Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 8877138, 15 pages
https://doi.org/10.1155/2021/8877138

Research Article

Dynamic Path Flow Estimation Using Automatic Vehicle Identification and Probe Vehicle Trajectory Data: A 3D Convolutional Neural Network Model

Can Chen, Yumin Cao, Keshuang Tang, and Keping Li

Key Laboratory of Road and Traffic Engineering in the Ministry of Education, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Keshuang Tang; tang@tongji.edu.cn

Received 1 October 2020; Revised 23 December 2020; Accepted 22 January 2021; Published 8 February 2021

Academic Editor: Wen LIU; wenliu@whut.edu.cn

Dynamic path flows, referring to the number of vehicles that choose each path in a network over time, are generally estimated with the partial observations as the input. The automatic vehicle identification (AVI) system and probe vehicle trajectories are now popular and can provide rich and complementary trip information, but the data fusion was rarely explored. Therefore, in this paper, the dynamic path flow estimation is based on these two data sources and transformed into a feature learning problem. To fuse the two data sources belonging to different detection ways at the data level, the virtual AVI points, analogous to the real AVI points (turning movements at nodes with AVI detectors), are defined and selected to statically observe the dynamic movement of the probe vehicles. The corresponding selection principles and a programming model considering the distribution of real AVI points are first established. The selected virtual AVI points are used to construct the input tensor, and the turning movement-based observations from both the data sources can be extracted and fused. Then, a three-dimensional (3D) convolutional neural network (CNN) model is designed to exploit the hidden patterns from the tensor and establish the high-dimensional correlations with path flows. As the path flow labels commonly with noises, the bootstrapping method is adopted for model training and the corresponding relabeling principle is defined to purify the noisy labels. The entire model is extensively tested based on a realistic road network, and the results show that the designed CNN model with the presented data fusion method can perform well in training time and estimation accuracy. The robustness of a model to noisy labels is also improved through the bootstrapping method. The dynamic path flows estimated by the trained model can be applied to travel information provision, proactive route guidance, and signal control with high real-time requirements.

1. Introduction

Unlike the static path flows, which represent the average path flows during a long period, the dynamic path flows represent the real-time path flows in a relatively small time interval and the corresponding estimation problem becomes more challenging. The dynamic path flow data have a wide range of applications, such as the analysis of user travel patterns, large-scale traffic network simulation, and traffic planning and management. The path flow and OD matrix estimates are sometimes similar and interdependent. The OD flows can be assigned to obtain path flows and the sum of several path flows can be used to obtain one specific OD flow. Due to the high cost and less efficiency of manual survey for directly observing path flows, a typical way has been widely used to indirectly estimate them from observed link flows.

For this way, numerous mathematical programming models have been established to solve the OD demands that are most consistent with observed link flows under certain assumptions, but the results remain unsatisfactory. The major reason is that the information provided by the observed links is limited. It is common for a network that the number of OD pairs is much larger than the number of observed links [1, 2], which is often referred to the under-specified problem. More information such as prior OD matrices is needed and the bi-level programming framework is mostly applied among existing studies [3, 4]. At the upper level, optimal solutions are searched according to certain objectives that minimize the distance between the possible estimates and observed values, including generalized least squares, entropy maximization, etc. At the lower level where dynamic traffic assignment (DTA) is usually conducted, estimated OD flows are mapped to observed link flows. Unfortunately, without observed trip information as constraints, the assumptions related to path choice and the driving behavior of a traveler using the DTA model may be inconsistent with realistic conditions [5].

Compared with link traffic counts, the AVI data and probe vehicle trajectories can provide more information about trips. The current AVI system is generally composed by radio frequency identification device-based detector, Bluetooth detector, and video-based detector. The unique vehicle's ID and passing time can be recorded with the vehicle passing the detector. By matching vehicles' IDs, the traffic counts and travel time of partial paths can be further obtained. In the current studies concerning path and OD flow estimations, the AVI information is generally incorporated in the following three ways: matched volume as prior observed OD data [6, 7] or observed counts of partial complete paths [2, 8]; the link choice [4] or lagged observation proportion based on travel time [6]; and recorded link volume [2, 8]. Compared with the AVI data, the probe vehicle trajectories are collected by GPS-enabled devices and the prior OD matrices and users' path choice behaviors could be further discovered based on the sampled and complete trajectories. Several researches have been conducted using either vehicle trajectories alone or as fused with the observed link flows [9, 10]. The information provided by the AVI system and the probe vehicles has different characteristics. Taking the video-based AVI system widely applied in China as an example, every vehicle passing the detector can ideally be detected, but the coverage rate and matched vehicles may still rapidly degenerate when the network size increases [8]. Recently, with the rapid development of online car-hailing market, more vehicle trajectories can be collected and the spatial coverage is considerably wide [7], but the comparatively low penetration rate remains a problem [11]. From the discussion, it is evident that the AVI and the probe vehicle systems have complementary characteristics in both spatial coverage and vehicle collection. Hence, fusing these two types of data is promising for dynamic path flow estimation but has rarely been explored.

Typically, dynamic path flow estimation is regarded as an optimization problem that searches for the most consistent path flow estimates and the solution is a high-dimensional time-dependent path flow matrix [12]. For mathematical programming models, the trip information obtained by fusing these two data is supposed to be described analytically. And the model accuracy may improve as rich information is added, yet the solving capacity of the model may become more difficult and inefficient. Considering the rich and implicit travel features stored in the AVI and the probe vehicles data, the path flows estimation can potentially be transformed from an optimization problem to a data-driven feature learning problem. Neural networks (NNs) have long been proven as an effective measure to learn hidden patterns from given samples and make further predictions.

Recently, with the emergence of deep learning algorithms, the NN has undergone a transition from shallow NN to deep NN (NN with multiple layers), which is much more powerful in learning complex and abstract features. Several researchers have explored the use of deep NNs for OD flow estimation. Huang et al. [7] used the long short-term memory model with time sequence of trajectory OD flows as the input. The temporal dependency between successive time steps is considered. The labels of partial OD pairs are provided by AVI data and the label propagation is adopted to infer uncovered OD pairs. An improved 3D-CNN model (Res3D), which can capture the features along both spatial and temporal dimensions, was designed by Tang et al. [13]. The model input is a cube of three stacked matrices, and for each matrix, the set of links installed by AVI detectors is used to represent the segment volume, matched volume, and matched travel time. The training samples are provided by a parallel simulation system and the model can be applied through transfer learning. In these two models, the AVI and trajectory data are still used separately as model inputs. The success of deep NNs relies heavily on supervised training and sufficient labeled data samples. The OD flow labels used by Huang et al. [7] and Tang et al. [13] are, respectively, obtained from AVI data and an exogenous simulator. However, it is difficult to guarantee that the label data are without noises, which is also common in the field of image recognition. And the label noise is potentially more harmful than feature noise, which affects the observed values of the feature [14]. It is inevitable to consider how to learn samples with noisy labels.

As suggested by Tang et al. [13], the 3D-CNN structure is effective for capturing the spatial-temporal patterns in link observation-based tensor. In this paper, the 3D convolutions are also introduced to design a deep NN for estimating the dynamic path flows, but the input tensor is constructed based on the turning movements of network nodes to express and fuse the multiple types of AVI and probe vehicle observations. The contributions of this study are highlighted below:

(1) The AVI and probe vehicle data produced by different detection ways are fused and the mobile detection of probe vehicles is changed to stationary detection by properly arranging the virtual AVI points. The principles for selecting the turning movements as virtual AVI points and a corresponding programming model are established, and with the distribution of real AVI points, data-driven feature learning characteristics and input tensor size related to model efficiency are comprehensively considered.

(2) Based on the selected virtual AVI points, the turning volumes, matched volumes, and matched travel time provided by AVI and probe vehicle data are represented and hierarchically combined in the input tensor to implicitly represent the travel patterns of networks. The concrete architecture of the 3D-CNN

model is designed to automatically recognize the hidden patterns behind these observations and establish a high-dimensional mapping to dynamic path flows.

(3) To cope with noisy path flow labels, especially with systematic errors, a self-correcting algorithm bootstrapping commonly applied for image classification tasks is extended to variable regression problem. The key is the re-labeling process gradually purifying the noisy labels, and the relationship between the generated path flows and the observed turning flows is used to determine whether to correct the original labels with model outputs.

2. Problem Statement

In this paper, we use the characters lowercase (a), bold lowercase ($\mathbf{a}$), uppercase (A), bold uppercase ($\mathbf{A}$), and uppercase calligraphy ($\mathcal{A}$) to represent scalar, vector, set, matrix, and tensor, respectively. Each set has an attribute of length that can be expressed by $|A|$.

Figure 1 illustrates the research background and relevant definitions and variables. We consider an urban road network specified by a directed graph $G = (N, L)$, where N denotes the node set of the network, $N = \{1, \ldots, n\}$, $n \in N$, and L denotes the link set of the network, $L = \{1, \ldots, l\}$, $l \in L$. P is the path set of the network, $P = \{1, \ldots, k\}, k \in P$. The nodes in N which directly connect to traffic zones are attracting or generating nodes and constitute the node set N^{se}. The remaining nodes are cross nodes that represent the real intersections and constitute the node set N^{c}. For each link of node, there are three circles. Along the approaching direction, the three circles represent left, straight, and right turns, from left to right. The white circle means this approach has no this turning movement; otherwise, the circle is filled with gray. A is the set of all turning movements, $A = \{1, \ldots, a\}, a \in A$. A^{se} is a subset of A and contains the turning movements of nodes in N^{se}. The purple circle represents a real AVI point where the turning vehicles can be detected by a video-based AVI system. Taking the link l as an example, several activated lines are drawn near the stop line in the shooting screens of cameras to automatically detect and record the passing vehicles' license plate numbers and time. The set of real AVI points is denoted by A^{r}.

The vehicle turning information for each a in A^{r} can be directly obtained, and Alibabai and Mahmassani [15] point out that using intersection turning movements as the basic field observation instead of links can result in more reliable estimates of dynamic OD matrices. Hence, the turning movements of nodes are selected as the basic network elements to construct the input tensor and represent the AVI and probe vehicle observations. However, compared with the stationary AVI detection points, the occurrence and movement of the probe vehicle are dynamic. It is necessary to study as to how to select the turning movements from A to constitute the virtual AVI point set denoted by A^{v}. For the virtual AVI point (circle filled with green), it can be thought that a virtual AVI detector is installed to specially detect the probe vehicles. The turning movement and passing time at the node for each probe vehicle with unique ID can also be got through the mapping between the GPS coordinates and links. The real and virtual AVI points can overlap (circles filled with yellow). Hence, for one network G, the selection of A^{v} based on the distribution of A^{r} is important not only for the representation of probe vehicle data but also for the fusion with AVI data.

The 3D-CNN is generally used to process consecutive video frames for action recognition. The corresponding input is a cube formed by stacking multiple contiguous frames together. For each frame, like a still image, it is represented by a three-dimensional tensor involving the height, width, and channels of the image. For CNNs applied to the transportation engineering field, in addition to the model architecture design, the first problem is to organize the traffic data based on the requirements of CNN. In the network traffic state estimation, it is common but a little "rough" to grid the network and use the heatmap as the input. Based on the virtual AVI point set A^{v}, the selection and combination of the turning movement-based observations provided by AVI and probe vehicle systems in input tensor of NN is important for the implicit representation of the road network.

The prior path flows commonly used in traditional mathematical models can be taken as one of the sources of path flow labels. From the viewpoint of estimation accuracy, it is desirable that the prior path flows should be a close estimate of the true path flows. However, Yang et al. [3] pointed out that prior OD flows may have random and systematic errors compared with the true values. This is also true for path flows. Some methods have been developed to deal with the problem of noisy labels and can be approximately divided into three categories: (1) noise-robust models, (2) data cleaning methods, and (3) noise-tolerant learning algorithms [14]. The noise-robust models rely on initializing the start point of gradient descent and early stopping mechanism to avoid (over) fitting to noisy labels. The second category aims to improve the quality of training data by identifying and removing the samples with incorrect labels. Considering the difficulties of obtaining training samples paired with labels and the desire to make full use of each sample's information, the third category's methods are focused and the bootstrapping method which has been used by Reed et al. [16] and Wu et al. [17] is extended from the classification problem to the variable regression problem.

3. Methodology

To address the problems summarized in the previous section, the modeling process is illustrated in Figure 2 and there are three main steps. In Step 1, the principles of optimally selecting the turning movements as virtual AVI points to fuse the AVI and the probe vehicle data is specified and modeled as a binary linear programming model. In Step 2, the AVI and probe vehicle observations based on turning movements are selected and organized in the input tensor. And the architecture of the designed 3D-CNN model and working process are explained. Finally, for path flow labels with errors, the bootstrapping method which can re-label

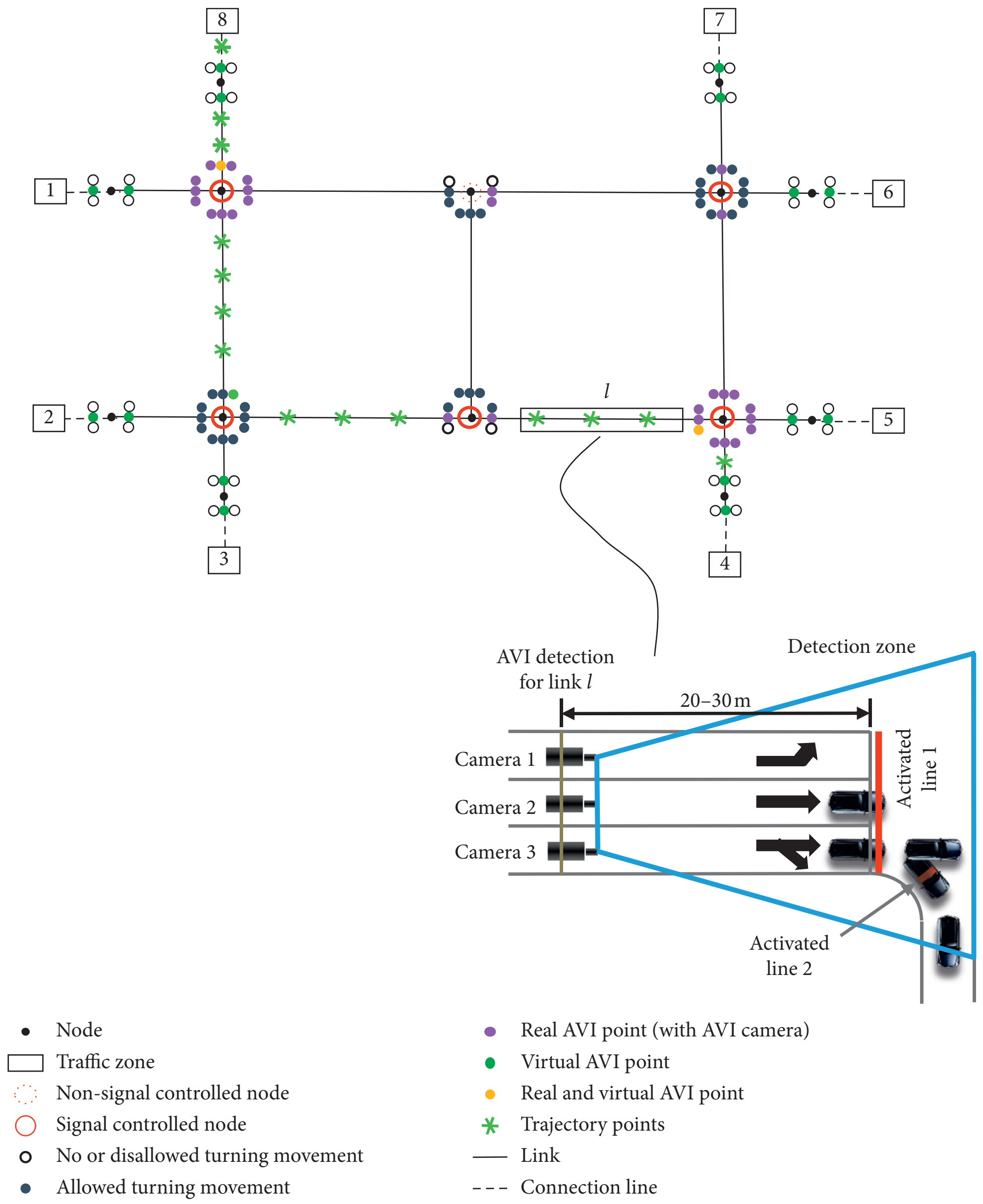


FIGURE 1: Illustration of the research problem.

and gradually purify the noisy labels is used to train the model.

Step 1. Virtual AVI Point Selection

The selected virtual AVI points are directly used to construct the input tensor of NN. Hence, except for the trip information of probe vehicles, the distribution of real AVI points, feature learning characteristics, and tensor size should also be considered.

Based on the algorithm proposed by Castillo et al. [2] about selecting links to be scanned for predicting path flows, a revised binary linear programming model is established. The model is shown as follows:

$$\min Z = \sum_{a\in A} \left(u_a^v + pa^{\max} * u_a^v * u_a^{se} - 0.5 * u_a^v * u_a^r\right). \quad (1)$$

subject to

$$\sum_{a\in A} u_a^v * b_a^k \geq 1, \quad \forall k \in P, \quad (2)$$

$$\sum_{a\in A} u_a^v * b_a^{k_1} * \mathrm{d}(k_1, k_2, a) \geq 1, \quad \forall k_1, k_2 \in P \text{ and } k_1 \neq k_2, \quad (3)$$

$$b_a^k = \begin{cases} 1, & \text{if path } k \text{ contains } a, \\ 0, & \text{otherwise}, \end{cases} \quad (4)$$

$$d(k_1, k_2, a) = \begin{cases} 1, & \text{if } b_a^{k_1} \neq b_a^{k_2}, \\ 0, & \text{otherwise}, \end{cases} \quad (5)$$

where u_a^v is the binary decision variable such that it takes the value 1 when the turning movement a is selected as the virtual AVI point, and 0, otherwise; u_a^{se} and u_a^r are 1 when a belongs to A^{se} and A^r, and 0, otherwise; $pa^{\max}$ is the maximum number of turning movements traversed by one path; b_a^k belongs to $\mathbf{B}$, which describes the relationship between the turning movements and each path of P.

The algorithm proposed by Castillo et al. [2] minimizes the number of scanned links with respect to two principles. Here, three principles are presented as follows:

(1) Any path k of P contains at least one virtual AVI point;

(2) For any two paths k_1 and $k_2$of P, each path can differ from the other through the minimum selected virtual AVI points of itself;

(3) Attempt to select the minimum virtual AVI points that are not in A^{se} set and make the selected points overlap more real AVI points.

The first principle is similar to that given by Castillo et al. [2] and realized by Constraint (2). For the second principle, Castillo et al. [2] allowed the scanned link which is in path k_1 and not in path k_2 or vice versa to distinguish the two paths. This means that for the two pairs of paths (k_1, k_2) and (k_2, k_1), the selected links can be the same. This is reasonable because the relationship between the path flows and observed counts is expressed by mathematical formulations and each path is guaranteed to have at least one scanned link. However, too many scanned links can potentially be assigned to several paths or paths with low demands. For the deep NN used in this study, the information of each path should be independent and integrated. Hence, each path should depend on the turning movements of itself to differ from other paths and for (k_1, k_2) and (k_2, k_1), the selected turning movements are different. Equations (3) and (5) can ensure this principle.

The last principle is reflected in the objective function (1), which includes all the three terms. To minimize the number of selected virtual AVI points, the decision variable u_a^v must be added as the first term. The second term is to constrain the selected virtual AVI points not from A^{se}, except in the cases where only the turning movements that belong to A^{se} can distinguish the two paths. This setting can help focus on the operation of probe vehicles in real intersections. The third term urges more overlaps between the virtual and the real AVI points. When the real AVI points are also selected as virtual AVI points, the objective value can be further reduced while satisfying the first two principles.

After solving the optimization model above, only a small part of A^{se} is selected and the dependency of paths and the passed turning movements cannot be completely expressed in the input tensor. This information is also important and useful for the estimation problem. Thus, the residual virtual points of A^{se} are also added to A^v. Then each path has its own start and terminal virtual points and the number of probe vehicles between the start/terminal point and each of the other passed points is the same. Please note that this selection method relies on the exogenous and static path information (path set P and turning movement-path matrix $\mathbf{B}$), which can be obtained through other data sources (e.g., the probe vehicle trajectories) or methods (e.g., the path flow assignment from known OD flows). It is suitable for stable network with not heavy traffic congestions [1, 3, 18]. If the effect of traffic congestion on travel time is significant, the path flow dynamics should be considered inside the model through the flow-related constraints.

Step 2. Input Tensor Construction and Model Architecture Design

3.1. Data Fusion and Input Tensor Construction.

There are several uniform and continuous time intervals for estimation denoted by $H = \{1, \ldots, h\}$, $h \in H$. The operation of network traffic within one time interval h can be seen as a video frame (also a still color image). The corresponding abstract expression is the three-dimensional tensor with the AVI and probe vehicle trajectory data merged, as shown in Figure 3. To implicitly express the travel patterns of one network, we define three main types of observations based on turning movements from these two data sources: turning volumes, matched volumes, and matched travel time. The last two observations can represent the local and global end-to-end trip information. As for the turning volumes, it helps to express richer information.

From Figure 3, we can see that there are three square matrices like the three channels of one color image and for each matrix, the rows and columns represent the selected turning movements of A^v. The matrix in channel 1, turning and matched volumes from AVI data, is represented by $\mathbf{Q}_h \in \mathbb{R}^{|A^v|\times|A^v|}$, where the diagonal element $Q_{a,a,h}$ $(Q \in \mathbb{R}^{|A^v|\times|A^v|\times|H|})$ denotes the number of detected vehicles passing turning movement a during time interval h and the off-diagonal element $Q_{a,a',h}$ denotes the number of matched vehicles from turning movement a to a'. Similar to Q_h, the matrix $Q'_h \in \mathbb{R}^{|A^v|\times|A^v|}$ in channel 2 represents the sampled turning vehicles and matched vehicles from the probe vehicle data. The matrix in the last channel, matched travel time, is represented by $\mathbf{T}_h \in \mathbb{R}^{|A^v|\times|A^v|}$, where only the off-diagonal elements may be nonzero, and $T_{a,a',h}$ $(T \in \mathbb{R}^{|A^v|\times|A^v|\times|H|})$ denotes the average travel time for the matched vehicles from the turning movement a to a' during interval h. Here, it is assumed that the travel time measured by the AVI system is more accurate than that of probe vehicles, because the AVI system can detect more regular vehicles. If the travel time from a to $a'$is positive, the reverse travel time and the number of matched vehicles are negative.

The stack of these three matrices forms a temporal cell $\Delta_h \in \mathbb{R}^{|A^v|\times|A^v|\times 3}$. Considering the complementary characteristics of AVI and the probe vehicle data in vehicle collection, the matrices $\mathbf{Q}_h$ and $\mathbf{Q}'_h$ are placed in adjacent channels. The travel time obtained from these two data sources is combined in one matrix $\mathbf{T}_h$ to avoid possible inconsistency between them. From the first channel to the last channel of Δ_h, we can see the matched regular vehicles

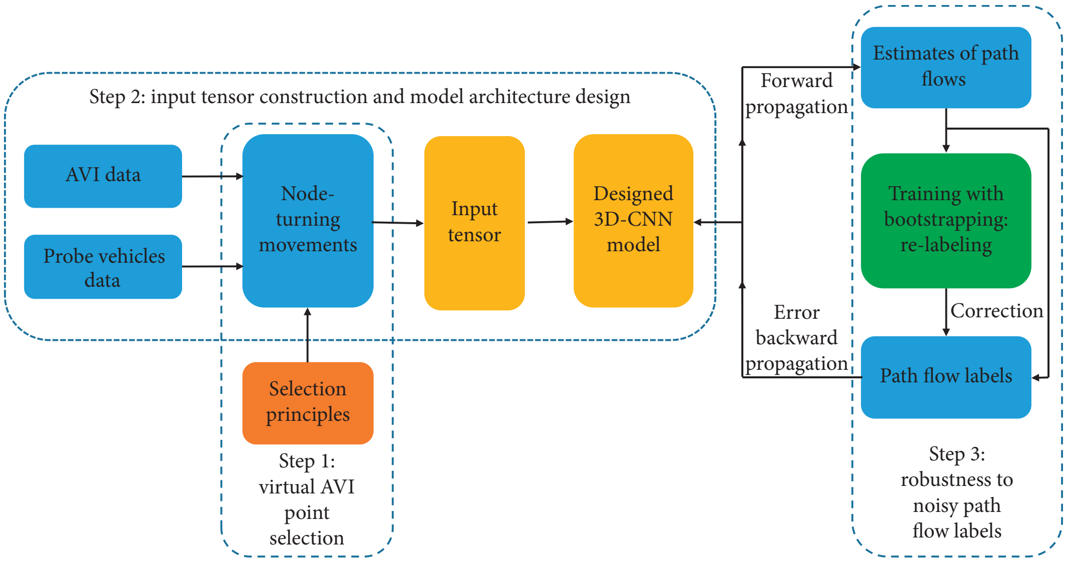

Step 2: input tensor construction and model architecture design
AVI data
Probe vehicles data
Node-turning movements
Input tensor
Designed 3D-CNN model
Selection principles
Step 1: virtual AVI point selection
Forward propagation
Error backward propagation
Estimates of path flows
Training with bootstrapping: re-labeling
Correction
Path flow labels
Step 3: robustness to noisy path flow labels

Figure 2: General modeling framework.

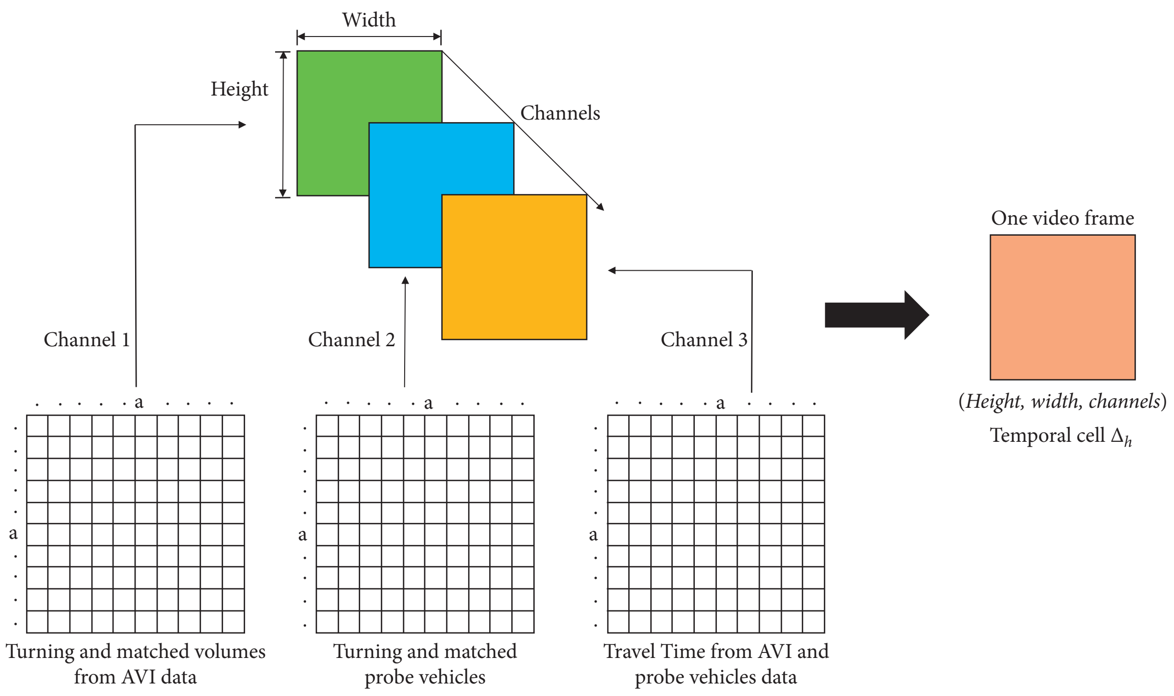

Width
Height
Channels
Channel 1
Channel 2
Channel 3
One video frame
(Height, width, channels)
Temporal cell Δ_h
a
Turning and matched volumes from AVI data
Turning and matched probe vehicles
Travel Time from AVI and probe vehicles data

Figure 3: The input tensor for 3D-CNN model.

from the AVI data, the matched probe vehicles, and the corresponding average travel time between any two turning movements in order. The trip information contained in the collected AVI and the probe vehicle data within interval h can be well organized and expressed hierarchically.

3.2. 3D-CNN Architecture Design. For one network, the prior and estimated path flow matrices are represented by $\mathbf{Y} \in \mathbb{R}^{|H| \times |P|}$ and $Y' \in \mathbb{R}^{|H| \times |P|}$. In a dynamic estimation problem, the generated $Y_{h,k}$ vehicles in the origin of path k will be observed in the following time intervals $h, h, \ldots, h + w'$, and $(w' + 1)$ is the maximum number of time intervals required to travel any path of the network [18]. Hence, for the estimate of one row vector $\mathbf{Y}_{h,:}$, the temporal cells from h to $h + w'$ are needed and can be seen as a sequence of video frames of network operation.

Referring to VGG-Net, which is a powerful two-dimensional CNN model presented by Simonyan and Zisserman [19], a 3D-CNN model is designed and shown in Figure 4. The VGG-Net has a very deep convolutional (conv.) architecture and the model capability is increased as the network becomes deeper, but it

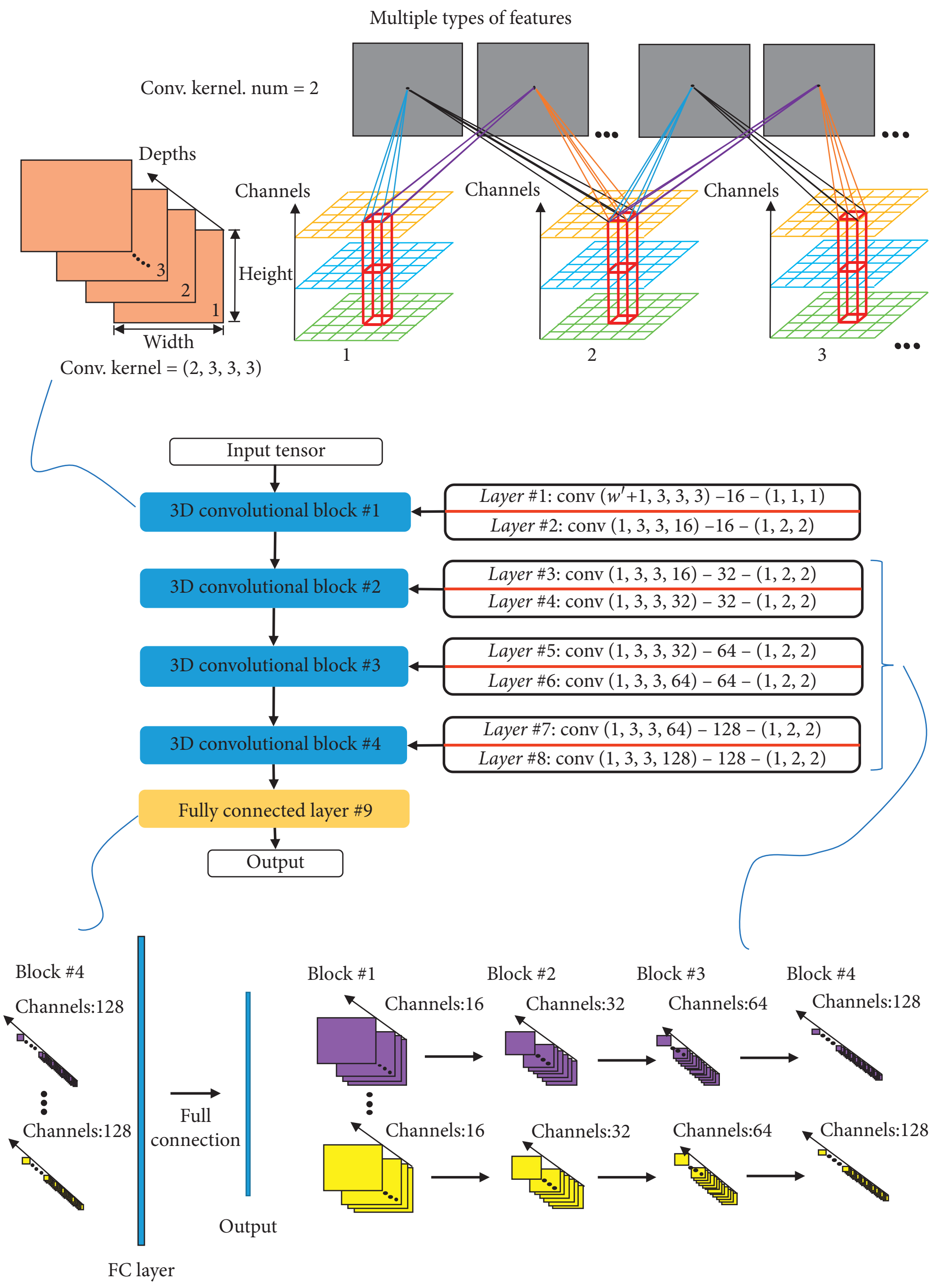

Multiple types of features
Conv. kernel. num = 2
Depths
Channels
Height
Width
Conv. kernel = (2, 3, 3, 3)
1
2
3
Input tensor
3D convolutional block #1
3D convolutional block #2
3D convolutional block #3
3D convolutional block #4
Fully connected layer #9
Output
Layer #1: conv (w'+1, 3, 3, 3) –16 – (1, 1, 1)
Layer #2: conv (1, 3, 3, 16) –16 – (1, 2, 2)
Layer #3: conv (1, 3, 3, 16) – 32 – (1, 2, 2)
Layer #4: conv (1, 3, 3, 32) – 32 – (1, 2, 2)
Layer #5: conv (1, 3, 3, 32) – 64 – (1, 2, 2)
Layer #6: conv (1, 3, 3, 64) – 64 – (1, 2, 2)
Layer #7: conv (1, 3, 3, 64) – 128 – (1, 2, 2)
Layer #8: conv (1, 3, 3, 128) – 128 – (1, 2, 2)
Block #4
Channels:128
Full connection
FC layer
Output
Block #1
Block #2
Block #3
Block #4
Channels:16
Channels:32
Channels:64
Channels:128

Figure 4: Designed 3D-CNN model.

leads to a heavier computational cost. Considering this tradeoff and experimental analysis, the number of 3D conv. blocks and weight layers (including conv. and fully connected (FC) layers) are reduced to 4 and 9, respectively. Each block has two conv. layers, and the corresponding parameters are denoted as "Conv (kernel size: depth, height, width, channels)–(number of kernels)–(conv. stride: depth, height, width)". Besides the first conv. layer, the conv. stride in the height and width directions, which is (1, 1) in VGG-Net, is changed to (2, 2). As the number of conv. layers decreases, the pooling players used in VGG-Net are completely removed, which may destroy the spatial features of maps.

The input of 3D-CNN containing several temporal cells Δ_h is denoted by $X \in \mathbb{R}^{|H|\times|A^v|\times|A^v|\times 3}$. As shown at the top of Figure 4, the 3D convolution kernel is also a cube and can move not only in height and width but also in depth. The kernel size on height and width (3, 3) is consistent with VGG-Net to extract spatial features. However, on depth, the kernel size is set as $(w' + 1)$ in the first conv. layer and 1 in the other conv. layers. Combining the moving stride on depth set as 1, it means that every $(w' + 1)$ time interval of X is treated as a group to extract the spatial-temporal features and estimate the path flows in the first interval. In addition, the number of feature maps for each group gradually increases in late conv. block with the rise of convolution kernels. The data-processing procedure in each block can be formulated as follows:

$$\begin{aligned} R^{(|H|-w',\ldots,16)} &= \mathrm{Conv\,3\,d\_1}\left(X^{(|H|,\ldots,3)}\right), \\ R^{(|H|-w',\ldots,128)} &= \mathrm{Conv\,3\,d\_4}\left(\mathrm{Conv\,3\,d\_3}\left(\mathrm{Conv\,3\,d\_2}\left(R^{(|H|-w',\ldots,16)}\right)\right)\right), \end{aligned} \tag{6}$$

where R is the output tensor of each conv. block.

After hierarchical feature extraction, combination, and transformation, multiple local and spatial-temporal traffic patterns are recognized and finally mapped to prior path flows through a single FC layer.

Step 3. Robustness to Noisy Labels

The key to bootstrapping is the designed judgment and the re-labeling process, which can be placed in the outer loop or incorporated in the loss function [17]. Through re-labeling the samples while training, more accurate labels may lead to a better model, which allows for further label clean-up, and the learner bootstraps itself in this way. An established framework of bootstrapping with the re-labeling process in the outer loop is described in Figure 5.

From Figure 5, you can see that there are two parts (2-1 and 2-2) in the training phase. For the pretraining part (2-1), its existence is not necessary and depends on whether the defined relabeling principle heavily relies on the output of the NN model itself. Taking the multi-class classification task of the NN model as an example, in the work of Liu et al. [20], a confidence policy curve, which is independent of the model output (the probabilities that the input sample belongs to each given class), is defined to determine the selection of training and prediction labels. But for the self-error-correcting CNN model proposed by Wu et al. [17], the label of one training sample would be re-labeled if the probability of the predicted label given by the model is greater than the threshold probability. In this case, the pretraining part is needed and the mini-batch stochastic gradient descent (SGD) algorithm is used. Combining the right part of Figure 5 and ignoring the elements marked in red, you can see there are t_e epochs and for each epoch, m_b samples are first randomly selected. Then the input tensor X^s and label matrix $\mathbf{Y}^s$ of each sample are stacked to form X and $\mathbf{Y}$, respectively. Finally, the SGD can optimize the model parameters based on the batch-size input tensor X and label data$\mathbf{Y}$. Through the pretraining with data sample set S_3 (the size of S_3 is not big but the labels of samples are of higher quality), the model parameters are initialized to enable the model to find the correct gradient descent direction during the next iterative training process.

In this study, the bootstrapping algorithm is used to handle the prior path flow labels with noises. Yang et al. [3] presented two indexes α and ρ, which indirectly measure the systematic and the random variations between the true OD matrix and the prior one. Compared with the random error, the systematic error may have a significant influence on model learning. The labels belong to different volume levels, similar to the images belonging to different classes. α is also used in OD matrix estimation to guarantee that the current flow levels are reproduced on an average [21]. And with the mini-batch SGD algorithm used, the gradient descent direction can still be found among the training samples whose labels contain random errors through iterative training. Hence, α is selected as the re-labeling indicator and considering the dynamic path flow estimation and the traffic counts of turning movements directly obtained from AVI system, the calculation of α within time interval h is changed as follows:

$$\alpha_h = \frac{1}{|A^r|} \sum_{a \in A^r} \frac{\sum_{w=h-w'}^{h} \sum_{k \in P} y_k^w \delta_{ha}^{wk}}{q_a^h}, \tag{7}$$

where q_a^h is the measured traffic counts of turning movement a within interval h; y_k^w is the flow of path k departing the origin in interval w; δ_{ha}^{wk} is defined as the contribution of y_k^w passing the turning movement a during interval h and the calculation can refer to the research of Ashok and Ben-Akiva [18].

The elements marked in red in Figure 5 represent the re-labeling process of part 2-2, which is key to the bootstrapping method. Combining equation (7), the calculation of α includes two steps: (1) estimate the flows of the turning movements (selected in X^s and with AVI detectors installed) based on the prior path flows $\mathbf{Y}^s$ and forward result of NN $\mathbf{Y}'^s$, respectively; (2) compare the estimated turning movement flows and the corresponding measured values in $\mathbf{X}^s$, and calculate the average ratio α. For the $\mathbf{Y}^s$ and $\mathbf{Y}'^s$, if the α'^s is closer to 1 than α^s, the original path flow labels are replaced by the estimated labels. It is noted that to use α, the number of temporal cells Δ_h of X^s must be larger than $(2 \times w' + 1)$. Only in this way, the path flows of more than $(w' + 1)$ time intervals in $\mathbf{Y}'^s$ can be estimated and α of at least one time interval can be calculated. For the α of several time intervals, the corresponding average value can be used in the re-labeling process.

3.3. Case Study

3.3. Case Study. In this section, a realistic urban road network in Qingdao, China, is used to validate the proposed model. Considering the designed 3D-CNN model as the

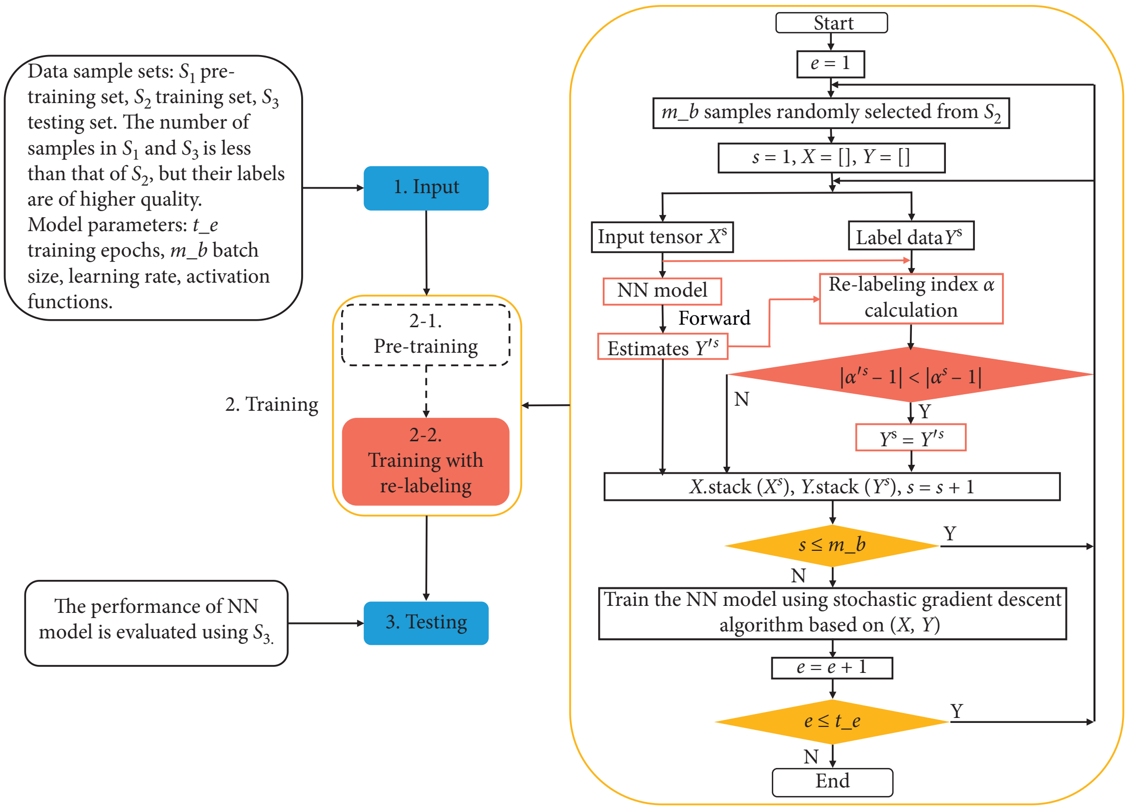

Data sample sets: S_1 pre-training set, S_2 training set, S_3 testing set. The number of samples in S_1 and S_3 is less than that of S_2, but their labels are of higher quality.
Model parameters: t_e training epochs, m_b batch size, learning rate, activation functions.
1. Input
2-1. Pre-training
2. Training
2-2. Training with re-labeling
The performance of NN model is evaluated using S_3.
3. Testing
Start
$e = 1$
m_b samples randomly selected from S_2
$s = 1, X = [], Y = []$
Input tensor X^s
Label data Y^s
NN model
Forward
Re-labeling index α calculation
Estimates Y'^s
$|\alpha'^s - 1| < |\alpha^s - 1|$
N
Y
$Y^s = Y'^s$
X.stack (X^s), Y.stack (Y^s), $s = s + 1$
$s \le m\_b$
Y
N
Train the NN model using stochastic gradient descent algorithm based on (X, Y)
$e = e + 1$
$e \le t\_e$
Y
N
End

Figure 5: Framework of model training using bootstrapping algorithm.

basis of the entire model, the architecture of deep NN is not separately tested. The method of turning movement selection and the robustness of the model with bootstrapping algorithm to noisy labels are focused.

3.4. Network and Dataset Description. The topology of the validating network is shown in Figure 6. The network contains 297 turning movements and the number of real AVI points is 70, accounting for 31.5% of the cross nodes' turns. To generate the large dataset required for model testing, the study of Tang et al. [13] was referred to and the micro-simulation model of validating the network is first established and calibrated to prepare the basic dataset. The model calibration process commonly includes four steps [22]: (1) ensuring key input data (e.g., digital road network, traffic management and control plan) are accurate; (2) calibrating the traffic demands; (3) correcting the related parameters of the simulation model (e.g., car-following, lane-changing parameters, and the distribution of desired speeds); and (4) selecting the evaluation indexes for validation. Steps 1 and 2 are the base and are completed by using the macro and micro simulation software VISUM and VISSIM together. The VISUM is good at modeling a large network and has the TFlowFuzzy module [23], which can correct the OD and path demand matrix based on the observed traffic data. Except for the road network image with high resolution for network modeling, the AVI data for the real AVI points in validating the network were also collected from 7:30 AM to 8:30 AM from November 25 to December 1, 2016. The average flows of these turning movements were used as observed values in the VISUM and the improved static OD matrix and assigned path flows were be obtained. The established macro-VISUM model can be directly imported into VISSIM to further complete the Steps 1, 3, and 4. Considering the micro-simulation model in this paper mainly used to validate the path flow estimation method, the related operational parameters like the distribution of desired speeds are focused and calibrated in Step 3. The errors between the estimated and the actual turning flows are used in Step 4 to evaluate the accuracy of the simulation model. The calculated average relative error is 6.26% less than 15%, and it can be accepted based on the work of Antoniou et al. [22].

Then, the calibrated dataset during the morning peak period is taken as the basis to generate rich scenarios covering noncongested and more congested periods, by adjusting the input volumes and adding the Gaussian noises. There are six levels of the VISSIM input volumes from −0.3 to 0.2 at 0.1 intervals. And the positive or negative volume level represents the corresponding ratio the basic input volumes will increase or decrease by. Under each level, the Gaussian noises (ranging from 5% to 30%) are superimposed on the input volumes and path choice probabilities eight times. For each time, the simulation model with added noises will run five times with different random seeds. Hence, there are 40 scenarios for each level and the total number of scenarios is 240. The number of paths is 311 and the market penetration for probe vehicles is set as 10%. The expanded dataset may be not real, but it is valid for model integrity training and testing.

Here, we define the estimation time interval to be 10 minutes and the one hour simulation period of any scenario can be divided into 6 intervals. The longest trip time among all scenarios is 17 minutes and the value of $(w' + 1)$ is thus 2. Considering at least $(2 \times w' + 1)$ temporal cells of input required for bootstrapping, the $X \in \mathbb{R}^{3\times|A^v|\times|A^v|\times 3}$ and the corresponding true path flows $\mathbf{Y}^+ \in \mathbb{R}^{2\times 311}$ can be paired to form a sample, and each scenario can produce four samples. The total number of samples is 960, and the sample size of pretraining, training, and testing sets are, respectively, 120, 720, and 120 samples.

3.5. Evaluation Metrics and Loss Function. Four error metrics are used for our evaluation: mean absolute error (MAE), relative MAE (%), root mean square error (RMSE), and relative RMSE (%). For m pairs of the true and estimated path flows ($\mathbf{Y}^+ \in \mathbb{R}^{|H|\times|P|}$, $Y' \in \mathbb{R}^{|H|\times|P|}$), the units of MAE and RMSE are veh/10 min, and the calculation formulas are as follows:

$$\begin{aligned}
\text{RMSE} &= \sqrt{\frac{\sum_{s=1}^{m}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\left|\mathbf{Y}_{h,k}^{+s} - \mathbf{Y}_{h,k}^{'s}\right|^2}{m\times|H|\times|P|}},\\
\text{RMSE}(\%) &= \frac{\text{RMSE}}{\left(\sum_{s=1}^{m}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\mathbf{Y}_{h,k}^{+s}/(m\times|H|\times|P|)\right)},\\
\text{MAE} &= \frac{\sum_{s=1}^{m}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\left|\mathbf{Y}_{h,k}^{+s} - \mathbf{Y}_{h,k}^{'s}\right|}{m\times|H|\times|P|},\\
\text{MAE}(\%) &= \frac{\sum_{s=1}^{m}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\left|\mathbf{Y}_{h,k}^{+s} - \mathbf{Y}_{h,k}^{'s}\right|}{\sum_{s=1}^{m}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\mathbf{Y}_{h,k}^{+s}}.
\end{aligned} \tag{8}$$

For a batch including m_b samples of which each is with a prior path flow $\mathbf{Y} \in \mathbb{R}^{|H|\times|P|}$, the mean squared error (MSE) is used as a loss function and the calculation is as follows:

$$\text{MSE} = \frac{\sum_{s=1}^{m\_b}\sum_{h=1}^{|H|}\sum_{k=1}^{|P|}\left(\mathbf{Y}_{h,k}^{s} - \mathbf{Y}_{h,k}^{'s}\right)^2}{m\_b}, \tag{9}$$

3.5.1. Test 1. Turning Movement Selection. For the target network, 110 virtual AVI points, accounting for 37% of all the turning movements, are selected by solving the binary programming model. The number of selected turning movements belonging to cross and noncross nodes is 78 and 32, respectively. Thirt-seven of the virtual AVI points also belong to A^r and the overlap ratio is 33.6%. The selection of noncross nodes' turns can reduce the overall overlap ratio. To completely reflect the dependency of paths and the passed points, the residual 43 turning movements of noncross nodes are finally added to the solution set of the programming model, as shown in Figure 7.

To evaluate the efficiency of the proposed method, two sets of virtual AVI points are defined as follows:

(1) A1: the 297 turning movements of network are all selected
(2) A2: the 153 turning movements selected by the proposed method

The designed 3D-CNN model is used and only the model input is changed according to different sets of turning movements. Hereafter, the 3D-CNN models with A1 and A2 sets are denoted as Model-A1 and Model-A2. The pretraining set is added to the training set and the total number of training samples is 840. During model training, the major hyper parameters, including the training epochs, learning rate, and batch size, were determined based on a grid search experiment. Here, the best combination is used for general evaluation. The number of training epochs, learning rate, and batch size is 5000, 1e-6, and 80, respectively. The train losses varying with epochs for Model-A1 and Model-A2 are shown in Figure 8. The corresponding training durations are marked next to the lines.

Notably, the A2 set is the subset of A1; therefore, the input tensor using A1 set naturally provides much more information. It means that after the same number of training epochs, the Model-A1 can extract more abstract features associated with the labels from the input tensor, which underlies the phenomenon that the train loss of Model-A1 drops faster, as presented in Figure 8. However, because of the larger size of the input tensor using A1 the set, more time is needed to complete the convolutional calculations of conv. layers. Model-A1 and Model-A2 ran in the same environment. The training time needed by Model-A1 is 111 min, which is approximately 4 times that of Model-A2. After 2250 epochs, the drop speed of Model-A2's train loss is higher than that of Model-A1. The final train loss of Model-A2 is 7.1% lower than that of Model-A1.

For the four evaluation metrics (RMSE, RMSE%, MAE, MAE%), the testing results of Model-A1 and Model-A2 on the entire testing set are (3.65 veh/10 min, 45.79%, 1.92 veh/10 min, 24.04%) and (3.58 veh/10 min, 44.93%, 1.88 veh/10 min, 23.55%), respectively. It can be seen that the four metrices of Model-A2 are improved compared with those of Model-A1. The results reveal that the turning movement selection method can identify the critical turning movements for path flow estimation and largely reduce the computation cost.

3.5.2. Test 2: Model Learning on Noisy Labels. According to the label noise analysis in Part 3 of Methodology, the path flow labels may have systematic and random errors. The prior path flow $\mathbf{Y}_{h,k}$ with noises can be generated by

$$\mathbf{Y}_{h,k} = \mathbf{Y}_{h,k}^{+} \times (1.0 - \eta) \times \left(1.0 - c_{vf}\sigma_{hk}\right), \tag{10}$$

where $\mathbf{Y}_{h,k}^{+}$ is the true path flow of path k during time interval h, σ_{hk} is the independent normal random variable of $N(0,1)$, η is the bias, and c_{vf} reflects the magnitude of random variation.

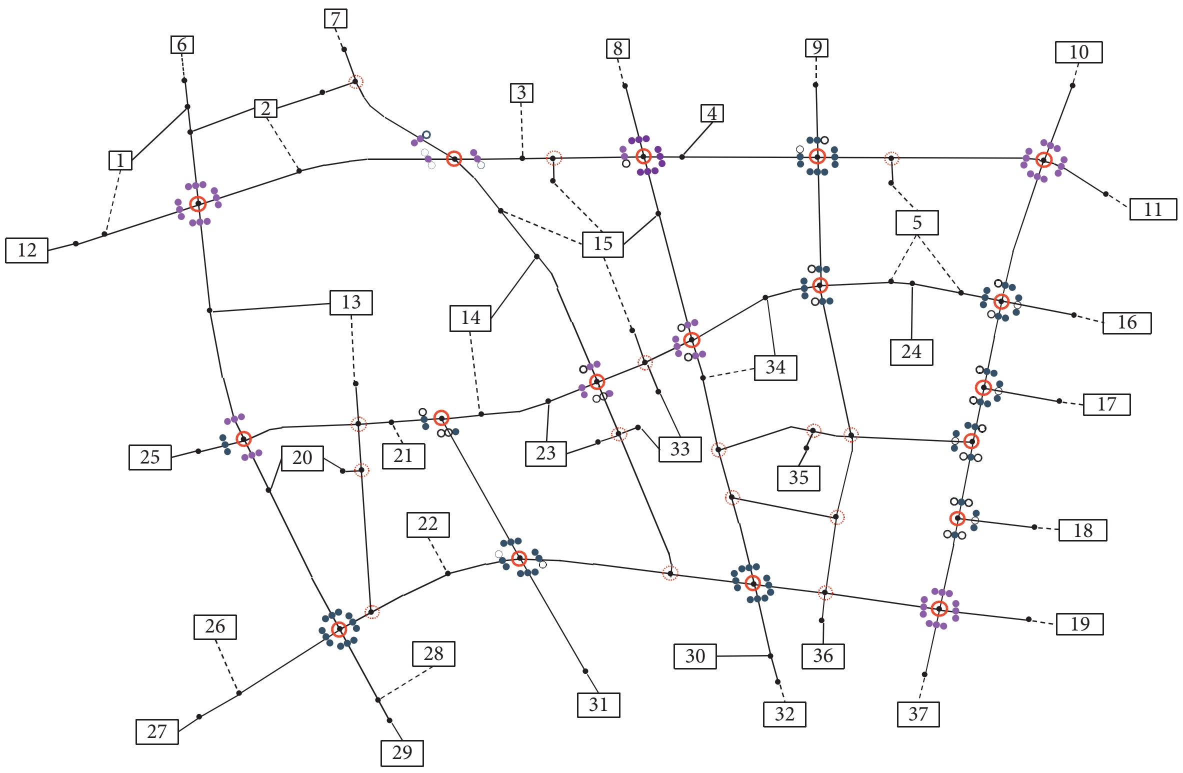


FIGURE 6: The topology of network and distribution of real AVI points.

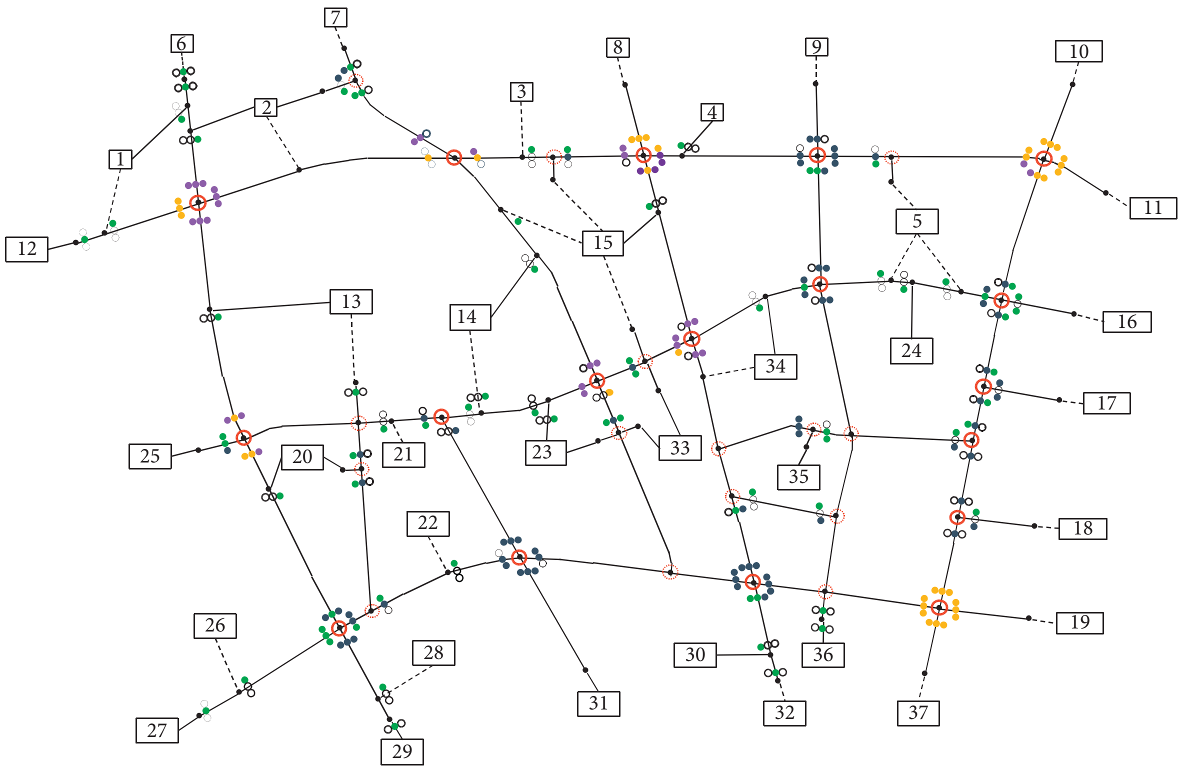


FIGURE 7: The distribution of optimally selected virtual AVI points.

Based on the results of Test 1, Model-A2 is used for validation here. Without using the bootstrapping training algorithm, Model-A2 is trained with training samples from pretraining and training sets whose labels are noisy. The number of training epochs and learning rate are kept the same, while the batch size is changed to 120. Table 1 presents

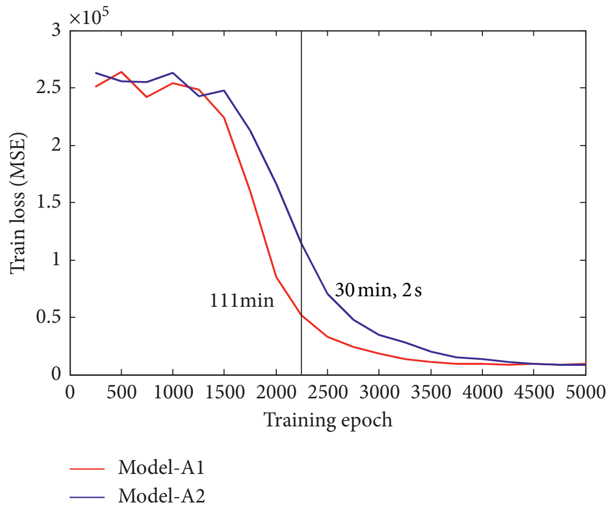


Figure 8: Train loss variation with increased train epochs.

Table 1: Performance of Model-A2 trained by samples with noisy labels.

η	0			0.2			0.4		
c_{vf}	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
RMSE veh/10 min	3.58	3.56	3.62	5.49	5.45	5.26	8.87	8.81	8.72
RMSE %	44.90	44.62	45.39	68.84	68.36	65.93	111.28	110.58	109.33
MAE veh/10 min	1.88	1.87	1.89	2.39	2.37	2.32	3.56	3.56	3.52
MAE %	23.54	23.48	23.76	29.95	29.73	29	44.69	44.6	44.12

the model performance in the testing set under different combinations of η and c_{vf}.

Table 1 reveals that the labels with systematic errors have a great influence on the model performance, while the model is generally insensitive to random errors. All four metrics increase significantly with an increase in η. However, under the same value of η, the variations of the four metrics are slight when c_{vf} increases from 0.2 to 0.6. It meets our expectation and the (stochastic) gradient descent algorithm is immune to training samples with random errors. However, it should be noted that owing to the over-parameterized NN, the CNN model has the capacity to (over) fit a subset of labels with random noises. The over-fitting should be avoided.

Considering the existence of the pretraining phase, the test for bootstrapping algorithm is divided into two parts. The model used in Part 1 is the pretrained Model-A2, while in Part 2, the pretraining phase is skipped. For the pre-training set, the labels of samples are just added to the random errors with the c_{vf} set as 0.2. Based on the model performance measurements of Table 1, the η and c_{vf} are, respectively, set as 0.4 and 0.2 to generate noises for the labels of samples in the training set. As for the percentage of noisy labels in the training set, 0.1, 0.3, 0.5, 0.7, and 0.9 are used.

In Part 1 and Part 2, the training sets with different percentages of noisy labels are used to train Model-A2 and the corresponding model performances on the testing set are listed in Table 2.

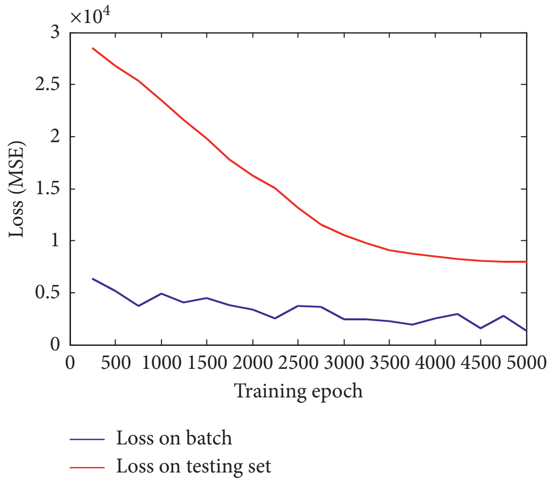
In Table 2, each parenthesis contains two metric values separated by a forward slash and the difference lies in whether the bootstrapping algorithm is used for Model-A2's training on the training set. For the pretrained Model-A2 used in Part 1, the corresponding four metrics on the testing set are 7.03 veh/10 min, 88.14%, 2.85 veh/10 min, and 35.7%. From the first values of parentheses in Part 1, it can be observed that the trained model performs gradually worse as the percentage of noisy labels in the training set increases. But for the Model-A2 trained on the training set with the bootstrapping algorithm, it performs better and the percentage of noisy labels has less influence on model performance. The trends of the values in Part 2 are similar to those in Part 1, but the values increase overall. It is not difficult to understand that the weight parameters are randomly initialized and the convergence speed is slow without the pretraining phase.

Take the label noise ratio of 70% as an example. For the models trained with bootstrapping in Part 1 and Part 2, the train loss, sample loss distribution within one epoch, and label replaced ratios are shown in Figure 9.

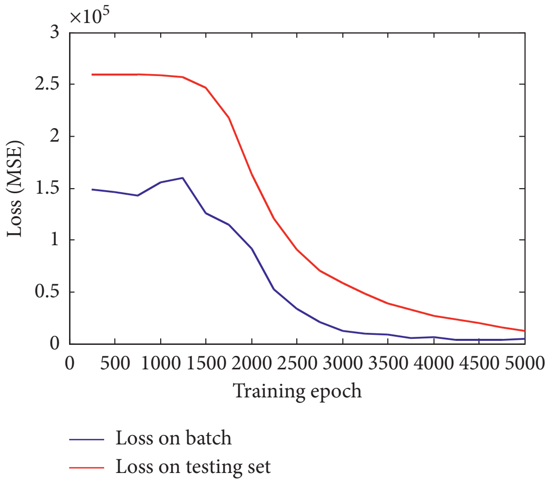
Figures 9(a) and 9(b) indicate that the loss on the testing set of Part 1 can converge to a good point within the total 5000 iterations, while that of Part 2 keeps decreasing. It reveals that the pretraining phase can effectively accelerate the convergence of the model. Owing to the pretraining phase, the label replacement ratio for training samples with noisy labels in Part 1 is nearly 100% within the entire training process, as shown in Figure 9(e). But in Figure 9(f),

Table 2: Test of bootstrapping algorithm for different percentages of noisy labels.

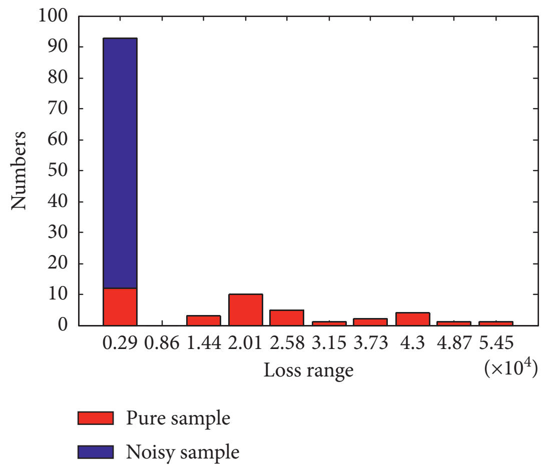
	Percentage	10%	30%	50%	70%	90%
Part 1 (with pretraining)	RMSE	(3.63/**3.55**)	(4.29/**3.55**)	(5.37/**3.56**)	(6.57/**3.58**)	(8.08/**4.45**)
	RMSE%	(45.56/**44.50**)	(53.84/**44.50**)	(67.35/**44.60**)	(82.37/**44.88**)	(101.35/**55.83**)
	MAE	(1.89/1.89)	(2.05/**1.89**)	(2.36/**1.89**)	(2.75/**1.91**)	(3.27/**2.06**)
	MAE%	(23.73/**23.72**)	(25.67/**23.72**)	(29.64/**23.75**)	(34.50/**23.94**)	(41.07/**25.89**)
Part 2 (without pretraining)	RMSE	(3.71/3.74)	(4.36/**3.8**)	(5.43/**3.99**)	(6.57/**4.48**)	(8.12/**7.79**)
	RMSE%	(46.60/46.87)	(54.72/**47.69**)	(68.18/**50.09**)	(82.42/**56.18**)	(101.89/**97.68**)
	MAE	(1.90/1.93)	(2.05/**1.94**)	(2.35/**1.97**)	(2.72/**2.04**)	(3.28/**2.68**)
	MAE%	(23.86/24.26)	(25.7/**24.34**)	(29.52/**24.8**)	(34.14/**25.65**)	(41.14/**33.66**)



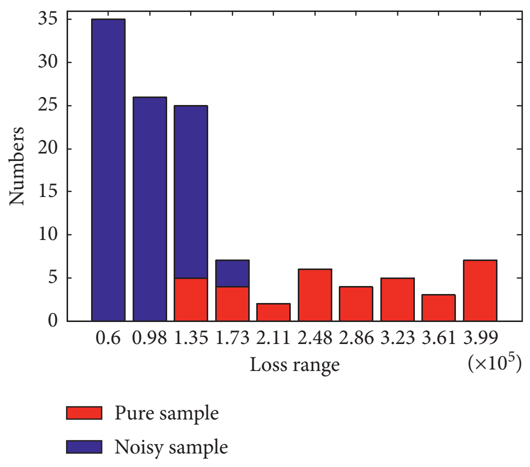
(a)



(b)

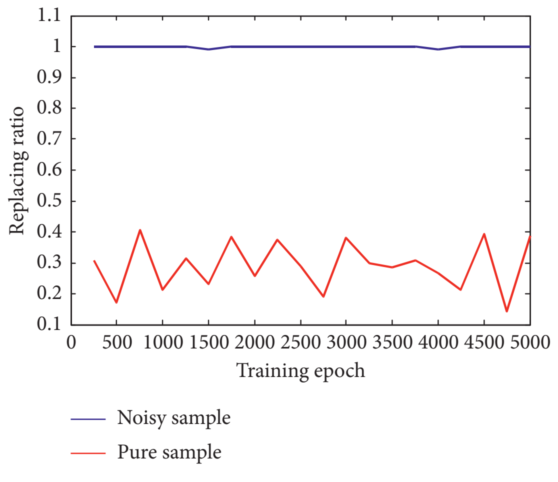


(c)

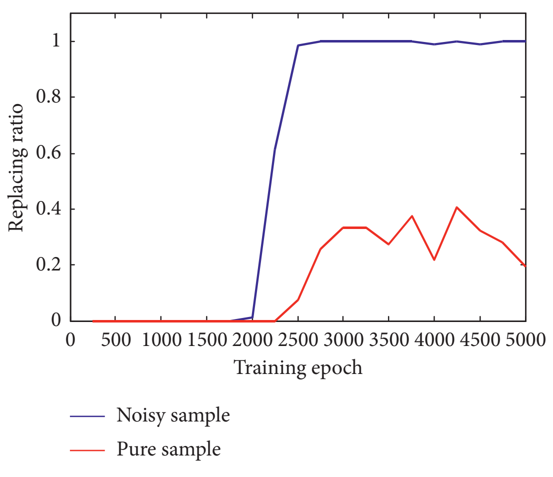


(d)

Figure 9: Continued.



(e)



(f)

Figure 9: Train losses, sample loss distributions, and label replacing ratios in Part 1 and Part 2. (a, e) Part 1, (b, f) Part 2, (c) No. 250 epoch of Part 1, and (d) No. 250 epoch of Part 2.

during the initial iterations, the label replacement ratio for noisy samples is nearly zero and it increases to 100% after 2000 epochs. For the sample loss distribution at the epoch No. 250 shown in Figure 9(d), the training samples with noisy labels are not re-labeled and occupy most of the training batch; therefore, they become the focus of regression and the losses of pure samples are naturally higher than those of Part 1. Figure 9(c) can be used to explain why the bootstrapping algorithm can reduce the influence of noisy labels. After resampling the samples and relabeling the corresponding noisy labels, in the next epoch, the losses for noisy samples are initially by zero and the start point of gradient descent is placed in the pure samples. The fluctuations of loss in the batch in Figures 9(a) and 9(b) are reasonable, because the percentages of pure and noisy samples are different in every resampled sample. With the improvement in model estimation accuracy, the label replacement ratio for pure samples is between 0.2 and 0.4.

4. Conclusions

In this study, to make full use of the rich and complementary individuals' trip information provided by AVI and probe vehicle data, and to avoid intractable mathematical program solution, the dynamic path flow estimation is treated as a data-driven feature learning problem and these two data sources are fused at the data level. A 3D convolution-based deep NN is designed, and the turning movements at network nodes are used to represent the AVI and the probe vehicle observations in the input tensor. The principles for selecting the key turning movements and a corresponding programming model are also proposed. To make the NN robust to the noisy path flow labels during model training, a self-correcting algorithm named bootstrapping, which can use the model outputs to correct the noisy labels based on the defined re-labeling principle, is established.

In the case study, a realistic urban road network was used and the corresponding microscopic simulation model was built and calibrated by VISSIM to generate large data samples. Two distinctive tests, numbered 1 and 2, were carried out to validate the turning movement selection and bootstrapping methods. In Test 1, the designed 3D-CNN model with the input tensor constructed by the selected turning movements achieved a MAE of 1.88 veh/10 min, and compared with the model with all the turning movements used in input tensor, the computational time and estimation accuracy were both improved. This reveals that the designed architecture of 3D-CNN model presents satisfactory performance, and the virtual AVI point selection method can retain the key information for each path and remove redundant information. In Test 2, the path flow labels were artificially superimposed with systematic and random errors to test the model robustness. Without overfitting, the NN model trained with the gradient descent algorithm is almost immune to the labels with random errors. Systematic errors were mainly considered and the bootstrapping can make the model more robust to different percentages of labels with errors. The pretraining phase is not necessary in this study, but it can help improve the convergence speed and estimation accuracy. The defined re-labeling criteria are important and can limit the final estimation accuracy.

Despite the promising results, the study has certain limitations and further works could be focused on three aspects to extend the topic: firstly, the dynamic path choice in congested networks should be considered in the virtual AVI points selection model; secondly, the influence of various modes of market penetration of probe vehicles and feature noises (e.g., the missing detection phenomenon in AVI system) should be further investigated; last but not least, a real-world validation with filed AVI data and probe vehicle trajectories has to be conducted.

Data Availability

The AVI and probe vehicle trajectory data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors thank the National Key Research and Development Program of China (2019YFB1600202) for supporting this research.

References

[1] Y. Nie and D.-H. Lee, "Uncoupled method for equilibrium-based linear path flow estimator for origin-destination trip matrices," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1783, no. 1, pp. 72–79, 2002.

[2] E. Castillo, J. M. Menéndez, and P. Jiménez, "Trip matrix and path flow reconstruction and estimation based on plate scanning and link observations," *Transportation Research Part B: Methodological*, vol. 42, no. 5, pp. 455–481, 2008.

[3] H. Yang, T. Sasaki, Y. Iida, and Y. Asakura, "Estimation of origin-destination matrices from link traffic counts on congested networks," *Transportation Research Part B: Methodological*, vol. 26, no. 6, pp. 417–434, 1992.

[4] X. Zhou and H. S. Mahmassani, "Dynamic origin-destination demand estimation using automatic vehicle identification data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 105–114, 2006.

[5] P. Krishnakumari, H. van Lint, T. Djukic, and O. Cats, "A data driven method for OD matrix estimation," *Transportation Research Part C: Emerging Technologies*, vol. 113, pp. 38–56, 2020.

[6] M. P. Dixon and L. R. Rilett, "Real-time OD estimation using automatic vehicle identification and traffic count data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 17, no. 1, pp. 7–21, 2002.

[7] T. Huang, Y. Ma, and Z. T. Qin, "Origin-destination flow prediction with vehicle trajectory data and semi-supervised recurrent neural network," in *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*, IEEE, Los Angeles, CA, USA, December 2019.

[8] J. Yang and J. Sun, "Vehicle path reconstruction using automatic vehicle identification data: an integrated particle filter and path flow estimator," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 107–126, 2015.

[9] X. Yang, Y. Lu, and W. Hao, "Origin-destination estimation using probe vehicle trajectory and link counts," *Journal of Advanced Transportation*, vol. 2017, pp. 1–18, 2017.

[10] R. Ásmundsdóttir, *Dynamic OD Matrix Estimation Using Floating Car Data*, Delft University of Technology, Delft, The Netherlands, 2008.

[11] W. Wong, S. C. Wong, and H. X. Liu, "Bootstrap standard error estimations of nonlinear transport models based on linearly projected data," *Transportmetrica A: Transport Science*, vol. 15, no. 2, pp. 602–630, 2019.

[12] C. Osorio, "Dynamic origin-destination matrix calibration for large-scale network simulators," *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 186–206, 2019.

[13] K. Tang, Y. Cao, C. Chen et al., "Dynamic origin-destination flow estimation using automatic vehicle identification data: a 3D convolutional neural network approach," *Computer-Aided Civil and Infrastructure Engineering*, vol. 2020, pp. 1–17, 2020.

[14] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2013.

[15] H. Alibabai and H. S. Mahmassani, "Dynamic origin-destination demand estimation using turning movement counts," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2085, no. 1, pp. 39–48, 2008.

[16] S. Reed, H. Lee, and D. Anguelov, "Training deep neural networks on noisy labels with bootstrapping," 2014, https://arxiv.org/abs/1412.6596.

[17] X. Wu, R. He, Z. Sun, and T. Tan, "A light CNN for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[18] K. Ashok and M. E. Ben-Akiva, "Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows," *Transportation Science*, vol. 36, no. 2, pp. 184–198, 2002.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[20] X. Liu, S. Li, and M. Kan, "Self-error-correcting convolutional neural network for learning with noisy labels," in *Proceedings of the 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, IEEE, Washington, DC, USA, June 2017.

[21] L. G. Willumsen, "Estimating time-dependent trip matrices from traffic counts," in *Proceedings of the 9th International Symposium on Transportation and Traffic Theory*, VNU Science Press Utrecht, Delft, The Netherlands, July 1984.

[22] C. Antoniou, J. Barcelò, and M. Brackstone, "Traffic simulation: case for guidelines," 2014.

[23] A. G. PTV, *How to Work with TFlow Fuzzy*, Official Manual, PTV Vision Visum, Karlsruhe, Germany, 2008.