bar{d}_t$, e.g. Euclidean distance function or entropy function,

$F_2\left(v_t, \bar{v}_t\right)$ represents the function of generalized distance measurement or errors between v_t and $\bar{v}_t$,

δ is the maximum number of time intervals during which the O-D flows leave origins and reach target links, i.e. v_t is made up of time-varying O-D flows $d_{t-\delta}, \ldots, d_{t-1}, d_t$, and $M\left(d_{t-\delta}, \ldots, d_{t-1}, d_t\right)$ represents the function of the dynamic mappings that describe the relationship between the time-varying O-D flows $d_{t-\delta}, \ldots, d_{t-1}, d_t$ and the observed link flows v_t.

B. PROPOSED FRAMEWORK

Figure 1 shows the proposed framework of the methodology with four major steps. Step 1 collects and prepares reliable traffic flow data by means of a series of data preprocessing operations. In Step 2, a CNN model is developed as the learner to capture the dynamic mappings between the estimated time-varying O-D flows and the assigned link flows. In Step 3, the CNN model trained with the prior O-D data set is used as the assigner and the offline O-D pattern (regular O-D matrix) is estimated as a template for real-time

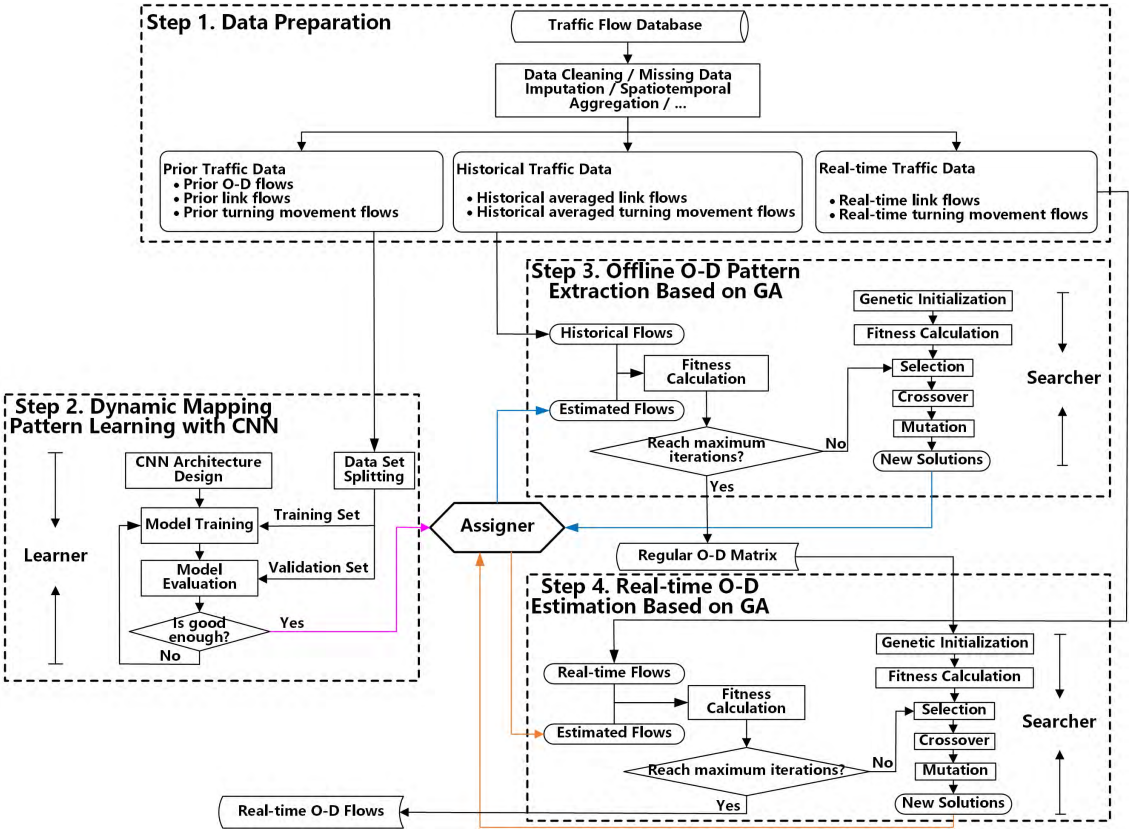

FIGURE 1. The proposed framework for real-time DODE.

O-D estimation. In Step 4, by taking the regular O-D matrix as the initialization solution, the GA combined with the assigner are utilized to estimate real-time O-D flows using real-time link flows and turning movement flows. Note that Step 2 and 3 are conducted in an offline manner, and could be monthly or seasonally updated, depending on traffic condition patterns for the network.

C. DATA PREPARATION

The aim of the data preparation is to provide clean and reliable traffic data for real-time O-D estimation. To this end, several preprocessing operations need to be carried out. For instance, data cleaning operation removes duplicate and noisy data records from raw data, while missing data imputation operation attempts to estimate missing values for incomplete traffic flow data. In addition, spatiotemporal aggregation operations are conducted to obtain the 15-min aggregated link flows and turning movement flows for DODE.

Three sets of traffic data are required for real-time DODE. The first is the prior O-D flows and corresponding link flows and turning movement flows. The prior data is used to train the CNN model to capture the dynamic mapping patterns, and can be obtained based on traffic simulation, trajectory reconstruction [37], or other prior O-D estimation methods. In this paper, we adopt the traffic simulation technique, as shown in the empirical analysis section. The second set consists of historical averaged link flows and turning movement flows at 15-min intervals by time-of-day and day-of-week for regular O-D matrices estimation. The third set is composed of real-time 15-min link flows and turning movement flows, which are utilized to estimate time-varying O-D flows in real time.

D. DYNAMIC MAPPING PATTERN LEARNING WITH CNN

A key role of a DTA model under a bilevel solution framework is to assign given O-D flows to different links for a given network. In other words, the DTA model aims to establish the dynamic mappings between the estimated O-D flows and the assigned link flows. With this in mind, a CNN is designed as the learner to learn the dynamic mapping patterns from the prior data set. As the dynamic mapping patterns could be subject to the change of different traffic control strategies (e.g. signal metering) and the occurrence of special traffic events (e.g. work zone), it is necessary to consider these specific traffic scenarios when establishing the CNN model. However, since a CNN model is capable of capturing various patterns from available data, it does not require to be trained separately for different traffic scenarios, provided the used data is able to fully reflect the corresponding patterns.

1) CNN ARCHITECTURE DESIGN

In essence, CNN is a special category of feed-forward artificial neural networks, and has been successfully applied to

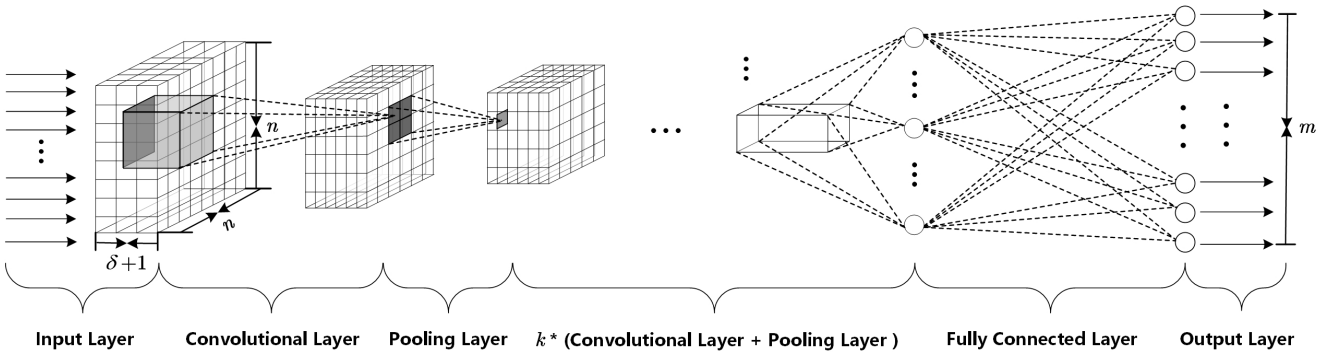

FIGURE 2. Architecture of the designed CNN.

various application fields. In contrast with the conventional feed-forward neural networks, e.g. multilayer perceptron (MLP), the CNN exploits spatially local correlation in data structure by connecting neurons of adjacent layers with a local connectivity pattern [38]. That is, each neuron in the current layer is only connected to a small region of the previous layer. The region is called the receptive field of the connected neuron. By scanning and stacking receptive fields of many layers, the CNN is able to extract complex and hierarchical features and patterns from data. Such an ability makes the CNN one of the most popular machine learning algorithms during the past decade [39].

A CNN is composed of a sequence of layers, including input layer, convolutional layer, pooling layer, fully connected layer, and output layer. These layers are assembled into a pipeline where data is input, transformed, correlated and finally output. Figure 2 illustrates the architecture of the designed CNN. It has one input layer, $k + 1$ convolutional layers, $k + 1$ pooling layers, one fully connected layer, and one output layer. The parameter k is empirically determined in the later experiments.

- Input layer and output layer

In the CNN model, the prior O-D flows are used as the inputs, while the outputs are the prior link flows and turning movement flows. As defined in Section III. A, the link flows v_t is made up of the time-varying O-D flows $d_{t-\delta}, \ldots, d_{t-1}, d_t$, where v_t has m elements, and d_t has $n \times n$ elements. Therefore, for the CNN, the input is encoded as an $n \times n \times (\delta + 1)$ tensor and the output is encoded as an $m \times 1$ vector. Note that the turning movement flows are included in v_t because the performance of a DODE model can be effectively improved by incorporating turning movement flows into the estimation process [40]. The parameter δ varies from network to network, and can be determined by computing the longest trip time in the study network. For instance, if the longest trip time regarding the network is 17 minutes and the estimation time interval is defined as 15-min, δ can be simply set as 2.

- Convolutional layer

The convolutional layer serves as feature extractors to learn multiple feature representations from the inputs of the layer. The inputs are often convolved with a set of filters to calculate new feature representations. The convolution process is equivalent to computing dot products between the filters and the inputs. The obtained feature representations are called feature maps each of which is a 2-dimensional array. After stacking the feature maps of all filters along the depth dimension, an $n \times n \times c$ tensor can be obtained, where c is the number of the filters. To introduce nonlinearity into the designed CNN, a nonlinear activation function is required to act on each element of the tensor. According to Glorot *et al.* [41], the rectified linear units (ReLU) function allows much smaller training times than the sigmoid and hyperbolic tangent functions, and is chosen as the activation function of the convolutional layers. The ReLU function is mathematically defined as $f(x) = max(0,x)$, where x represents an element of the input tensor. In addition, to control the size of the output, zero padding strategy is adopted to symmetrically add zeros around borders of the input tensor.

- Pooling layer

The pooling layer is to reduce the spatial dimension of the feature maps, and hence to control overfitting while achieving spatial and scale invariance to input distortions and translations [42]. Max pooling is one of the most common pooling strategies, which operates independently on each feature map and resizes it spatially. Formally, the max pooling strategy selects the largest element within each receptive filed such that

$$y_{a,b}(c) = \max_{(p,q)\in R_{a,b}} x_{p,q}(c), \tag{3}$$

where

$R_{a,b}$ is the receptive field associated with the c^{th} feature map,

$y_{a,b}(c)$ is the output obtained by conducting the pooling operation on $R_{a,b}$, and

$x_{p,q}(c)$ represents the element of the c^{th} feature map at location (p, q) contained by $R_{a,b}$.

- Fully connected layer

The fully connected layer aims to interpret the extracted features and perform high-level reasoning by establishing full

connections between the output of the previous layer and the input of the next layer. In order to establish mappings from the extracted features to final outputs, a corresponding mapping function is required. Because of the existence of the nonlinearity relationship between the time-varying O-D flows and the link flows, a nonlinear function named Smooth Adaptive Activation Function (SAAF) is adopted because it can decrease model bias and complexity simultaneously [43]. The SAAF is in essence a piecewise polynomial. Given $\xi+1$ real numbers $a_1, a_2, \ldots, a_{\xi+1}$ in ascending order and a non-negative integer η, the SAAF is defined as:

$$f(x) = \sum_{c=0}^{\eta-1} \alpha_c \mathrm{p}^c(x) + \sum_{l=1}^{\xi} \beta_l \mathrm{b}_l^{\eta}(x), \tag{4}$$

$$\mathrm{p}^c(x) = \frac{x^c}{c!}, \tag{5}$$

$$\mathrm{b}_l^{\eta}(x) = \underbrace{\int \int \cdots \int_0^x}_{\eta \text{ times}} \mathrm{b}_l^0(\gamma)\, d^{\eta}\gamma, \tag{6}$$

$$\mathrm{b}_l^0(x) = \mathrm{I}(a_l \le x < a_{l+1}), \tag{7}$$

where

η is the degree of polynomial segments,
a_l represents the break points,
p^c and b_l^{η} are basis functions,
α_c and β_l are learned parameters, and
$\mathrm{I}(\cdot)$ is the indicator function.

From the definition, we can see that the SAAF is a linear combination of the defined basis functions with adjusted weights. As a piecewise quadratic SAAF can already achieve satisfactory results, η is set as 2 and a_l is distributed in $[-1.1, 1.1]$, as suggested by Hou *et al.* [43]. The number of segments ξ is randomly chosen as 15 based on the proof that the model complexity can be bounded regardless of the number of polynomial segments [43].

2) CNN INITIALIZATION

CNN initialization, namely weight initialization, has a profound impact on both the convergence rate and generalization ability of the designed network. Large weights lead to divergence, while small weights do not allow the network to learn. In this study, a robust initialization method presented by He *et al.* [44] is adopted. It initializes the biases to be 0 and samples the initial weights W_l at corresponding layers from the zero-mean Gaussian distribution whose standard deviation is $\sqrt{2/n_l}$, that is, $W_l \sim \mathcal{N}\left(0, \frac{2}{n_l}\right)$. Here, n_l is the number of connections from the previous layer.

3) CNN TRAINING

After the architecture design and network configuration, the CNN model is trained with the back propagation algorithm that uses gradient descent optimization strategy to update the parameters with the aim of minimizing the loss function:

$$\mathcal{L}(\theta) = \frac{1}{2N} \sum_{h=1}^{N} \left(\hat{y}_h - y_h\right), \tag{8}$$

where

$\hat{y}_h$ is the predicted value of the h^{th} sample, representing the assigned link flows and turning movement flows,
y_h is the true value of the h^{th} sample, representing the observed link flows and turning movement flows, and
N is the size of the training set.

A variety of gradient descent optimization strategies have been proposed in recent years. Among them, Adam [45] is the most straightforward and popular one. Denote θ_s as the learned parameter vector at timestep s, the update rule of the Adam is described by the following equations:

$$\theta_s = \theta_{s-1} - \alpha \cdot \frac{\omega_s}{1-\beta_1^s} \Big/ \left(\sqrt{\frac{v_s}{1-\beta_2^s}} + \epsilon\right), \tag{9}$$

$$\omega_s = \beta_1 \cdot \omega_{s-1} + (1-\beta_1) \cdot g_s, \tag{10}$$

$$v_s = \beta_2 \cdot v_{s-1} + (1-\beta_2) \cdot g_s^2, \tag{11}$$

$$g_s = \nabla_\theta \mathcal{L}_s(\theta_{s-1}), \tag{12}$$

where

g_s is the gradient of loss function $\mathcal{L}_s$ regarding θ at timestep s,
ω_s is the 1^{st} moment (mean) of the gradient, $\omega_0 = 0$,
v_s is the 2^{nd} raw moment (uncentered variance) of the gradient, $v_0 = 0$,
$\beta_1, \beta_2 \in [0, 1)$ are the exponential decay rates for the moment estimates, and the default values are set as: $\beta_1 = 0.9$, $\beta_2 = 0.999$,
α is the step size that controls the update rate of θ, and the default value is set as 0.001,
ϵ is a small constant for avoiding the divisor is 0 and commonly set as 10^{-8}.

The parameter vector θ is iteratively updated during the training phase. In addition, a validation data set is required to evaluate the performance of the trained model during each training epoch. When the model achieves the best performance on the validation set, the training will be terminated.

E. OFFLINE O-D PATTERN EXTRACTION BASED ON GA

The offline O-D pattern extraction is to estimate a template O-D matrix based on historical averaged link flows and turning movement flows. To this end, an O-D pattern estimation algorithm based on GA is presented. Figure 3 shows the pseudocode of the algorithm. The main input is a set of historical averaged flows (i.e. link flows and turning movement flows) collected over T time intervals of the study period. The output is an O-D pattern that is composed of the regular O-D matrices regarding the T time intervals.

The method starts at generating a set of random solutions $D = \{D_h \mid 1 \le h \le N\}$ in a constrained search space $\mathcal{R} = \{\tau_c\}_{c=1}^{n*n*T}$, where τ_c is the range that the c^{th} of $(n*n*T)$ O-D flow falls into. Here, n is the number of O-D pairs in the network. In other words, each solution of the GA is encoded as a group of integers representing $(n*n*T)$ O-D flows during the T time intervals. To measure the quality of each solution, a proper fitness function is required, which is

Algorithm 1: Offline O-D Pattern Extraction Based on Heuristic Genetic Search

Input: Historical averaged link flows $V^* = \{\bar{v}^*_1, \dots, \bar{v}^*_{T-1}, \bar{v}^*_T\}$ associated with T time intervals; predefined search space $\mathcal{R}$; population size N; crossover probability μ; mutation probability λ; number of maximum iterations I

Output: O-D pattern $D^* = \{\bar{d}_1, \dots, \bar{d}_{T-1}, \bar{d}_T\}$ associated with T time intervals $1, \dots, T-1, T$

Process:

Create initial population $D = \{D_1, D_2, \dots, D_N\}$ by randomly generating solutions in $\mathcal{R}$
$D' \leftarrow \emptyset$ *# D' is a set to store candidate solutions generated by selection, crossover and mutation*
$\rho \leftarrow \{\rho_h = 0 \mid 1 \le h \le N\}$ *# ρ is a fitness set associated with the generated solutions*
for $i' = 1, 2, \dots, I$ **do**
  for $h = 1, 2, \dots, N$ **do** *# step 5 ~ step 7 are to compute the fitness for a given solution*
    $\rho_h \leftarrow computeFitness(D_h, V^*)$
  end for
  for $k' = 1, 2, \dots, N/2$ **do** *# step 8 ~ step 12 are to generate new solutions by selection and crossover*
    $\langle D_{oldA}, D_{oldB} \rangle \leftarrow select(D, \rho, k')$
    $\langle D_{newA}, D_{newB} \rangle \leftarrow crossover(D_{oldA}, D_{oldB}, \mu, k')$
    add D_{newA} and D_{newB} to D'
  end for
  for $l = 1, 2, \dots, N$ **do** *# step 13 ~ step 16 are to generate new solutions by mutation*
    $D_{newC} \leftarrow mutate(D'_l, \lambda)$
    replace D'_l with D_{newC} in D'
  end for
  $D \leftarrow D'$
  $D' \leftarrow \emptyset$
end for
$D^* \leftarrow \underset{1 \le l \le N, D_l \in D}{\operatorname{argmax}} fitness(D_l)$

FIGURE 3. Pseudocode of the presented offline O-D pattern extraction algorithm.

defined as follows:

$$fitness\,(D_h) = \frac{1}{\sum_{t=\delta+1}^{T} \left(\frac{1}{m} \sum_{j=1}^{m} \sqrt{\frac{2\left(\hat{v}_t^j - \bar{v}^{*j}_t\right)^2}{\hat{v}_t^j + \bar{v}^{*j}_t}} \right)}, \quad (13)$$

where

$\hat{v}_t^j$ represents the assigned historical averaged flow on the j^{th} link during the time interval t for the solution D_h,

$\bar{v}^{*j}_t$ represents the observed historical averaged flow on the j^{th} link during the time interval t for the solution D_h, and m is the number of monitored links in the network.

As the equation shows, the fitness of a solution is defined as the reciprocal of GEH statistic which contains both information from the absolute error and the absolute percentage error [46]. The higher the fitness, the higher the probability of the solution being selected. After computing the fitness of all candidate solutions, the probability of a solution being selected is calculated as:

$$p\,(D_h) = \frac{fitness\,(D_h)}{\sum_{h=1}^{N} fitness\,(D_h)}. \quad (14)$$

According to the value of p, a pair of solutions are selected to produce a new offspring by conducting the crossover operation with a probability of μ. During this process, a crossover point is randomly determined, and all values beyond this point in either solution are swapped between the two parent solutions. The selection and crossover operations are repeated $N/2$ times to make the population size unchanged. In some cases, the solutions in the initial population have the same value at a particular position, making all future offspring cannot change this value in the whole search process. To prevent this problem, the mutation operation is conducted to introduce some randomness into the search process based on the mutation probability λ. In this situation, the value at the mutation position c is randomly set as an integer within the given range τ_c. The above process is repeated until the maximum number of iterations I is reached. The final solution D^* is determined as the solution with the highest fitness in the candidate solution set D.

F. REAL-TIME O-D ESTIMATION BASED ON GA

In a realistic network, traffic count on a given link is usually contributed by the O-D flows during several continuous time intervals. Therefore, an algorithm for real-time dynamic O-D estimation is presented. The pseudocode of the algorithm is depicted in Figure 4.

In contrast with the offline O-D pattern extraction algorithm, the real-time algorithm takes the real-time link and turning movement flows and the extracted O-D pattern D^* as the inputs. During initialization stage, a set of N solutions are produced by generating random integers in the range constrained by D^*. For instance, an O-D flow in a certain time interval is 50 (veh/15 min), the range is defined as an integer interval which allows the flow to fluctuate around it about 50 percent, i.e. [25, 75]. This initialization strategy ensures the method to produce reasonable O-D solutions. After that, the generated solutions are evaluated by the following fitness

Algorithm 2: Real-time O-D Estimation Based on Heuristic Genetic Search

Input: Observed real-time link flows $\bar{v}_t$ in time interval t; O-D pattern $D^* = \{\bar{d}_1, \ldots, \bar{d}_{T-1}, \bar{d}_T\}$; population size N; crossover probability μ; mutation probability λ; number of maximum iterations I

Output: Real-time time-dependent O-D flows $\widetilde{D} = \left\{\tilde{d}_{t-\delta}, \ldots, \tilde{d}_{t-1}, \tilde{d}_t\right\}$ associated with δ time intervals $t-\delta, \ldots, t-1, t$

Process:

Create initial population $D = \{D_1, D_2, \ldots, D_N\}$ according to the O-D pattern D^*
$D' \leftarrow \emptyset$ *# D' is a set to store candidate solutions generated by selection, crossover and mutation*
$\rho \leftarrow \{\rho_h = 0 \mid 1 \le h \le N\}$ *# ρ is a fitness set associated with the generated solutions*
for $i' = 1, 2, \ldots, I$ **do**
  for $h = 1, 2, \ldots, N$ **do** *# step 5 ~ step 7 are to compute the fitness for a given solution*
    $\rho_h \leftarrow computeFitness(D_h, \bar{v}_t)$
  end for
  for $k' = 1, 2, \ldots, N/2$ **do** *# step 8 ~ step 12 are to generate new solutions by selection and crossover*
    $\langle D_{oldA}, D_{oldB}\rangle \leftarrow select(D, \rho, k')$
    $\langle D_{newA}, D_{newB}\rangle \leftarrow crossover(D_{oldA}, D_{oldB}, \mu, k')$
    add D_{newA} and D_{newB} to D'
  end for
  for $l = 1, 2, \ldots, N$ **do** *# step 13 ~ step 16 are to generate new solutions by mutation*
    $D_{newC} \leftarrow mutate(D'_l, \lambda)$
    replace D'_l with D_{newC} in D'
  end for
  $D \leftarrow D'$
  $D' \leftarrow \emptyset$
end for
$\widetilde{D} \leftarrow \begin{cases} \bar{d}_t & if \quad 1 \le t \le \delta \\ \underset{1 \le l \le N, D_l \in D}{\operatorname{argmax}} \; fitness(D_l) & if \quad \delta + 1 \le t \le T \end{cases}$

FIGURE 4. Pseudocode of the presented real-time dynamic O-D estimation algorithm.

function:

$$fitness\,(D_h) = \frac{1}{\left(\frac{1}{m}\sum_{j=1}^{m}\sqrt{\frac{2\left(\hat{v}_t^j - \bar{v}_t^j\right)^2}{\hat{v}_t^j + \bar{v}_t^j}}\right)}, \tag{15}$$

where

$\hat{v}_t^j$ represents the assigned real-time link flows and turning movement flows on the j^{th} link in the time interval t for the solution D_h, and

$\bar{v}_t^j$ represents the observed real-time link flows and turning movement flows on the j^{th} link in the time interval t for the solution D_h.

The following selection-crossover-mutation steps are as similar as that of the offline version. After conducting the maximum number of iterations I, the algorithm is terminated, and the optimal solution is determined as the solution with the highest fitness while $\delta + 1 \le t \le T$. While $1 \le t \le \delta$, the corresponding flow vector $\bar{d}_t$ in D^* is used as the alternative solution because during these δ time intervals, no sufficient historical O-D flows can be used to estimate the dynamic O-D flows.

IV. EMPIRICAL ANALYSIS

A. NETWORK DESCRIPTION

The proposed framework is evaluated using a realistic urban road network, which covers the downtown area of Kunshan, China. There are 32 intersections and 143 links in the network. Among the intersections, 27 of them are signal controlled, while the remaining are non-signal controlled.

Traffic flow data were archived through three data collection ways. The first is the use of microwave detectors to collect link flows. The second is the use of high-resolution video detectors to collect turning movement flows. The third is the use of automatic license plate system to collect travel time information. As a consequence, a total of 111 links and 127 turning movements were monitored in the network. In addition, based on the division of the functional district of the network, a total of 25 traffic analysis zones were defined for the dynamic O-D estimation task, as shown in Figure 5.

B. DATA DESCRIPTION

As mentioned above, a total of 111 link flows and 127 turning movement flows were collected for estimating dynamic O-D flows in the study network. The study period covers the morning hours from 6:00 AM to 11:00 AM in June 5 to June 19, 2017. As the estimation time interval is defined as 15-min, the study period was finally divided into 20 time intervals, denoted as TI1, TI2, …, TI20, respectively.

By inspecting the automatic license plate data, we found the longest trip time in the study network is 23 minutes, and the parameter δ was thus set as 2 in the DODE process. Meanwhile, the traffic flow data collected from June 5 to June 18 was used for offline O-D patter extraction, while the corresponding data collected in June 19 was used for real-time DODE performance evaluation.

A key problem in the DODE modeling process is how to acquire prior O-D flows and the corresponding link flows and turning movement flows to train the CNN model.

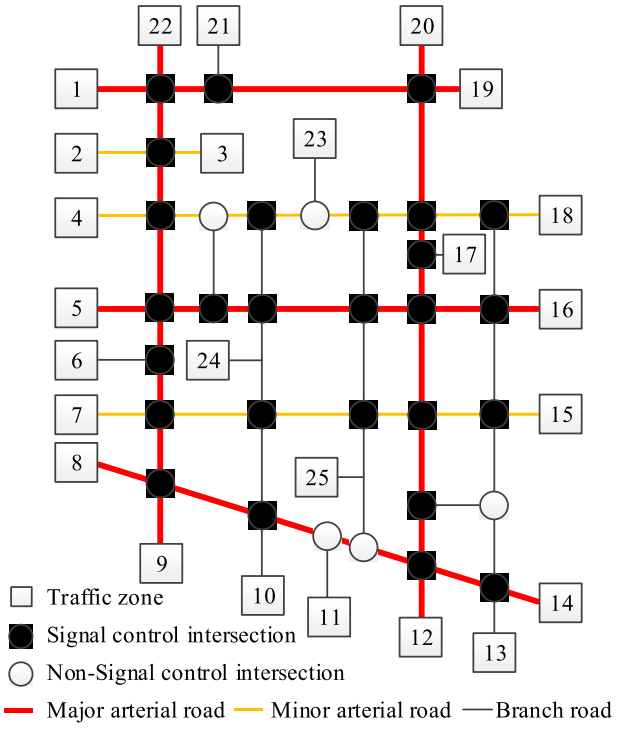

FIGURE 5. Network topology representation.

Traffic survey is one of the most common means for prior O-D data collection. However, such means is unsuitable for our task because numerous samples are required to train a satisfactory CNN model, making it cost-expensive to collect so many samples. Another potential means is to use vehicle trajectory reconstruction technique to obtain prior O-D data. It should be noted that extra cares need to be taken for modeling the market penetration rate factor when using this technique. In this study, we present a simulation-based method to establish the prior data set. The main procedure of the method is described as follows:

a. For a given O-D pair at timestamp t, estimate a raw range of its flow value according to a prior estimator (e.g. a license plate matching estimator or an earlier O-D estimator) or a traffic survey;

b. Generate a random integer within the range and use it as the flow value of the O-D;

c. Repeat steps *a* and *b* for all O-D pairs to generate an O-D matrix X^t (represented as a vector) at timestamp t;

d. Repeat step *c* to generate an O-D matrix for each of the T time intervals in the study period, and finally, we can get an O-D matrix set $\left\{\mathrm{X}^1, \ldots, \mathrm{X}^{T-1}, \mathrm{X}^T\right\}$;

e. Assign the generated O-D matrices to the network using a simulation software, e.g. PARAMICS, and extract the corresponding link flows and turning movement flows $\left\{\mathrm{Y}^1, \ldots, \mathrm{Y}^{T-1}, \mathrm{Y}^T\right\}$;

f. Join $\mathrm{X}^{t-\delta}, \ldots, \mathrm{X}^{t-1}, \mathrm{X}^t$ and Y^t together to produce a prior O-D sample $\mathrm{S}^t = \left(\mathrm{X}^{t-\delta}, \ldots, \mathrm{X}^{t-1}, \mathrm{X}^t, \mathrm{Y}^t\right)$ at timestamp t, where $\delta + 1 \leq t \leq T$;

g. Repeat step *f* to produce a prior O-D sample for each of $(T - \delta)$ time intervals;

h. Generate N prior O-D samples to constitute the prior O-D data set $\left\{\mathrm{S}_i^t\right\}_{i=1|t=\delta+1}^{N|T}$ by repeating step *g*.

In the study, N is set as 1100, that is, a total of 1100 prior data samples were generated using the above way. Among them, 1000 data samples were used as the training set of the CNN model, while the remaining 100 data samples were used as the test set for model evaluation. N was set as a relatively small number because it needs to take a time-consuming simulation process to generate a prior data sample, while the generated data set with $N = 1100$ is sufficient enough to train a well-performed CNN model.

C. PERFORMANCE MEASURES

To evaluate the proposed framework, three performance measures are adopted:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|, \tag{16}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{17}$$

$$\mathrm{GEH} = \sqrt{\frac{2\left(\hat{y}_i - y_i\right)^2}{\hat{y}_i + y_i}}, \tag{18}$$

where

y_i is the actual value of the i^{th} sample for the considered variable,

$\hat{y}_i$ is the estimated value of the i^{th} sample for the considered variable, and

n is the number of samples.

The MAE and MAPE measures are used to quantify the difference between the actual O-D flows and the estimated O-D flows, while all of the three measures are utilized to measure the difference between the actual link flows and the assigned link flows.

D. CNN MODEL VALIDATION

To validate the CNN model, its overall performance on the test set of the prior traffic data was first evaluated. After that, the effect of the number of network layers and the effect of the size of training set on the model performance are separately explored.

1) OVERALL PERFORMANCE EVALUATION

For the CNN model, the input is a $25 \times 25 \times 3$ tensor, and the output is a 238×1 vector which contains 111 link flows and 127 turning movement flows. For illustration purpose, we will use 'link' flows to represent 'link + turning movement' flows afterward.

As mentioned earlier, 100 samples are used to evaluate the CNN model. Thus, a total of 23800 assigned link flows can be estimated. Figure 6 shows the comparisons of the assigned (estimated) and observed (true) link flows during six typical time intervals, i.e. TI3, TI4, TI9, TI10, TI19 and TI20. The time period during from 6:30 AM to 7:00 AM (TI3 and TI4) is the stage when the traffic in the network is becoming

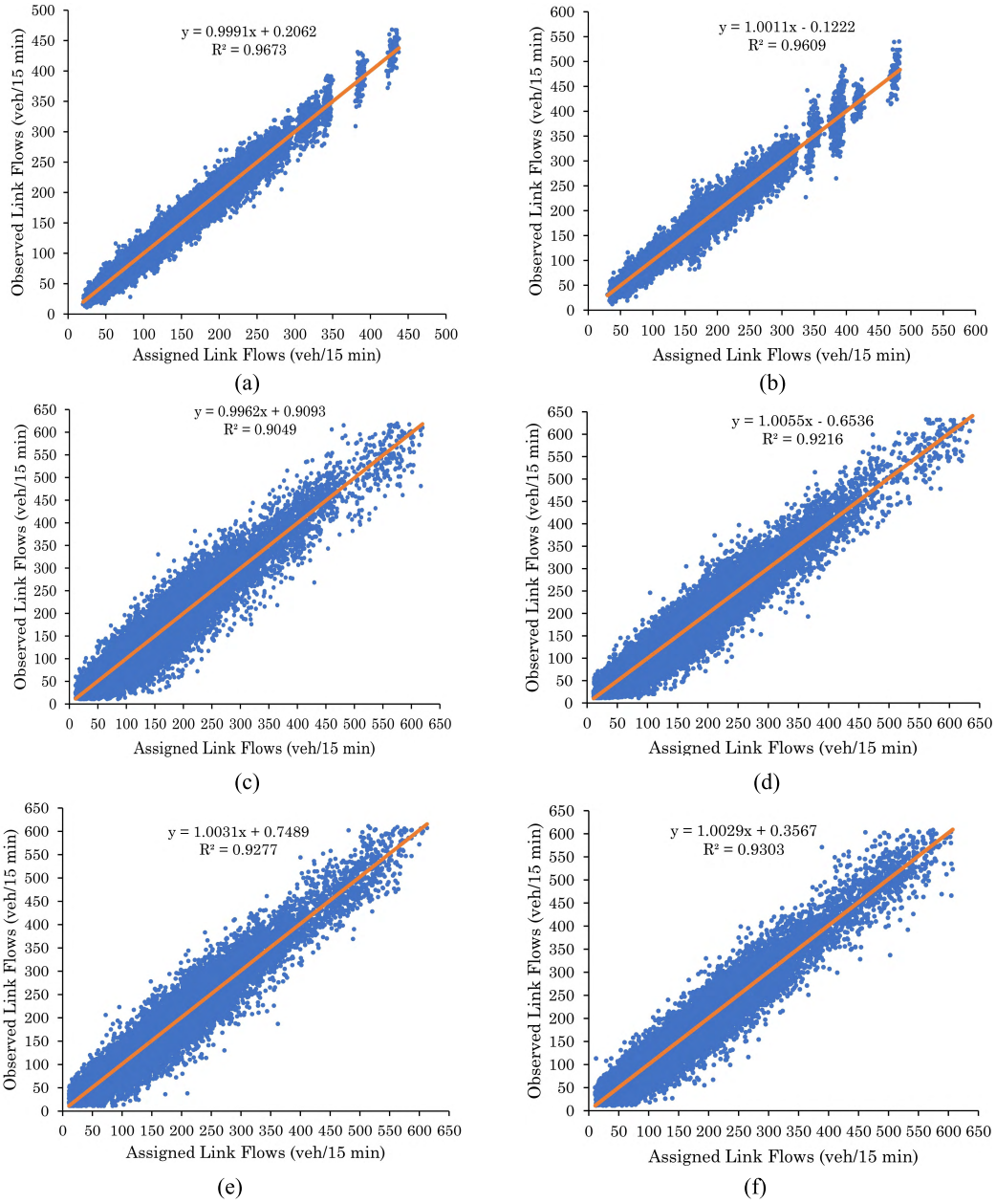

FIGURE 6. Comparisons of the assigned and observed flows of the monitored links and turning movements during typical time periods. (a) TI3 (6:30 AM ∼ 6:45 AM). (b) TI4 (6:45 AM ∼ 7:00 AM). (c) TI9 (8:00 AM ∼ 8:15 AM). (d) TI10 (8:15 AM ∼ 8:30 AM). (e) TI19 (10:30 AM ∼ 10:45 AM). (f) TI20 (10:45 AM ∼ 11:00 AM).

busy, while the period during from 8:00 AM to 8:30 AM (TI9 and TI10) is the most crowded time in the morning. Distinct from the former two periods, in the period during from 10:30 AM to 11:00 AM (TI19 and TI20), the crowded traffic begins to calm down, and flows on the links gradually decrease.

As seen from the figure, the CNN model performs well on the test set. More specifically, all slopes of the fitted linear regression models are approximate to 1.0 and the data points gathered around the regression line, meaning most of the assigned link flows are close to the true link flows. Meanwhile, all coefficients of determination, denoted by R^2, are greater than 0.9, demonstrating at least 90 percent of the data points in each subfigure can be well explained by the fitted regression models. Therefore, it can be seen that the CNN model has learned and captured the patterns that characterize the dynamic mappings between prior time-varying O-D flows and the corresponding link flows. In addition, the R^2 of TI9 and TI10 are less than that of the other time intervals, implying the model performs relatively poorer during the two time intervals. This can also be illustrated by the fact that the data points associated with TI9 and TI10 are more scattered

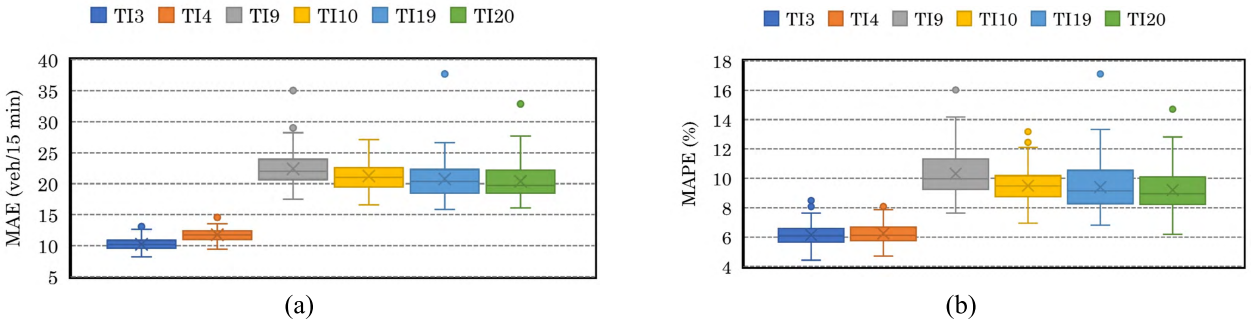

FIGURE 7. MAE and MAPE distributions of the CNN model.

than that of the remaining time intervals. The underlying reason is that during TI9 and TI10, the traffic was more congested than that of the other four time intervals, and hence caused more uncertainties in user route choices, leading to more complicated patterns that are challenging to precisely capture by the model.

To go further in evaluating the overall performance of the CNN model, the MAE and MAPE measures of the assigned link flows from all 625 O-D pairs were calculated, respectively. Figure 7 provides the distributions of the two measures. As the figure shows, the model can produce accurate estimation results during TI3 and TI4 since most MAEs fall into the range between 10 and 15 vehicles every 15 minutes, and both of the averaged MAPEs approximate to 6%. Unsurprisingly, the model performances associated with the other four time intervals are relatively deteriorated. Even so, the averaged MAE and MAPE associated with the most congested time interval TI9 are 23 (veh/15 min) and 11%, respectively, which are both acceptable in real applications. It should be noted that there are several outliers in both measure distributions. Specifically, three evident outliers associated with TI9, TI19 and TI20 can be identified in Figure 7(a). Their values are all greater than 30 (veh/15 min). Similarly, the outliers associated with the same time intervals can also be found in Figure 7(b), and their MAPEs are all greater than 14%. These outliers might be partially explained by the uncertainties resulting from traffic fluctuations in the network.

2) INFLUENCE OF DIFFERENT NUMBER OF NETWORK LAYERS

The number of network layers plays an important role in CNN modeling. Generally, with the increase of the number of network layers, the generalization ability of the model could be further improved. To evaluate the model performance with different number of network layers, the CNN model with five different network layers was tested. As the convolutional layer and the pooling layer are the most key layers that influence the accuracy of CNNs, the CNN model is mainly distinct in the number of the combination of the convolutional and pooling layers, i.e. the parameter k depicted in Figure 2. In the study, k is set as 0, 1, 2, 3, and 4, respectively. In other words, we can check the performance of the CNN model with 5, 7, 9, 11, and 13 network layers through the experiment.

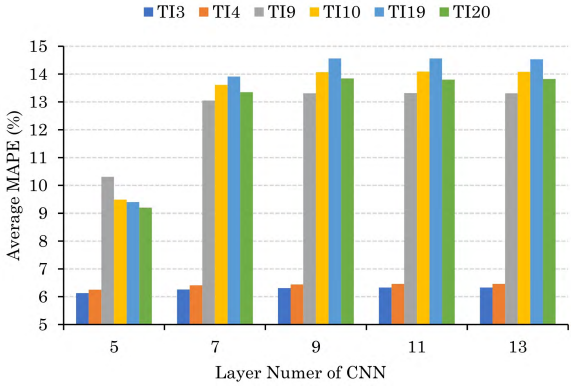

FIGURE 8. Model performance with different number of network layers.

Figure 8 shows the performance comparisons of the CNN model over the test set. It is a little counterintuitive that with the increase of the number of network layers, the average MAPEs of the model does not decrease as expected, especially for TI9, TI10, TI19 and TI20. One possible explanation is that the model with 5 layers is already able to capture the potential patterns in the training set, whereas the increase of the network layer number increases the model complexity, resulting to overfitting to some degree. It should be noted that the performance of a CNN model is closely associated with various settings of CNNs. Therefore, the model performance may be improved by using some advanced parameter adjustment strategies. However, this is beyond the scope of this paper. Considering the 5-layer CNN model can already achieve satisfactory performance with the least training cost, the number of layers is fixed as 5 in the later experiments.

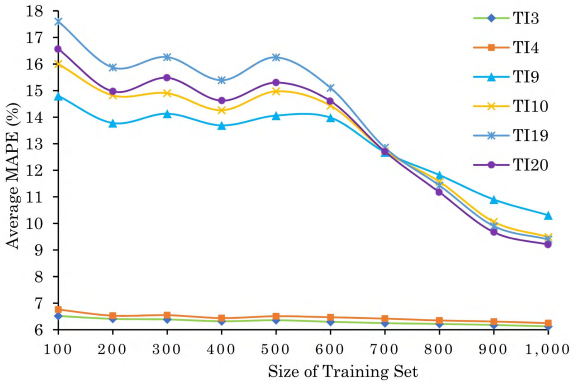

FIGURE 9. Model performance with different size of training set.

3) INFLUENCE OF DIFFERENT SIZE OF TRAINING SET

Another factor to affect the model performance is the size of training set. With this in mind, we tested the CNN model with different training sets, which are distinct in the number of training data samples. The comparison results are illustrated in Figure 9. It is observed that the model performs well when the time intervals are TI3 and TI4. Meanwhile, for the other four time intervals, the model performances have been gradually improved with the increase of the size of the training set. More specifically, when the number of the training samples is less than 700, the model cannot achieve satisfactory results. However, when the sample number raises to 900, the average MAPE begins to be lower than 10%. In addition, it is worth noting that the model performance associated with TI9 is better than that of TI10, TI19 and TI20 when the sample number is below 700, whereas it achieves the worst accuracy eventually. A potential reason is that more traffic congestion could occur in TI9, making the model harder to give accurate estimations because of the complicated dynamic mapping relationship between the O-D flows and the link flows. As the comparison results show, with the increase of the size of the training set, the model performance is further improved. In view of this, the size of the training set is set as 1000 in the later experiments.

E. DODE RESULTS ANALYSIS

Collecting ground-truth O-D flow data is a quite challenging task in reality. Therefore, a traffic simulation model based on Q-PARAMICS was built as the testbed to validate the DODE performance of the proposed framework. The model has been fully calibrated and tested in [21]. In this sense, the link flows and turning movement flows collected from the installed traffic sensors are treated as the observed link flows and turning movement flows, while the O-D flows obtained from the simulation model are treated as the true O-D flows.

1) CONVERGENCE ANALYSIS OF GA

The parameters of the GA for offline O-D pattern extraction were set as: population size $N = 800$, crossover probability $\mu = 0.9$, mutation probability $\lambda = 0.09$, and maximum number of iterations $I = 10000$. The parameters of the GA for real-time dynamic O-D estimation were set as the same as the offline version.

To go through the evolution process of the GA for real-time O-D estimation, the maximum fitness and average fitness in each GA iteration of the real-time O-D estimation algorithm were both recorded, as depicted in Figure 10. From the two subfigures, we can see that the two types of fitness curves have similar convergence trends, which both asymptotically converge to the corresponding peak value, meaning the generated solutions in each iteration evolve toward the optimal solution. Besides, the curves associated with TI3 and TI4 converge earlier than that of the other four time intervals. This reflects the fact that it is more challenging to estimate dynamic O-D flows under congested traffic conditions, as mentioned earlier. Note that the search process in effect can be terminated at about the 6500$^{\text{th}}$ iteration for all of the six time intervals.

The two GA-based models mentioned above were implemented using the MATLAB programming language, and their running times were recorded using a computer equipped with a 3.2 GHz 2-core CPU and a 16 GB RAM. The estimation (assignment) time of the trained CNN model is 0.0096 seconds per sample, significantly faster than that of the simulation-based model which needs about 8.3 seconds for each assignment. This is due to the fact that the CNN model was trained in an offline manner and used to implement an online traffic assignment is instantaneous.

To go further in evaluating the computational efficiency of the DODE process, the DODE computation times regarding the six typical time intervals mentioned earlier were recorded as 202, 197, 736, 741, 711, and 706 seconds, respectively. Note that the time of the DODE process for each time interval was computed as the elapsed time during which the maximum fitness was reached. It can be observed that the computation time of all DODE models are less than 15 minutes, meaning they can be perfectly applied to real-time traffic management.

2) DODE PERFORMANCE EVALUATION AND ANALYSIS

The accuracy of an O-D estimation model is easily impacted by the O-D pairs with large traffic demands. Therefore, 24 O-D pairs whose average flow is greater than 50 (veh/15 min) were chosen to evaluate the performance of the proposed real-time DODE algorithm.

Table 1 shows the relevant evaluation results. The second column of the table represents the average flow of the selected O-D pairs in 18 time intervals (TI3-TI20) of the study period. The third column lists the number of intervals within which the average flow is less than 5 (veh/15 min). The fourth and fifth columns show the MAEs and MAPEs calculated based on the estimated and observed O-D flows, respectively. It is easy to see that the MAEs of all selected pairs are no more than 7 (veh/15 min), while the MAPEs of most of the selected O-D pairs are less than 15%, demonstrating

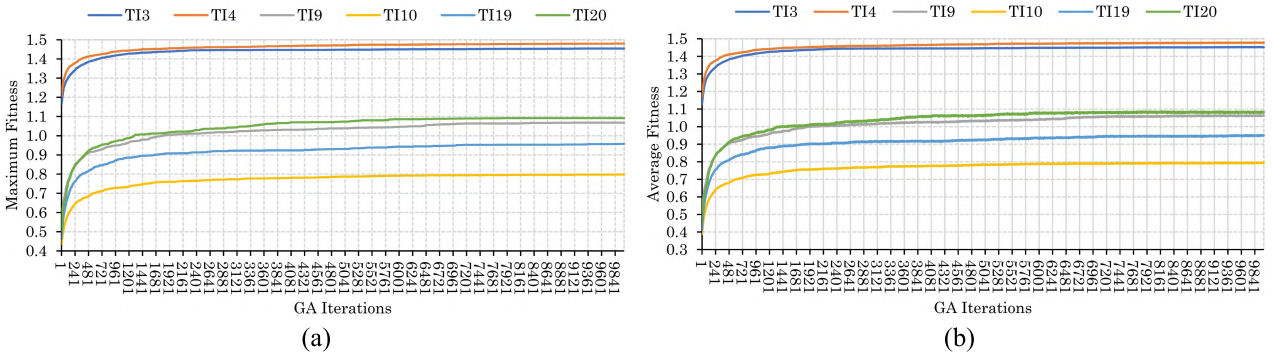

FIGURE 10. Convergence evolution of the real-time dynamic O-D estimation algorithm.

TABLE 1. Evaluation results for the proposed DODE algorithm on the selected O-D pairs.

O-D pair	Average flow (veh/15 min)	Number of intervals with average flow <= 5	MAE (veh/15 min)	MAPE (%)
1-18	78	0	6	8.21
5-16	153	0	6	4.17
6-9	71	0	6	10.59
7-15	60	0	4	7.96
8-9	56	0	4	12.93
8-11	68	0	4	7.28
10-23	57	0	5	11.07
11-14	83	0	5	6.95
12-19	118	0	5	4.20
13-10	91	0	5	5.50
13-18	61	0	4	10.15
15-7	75	0	6	8.40
16-5	82	0	5	6.24
16-12	64	0	6	11.23
18-13	51	1	4	7.10 (6.71)
19-2	68	0	6	10.73
19-20	65	7	3	4.09 (18.33)
20-12	110	0	6	6.49
20-17	53	1	6	15.06 (17.00)
20-24	56	1	4 (3)	6.31 (5.96)
20-25	54	1	6 (5)	12.27 (14.36)
21-19	62	0	5	10.49
22-23	110	0	7	6.80
23-10	103	0	5	5.03

Note: The figures in the parentheses in the fourth and fifth columns were calculated in the cases where the flows with value of less than 5 (veh/15 min) were included.

the proposed algorithm is capable of providing accurate estimates. Note that the figures in the parentheses in the fourth and fifth columns were calculated in the cases where the flows with value of less than 5 (veh/15 min) were included. These figures were given here because the existence of small flows can significantly impact the values of the MAE and MAPE measures, leading to an unreliable estimation of the model performance. For example, for the O-D pair 19-20, if the small flows associated with seven time intervals are included, the MAPE will be increased from 4.09% to 18.33%. However, as can be seen from Figure 11(f), when the average flow is smaller than 5 (veh/15 min), the estimates given by the proposed algorithm are still acceptable.

To go further in inspecting the performance of the DODE algorithm, the top eight O-D pairs in terms of the MAPE in descending order were selected and their estimated and observed O-D flows during the six typical time intervals were compared, as shown in Figure 11. It is observed that the model

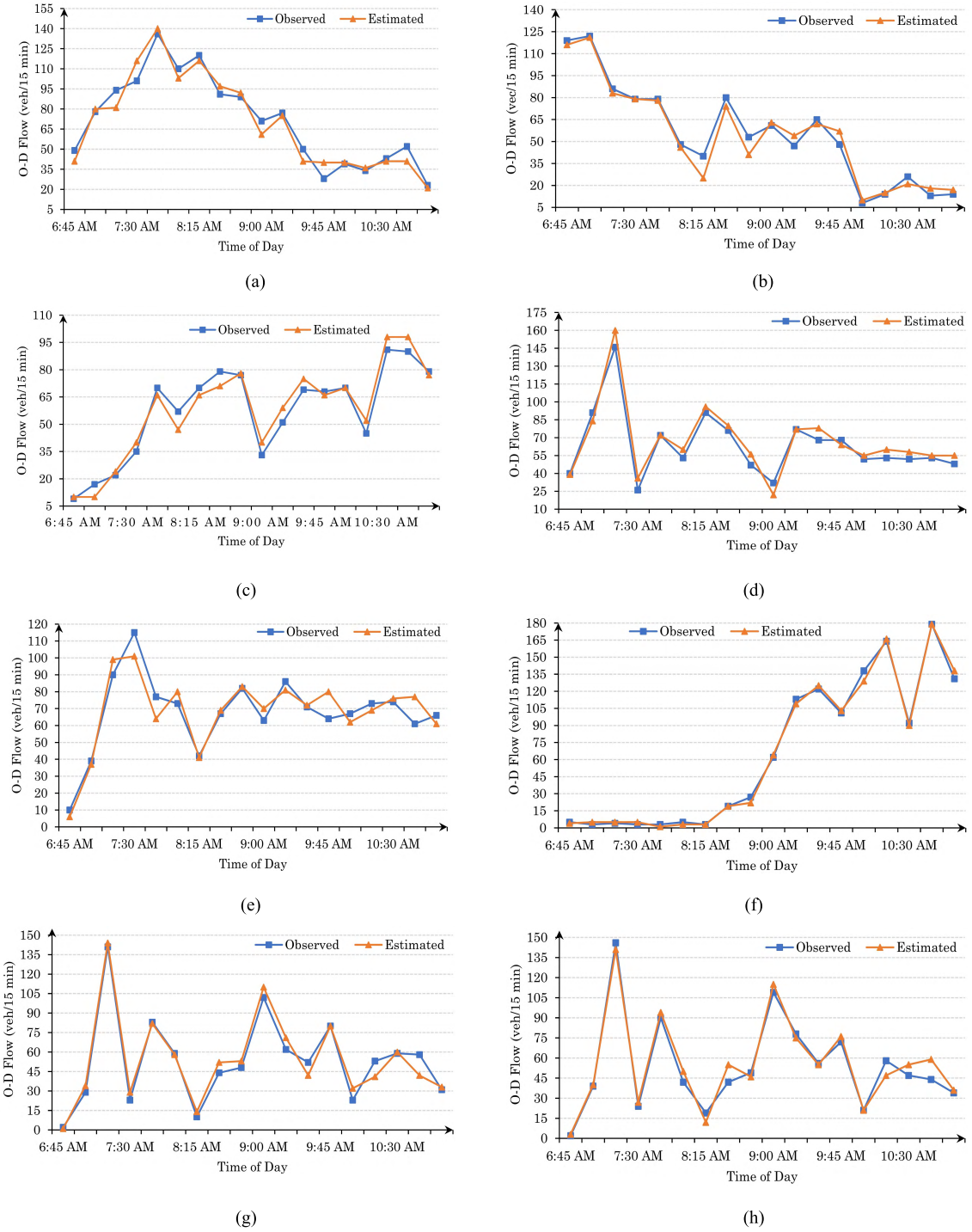

FIGURE 11. Comparisons of the estimated and observed O-D flows of the selected O-D pairs. (a) O-D pair 6-9. (b) O-D pair 8-9. (c) O-D pair 10-23. (d) O-D pair 16-12. (e) O-D pair 19-2. (f) O-D pair 19-20. (g) O-D pair 20-17. (h) O-D pair 20-25.

achieves satisfactory performances since the estimated and observed O-D flow curves for each selected O-D pair are very close to each other. It is also worth noting that though the MAPE of the O-D pair 19-20 is 18.33%, the estimated and observed flow curves have strong consistency, even in the cases where the observed flows are less than 5 (veh/15 min).

Comparisons between the estimated and observed flows on all 625 O-D pairs during the six typical time intervals were also conducted. The relevant results are shown in Figure 12. As seen, both of the regression coefficient and the coefficient of determination associated with all of the typical time intervals are very close to 1.0, demonstrating the proposed

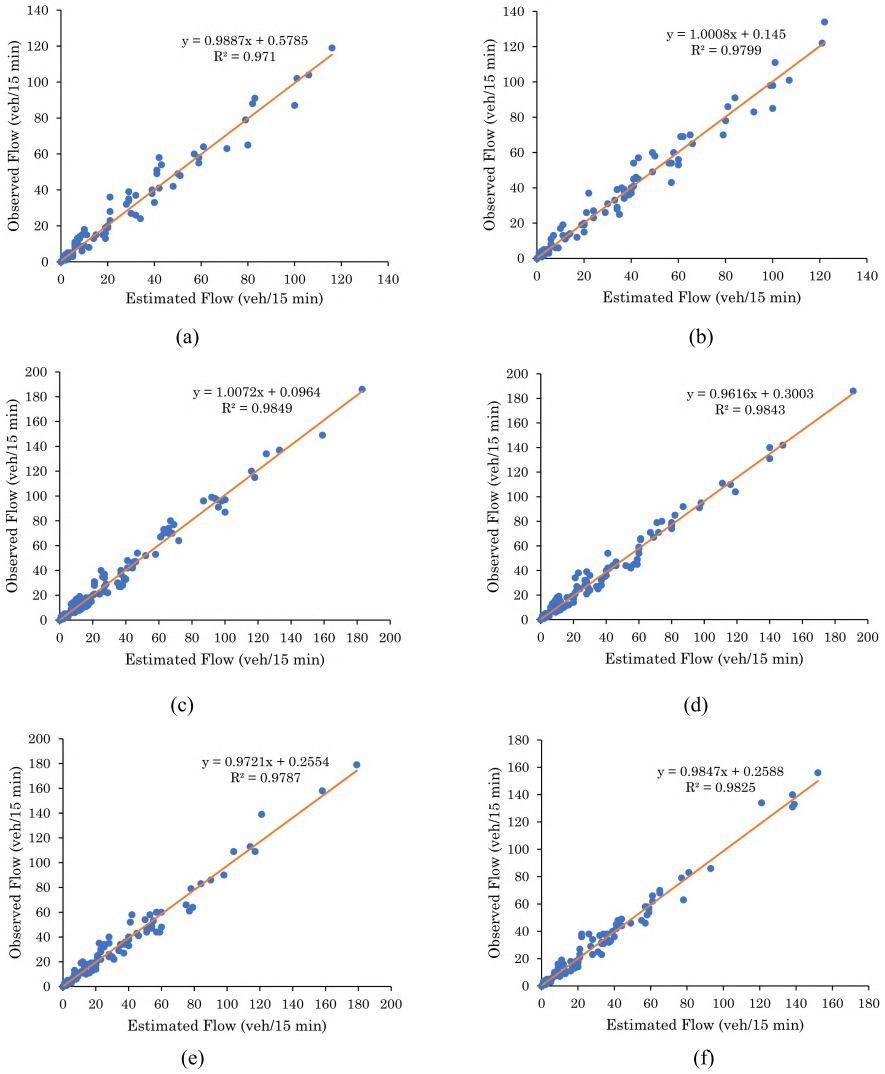

FIGURE 12. Comparisons of the estimated and observed O-D flows in typical time periods. (a) TI3 (6:30 AM ∼ 6:45 AM). (b) TI4 (6:45 AM ∼ 7:00 AM). (c) TI9 (8:00 AM ∼ 8:15 AM). (d) TI10 (8:15 AM ∼ 8:30 AM). (e) TI19 (10:30 AM ∼ 10:45 AM). (f) TI20 (10:45 AM ∼ 11:00 AM).

O-D estimation algorithm is accurate, and can be used for practical applications.

V. CONCLUSIONS

The real-time estimation of dynamic O-D flows remains a challenging research topic due to the significant complexity in jointly solving both of the DTA and DODE problems. In this study, a learning-assigning-searching solution framework using machine learning algorithms is presented to estimate dynamic O-D flows in real time. A CNN model is developed to capture the patterns that characterize the dynamic mappings between the estimated time-varying O-D flows and the assigned link flows. Next, an offline O-D pattern extraction algorithm and a real-time O-D estimation algorithm are separately developed by using the designed GAs as the searcher and the established CNN model as the assigner. The framework was evaluated with a realistic network in Kunshan, China, and its effectiveness was demonstrated by a series of experiments.

Future work could be focused on the following aspects. First, it is worth exploring the applicability of the proposed framework on a large-scale road network. Second, as the developed CNN model cannot provide physical meanings of the assignment process as done by the existing analytical and simulation-based DTA models, some more advanced machine learning algorithms regarding interpretation modeling, e.g. Liu *et al.* [47], could be tailored to the proposed framework. Third, it is interesting to explore the use of vehicle trajectory data to establish the prior traffic data. A potential benefit is that it can subtly avoid extensive simulation calibration work. The recent study by Zhang *et al.* [48] regarding how to efficiently calibrate a large-scale traffic simulator provides another way to enhance the current work. At last but not least, for the DTA modeling, a mesoscopic simulation

model is commonly preferred because of its high computational efficiency [7]. Therefore, the DTA models based on mesoscopic simulation could be integrated into the proposed framework in the future.

REFERENCES

[1] E. Cascetta, D. Inaudi, and G. Marquis, "Dynamic estimators of origin-destination matrices using traffic counts," *Transp. Sci.*, vol. 27, no. 4, pp. 363–373, 1993.
[2] X. Zhou and H. S. Mahmassani, "Dynamic origin-destination demand estimation using automatic vehicle identification data," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 105–114, Mar. 2006.
[3] S. Carrese, E. Cipriani, L. Mannini, and M. Nigro, "Dynamic demand estimation and prediction for traffic urban networks adopting new data sources," *Transp. Res. C, Emerg. Technol.*, vol. 81, pp. 83–98, Aug. 2017.
[4] H. Yang, T. Sasaki, Y. Iida, and Y. Asakura, "Estimation of origin-destination matrices from link traffic counts on congested networks," *Transp. Res. B, Methodol.*, vol. 26, no. 6, pp. 417–434, 1992.
[5] H. Yang, "Heuristic algorithms for the bilevel origin-destination matrix estimation problem," *Transp. Res. B, Methodol.*, vol. 29, no. 4, pp. 231–242, 1995.
[6] Y. Yin, "Genetic-algorithms-based approach for bilevel programming models," *J. Transp. Eng.*, vol. 126, no. 2, pp. 115–120, 2000.
[7] *A Primer for Dynamic Traffic Assignment*, ADB Transp. Netw. Model. Committee, Transp. Res. Board, Washington, DC, USA, 2010.
[8] X. Wu, J. Guo, K. Xian, and X. Zhou, "Hierarchical travel demand estimation using multiple data sources: A forward and backward propagation algorithmic framework on a layered computational graph," *Transp. Res. C, Emerg. Technol.*, vol. 96, pp. 321–346, Nov. 2018.
[9] Y. Nie and H. M. Zhang, "A variational inequality formulation for inferring dynamic origin-destination travel demands," *Transp. Res. B, Methodol.*, vol. 42, nos. 7–8, pp. 635–662, 2008.
[10] W. Shen and L. Wynter, "A new one-level convex optimization approach for estimating origin–destination demand," *Transp. Res. B, Methodol.*, vol. 46, no. 10, pp. 1535–1555, 2012.
[11] J. T. Lundgren and A. Peterson, "A heuristic for the bilevel origin–destination-matrix estimation problem," *Transp. Res. B, Methodol.*, vol. 42, no. 4, pp. 339–354, 2008.
[12] C. C. Lu, X. Zhou, and K. Zhang, "Dynamic origin–destination demand flow estimation under congested traffic conditions," *Transp. Res. C, Emerg. Technol.*, vol. 34, pp. 16–37, Sep. 2013.
[13] L. G. Willumsen, "Estimating time-dependent trip matrices from traffic counts," in *Proc. 9th Int. Symp. Transp. Traffic Theory*, Delft, The Netherlands, 1984, pp. 397–411.
[14] M. L. Hazelton, "Statistical inference for time varying origin–destination matrices," *Transp. Res. B, Methodol.*, vol. 42, pp. 542–552, Jul. 2008.
[15] A. Chen, P. Chootinan, W. Recker, and H. M. Zhang, "Development of a path flow estimator for deriving steady-state and time-dependent origin-destination trip tables," UC Berkeley, California Partners Adv. Transp. Technol., Berkeley, CA, USA, Sep. 2004, pp. 9–10. Accessed: Nov. 9, 2018. [Online]. Available: https://merritt.cdlib.org/d/ark%3A%2F13030%2Fm50g3mcq/2/producer%2FPRR-2004-29.pdf
[16] I. Okutani, "The Kalman filtering approaches in some transportation and traffic problems," in *Proc. 10th Int. Symp. Transp. Traffic Theory*, Cambridge, MA, USA, Jul. 1987, pp. 397–416.
[17] K. Ashok, "Dynamic origin-destination matrix estimation and prediction for real-time traffic management system," in *Proc. 12th Int. Symp. Transp. Traffic Theory*, Berkeley, CA, USA, 1993, pp. 465–484.
[18] K. Ashok and M. E. Ben-Akiva, "Alternative approaches for real-time estimation and prediction of time-dependent origin–destination flows," *Transp. Sci.*, vol. 34, no. 1, pp. 21–36, 2000.
[19] H. D. Sherali and T. Park, "Estimation of dynamic origin–destination trip tables for a general network," *Transp. Res. B, Methodol.*, vol. 35, no. 3, pp. 217–235, 2001.
[20] X. Zhou and H. S. Mahmassani, "A structural state space model for real-time traffic origin–destination demand estimation and prediction in a day-to-day learning framework," *Transp. Res. B, Methodol.*, vol. 41, no. 8, pp. 823–840, 2007.
[21] Z. Lu, W. Rao, Y.-J. Wu, L. Guo, and J. Xia, "A Kalman filter approach to dynamic OD flow estimation for urban road networks using multi-sensor data," *J. Adv. Transp.*, vol. 49, no. 2, pp. 210–227, 2015.
[22] H. Kim, S. Baek, and Y. Lim, "Origin-destination matrices estimated with a genetic algorithm from link traffic counts," *Transp. Res. Rec., J. Transp. Res. Board*, no. 1771, pp. 156–163, Jan. 2001.
[23] S. Baek, H. Kim, and Y. Lim, "Multiple-vehicle origin–destination matrix estimation from traffic counts using genetic algorithm," *J. Transp. Eng.*, vol. 130, no. 3, pp. 339–347, 2004.
[24] A. Stathopoulos and T. Tsekeris, "Hybrid meta-heuristic algorithm for the simultaneous optimization of the O–D trip matrix estimation," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 19, no. 6, pp. 421–435, 2004.
[25] I. Yun and B. Park, "Estimation of dynamic origin destination matrix: A genetic algorithm approach," in *Proc. IEEE Intell. Transp. Syst.*, Vienna, Austria, Sep. 2005, pp. 522–527.
[26] L. Kattan and B. Abdulhai, "Noniterative approach to dynamic traffic origin–destination estimation with parallel evolutionary algorithms," *Transp. Res. Rec., J. Transp. Res. Board*, no. 1964, pp. 201–210, Jan. 2006.
[27] B. Park and K. Zhu, "Time-dependent origin-destination estimation: Genetic algorithm-based optimization with updated assignment matrix," *KSCE J. Civil Eng.*, vol. 11, no. 4, pp. 199–207, 2007.
[28] S. Huang, A. W. Sadek, and L. Guo, "Computational-based approach to estimating travel demand in large-scale microscopic traffic simulation models," *J. Comput. Civil Eng.*, vol. 27, no. 1, pp. 78–86, 2012.
[29] D. K. Merchant and G. L. Nemhauser, "A model and an algorithm for the dynamic traffic assignment problems," *Transp. Sci.*, vol. 12, no. 3, pp. 183–199, 1978.
[30] M. Carey, "Optimal time-varying flows on congested networks," *Oper. Res.*, vol. 35, no. 1, pp. 56–69, 1987.
[31] T. L. Friesz, J. Luque, R. L. Tobin, and B. W. Wie, "Dynamic network traffic assignment considered as a continuous time optimal control problem," *Oper. Res.*, vol. 37, no. 6, pp. 893–901, 1989.
[32] H. S. Mahmassani, "Dynamic network traffic assignment and simulation methodology for advanced system management applications," *Netw. Spatial Econ.*, vol. 1, nos. 3–4, pp. 267–292, Sep. 2001.
[33] T. Toledo and T. Kolechkina, "Estimation of dynamic origin–destination matrices using linear assignment matrix approximations," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 618–626, Jun. 2013.
[34] K. Sohn and D. Kim, "Dynamic origin–destination flow estimation using cellular communication system," *IEEE Trans. Veh. Technol.*, vol. 57, no. 5, pp. 2703–2713, Sep. 2008.
[35] C. Antoniou, M. Ben-Akiva, and H. N. Koutsopoulos, "Nonlinear Kalman filtering algorithms for on-line calibration of dynamic traffic assignment models," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 4, pp. 661–670, Dec. 2007.
[36] H. X. Liu, W. Ma, J. X. Ban, and P. Mirchandani, "Dynamic equilibrium assignment with microscopic traffic simulation," in *Proc. IEEE Intell. Transp. Syst.*, Vienna, Austria, Sep. 2005, pp. 676–681.
[37] W. Rao, Y.-J. Wu, J. Xia, J. Ou, and R. Kluger, "Origin-destination pattern estimation based on trajectory reconstruction using automatic license plate recognition data," *Transp. Res. C, Emerg. Technol.*, vol. 95, pp. 29–46, Oct. 2018.
[38] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transp. Res. C, Emerg. Technol.*, vol. 86, pp. 360–371, Jan. 2018.
[39] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.
[40] H. Alibabai and H. S. Mahmassani, "Dynamic origin-destination demand estimation using turning movement counts," *Transp. Res. Rec., J, Transp. Res. Board*, no. 2085, pp. 39–48, Jan. 2008.
[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *J. Mach. Learn. Res.*, vol. 15, no. 4, pp. 315–323, 2011.
[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[43] L. Hou, D. Samaras, T. Kurc, Y. Gao, and J. Saltz, "ConvNets with smooth adaptive activation functions for regression," in *Proc. 20th Int. Conf. Artif. Intell. Statist.* Tallahassee, FL, USA, JMLR: W&CP, vol. 54, 2017, pp. 430–439.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Washington, DC, USA, Jun. 2015, pp. 1026–1034.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, pp. 7–9.

[46] A. R. de Villa, J. Casas, M. Breen, and J. Perarnau, "Static OD estimation minimizing the relative error and the GEH index," *Procedia Soc. Behav. Sci.*, vol. 111, pp. 810–818, Feb. 2014.
[47] Q. Liu, B. Wang, and Y. Zhu, "Short-term traffic speed forecasting based on attention convolutional neural network for arterials," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 11, pp. 999–1016, 2018.
[48] C. Zhang, C. Osorio, and G. Flötteröd, "Efficient calibration techniques for large-scale traffic simulators," *Transp. Res. B, Methodol.*, vol. 97, pp. 214–239, Mar. 2017.

JISHUN OU is currently pursuing the Ph.D. degree with the Intelligent Transportation System Research Center, Southeast University, China. He was a Visiting Scholar with the Department of Civil and Architectural Engineering and Mechanics, The University of Arizona, USA, during 2016–2017. His current research interests include traffic big data analysis and application, urban road network modeling, and intelligent transportation systems.

JIAWEI LU is currently pursuing the Ph.D. degree with the School of Sustainable Engineering and the Built Environment, Arizona State University, USA. His research interests include transportation network modeling, traffic simulation, and distributed traffic control.

JINGXIN XIA received the Ph.D. degree in transportation engineering from the University of Kentucky, USA, in 2006. He is currently a Professor with the Intelligent Transportation System Research Center, Southeast University, China. He has published more than 40 peer-reviewed papers so far. His main research interests include traffic flow theory, transportation network modeling, traffic signal control, and intelligent transportation systems.

CHENGCHUAN AN is currently pursuing the Ph.D. degree with the Intelligent Transportation System Research Center, Southeast University, China. From 2014 to 2016, he was a Visiting Student with the Department of Civil and Architectural Engineering and Mechanics, The University of Arizona, USA. His current research interests include traffic signal optimization, urban traffic flow modeling, and traffic data mining.

ZHENBO LU received the Ph.D. degree in traffic information engineering and control from Southeast University, China, in 2011, where he is currently an Associate Professor with the Intelligent Transportation System Research Center. His main research interests include transportation planning, traffic simulation, and intelligent transportation systems.