Transportation Research Part C 116 (2020) 102644

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A deep convolutional neural network based approach for vehicle classification using large-scale GPS trajectory data☆

Sina Dabiri[a,b,*], Nikola Marković[c], Kevin Heaslip[a], Chandan K. Reddy[b]

[a] Department of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA, USA
[b] Department of Computer Science, Virginia Tech, Arlington, VA, USA
[c] Department of Civil and Environmental Engineering, University of Utah, Salt Lake City, UT, USA

ARTICLE INFO

Keywords:
Deep learning
Vehicle classification
GPS data
Convolutional neural networks

ABSTRACT

Transportation agencies are starting to leverage increasingly-available GPS trajectory data to support their analyses and decision making. While this type of mobility data adds significant value to various analyses, one challenge that persists is lack of information about the types of vehicles that performed the recorded trips, which clearly limits the value of trajectory data in transportation system analysis. To overcome this limitation of trajectory data, a deep **C**onvolutional **N**eural **N**etwork for **V**ehicle **C**lassification (**CNN-VC**) is proposed to identify the vehicle's class from its trajectory. This paper proposes a novel representation of GPS trajectories, which is not only compatible with deep learning models, but also captures both vehicle-motion characteristics and roadway features. To this end, an open source navigation system is also exploited to obtain more accurate information on travel time and distance between GPS coordinates. Before delving into training the CNN-VC model, an efficient programmatic strategy is also designed to label large-scale GPS trajectories by means of vehicle information obtained through Virtual Weigh Station records. Our experimental results reveal that the proposed CNN-VC model consistently outperforms both classical machine learning algorithms and other deep learning baseline methods. From a practical perspective, the CNN-VC model allows us to label raw GPS trajectories with vehicle classes, thereby enriching the data and enabling more comprehensive transportation studies such as derivation of vehicle class-specific origin-destination tables that can be used for planning.

1. Introduction

The advances in Global Position Systems (GPS) and ever-increasing market penetration of GPS-equipped devices (e.g., smart phones) enabled generation of massive spatiotemporal trajectory data that capture human mobility patterns. Availability of large-scale trajectory data motivated transportation agencies to leverage such information for measuring transportation network performance and supporting their decision making. However, even though this type of mobility data adds significant value to various transportation analyses, one challenge that persists is lack of information about the classes of vehicles that performed the recorded trips. Namely, users of different GPS navigation apps (e.g., Google Maps) typically do not specify the type of vehicle they are driving, making it unclear to a transportation analyst whether a recorded trip was performed by a passenger car, small commercial vehicle, or

☆ This article belongs to the Virtual Special Issue on "Machine learning".

* Corresponding author.
E-mail address: sina@vt.edu (S. Dabiri).

https://doi.org/10.1016/j.trc.2020.102644
Received 25 November 2018; Received in revised form 10 April 2020; Accepted 12 April 2020

Table 1
FHWA Vehicle Classification System.

#Class	Class Definition	#Class	Class Definition
1	Motorcycles	8	Four or Fewer Axle Single-Trailer Trucks
2	Passenger Cars	9	Five-Axle Single-Trailer Trucks
3	Other Two Axle, Four tires, Single-Unit	10	Six or More Axle Single-Trailer Trucks
4	Buses	11	Five or Fewer Axle Multi-Trailer Trucks
5	Two-Axle, Six-Tire, Single-Unit Trucks	12	Six-Axle Multi-Trailer Trucks
6	Three-Axle Single-Unit Trucks	13	Seven or More Axle Multi-Trailer Trucks
7	Four or More Axle Single-Unit Trucks	–	–

even an eighteen-wheeler truck. Obviously, having this piece of information would be extremely useful for a wide range of applications in Intelligent Transportation Systems (ITS) including emission control, pavement designation, traffic flow estimation, and urban planning (Marković et al., 2018). Furthermore, unlike fixed-point traffic flow sensors (e.g., loop detectors and video cameras), GPS sensors are not limited by sparse coverage and can be considered as an efficient and ubiquitous data source. Accordingly, our main objective in this paper is to enrich the increasingly available trajectory data by developing a model that is capable of identifying the type of a vehicle that produced a GPS trajectory.

However, the definition of vehicle categories is highly contingent on the ITS application and data-collection sensors. For example, vehicles can be classified based on the number of axles or weights for pavement designation and ETC systems. Among various categorizations, the vehicle-classification proposed by the US Federal Highway Administration (FHWA) in the mid 1980s has been accepted as the most standard scheme and served as the basis for a majority of state-counting efforts. The number of axles, the spacing between axles, the first axle weight, and gross vehicle weight are the main factors for categorizing vehicles in the FHWA system. Table 1 provides information on the 13 vehicle categories according to the FHWA definitions. The FHWA vehicle classification might slightly change from State to State depending on the types of vehicles that are allowed to commute in that State. Nonetheless, in real-world situations, vehicles can be regrouped to more coarse-grained classes, such as {light, medium, heavy} (Simoncini et al., 2018), {passenger cars, trucks} (Sun and Ban, 2013), and {sedan, pickup truck, SUV/minivan} (Kafai and Bhanu, 2012).

The common approach for classifying vehicles is based on fixed-point traffic sensors, which are categorized into two groups: (1) in-roadway sensors that are embedded in pavement or attached to road surfaces, including inductive-loop detectors, magnetometers, and Weigh-In-Motion (WIM) systems, (2) over-roadway sensors that are mounted above the surface, including microwave radar sensors, laser radar sensors, and ultrasonic infrared sensors. When a vehicle crosses them, these sensors are capable of calculating the vehicle's axle information (e.g., the number of axles and the space between them). Video image processor (VIP) is another over-roadway sensor that has extensively been deployed for the vehicle classification, in particular due to advances in vision-based techniques (Gupte et al., 2002; Kafai and Bhanu, 2012). After a vehicle is detected through VIP, computer-vision algorithms are exploited to first extract robust features from images and/or videos, and then classify the vehicle into pre-defined groups using supervised learning algorithms. However, there are two major shortcomings regarding the fixed-point traffic flow sensors including: (1) the installation and maintenance of such technologies not only requires extra costs but also necessitates road closure and physical changes to road infrastructure, (2) they can be used only for the location where they were installed while many ITS applications require dynamic vehicle classification across a wide area.

One cost-effective way to address the above-mentioned issues is to use well-established positioning tools such as Global Position Systems (GPS) that enable to record spatiotemporal information of vehicles' trips. Unlike traffic flow sensors, GPS infrastructures does not impose extra costs as the system has already been designed for purposes not related to traffic management. Furthermore, GPS data are not limited by sparse coverage and can be considered as an efficient and ubiquitous data source for several transportation-domain applications including traffic state estimation, traffic incident detection, mobility pattern extraction, travel demand analysis, and transport mode inference (Dabiri and Heaslip, 2018c). Accordingly, this study seeks to harness the GPS data for addressing the vehicle-classification problem.

Before proceeding to development of a vehicle-classification model, for the first time, we label a large-scale GPS trajectory dataset according to the fine-grained FHWA vehicle categories. A vehicle's GPS trajectory is constructed by connecting a sequence of GPS points, recorded by means of an on board GPS-enabled device, where a GPS point contains information about the device's geographic location at a particular moment. The GPS data are recorded from vehicles that have traveled in the State of Maryland over the period of four months in 2015. To this end, a separate WIM dataset that contains the FHWA class information of the vehicles that passed Virtual Weight Stations (VWS) is used. An efficient programmatic-labeling strategy is designed to assign the vehicle class information from the WIM dataset to the GPS trajectories that have passed the corresponding VWS. Our programmatic labeling is accurate enough while saving costs and times compared to manual labeling, which is too expensive and time-consuming.

Having a large volume of labeled GPS trajectories, the GPS-based vehicle-classification problem can be examined for several vehicle-categorization scenarios. However, a raw GPS trajectory contains only a sequence of GPS points without any meaningful correlations and explicit features for feeding into supervised models. Feature engineering is common approach for extracting some hand-crafted features using the descriptive statistics of motion characteristics such as maximum speed and acceleration. Nonetheless, feature engineering not only requires expert knowledge but also involves biased engineering justification and vulnerability to traffic and environmental conditions (Dabiri and Heaslip, 2018b). For example, different types of vehicles may have the same speed in a traffic jam. Automated feature learning methods such as deep learning architectures are a remedy for resolving shortcomings with

hand-designed features. But, a raw GPS trajectory with a sequence of GPS coordinates and timestamps neither has an adaptable structure for passing into deep-learning algorithms nor conveys meaningful information (Dabiri and Heaslip, 2018b; Dabiri et al., 2019). Accordingly, before deploying deep-learning algorithms, a novel GPS representation is designed to convert the raw GPS trajectory into a structure that contains both vehicle-motion characteristics and roadway-geometric information associated with the GPS trajectory. Afterward, a deep Convolutional Neural Network for Vehicle-Classification (CNN-VC) is proposed to identify the vehicle class from its GPS trajectory. The developed classification method can be subsequently deployed to identify vehicle classes of the remaining GPS trajectories in the original dataset, which adds a very important dimension to the dataset that is widely used to support analysis of the Maryland State Highway Administration (e.g., derive class-specific trip tables or link-based traffic volumes). Accordingly, this paper makes the following contributions:

- **Labeling a large-scale GPS trajectory dataset based on the FHWA classification scheme**. An efficient programmatic approach is developed to label GPS trajectories by means of vehicle class information obtained from VWS vehicle records. Information from an open source routing engine is also exploited to improve the accuracy of our labeling approach.
- **Designing a new representation for raw GPS trajectories**. First, a GPS trajectory is considered as a sequence of GPS legs, where each leg is the route segment between two consecutive GPS points. Then, a feature vector is computed for every GPS leg, which is a combination of motion-related and road-related features. Concatenating the feature vector corresponding to all GPS legs generates an efficient representation for each GPS trajectory.
- **Developing a deep Convolutional Neural Network for Vehicle-Classification (CNN-VC)**. For the first time in this domain, a CNN-based deep-learning model is developed to identify vehicle's class from the proposed GPS representation. The CNN-VC comprises a stack of CNN layers for extracting abstract features from the GPS representation, a pooling operation for encapsulating the most important information, and a softmax layer for performing the classification task.
- **Conducting an extensive set of experiments for evaluating the proposed model**. In addition to the classical machine-learning techniques, several state-of-the-art deep-learning algorithms are developed to be used as baselines. Experimental results reveal that our CNN-VC model clearly outperforms both classical-supervised algorithms and deep-learning architectures. The models were compared on the problem of differentiating between FHWA vehicle classes 2 and 3, as well as the problem of identifying the light-medium-heavy duty vehicles.

The rest of this article is organized as follows. After reviewing related works in Section 2, the datasets and the labeling procedure are described in Sections 3 and 4, respectively. The details of our proposed framework are elaborated in Section 5. In Section 6, numerous experiments are conducted to evaluate the performance of the proposed framework. Finally, conclusions are drawn in Section 7.

2. Related works

Since vehicle classification is essential for many tasks in ITS, a considerable amount of literature has been published on developing frameworks for identifying the vehicle's class. Two primary types of sensors that have been used to develop vehicle-classification systems include: (1) fixed-point traffic flow sensors, and (2) floating sensors using GPS data. In this section, after briefly reviewing the literature on the first group, the few studies on the vehicle classification using GPS data are examined.

2.1. Fixed-point sensors for vehicle classification

A large and growing body of literature has addressed the vehicle-classification problem using various fixed-point traffic data-collection systems over the past decades. In-roadway (e.g., loop detectors, magnetic sensors, and piezoelectric sensors) and over-roadway (e.g., radar sensors, infrared sensors, and VIP) are the two main types of traffic flow sensors that have been exploited for classifying vehicles. Since in-roadway sensors interrupt traffic flow during installation and maintenance, they are more appropriate for urban areas rather than high-speed roads such as freeways. On the other hand, over-roadway sensors are more appropriate for capturing vehicles information in high-speed, high-capacity roads as they are much less disruptive to traffic flows (Sun and Ban, 2013). The idea behind vehicle classification using fixed-point sensors is to capture vehicles' physical characteristics (e.g., axle configuration and length) and then categorize vehicles according to a pre-defined set of classes. A dual-loop detector system, as an example of in-roadway sensors, can simply estimate the length of a vehicle by computing the speed and occupancy time and establish a length-based classification system (Coifman and Kim, 2009). A VIP system, as an example of on-roadway sensors, models vehicles as rectangular patches in a sequence of images recorded by a camera. Afterward, vehicles are classified to several groups using visual features (e.g., width, length, area, and perimeter) computed by traditional vision-based techniques (Kafai and Bhanu, 2012) or abstract automatic features learned by deep-learning algorithms (Dong et al., 2015). Sensors can also be integrated to form a more robust classifier system for fine-grained classification. A WIM system, which is comprised of piezoelectric sensors, VIP sensors, loop detectors, etc., is capable of classifying vehicles based on the 13-category FHWA classification scheme (Hallenbeck et al., 2014). Further details on pros and cons of various vehicle-classification systems using fixed-point technologies are available in (Sun and Ban, 2013). Since the specific objective of this study is to develop a GPS-based vehicle-classification system, we focus more on the studies that have harnessed GPS data for classifying vehicles.

2.2. GPS-based vehicle-classification

In spite of the popularity in fixed-point traffic flow sensors, they incur a high initial cost for installation and a permanent cost for maintenance. Furthermore, they are unable to spatially cover a wide area and in turn their usage is limited to the installation location. Probe vehicles equipped with positioning tools such as GPS can be a viable alternative to fixed-point sensors, due to the fast-growing market-penetration rate of GPS-enabled devices (e.g., smart phones). GPS trajectories, which summarize vehicle's spatio-temporal information, can be recorded when vehicles are performing regular trips. Over the past decade, much research has been carried out to harness GPS trajectories for several mobility applications, ranging from vehicle fleet management, human mobility behavior, and inferring significant locations to travel mode detection and travel anomaly detection (Dabiri and Heaslip, 2018c). A comprehensive and systematic review on various aspects of trajectory data mining including trajectory data preprocessing, trajectory data management, and several trajectory mining tasks is available in (Zheng, 2015). Although numerous studies leveraged GPS data in various mobility applications, only a few studies have developed vehicle-classification systems based on GPS trajectories.

Sun and Ban (2013), for the first time, utilized GPS data to distinguish passenger cars from trucks, yet on a very small dataset with 52 samples of passenger cars and 84 samples of trucks. Compared to speed-related features, they found that the features related only to acceleration and deceleration (e.g., the proportions of acceleration and deceleration larger than 1 m/s^2, and the standard deviations of acceleration and deceleration) result in a better classification performance. Acceleration-based features are then passed into the Support Vector Machine (SVM) classifier for the classification task. In a recent study by Simoncini et al. (2018), a deep-learning framework based on Recurrent Neural Networks (RNN) was developed to classify vehicles into three categories: light-duty, mid-duty, and heavy-duty. The input data for the RNN model is a sequential point-wise GPS information. Such a sequence is created by computing a set of point-wise features including distance, time, speed, acceleration, and road type for every GPS point in a GPS track. Their proposed model was trained using almost 1 million GPS tracks collected by a fleet intelligence company. Note that a growing body of literature has recently leveraged the power of deep learning algorithms for solving various applications in the transportation domain (Dabiri and Heaslip, 2018a; Sekuła et al., 2018).

A similar research track belongs to travel mode detection using GPS trajectories. There the objective is to identify transportation mode(s) used by travelers for making their regular trips based on GPS data. Analogous to the vehicle-classification problem, the problem of travel mode detection consists of two main steps: (1) extracting features from GPS logs, either by means of feature engineering (Zheng et al., 2008) or automated feature learning methods (Dabiri and Heaslip, 2018b), (2) feeding features to learning algorithms for the classification task. However, it is obvious that distinguishing between the GPS patterns of transportation modes (e.g., walk and car) is less challenging than discriminating between GPS patterns of vehicles (e.g., passenger car and pickups).

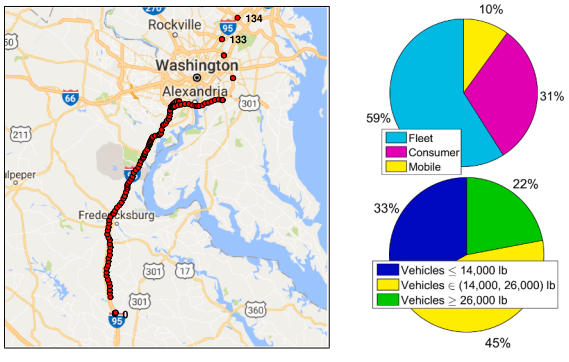
To the best of our knowledge, no GPS dataset annotated with the FHWA vehicle categories is available. Thus, we first design an efficient time-based-matching process for labeling a large number of GPS trajectories by the help of a separate WIM dataset, which contains the FHWA class information. Furthermore, a novel representation for GPS tracks is proposed to consider both kinematic motion characteristics of vehicles and roadway features for all road segments between every two consecutive GPS points in a GPS track. Finally, for the first time, a convolutional-based deep-learning model is applied to the new GPS representation for discriminating GPS trajectories according to their vehicle classes.

3. Data

This section describes the GPS trajectory data, VWS data, and Open Source Routing Machine (OSRM) that will be instrumental in preparing a dataset to train vehicle classification models.

3.1. GPS trajectory

Trajectory data used in this paper comes from one of the leading GPS data providers in North America, which collects data from



(a) Sample trajectory (b) Trip info

Fig. 1. A sample GPS trajectory and info about 20 million trips (i.e., vehicle weights and data providers).

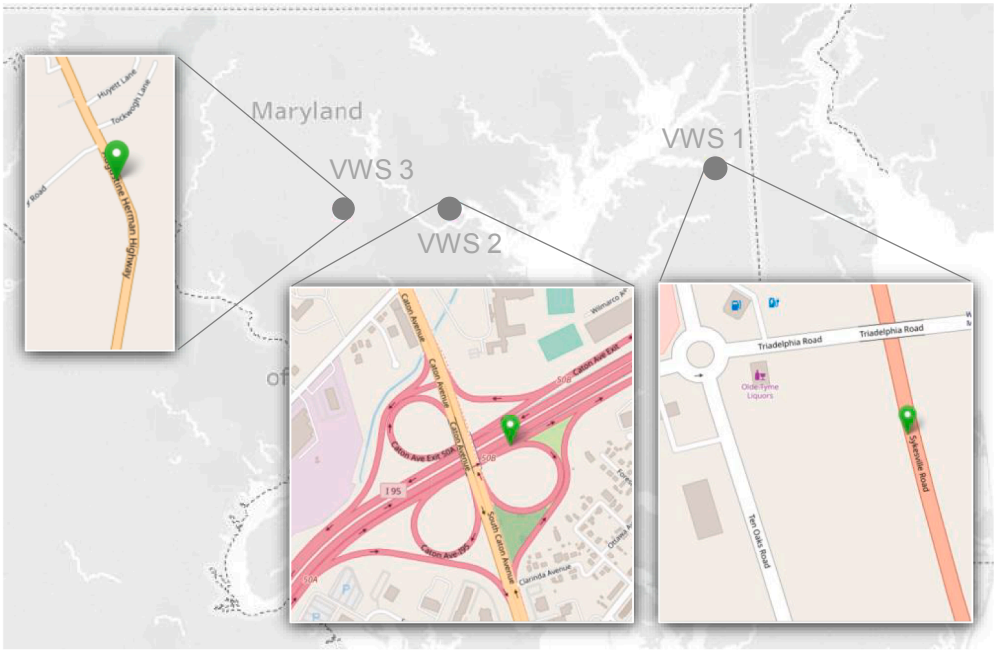


Fig. 2. The location and roadway network associated with three Virtual Weight Stations (VWS) in the state of Maryland, used for the labeling process. The green map markers depict the location of VWS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

several hundred million vehicles and devices across the globe. As a sample data, Fig. 1(a) presents a single GPS trajectory, which consists of vehicle locations and corresponding timestamps. The considered dataset includes 20 million GPS trajectories over a period of four months in 2015: February, June, July, and October. Every GPS trajectory is defined by a unique trip ID and a device ID. Trajectories have also been categorized into three groups based on the vehicle gross weight: (1) below 14,000 lbs (FHWA classes 1–3), (2) between 14,000 lbs and 26,000 lbs (FHWA classes 4–6), and (3) above 26,000 lbs (FHWA classes 7–12). These trajectories span over the entire state of Maryland and include a relatively high percentage of vehicles with weights above 14,000 lbs, as shown in Fig. 1(b). The average time lapse between two consecutive GPS points is 17 s.

3.2. Virtual weight stations data

A VWS is a roadside enforcement facility that eliminates the need for continuous staffing by automatically identifying and recording vehicles' characteristics on a real-time basis. Each station is comprised of several components, including WIM sensors (for computing vehicles' weights and axle configuration), camera system (for real-time identification of vehicles), screening software (for integrating data from WIM and camera systems), and communication infrastructure (for transferring the recorded information to authorized personnel such as mobile enforcement teams). Accordingly, WIM sensors in a VWS are capable of determining the approximate gross weight and axle configurations (e.g., weight, number, and spacing) in real-time, which are in turn used for the vehicle classification (Capecci et al., 2009).

The VWS records considered in this study include information collected at 7 VWS in the state of Maryland over the period of four months in 2015: February, June, July, and October. However, due to programmatic nature of the labeling strategy described in the following sections, only three stations installed on road segments with one lane per direction are utilized for the labeling purpose. Fig. 2 illustrates the approximate location and the roadway network around these three VWS. For every vehicle crossing WIM sensors, various information has been recorded including: crossing time, crossing lane, class, number of axles, speed, gross weight, and length. The class definitions are based on the FHWA vehicle-classification system, as described in Table 1. Class 1 and 13 are not available in this dataset. Among all attributes, only vehicle's crossing time, class, and gross-weight features are deployed in our labeling process. It should be noted that there is no pre-relation between the vehicles in GPS and VWS datasets. Indeed, the main goal of our labeling strategy is to match vehicles in these two datasets.

3.3. Open source routing machine

Open Source Routing Machine (OSRM) is a web-based navigation system that computes the fastest path between an origin-destination pair, using the Open Street Map (OSM) data. The other functional services that OSRM provides, include map-matching, nearest matching, traveling salesman problem solving, and generating vector tiles. All of these services are accessible through its Application Programming Interface (API) methods. The main advantage of the OSRM APIs is that it can be used for free and without

usage limits, which makes it a valuable resource for research purposes. The following OSRM services are exploited throughout this study for both labeling process and the vehicle-classification system:

- *Map-matching*: Given a set of GPS points (e.g., a GPS trajectory), the map-matching is the service that snaps the GPS points to the OSM network in the most plausible way. Attaching GPS points to the most likely nodes in the road network alleviates location errors associated with the GPS technology. Erroneous GPS logs, that cannot be matched successfully, are considered as outliers and discarded. When map-matching is completed, a variety of fine-grained information between every two consecutive matched points are provided including travel distance, travel time, number of turn movements, number of intersections, etc.
- *Nearest*: The nearest service snaps a GPS coordinate to the nearest location in the traffic network. This service is particularly useful when the accurate location of a GPS log, rather than a GPS trajectory, matters. The name of street, to that the GPS log is snapped, is the most useful information provided by the nearest service.
- *Route*: The route service finds the fastest route between two GPS coordinates. Although the map-matching service can provide the similar information, the route service is a more optimized way when the goal is to compute travel time and distance for only one origin-destination pair.

4. Labeling GPS trajectories

This section develops an efficient scheme for labeling a large-scale GPS trajectory dataset based on the FHWA classification scheme. To the best of our knowledge, no GPS dataset has yet been labeled according to a fine-grained vehicle-class system. Such dataset could be exploited in a variety of trajectory mining and mobility studies besides the vehicle-classification problem. For example, knowing vehicle classes for all the recorded trips would enable derivation of trip tables for different vehicle classes (e.g., separate for passenger and commercial vehicles) and estimation of not only aggregate link-based volumes but their vehicle compositions.

Clearly, manual labeling of a large-scale dataset would be expensive and time-consuming task, which calls for a programmatic labeling approach. To this end, vehicle class information obtained from VWS records is deployed as an auxiliary dataset for labeling GPS trajectories. First, let us begin with the definition of a vehicle's GPS trajectory, which is used throughout this study:

Definition 1 (*Vehicle's GPS trajectory***).** A vehicle's GPS trajectory T is defined as a sequence of time-stamped GPS points $p \in T$, $T = [p_1, ..., p_N]$. Each GPS point p is a tuple of latitude, longitude, and time, $p = [lat, lon, t]$, which identifies the geographic location of point p at time t.

The key idea behind our proposed labeling process is to compute the crossing time-window of a vehicle's GPS trajectory from a VWS and then identify its class among those vehicles that crossed the station at the computed time-window. Thus, our proposed labeling strategy consists of the following five major steps that should be taken in sequence:

1. **Retrieving initial GPS trajectories.** GPS trajectories that have not potentially crossed one of the three stations are filtered out from the whole GPS dataset. This eliminates the need of involving a large volume of GPS trajectories that obviously have not crossed any of VWS and in turn there is no chance to be labeled. Temporarily removing these GPS trajectories from our labeling pipeline significantly reduces the computation time in the next steps.
2. **Applying filtering criteria.** Every GPS trajectory, obtained from the first step, is re-examined through several filtering rounds to guarantee that the GPS trajectory has truly crossed one of the three VWS. The GPS trajectories that have crossed a VWS and their corresponding VWS are determined and used in the next step.
3. **Predicting crossing time-window.** For every identified trajectory in the previous step, the time-window that the GPS trajectory has crossed the VWS is predicted.
4. **Labeling GPS trajectories.** The class information of all vehicles that have crossed the VWS during the predicted crossing time-window are retrieved. The GPS trajectory is labeled only if all crossing vehicles have the same class.
5. **Augmentation of labeled data.** Trajectories with the same device ID have been operated with the same vehicle. Hence, after labeling a portion of trajectories retrieved from the first step, trajectories with the same device ID as the trajectories labeled in the previous step are fetched from the database containing all 20 million GPS traces and labeled accordingly.

The above five steps are explained in the following five sections.

4.1. Step 1: Retrieving initial GPS trajectories

Using the OSM API, only trajectories that include a GPS point within a two mile radius from the 3 VWS are queried for the subsequent analysis. This reduces the number of trajectories from the initial 20 million to about 300,000 trajectories. The rationale behind this quick step is to dramatically expedite the processing time in steps 2–4, as the most computationally-intensive steps in our labeling pipeline.

4.2. Step 2: Applying filtering criteria

The retrieved trajectories from step 1 have not necessarily passed one of the three stations due to several reasons. For example,

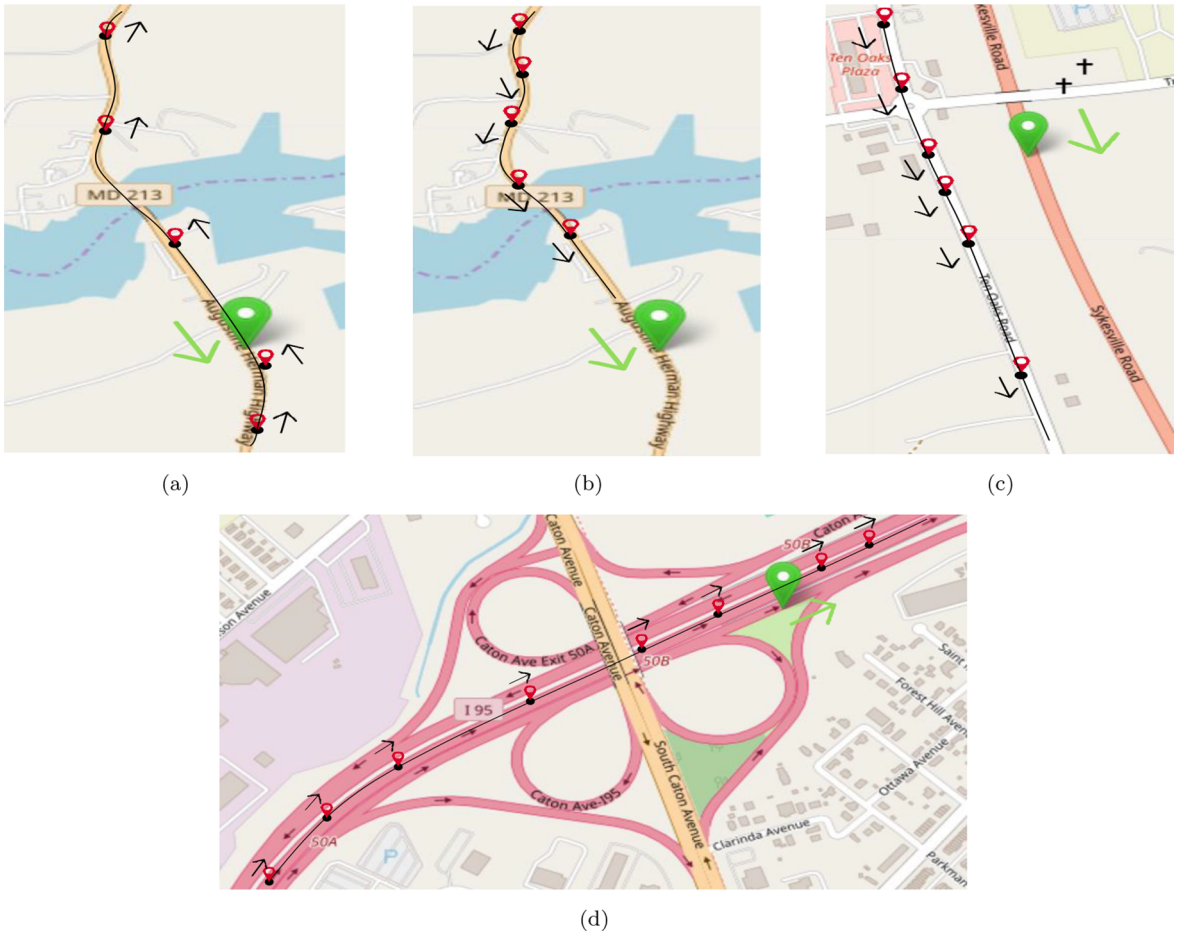


Fig. 3. Examples of GPS trajectories that have failed to pass one of the filtering criteria. The green and red markers indicate the VWS and GPS point locations, respectively. The green and black arrows indicate the road direction associated with the VWS and GPS trajectory, respectively. (a) GPS trajectory is moving in a reverse direction with respect to the VWS. (b) GPS trajectory is moving along a parallel road with respect to the VWS. (c) GPS trajectory approaches to the VWS, yet not crossing. (d) GPS trajectory is moving along an alternative route, rather than the VWS road. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

both directions of a roadway in OSM are often represented by one 'way' component, in which 'way' is one of the basic elements in OSM that are used to represent linear features such as roads. Therefore, a trajectory found in the first step might have a reverse direction with respect to the VWS, as shown in Fig. 3(a). Also, some of the retrieved trajectories are only close to a VWS without passing the station. Accordingly, we implement several strong filtering criteria to ensure a GPS trajectory has crossed the station. The possibility of a GPS trajectory crossing a station is examined one-by-one for all three stations. Since stations are far apart, we assume that a GPS trajectory can cross only one station within a few hours. Thus, if a GPS trajectory passes all criteria for the first station, the remaining stations will not be examined for that particular trajectory. Finally, our filtering criteria in this step have been designed to simultaneously obtain the GPS points of a trajectory that are adjacent to the crossed VWS, which is an essential information for the next step (i.e., predicting and matching the crossing time-window). This is another way to improve the overall efficiency of our labeling scheme. Before proceeding to the filtering criteria, the following notation is described for an easier understanding of the filtering approach.

Definition 2 (*Before/after GPS point lists*). Let a VWS be represented by S. The before/after GPS point lists of the trajectory T with respect to the station S, denoted as BL_{TS}/AL_{TS}, is defined as all GPS points in T in which their timestamp attribute are less/greater than the approximate time that the GPS trajectory T might have passed the station S. This crossing time is denoted as C_{TS}. Note that the before/after lists are defined based on the *time*, not the location. BL_{TS}/AL_{TS} are mathematically defined as follows:

$$BL_{TS} = \{p[t] \leqslant C_{TS} | p \in T \text{ and the station is } S\}$$
$$AL_{TS} = \{p[t] \geqslant C_{TS} | p \in T \text{ and the station is } S\}$$

Every GPS trajectory T is looped over the three VWS and, for every VWS, the following filtering criteria are examined in sequence. If T fails to pass a criterion, T is discarded for labeling and the remaining criteria are not examined. Thus, the GPS trajectory T crosses

a VWS S only if it passes all the filtering criteria.

1. The GPS trajectory T needs to have at least one GPS point located before the station S in order to cross the station. For example, since the station 1 is installed on the southbound, the GPS trajectory T is crossing the station 1 only if a GPS point with the latitude greater than the latitude of S is found in T. If any GPS point found, then the closest GPS point to the S is considered as the before-GPS-point. Note that, since the trip direction of a vehicle might change several times along the trip path, not all GPS points geographically located before the station are considered in the before list BL_{TS}. The GPS points that are recorded in a time order before the before-GPS-point form BL_{TS}.
2. At least one GPS point needs to exist after the before-GPS-point and located after the station S in the GPS trip T; otherwise, T has not crossed the station S. If exists, this GPS point is considered as the after-GPS-point. An example of a GPS trajectory that does not have an after-GPS-point (i.e., has not crossed the station) is depicted in Fig. 3(b).
3. The distance of before-GPS-point and after-GPS-point to the station S must be less than a specified threshold; otherwise, T has not crossed the station S. For example, it might pass a street parallel to the station S, as shown in Fig. 3(c). The threshold value in our labeling process is set to 2 miles. Note that if a GPS trajectory is passing through a parallel street yet its distance to the station is less than the specified threshold, the GPS trajectory is detected and discarded by the next filtering criterion.
4. Due to the existence of parallel and similar roadways around a station, there is still a chance that a GPS trajectory passes all the previous criteria without actually crossing the station. This situation particularly happens for the station 2, in which the road geometry around this station is very complex with several grade-separated road junctions. This complex interchange contains road sections that are parallel with the VWS lane. This causes to detect many GPS trajectories that have passed the previous criteria while they have not crossed the station 2, as shown in Fig. 3(d). Accordingly, more restricted criteria are implemented. In this case, we utilize the nearest service in OSRM to snap the GPS points of T, that are close to a VWS, into the nearest coordinates in the traffic network. The names of roads associated to each snapped GPS point are then retrieved. The GPS trajectory T has crossed the VWS only if it has traversed the road sections that end up to the VWS location.

It is worth noting that the third filtering criterion could be omitted; however, it helps reduce the number of trajectories that need to be map matched in the last step.

4.3. Step 3: Predicting time-window

We have thus far obtained some GPS trajectories that have crossed one of the three VWS. The before and after GPS point lists have also been identified for each GPS trajectory with respect to its corresponding station. Having the GPS trajectory T and the crossed station S, the following steps are performed to approximate the time-window that T has crossed S. Fig. 4 illustrates the process of predicting the time-window for a sample GPS trajectory.

1. **Computing the approximate crossing time**. Using the OSRM route service, the fastest route between the before-GPS-point (or alternatively the after-GPS-point, whichever is closer to the station S) and the station S is computed. The approximate crossing time of station S by the GPS trajectory T is then computed as follows:

$$C_{TS} = \begin{cases} p_b[t] + TR_{osrm}, & \text{if the ""before"" point is closer} \\ p_a[t] - TR_{orsm}, & \text{if the ""after"" point is closer} \end{cases} \quad (1)$$

where $p_b[t]$ and $p_a[t]$ are the timestamps for the before-GPS-point and after-GPS-point, respectively. TR_{orsm} is the travel time

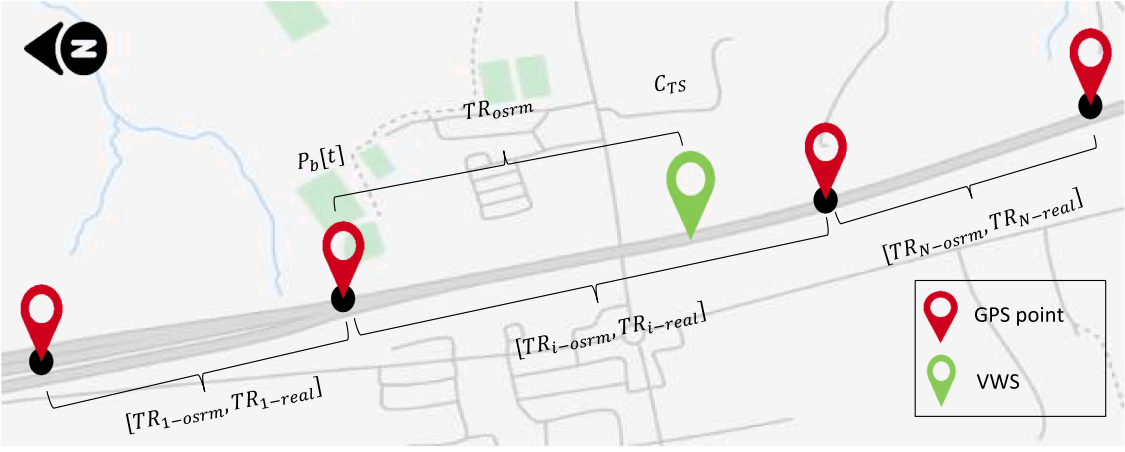


Fig. 4. Predicting the crossing time-window from a VWS for a GPS trajectory. Notations are referred inside the text.

duration between either $p_b[t]$ or $p_a[t]$ to the station S, retrieved from the OSRM route service.

2. **Computing the travel-time error.** Travel time retrieved from OSRM does not exactly match with the real travel time recorded by GPS-enabled devices. Thus, we need to obtain the travel-time error introduced by OSRM, which is then used as the time-window around C_{TS}. Two immediate before and after GPS points from the BL_{TS} and AL_{TS} are selected and concatenated to form a GPS sequence for the time-window prediction. After snapping the GPS sequence into the traffic network using the map-matching service, the travel times computed by OSRM between every two consecutive GPS points in the GPS sequence are obtained. Comparing OSRM travel times against real travel times computed from the timestamp attribute of GPS points results in the average error as follows:

$$MAE = \frac{\sum_{i=1}^{N} | TR_{i-real} - TR_{i-osrm} |}{N} \tag{2}$$

where TR_{i-real} and TR_{i-osrm} are the real and OSRM-based travel times between every two consecutive GPS points, respectively. MAE is the mean absolute error in approximating C_{TS}. In other words, MAE shows the error imposed by OSRM when computing C_{TS} according to Eq. (1). In computing the travel-time error, we particularly use very few points (i.e., two immediate GPS points from the before and after lists) around the VWS to consider traffic conditions and driver behavior only around that specific location, rather than over the whole GPS trajectory.

3. **Computing crossing time-window.** Finally a crossing time-window is constructed around the C_{TS} using the average travel-time error as follows:

$$C_{TS} - MAE \leqslant C_{TS} \leqslant C_{TS} + MAE \tag{3}$$

4.4. Step 4: Labeling GPS trajectories

After predicting the time window for C_{TS}, we assign the class label to the T according to the following steps:

1. As mentioned, the raw GPS data have categorized into three weight groups by the GPS provider. The vehicle records that their crossing time and gross weight attributes fall in the predicted time-window and the original weight category, respectively, are fetched from the VWS vehicle records associated with the station S. Note that no vehicle records might be retrieved from the VWS dataset as the predicted crossing time-window is commonly very short due to the low amount of error in Eq. (3). Low travel-time error is a good sign about our crossing-time prediction accuracy.
2. The GPS trajectory T is labeled to a vehicle class **only if** all vehicle records retrieved in the last step are the same. Although this leads to a smaller number of labeled GPS trajectories, such a conservative approach results in very accurate labeling. The GPS trajectory is discarded as an unlabeled trajectory if a unique class is not found among all retrieved vehicle records. The 'Before Augmentation' column in Table 2 shows the number of GPS trajectories that have been labeled using our labeling approach. Only 6,906 trajectories out of almost 300,000 initial GPS trajectories, retrieved from the first step of the labeling approach, have been labeled through our proposed strategy. The low number of labeled data at this stage, which mainly stems from applying strong constraints, is a good sign that our approach only labels GPS trajectories that have truly crossed a VWS. Also, for the trajectories with the same device ID, we were able to verify that they were labeled with the same FHWA class.

4.5. Step 5: Augmentation of labeled data

Finally, in order to augment labeled data, the GPS trajectories having the same device ID as the trajectories labeled in the previous step are extracted from the dataset containing 20 million trajectories and annotated with corresponding vehicle classes. As can be

Table 2
Number of labeled GPS trajectories per vehicle class obtained in various stages.

Vehicle Class	Before Augmentation	After Augmentation	After Processing (Final)
Class 2	3306	83,917	42,473
Class 3	46	8819	5961
Class 4	86	12,240	8128
Class 5	1372	302,960	100,000
Class 6	208	40,361	24,610
Class 7	356	43,717	29,178
Class 8	100	13,006	7968
Class 9	630	17,516	10,777
Class 10	16	105	85
Class 11	3	20	16
Class 12	1	689	284
Total	6906	523,350	229,480

seen in Table 2, extracting trajectories with the same device ID significantly augments the number of labeled trajectories, from almost 7000 to 500,000. Due to the augmentation opportunity, we deliberately implemented strong filtering criteria and labeling constraints so as to obtain labeled GPS data with the highest possible quality out of a programmatic approach. Moreover, accurate map matching of the labeled trajectories was also validated through numerous visualizations. It should be noted that the 'After Augmentation' column in Table 2 summarizes the output of our labeling process while the 'After Processing (Final)' column shows the number of labeled GPS data per class used for developing CNN-VC after applying the pre-processing steps described in Section 6.

4.6. Scalabitlty of the labeling approach

As described, our proposed labeling approach is a computationally-extensive task since labeling every GPS trajectory demands numerous computation steps. This makes the labeling of such a large-scale GPS data unscalable without using parallel computing systems. Apache Spark, a distributed computing engine that processes data in parallel, is deployed to make this procedure scalable by expediting the process 13× faster. For instance, the computation time for labeling one month of GPS data reduced from 8 h to 35 min by means of Apache Spark installed on a 16-core desktop.

5. The proposed model

The proposed CNN-VC is comprised of two main components: (1) New representation for GPS trajectories, and (2) Convolutional Neural Network (CNN). In this section, these two components and the way they interact with each other for identifying vehicles' classes from GPS trajectories are elaborated. The process for training our CNN-VC is also described in this section.

5.1. GPS trajectory representation

As defined, a GPS trajectory contains only a series of chronologically ordered points, with only coordinate and timestamp information for each point. Although this numerical ordered list can be blindly fed into traditional supervised algorithms (e.g., SVM), its structure is not adaptable for deep-learning models such as CNN and RNN (Dabiri and Heaslip, 2018b; Dabiri et al., 2019). Therefore, a novel representation of GPS trajectories suitable for deep learning algorithms is proposed, which accounts for both the vehicle's motion features and roadway characteristics.

Unlike other studies (Dabiri and Heaslip, 2018b; Simoncini et al., 2018), we leverage the OSRM APIs to obtain more accurate motion information and access to fine-grained roadway features. First, GPS coordinates in a raw GPS trajectory are snapped into the traffic network using the OSRM map-matching service. The matching process is accompanied by several advantages: (1) Every GPS point is snapped to the nearest node in the traffic network, which in turn alleviates errors that may be attributed to the GPS technology (e.g., satellite orbits, receiver clocks, and atmosphere effects) and the GPS data provider (i.e., rounding the latitude/longitude decimal numbers to the 4th decimal number), (2) The outlier GPS points are removed if they cannot be appropriately matched, (3) The sequence of matched points represents a real and plausible route in the network. The matched GPS trajectory can be imagined as a sequence of GPS legs, denoted by $lg \in T$, $T = [lg_1, ..., lg_N]$, where every lg is the route segment between two consecutive matched GPS points. Such a structure is analogous to the sentence structure in Natural Language Processing (NLP) tasks, in which a sentence is comprised of a word sequence. Our main objective in this section is to represent every lg with a numerical feature vector. The feature vector corresponding to a GPS lg posses two types of features: motion-related and road-related features.

The following motion features are computed for every lg in a GPS trajectory T:

- *Distance*, denoted by D: Distance is the accurate map-based length traveled by lg, which is extracted from the OSRM API. Unlike other studies (Dabiri and Heaslip, 2018b; Simoncini et al., 2018), we do not compute the geodesic distance (i.e., the direct distance between two locations) using well-known formulas such as Haversine and Vincenty. Instead, the map-based distance is computed using the OSRM API to take the road network configuration into account.
- *Duration*, denoted by Δt: Duration is the travel time for lg. The travel time is calculated based on the original timestamp information of GPS points, rather than the travel time computed by OSRM.
- *Speed*, denoted by S: Speed is the rate of change in distance for traveling the road segment of lg that shows how fast a driver is moving. Speed is calculated based on the map-based distance D and the real duration Δt.
- *Acceleration*, denoted by S: Acceleration is the rate of change in speed. The acceleration for the current lg is calculated based on the speed change of the current and subsequent legs. Accordingly, the last lg in T will be discarded.
- *Bearing rate*, denoted by BR. Bearing rate for lg indicates to what extent the vehicle's direction has changed between the start and the end points in lg. BR is the difference between bearings of the starting and end points. The bearings of start/end points are the clockwise angle from true north to the direction of the travel immediately after/before the start/end points of the lg. Likewise to distance D, the bearing values are retrieved from OSRM so as to obtain more accurate information. Note that bearing rate is an important motion feature that varies among vehicles (Zheng et al., 2008; Dabiri and Heaslip, 2018b). For example, a semi-trailer may not travel in a route segment that requires a sharp change in the heading direction.

The overall characteristics of the roadway that a vehicle uses for performing a trip is a discerning factor for developing a robust vehicle-classification system. It is apparent that various types of vehicles (e.g., passenger cars versus trucks) have different transport-infrastructure preferences for making their regular trips. Furthermore, vehicle regulations may ban some vehicles (e.g., semi-trailer

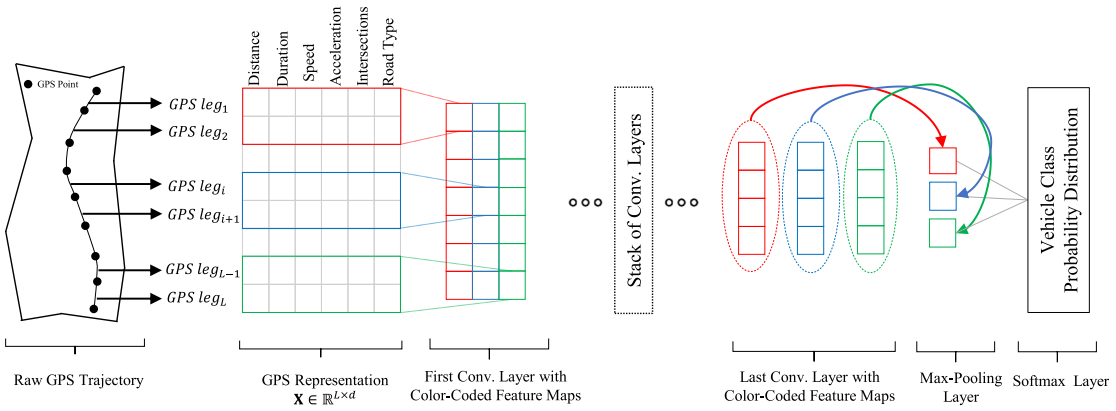


Fig. 5. The structure of the proposed GPS representation and CNN-VC model. The CNN-VC comprises a stack of convolutional layers followed by max-pooling and softmax layers. Each color code in the convolutional layer is corresponding to one filter and its feature map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

trucks) from traversing in dense urban environment. Accordingly, the following roadway characteristics are extracted from the OSRM API for every *lg* in a GPS trajectory *T*.

- *Number of intersections*, denoted by *IN*. Intersection is defined as any cross-path the vehicle passes when traveling along *lg*. *IN* is the number of all intersections in *lg*.
- *Number of maneuvers*, denoted by *M*. Maneuver is defined as any type of movement change that a vehicle needs to take when traversing *lg*. Turning right/left, traversing roundabout, taking a ramp to enter a highway, and merging onto a street are the examples of maneuver.
- *Road type*, denoted by *R*. Although a wide range of schemes are available for classifying roadways, a coarse-grained scheme is selected to divide roads into three groups: (1) Low-capacity roads including alley, street, avenue, boulevard, etc. (2) High-capacity roads including arterial, expressway, turnpike, parkway. (3) Limited-access grade-separated roads including freeway, highway, motorway, beltway, thruway, bypass, etc. The road type associated to each GPS *lg* can be extracted using the OSRM nearest service. Since *R* is represented as one-hot encoding, a coarse-grained road-type scheme avoids having a high-dimensional sparse feature vector.

The feature vector corresponding to every GPS *lg* is then created by concatenating the above-described motion and road features. Let $x_i \in \mathbb{R}^d$ represent the feature vector corresponding to the *i*-th leg in a trajectory, where *d* is the feature-vector dimension. Horizontally concatenating feature vectors of all legs in the GPS trajectory results in the new matrix representation, denoted by $\mathbf{X} \in \mathbb{R}^{L\times d}$, where *L* is the number of legs in the GPS trajectory. The structure of $\mathbf{X} \in \mathbb{R}^{L\times d}$ is shown in Fig. 5 and is used as the input matrix for deep-learning models. Since deep learning requires $\mathbf{X}$ for all samples in a training batch to have a fixed size, we either truncate long GPS trajectories or pad short ones with zero values into the fixed size *L*. Our observation indicates that setting *L* to a high percentile value (i.e., a value between 75 and 85%) of the number of legs in all GPS trajectories improves the overall model performance as long as trajectories can be represented by more motion and roadway features.

5.2. Convolutional neural network

CNN is a sophisticated network that is capable of capturing local correlations in spatial structures. In each convolutional layer of CNN, local correlations between adjacent input values are perceived by convolving a filter across the whole surface of an input volume. Considering $\mathbf{X}$ as a spatial representation, CNNs can obtain the correlation between consecutive GPS legs by convolving their corresponding feature vectors. The output of the convolution operation performed by each filter is called a feature map. Concatenating feature maps corresponding to multiple filters results in a new volume, which is an abstract representation for the GPS trajectory. The subsequent convolution layer will extract a more abstract representation of the output volume from the previous layer.

Fig. 5 depicts our proposed architecture for identifying the vehicle class associated with a GPS trajectory. As can be seen in Fig. 5, a vehicle's class is detected by passing its GPS trajectory $\mathbf{X}$ through multiple sets of convolutionsal layers that are stacked together. The convolution operation in each layer involves a filter $F \in \mathbb{R}^{n\times d}$, where *n* is the number of consecutive GPS legs in the GPS trajectory and *d* is the filter width equal to the leg's feature-vector dimension. It is very important to set the width of the filter equal to the feature-vector dimension since the principal goal is to extract spatial correlation between consecutive legs. Sliding the filter *F* across the matrix $\mathbf{X}$ produces a feature map $\mathbf{h} \in \mathbb{R}^{(L-n+1)\times 1}$, where each element h_i of the feature map is computed as

$$h_i = f\,(F \circ X_{i:i+n-1} + b), \tag{4}$$

where $\circ$ is the dot product between the parameters of filter *F* and the entries of submatrix $\mathbf{X}_{i:i+n-1}$, *b* is a bias term, and *f* is a non-

linear activation function. Rectified Linear Units (ReLU) is utilized as the non-linear activation function f throughout this paper. $\mathbf{X}_{i:i+n-1}$ refers to concatenation of n consecutive GPS legs at position i in the GPS trajectory. Using K filters of the same size as F, K feature maps are generated. Vertical concatenation of the created feature maps results in a new GPS representation matrix $\mathbf{X}_{new} \in \mathbb{R}^{(L-n+1)\times K}$, formally defined as

$$\mathbf{X}_{new} = [\mathbf{h}_1, \cdots, \mathbf{h}_{L-n+1}]. \tag{5}$$

The i-th row of the matrix $\mathbf{X}_{new}$ is an abstract feature vector for the n consecutive legs in the GPS trajectory. $\mathbf{X}_{new} \in \mathbb{R}^{(L-n+1)\times K}$ is used as the input volume for the next convolutional layer, where the same convolutional operation is applied. Our network can comprise several sets of convolutional layers, where every set has two convolutional layers with the same filter size and number of filters (i.e., n and k). However, the filter size and the number of filters may vary among layer sets. Finally, a max-pooling layer is applied to each feature map of the last convolutional layer. The rationale behind applying the max-pooling layer is to take one feature with the highest value (probably as the most important one) from each feature map. Such a max-pooling operation generates a feature vector that summarizes the high-level, abstract representation of a GPS trajectory. As the last step, the final vector representation is directly passed into the softmax function to perform the classification task by generating a probability distribution over vehicle classes for each GPS trajectory T, denoted by $P = \{p_1, \cdots, p_C\}$, where C is the number of vehicle classes. It should be noted that no fully-connected layers are used between the convolutional layers and the softmax layer (because their use consistently reduced performance of the model).

5.2.1. Training and regularization

Our proposed architecture is trained by minimizing the categorical cross-entropy as the loss function, which is formulated for every GPS trajectory T as

$$\mathscr{L}(\theta) = -\sum_{i=1}^{H} y_{T_i} \log(p_{T_i}), \tag{6}$$

where θ are all learnable parameters in the model. $y_{T_i} \in \mathbf{Y}$ is a binary indicator which is equal to 1 if the class i is the true vehicle class for the GPS trajectory T and 0, otherwise. $\mathbf{Y}$ is the true label for T, represented as one-hot encoding. p_{T_i} is predicted probability that the GPS trajectory T has been performed by the vehicle class i. The cross-entropy loss is averaged across the training batch in each iteration. The Adam optimizer is used to update model parameters in the back-propagation process. Adam is a well-suited optimization technique for problems with large dataset and parameters that has recently seen broader adoption for deep learning applications (Kingma and Ba, 2014). We use Adam's default settings as provided in the aforementioned paper: $learningrate = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The network weights are initialized by following the scheme proposed in (Glorot and Bengio, 2010).

The vehicle-classification problem often suffers from an imbalanced distribution among vehicle classes, where the number of samples belonging to each class does not constitute an equal portion of the training dataset. Training an imbalanced dataset in the same way as a balanced one and without implementing appropriate strategies results in a poor classifier that is not robust against non-dominant classes. Therefore, the following strategies are implemented in our training process in order to achieve a robust classifier:

- *Random over-sampling.* The number of GPS trajectories in all minority classes increases to the number of GPS trajectories in the majority class by randomly replicating them.
- *Balancing training batch.* Equal proportion of GPS trajectories from each class forms the training batch in each training iteration.
- *Cost-sensitive learning.* Higher weights are assigned to minority classes in the cross-entropy loss, Eq. (6), where weights are determined according to their class proportion in the dataset. This strategy penalizes misclassifications of the minority classes more heavily than the majority classes.

It is worth noting that the random over-sampling and balancing the training batch significantly improves our model performance while the cost-sensitive learning has a low impact.

Dropout and early-stopping are two types of regularization that are applied to our architecture. The dropout layer is added before the last layer (i.e., the softmax layer) with setting the dropout ratio to 0.5. In the early-stopping methods, the training is stopped if the performance metric does not improve on the validation set after two consecutive epochs. The model with the highest validation score is restored for applying on the test set.

6. Experimental results

In this section, first, several variants of the labeled GPS data are created. Afterward, the performance of our proposed model will be evaluated on the created datasest by comparing against several classical and deep-learning models. Robust classification metrics are used to assess the prediction quality of our proposed model. The quality of our proposed GPS representation is also investigated. Finally, the overall quality of the proposed model for predicting different types of vehicle categorization is discussed.

6.1. Datasets definitions and preparations

The large-scale GPS trajectories, that are labeled according to the proposed strategy in Section 3, is exploited for model evaluation. Before using the GPS data for training purposes, the following pre-processing steps are applied to each GPS trajectory so as to prepare them for generating the proposed GPS representation, introduced in Section 5, and remove erroneous GPS points caused by error sources in the GPS technology (e.g., satellite or receiver clocks).

- GPS points whose speed and/or acceleration exceeds a realistic value for vehicles moving in the US road networks are identified and discarded. The maximum speed and acceleration are set to 54 m/s and 10 m/s^2, respectively.
- After removing the GPS points with unrealistic speed and acceleration, the GPS trajectories containing less than 4 points are disregarded. Similarly, GPS trajectories shorter than 600 meters or 10 min were disregarded as well.
- The maximum length (i.e., the number of GPS legs) is set to 70, which is equivalent to the 80 percentile of the number of GPS legs in all GPS trajectories.
- After converting raw GPS trajectories into the proposed GPS representation, except for the road type binary features, all other types of features are standardized by subtracting the mean value and dividing by the standard deviation.

The specific filtering criteria were chosen to remove erroneous data points and short trips, but without significantly reducing the size of dataset. The number of labeled GPS trajectories per each vehicle class after applying the above-mentioned processing steps has been shown in the last column of Table 2. The processed GPS data are used for building our proposed model and baselines.

After pre-processing GPS trajectories, several variants of the original labeled dataset are created by re-grouping vehicle classes so as to examine and validate the proposed model. The vehicle-class definition is the only difference among the following datasets.

- **2&3** This dataset contains GPS trajectories only from FHWA class 2 and class 3. The proposed model needs to discriminate passenger cars (e.g., sedans, coupes, and station wagons) from all other two-Axle, four-tire single unit vehicles (e.g., campers, motor homes, and ambulances). As mentioned, the vehicles with class 2 and 3 of the FHWA system are categorized into one group (i.e., vehicles with gross weight less than 14,000 lbs) according to the GPS provider classification system. Hence, this type of problem is particularly useful for subdividing GPS trajectories with the gross weight under 14,000 lbs into the class 2 and class 3 of the FHWA system.
- **light&heavy** This dataset categorizes all GPS trajectories into two classes: (1) light-duty, which contains vehicles from class 2 and class 3, and (2) heavy-duty, which contains vehicles from all the other classes. This definition was first introduced in (Sun and Ban, 2013) and also used in (Simoncini et al., 2018).
- **light&medium&heavy** This dataset categorizes all GPS trajectories into three classes: (1) light-duty, which contains vehicles from FHWA class 2, (2) medium-duty, which contains vehicles from FHWA classes 3–6, (3) heavy-duty, which contains vehicles from FHWA class 7 and above. This classification is based on Gross Vehicle Weight Rating (GVWR) system, reported in the US Department of Energy Portal. Note that the corresponding FHWA vehicle classes in each category of GVWR is slightly different from weight category system defined by the GPS provider.

6.2. Baselines

Two types of baseline methods are considered for comparison: (1) classical-supervised methods on the top of hand-crafted features, (2) deep-learning models on the top of GPS representation proposed in Section 5.

6.2.1. Classical-supervised baselines

In the first group, widely used supervised algorithms in the literature of GPS-based vehicle classification and travel mode detection are deployed including K-Nearest Neighborhood (KNN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), as a representative of ensemble algorithms, and Multilayer Perceptron (MLP), as a regular fully-connected neural networks. Since the proposed GPS representation is not an appropriate input for these algorithms, a set of robust and efficient hand-designed features are extracted from each GPS trajectory. These features, which are introduced in the seminal study by Zheng et al. (2008) include the trajectory's length, mean speed, expectation of speed, variance of speed, top three speeds, top three accelerations, heading change rate, stop rate, and speed change rate. These are the most robust and acceptable hand-designed features in this domain that have been used by various studies. Note that since the MLP technique is fed by these hand-designed features, we group it as a classical baseline in spite of its deep nature. Also, it is worth recalling that the SVM was employed for GPS trajectory-based vehicle classification in Sun and Ban (2013), which leveraged acceleration-based features to differentiate between cars and trucks.

6.2.2. Deep-learning baselines

With the respect to the second group, several deep-learning models are developed for comparison. Major training blocks for building deep-learning baselines are CNN, RNN, and attention mechanism.

Recurrent Neural Network. An RNN is a chain-like neural network architecture that allows information to be passed from one step to another step. Thus, they constitute a series of repeating modules of neural networks, where each module consists of a hidden state that operates on sequential data like GPS trajectory representation $\mathbf{X}$ (Cho et al., 2014). For our proposed GPS representation, as shown in Fig. 6, each module looks at the current input x_i, which is the feature vector corresponding to the i-th GPS leg in the GPS

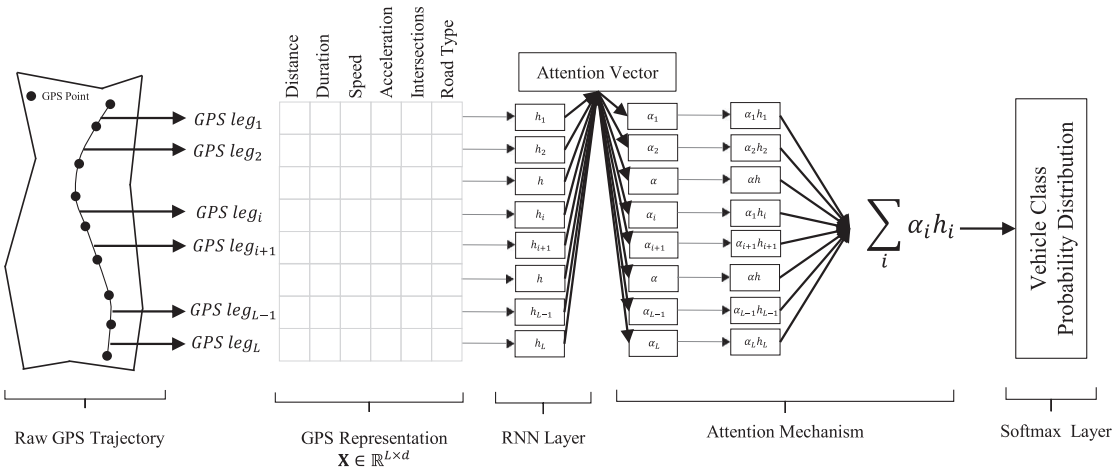


Fig. 6. The structure of an RNN layer with attention mechanism. Analogously, the attention mechanism can be applied to the convolutional layers.

trajectory, and the previous hidden state to update the hidden state at the current module through a set of non-linear functions. Long Short Term Memory network (LSTM) is the most widely used module in RNN layers that utilizes several non-linear functions to update the hidden state in each RNN module. The hidden state in the last layer is often used as the output of an RNN layer. Further details on the RNN and LSTM models can be found in (Cho et al., 2014; Hochreiter and Schmidhuber, 1997). Also, it is worth noting that Simoncini et al., 2018 also leverage a RNN-based model with a stack of LSTM layers on top of each others. Our RNN uses only one LSTM layer because adding more layers did not notably improve the results on our data (unlike in the case of the CNN-based architectures).

Attention mechanism. Attention mechanism, first introduced by (Bahdanau et al., 2014) is a new trend in deep-learning models that have received much interest in recent years. Attention helps a neural network to learn the importance of various regions of an input and focus more on certain input parts for performing the task-at-hand. It seeks to emulate the human's visual attention mechanism, which often pays more attention to only a small amount of information presented in visual scenes while doing their routine chores. Although a number of studies have assessed the efficacy of the attention mechanism in several applications including neural machine translation (Bahdanau et al., 2014), text classification (Du et al., 2017), text summarization (Rush et al., 2015), and caption generation (Xu et al., 2015), no study has examined the attention mechanism in the GPS trajectory mining tasks.

In particular for our application, the ultimate goal in the attention layer is to compute an importance weight for each abstract leg. Let $u \in \mathbb{R}^K$ be the attention vector, where K is the number of feature maps in the last CNN/RNN layer. Let $\mathbf{X}_{last}$ be the output of the last convolutional layer, in which $x_i \in \mathbb{R}^K$ is the feature vector corresponding to the i-th abstract GPS leg in $\mathbf{X}_{last}$. The importance of the abstract leg x_i is measured by computing its similarity with the attention vector u, and then get a normalized importance weight α_i through a softmax function. Elements in the attention vector u are trainable weights that are jointly learned with other network weights during the training process. Finally, the weighted representation of abstract legs are summed up to generate the attention vector representation, which is fed into a softmax layer for performing the classification task. Fig. 6 depicts how the attention mechanism works on the top of an RNN layer. Further details on the attention mechanism are available in the seminal studies by Bahdanau et al. (2014),Yang et al. (2016).

Using CNN, RNN, and attention mechanism, the following deep-learning baselines are used, where all trained on the top of our proposed GPS representation:

- **RNN**: A single RNN layer with LSTM unit is used. The hidden state in the last LSTM unit is directly fed into a softmax layer for the classification task.
- **RNN-Attention**: The model structure is the same as the **RNN** baseline while an attention mechanism is used between the RNN and softmax layers, as shown in Fig. 6.
- **Parallel-CNN**: Considering a convolutional layer followed by a max-pooling layer as one convolutional unit, several units are operated on the same GPS representation on a parallel way. The abstract feature-vector generated by each unit is then concatenated to create the final feature vector, which is fed into the softmax layer for the classification task. Note that convolutional and max-pooling operations are similar to our proposed model in Section 5. The primary difference between this baseline and our proposed architecture is that convolutional layers in our model are stacked on the top of each other and in turn operated sequentially rather than in a parallel way.

6.2.3. Performance evaluation

As mentioned, the vehicle classification represents an imbalanced classification problem. In this situation, a predictive model could simply achieve high accuracy by predicting all unseen records as the dominant class and considering samples in the minority class as noises. A robust and high-quality model needs to have a good prediction capability for both majority and minority classes. For

imbalanced classification problems, recall is much more reliable performance metric that implicitly indicates the model accuracy for retrieving true positives corresponding to each class. Since we are seeking a model that is able to retrieve a majority of true positives for all vehicle classes, average recall is selected as the main performance measure. Average recall is computed based on one specific probability threshold (e.g., 0.5). However, the threshold value for classification might be subjected to change depending on application objectives. Area Under the Receiver Operating Characteristic curve (AUROC) is another reliable metric that, like recall, measures the model accuracy for each class. For a binary-class problem, Receiver operating characteristic Curve is created by plotting true positive rate (i.e., recall for positive class) versus false positive rate (i.e., 1- recall for negative class), for various threshold settings. The following performance metrics are used for models evaluation:

- *AVE-Recall.* AVE-Recall is the average recall values of all vehicle classes in a dataset, which is defined as below: $AVE - Recall = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FN_c}$where TP_c and FN_c are the number of true positives and false negatives in the test set related to the class c. C is the total number of vehicle class in a dataset.
- *AUROC.* First, AUROC is computed for each class, and then the average AUROC for all classes is reported. For computing AUROC for each class in a multi-class problem, the subject class is considered as positive while the remaining assigned as negative.
- *Accuracy*. Accuracy is computed as the fraction of GPS trajectories in the test set that are correctly classified. While accuracy is not considered as the main metric in this study on account of imbalanced distribution issue, we report this metric because it is the most common classification metric.

In all our experiments, models are trained and tested using stratified 5-fold cross-validation and average values along with the standard deviations of the results on all 5-fold are reported. In other words, we create 5 train/test splits from the whole dataset. In each split, 4 folds are used for training while holding out the remaining onefold as the test set. Before obtaining the average results on the 5 test folds, models' hyperparameters are tuned. To this end, one fold out of the 4-fold training is used as the validation set for each split. The hyperparameter combination that on average achieves the highest performance on the validation sets of all splits is used for the final training and testing on 5 train/test splits. In this case, the test fold in each split plays no role in tuning hyperparameters. Next, using the obtained hyperparameters, models are trained on the folds and tested on one fold for each train/test split and the average results are reported. By doing this, every GPS trajectory in the whole dataset is used in the test set at least and at most once. Only for implementing the early-stop method in deep-learning models, 10% of the training data in each split is randomly selected as the validation set for the early-stopping procedure using the stratified random selection.

All data processing and models are implemented within Python programming language. The deep-learning architectures are learned in TensorFlow with the GPU support. Classic machine-learning algorithms are implemented using the scikit-learn library. The source codes related to all data processing and models utilized in this study are available at: https://github.com/sinadabiri/CATT-DL-Vehicle-Classifcation-GPS.

6.3. Hyperparameter settings

The number of convolutional sets, denoted as D, the filter size n, and the number of filters K in each layer are the primary hyperparameters in our proposed architecture. The grid search is a common approach for tuning hyperparameters that exhaustively considers all parameter combinations, yet it may not be an efficient way for deep-learning models with many hyperparameters. Instead, a manual search is conducted by designing a variety of model configurations and selecting the best parameter combination. We tune the hyperparameters based on the AVE-Recall metric by examining the model performance on only the 2&3 dataset. Table 3 shows the average of AVE-Recall along with the standard deviation over 5 folds for several CNN-VC configurations. A wide range of configurations has been designed by varying the number of convolutional sets, filter size, and number of filters in the range of $D \in [1, 2, 3]$, $n \in [1, 2, 3, 4]$, and $K \in [25, 50, 75, 100, 125]$, respectively. As can be seen from Table 3, setting the filter size and the number of filters in each convolutional layer to 2 and 100, respectively, achieves the optimum performance. Furthermore, increasing

Table 3
AVE-Recall for several CNN-VC configurations. The convolutional layer hyperparameters are denoted as conv(filter size)-(number of filters). Each conv set consists of two stacked convolutional layers with the same hyperparameters. NA: Not Available

#Config	Conv-set 1	Conv-set 2	Conv-set 3	Max-Pool	AVE-Recall
1	Conv2-25	NA	NA	Available	0.793(± 0.003)
2	Conv2-50	NA	NA	Available	0.805(± 0.007)
3	Conv2-75	NA	NA	Available	0.807(± 0.004)
4	**Conv2-100**	**NA**	**NA**	**Available**	**0.821**(± 0.005)
5	Conv2-125	NA	NA	Available	0.811(± 0.007)
6	Conv1-100	NA	NA	Available	0.809(± 0.008)
7	Conv3-100	NA	NA	Available	0.798(± 0.007)
8	Conv4-100	NA	NA	Available	0.777(± 0.012)
9	Conv2-100	Conv2-100	NA	Available	0.789(± 0.010)
10	Conv2-100	Conv2-100	Conv2-100	Available	0.784(± 0.016)
11	Conv2-50	Conv3-75	Conv4-100	Available	0.791(± 0.004)
12	Conv2-100	NA	NA	NA	0.776(± 0.007)

Table 4
Comparison of AVE-Recall values on created datasets using classical-supervised algorithms, deep-learning baselines, and our proposed CNN-VC model.

Model\Dataset	2&3	light&heavy	light&mid&heavy
KNN	0.566 (± 0.005)	0.568 (± 0.001)	0.392 (± 0.001)
SVM	0.623 (± 0.024)	0.582 (± 0.010)	0.421 (± 0.014)
DT	0.576 (± 0.004)	0.616 (± 0.001)	0.438 (± 0.001)
RF	0.555 (± 0.004)	0.637 (± 0.002)	0.467 (± 0.002)
MLP	0.649 (± 0.012)	0.624 (± 0.010)	0.437 (± 0.006)
RNN	0.766 (± 0.048)	0.732 (± 0.002)	0.615 (± 0.011)
RNN + Attention	0.776 (± 0.014)	0.732 (± 0.002)	0.619 (± 0.002)
CNN-Parallel	0.752 (± 0.008)	0.701 (± 0.004)	0.573 (± 0.003)
CNN-VC (ours with attention)	0.812 (± 0.011)	0.735 (± 0.003)	0.614 (± 0.002)
CNN-VC (ours)	**0.819** (± 0.005)	**0.741** (± 0.005)	**0.631** (± 0.004)

the number of convolutional sets, as used in configurations #9–10, does not boost the performance. Thus, a network with one convolutional set is selected to avoid the model complexity and reduce the training time. Even using multiple sets of convolutional layers with different filter sizes and number of filters (i.e., configuration #11) does not ameliorate the model performance. Finally, comparison between configurations #4 and #12 demonstrates that using a max-pooling operation on the top of the last convolutional layer significantly improves the prediction quality by increasing the AVE-Recall by more than 4%.

6.4. Comparison results

Tables 4,5 summarize the average results of the stratified 5-fold cross-validation along with the standard deviation for our CNN-VC model and baselines in terms of AVE-Recall, AUROC, and accuracy, respectively. In addition, the impact of the attention mechanism in our CNN-VC model is investigated by adding the attention layer, as shown in Fig. 6, on the top of the convolutional layers. Since the attention layer summarizes the high-level GPS representation into a feature vector for passing into the softmax layer, the max-pooling operation is taken out from the network. Like our CNN-VC model, the hyperparameters associated with every baseline is first tuned on the 2&3 datasets. Then, using the optimum combination of hyperparameters, every model is trained and tested on the three datasets: 2&3, light&heavy, and light&medium&heavy.

Considering AVE-Recall and AUROC as the most reliable metrics, Tables 4,5 clearly show the superiority of our CNN-VC model in comparison with both classical-supervised and deep-learning models for all three datasets. With respect to AVE-Recall, the CNN-VC model achieves on average 19% and 4% better performance compared to the classical-supervised and deep-learning models, respectively. Analogously, the CNN-VC surpasses the baselines by obtaining on average 22% and 3% higher AUROC in comparison with the classical-supervised and deep-learning models, respectively.

What is striking about Tables 4,5 is the significant superiority of deep-learning models compared to classical machine-learning techniques. One of the key differences between these two architectures is the structure of their input volume, the proposed GPS representation versus a set of hand-crafted features. Since the ultimate performance of a machine-learning algorithm is highly contingent on the quality of its input representation, the superiority of deep-learning models strongly demonstrates the effectiveness of our proposed GPS representation. Another interesting finding is that using the attention mechanism on the top of RNN and CNN blocks could not improve the model performance. This is a compelling reason to utilize the pooling operation instead of the attention mechanism in our proposed network. The comparison between RNN and CNN based models reveals that although the RNN architecture generates competitive results, our CNN-VC outperforms by improving the AVE-Recall and AUROC on average 3% and 2%, respectively. Finally, the clear superiority of CNN-VC compared to CNN-Parallel demonstrates that stacking convolutional layers in sequence ameliorates the overall performance of the model, against deploying convolutional layers in parallel. The explanation for

Table 5
Comparison of AUROC values on created datasets using classical-supervised algorithms, deep-learning baselines, and our proposed CNN-VC model.

Model\Dataset	2&3	light&heavy	light&mid&heavy
KNN	0.585 (± 0.005)	0.593 (± 0.001)	0.557 (± 0.001)
SVM	0.692 (± 0.022)	0.649(± 0.009)	0.603 (± 0.010)
DT	0.576 (± 0.004)	0.619 (± 0.001)	0.575 (± 0.001)
RF	0.737 (± 0.004)	0.687 (± 0.002)	0.634 (± 0.001)
MLP	0.685 (± 0.012)	0.655 (± 0.011)	0.591 (± 0.007)
RNN	0.873 (± 0.044)	0.830 (± 0.004)	0.797 (± 0.007)
RNN + Attention	0.884 (± 0.008)	0.831 (± 0.004)	0.801 (± 0.001)
CNN-Parallel	0.845 (± 0.006)	0.789 (± 0.004)	0.758 (± 0.002)
CNN-VC (ours with attention)	0.909 (± 0.003)	0.832 (± 0.001)	0.798 (± 0.002)
CNN-VC	**0.910** (± 0.003)	**0.840** (± 0.002)	**0.810** (± 0.002)

Table 6
Comparison of Accuracy values on created datasets using classical supervised algorithms, deep-learning baselines, and our proposed CNN-VC model.

Model\Dataset	2&3	light&heavy	light&mid&heavy
KNN	0.672 (± 0.002)	0.611 (± 0.001)	0.395 (± 0.001)
SVM	0.658 (± 0.019)	0.534 (± 0.027)	0.337 (± 0.055)
DT	0.814 (± 0.002)	0.737 (± 0.001)	0.511 (± 0.002)
RF	**0.869** (± 0.002)	**0.790** (± 0.001)	0.589 (± 0.002)
MLP	0.493 (± 0.053)	0.764 (± 0.058)	0.602 (± 0.023)
RNN	0.763 (± 0.165)	0.751 (± 0.022)	0.607 (± 0.048)
RNN + Attention	0.853 (± 0.022)	0.749 (± 0.021)	0.587 (± 0.025)
CNN-Parallel	0.720 (± 0.041)	0.670 (± 0.029)	0.562 (± 0.036)
CNN-VC + Attention	0.811 (± 0.043)	0.771 (± 0.011)	0.596 (± 0.019)
CNN-VC	0.856 (± 0.016)	0.771 (± 0.025)	**0.610** (± 0.004)

this outcome is as follows. The VCC-Parallel can be observed as an ensemble model where the weak learner is a one-layer CNN. On the other hand, our proposed model is comprised of a series of CNN layers (a deep CNN network), where the output of one layer is the input for the next layer. That being said, the proposed model takes advantage of the deep architecture and is able to generate more efficient abstract representation for the final classification, compared to VCC-parallel that averages over a set of one-layer CNN networks (shallow models).

Although the accuracy is not a reliable metric for the imbalanced problems, we do report the accuracy results in Table 6 to evaluate the overall performance of our model in terms of this widely used metric. As can be seen, except for RF, our CNN-VC model works better than all baselines for three datasets. While the model is forced to have close prediction quality for all vehicle classes, it achieves high and reasonable accuracy as well. The high accuracy value in contrast to low AVE-Recall and AUROC values for some baselines (e.g., RF and DT) is another proof that using only accuracy measure for imbalanced problems is not sufficient for model evaluation.

6.5. Effect of GPS representation

As noted, the comparison results have already demonstrated the effectiveness of our proposed GPS representation. In addition, the effect of motion features and roadway features are investigated by training our model on the top of different variants of GPS representation: (1) only motion features, (2) only roadway features, and (3) combination of motion and roadway features. Table 7 shows the average results for variants of GPS representation on the 2&3 dataset. From Table 7, it is apparent that the impact of motion features in improving the model quality is more significant. However, fusing motion features with roadway characteristics improves the prediction quality by more than 2% in terms of AVE-Recall, as the main metric value. Also, it should be noted that the GPS representation structure is very adaptable to be combined with information from other sources (e.g., environmental data) upon availability.

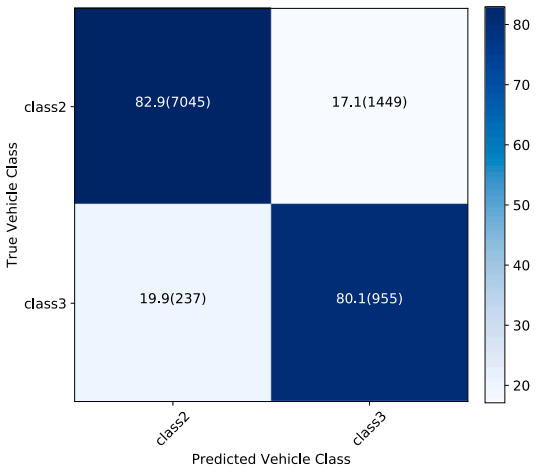
6.6. Model performance interpretation

An interesting analysis is to investigate the model prediction quality for every vehicle class. Confusion matrix, as shown in Fig. 7 for all types of datasets, allows us to visualize our CNN-VC quality at retrieving true positives for each vehicle class. Values in diagonal elements of Fig. 7 represent the percentage of GPS trajectories per each class in the test set that the CNN-VC model has correctly predicted (i.e., the recall value for each vehicle class). On the other hand, the off-diagonal values in each row shows the misclassification rate. The values in the parenthesis represent the number of true GPS trajectories that were assigned to various classes.

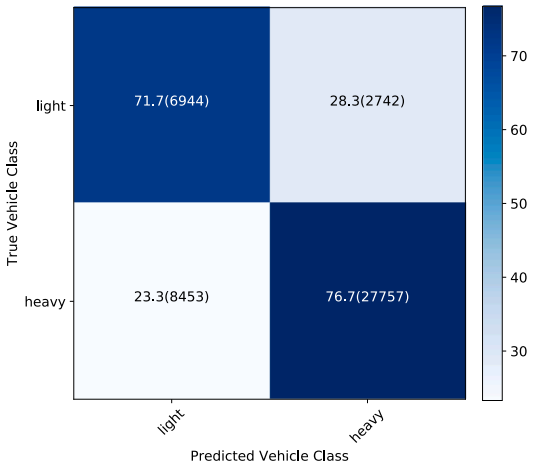
As shown in Fig. 7(a), the CNN-VC achieves almost the same performance for correctly predicting both the majority vehicle class (i.e., class 2) and the minority vehicle class (i.e., class 3), with recall values around 81%. It should be noted that the FHWA class 2 and class 3 include vehicles with similar operating functions and in turn similar moving patterns. For example, pickups and vans, which are categorized as the FHWA class 3, may be utilized as passenger cars, which is the main function of vehicles in the FHWA class 2. Accordingly, a high-quality performance for discriminating between the FHWA class 2 and 3 is not a trivial task, yet it has been

Table 7
Performance comparison of the CNN-VC model with variants of GPS representation on the 2&3 dataset

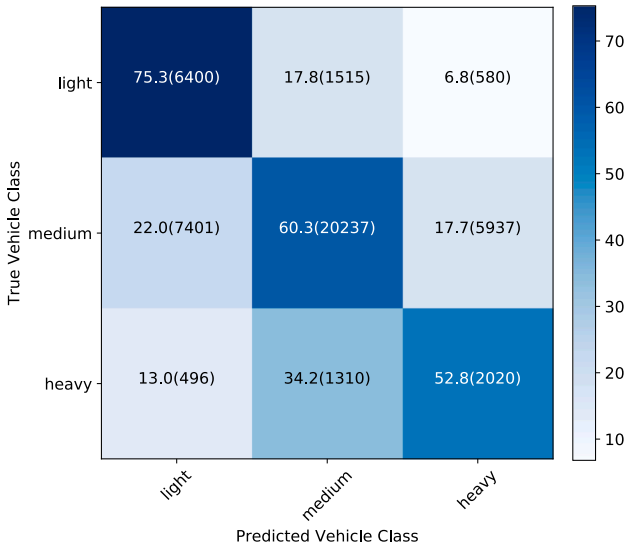
GPS Representation	AVE-Recall	AUROC	Accuracy
Only motion features	0.794 (± 0.011)	0.896 (± 0.004)	0.859 (± 0.029)
Only roadway features	0.690 (± 0.005)	0.771 (± 0.005)	0.667 (± 0.023)
Motion and roadway features	**0.819** (± 0.005)	**0.910** (± 0.003)	**0.856** (± 0.016)



(a) Confusion matrix for 2&3 dataset



(b) Confusion matrix for light&heavy dataset



(c) Confusion matrix for light&medium&heavy dataset

Fig. 7. Confusion matrices of the CNN-VC model for three datasets.

attained by the CNN-VC model. Since the raw GPS data collected by the GPS provider categorizes these two classes in the same weight group (i.e., less than 14,000 lbs), a practical benefit of the classifier on the 2&3 dataset is to distinguish the GPS trajectories of class 2 from class 3 across the entire dataset including 20 million GPS traces.

In a similar fashion, it can be observed from Fig. 7(b) that the proposed model has also a reasonable performance when vehicles are categorized into light-duty versus heavy-duty. The model's capability for detecting both the non-dominant class (i.e., light class in this dataset) and the dominant-class (i.e., heavy class in this dataset) are roughly similar, with the recall values around 74%. It should be emphasized that even the FHWA 13-category rule sets have some difficulty in distinguishing between vehicle categories when the information on the number, weight, and spacing of axles are available by means of traffic flow sensors (Hallenbeck et al., 2014). For example, a pickup truck with conventional two-tire real axle (FHWA class 3) cannot be distinguished from the similar truck yet with dual tries on each side of its rear axle (FHWA class 5) when they are empty. This problem is exacerbated when the information related to the vehicle is limited to only a sequence of GPS records, rather than vehicle shape and weight characteristics. Considering such issues, an accuracy around 74% for both light-duty and heavy-duty vehicle classes can be considered as an acceptable performance while all information is limited to the vehicle mobility pattern obtained from GPS records.

As expected and shown in Fig. 7(c), the performance quality is worsened by increasing the number of vehicles in the light& medium&heavy dataset. The CNN-VC achieves a good performance in distinguishing light (i.e., FHWA class 2 and 3) and medium (i.e., FHWA class 3–6) vehicles. However, the model encounters difficulty for discriminating the heavy vehicles from medium ones, where 34% of heavy-duty vehicles are misclassified as medium-duty. This low-quality performance can be attributed to low sampling rate of the GPS data and/or overall prediction ability of the employed machine learning models; however, it is worth noting that

distinguishing between medium and heavy vehicles is a non-trivial task even when information about vehicles' axles is available. For instance, a light truck pulling utility trailer, which is classified as the FHWA class 3 and medium-duty class, may have similar axle configuration to a truck pulling a heavy single-axle trailer, which is classified as the FHWA class 8 and heavy-duty class (Hallenbeck et al., 2014). Without access to the axle weight information, these two trucks cannot be differentiated from each other even using fixed-point traffic flow sensors. Furthermore, trucks size and weight configurations might vary among different States and manufacturers, depending on State laws and companies profit strategies although they are designed for exactly the same purposes. Considering the boundary vehicles types (e.g., the FHWA class 6 as the medium-duty class against the FHWA vehicle class 7 as the heavy-duty class), that have almost the same operation capabilities, is sufficient to intuitively understand the difficulty in distinguishing these boundary classes using only GPS information.

In summary, the GPS sensor is a reliable alternative to overcome the shortcomings of fixed-point sensors by reducing the installation and maintenance costs, removing the spatial coverage limitation, and opening the opportunity for online monitoring the traffic network by inferring the vehicle-class distribution. However, it is obvious that GPS trajectories can only represent the mobility patterns of moving vehicles around the road network. Lack of information on the axle and weight configuration, which cannot be extracted from GPS data, makes the GPS-based vehicle classification harder than alternative methods, in particular for fine-grained classification scheme with several vehicle categories. While the results in this study show that building a coarse-grained vehicle-classification scheme using GPS data is achievable, creating a robust model for the fine-grained classification scheme requires additional information. A potential solution for improving the model performance while preserving the advantages of the GPS data (e.g., wide-area spatial coverage) is to fuse other information such weather conditions, that manifests the road network with more details, with the GPS data. As described in Section 5, our proposed GPS representation has the flexibility for including more information about the vehicle and road network along the GPS path. Lastly, it is worth noting that the vehicle classification results reported in this paper are based on the GPS trajectory data with the average sampling frequency of about 4 points per minute. Higher sampling rate would allow the model to better capture any variations in vehicle motion characteristics (e.g., acceleration, speed) and would most likely improve the model accuracy.

6.7. Practical applications

The proposed method can be readily used to label all 20 million trajectories that Maryland State Highway Administration is using to support its analysis and decision making. Such enriching of the trajectory data would enable more comprehensive studies that require information about vehicle classes. Examples include, but are not limited to:

- Derivation of origin-destination tables for individual FHWA vehicle classes, which is especially useful for planning efforts where transportation analysts need to distinguish between travel patterns of passenger cars used mainly for commuting and commercial vehicles;
- Examining whether trajectories associated with heavy trucks are observed in downtown areas or along the routes with weight restrictions, which would indicate the need for additional enforcement in those areas in order to establish the desired level of safety;
- Analyzing whether trajectories attributed to large trucks are deviating from locations with weigh-in-motion systems, which would suggest the need for deploying mobile patrols in these regions to enforce weight limits;
- Estimation of FHWA-class-specific volumes along different road links in a transportation network, which would allow transportation analysts to more accurately measure performance as it would enable them to distinguish between vehicle-hour and truck-hour delays;
- More accurate estimation of emissions based on trajectory data, where individual tracks can now be attributed to different vehicle classes (e.g., passenger cars vs. trucks), which obviously makes a big difference in estimating transportation-related emissions.

7. Conclusion

Vehicle classification, as an essential step in a variety of ITS applications, has been addressed for decades using fixed-point traffic flow sensors. However, such conventional senors suffer from major shortcomings including high maintenance cost and low spatial coverage. To address these shortcomings, we aimed to leverage the spatiotemporal information of vehicles' trips retrieved through on board GPS-enabled devices, for classifying vehicles into various categories. First, we designed an efficient programmatic approach to label a large scale GPS data with the aid of an auxiliary resource, (i.e., VWS vehicle records). Using the labeled GPS data, we proposed a deep convolutional neural network (CNN-VC) for identifying the vehicle class from its GPS trajectory. A novel representation was designed to convert the GPS trajectory into an input matrix for deep learning. Each row in the new representation corresponds to motion-related and roadway-related features for the segment between two consecutive GPS points. Afterward, a stack of convolutional layers, followed by a max-pooling operation, were applied on the top of the proposed GPS representation so as to compute the abstract representation of the GPS trajectory for the classification task using the softmax operation. Our extensive experiments clearly demonstrated the superiority of the CNN-VC model in comparison with several state-of-the-art classical-supervised and deep-learning techniques.

The proposed deep learning approach is particularly convenient from the practical perspective because it eliminates the need for tedious feature engineering, while also providing superior performance thanks to the large number of parameters inherent to the proposed deep learning architecture. On the other hand, while the proposed framework provides a ubiquitous and cost-effective

alternative to using fixed-point sensors, a major drawback of the proposed GPS-based vehicle classification approach is that it is currently much less accurate than the classification done by the fixed-point sensors. This, however, may change as more and more high-frequency GPS trajectory data becomes available to train deep learning vehicle classification models.

Acknowledgement

The first two authors were supported by the Center for Advanced Transportatin Technology (CATT) at the University of Maryland. The authors are also grateful to Subrat Mahapatra and the Maryland State Highway Administration who provided the GPS trajectory data used in the paper. This support is gratefully acknowledged, but it implies no endorsement of the findings.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.trc.2020.102644.

References

Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
Capecci, S., Krupa, C., Systematics, C., 2009. Concept of operations for virtual weigh station. Technical report. United States. Federal Highway Administration.
Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
Coifman, B., Kim, S., 2009. Speed estimation and length based vehicle classification from freeway single-loop detectors. Transp. Res. Part C: Emerg. Technol. 17 (4), 349–364.
Dabiri, S., Heaslip, K., 2018a. Developing a twitter-based traffic event detection model using deep learning architectures. Expert Syst. Appl.
Dabiri, S., Heaslip, K., 2018b. Inferring transportation modes from GPS trajectories using a convolutional neural network. Transp. Res. Part C: Emerg. Technol. 86, 360–371.
Dabiri, S., Heaslip, K., 2018c. Transport-domain applications of widely used data sources in the smart transportation: A survey. arXiv preprint arXiv: 1803.10902.
Dabiri, S., Lu, C.-T., Heaslip, K., Reddy, C.K., 2019. Semi-supervised deep learning approach for transportation mode identification using gps trajectory data. IEEE Trans. Knowl. Data Eng.
Dong, Z., Wu, Y., Pei, M., Jia, Y., 2015. Vehicle type classification using a semisupervised convolutional neural network. IEEE Trans. Intell. Transp. Syst. 16 (4), 2247–2256.
Du, J., Gui, L., Xu, R., He, Y., 2017. A convolutional attention model for text classification. In: National CCF Conference on Natural Language Processing and Chinese Computing. Springer, pp. 183–195.
Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256.
Gupte, S., Masoud, O., Martin, R.F., Papanikolopoulos, N.P., 2002. Detection and classification of vehicles. IEEE Trans. Intell. Transp. Syst. 3 (1), 37–47.
Hallenbeck, M.E., Selezneva, O.I., Quinley, R., 2014. Verification, refinement, and applicability of long-term pavement performance vehicle classification rules. Technical report.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Kafai, M., Bhanu, B., 2012. Dynamic Bayesian networks for vehicle classification in video. IEEE Trans. Industr. Inf. 8 (1), 100–109.
Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization arXiv preprint arXiv: 1412.6980.
Marković, N., Sekuła, P., Vander Laan, Z., Andrienko, G., Andrienko, N., 2018. Applications of trajectory data from the perspective of a road transportation agency: Literature review and maryland case study. IEEE Trans. Intell. Transp. Syst. 99, 1–12.
Rush, A.M., Chopra, S., Weston, J., 2015. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv: 1509.00685.
Sekuła, P., Marković, N., Laan, Z.V., Sadabadi, K.F., 2018. Estimating historical hourly traffic volumes via machine learning and vehicle probe data: A Maryland case study. Transp. Res. Part C: Emerg. Technol. 97, 147–158.
Simoncini, M., Taccari, L., Sambo, F., Bravi, L., Salti, S., Lori, A., 2018. Vehicle classification from low-frequency GPS data with recurrent neural networks. Transp. Res. Part C: Emerg. Technol. 91, 176–191.
Sun, Z., Ban, X.J., 2013. Vehicle classification using GPS data. Transp. Res. Part C: Emerg. Technol. 37, 102–117.
Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057.
Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E., 2016. Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489.
Zheng, Y., 2015. 'Trajectory data mining: An overview', ACM Trans. Intell. Syst. Technol. 6 (3), 29:1–29:41.
Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y., 2008. Understanding mobility based on GPS data. In: Proceedings of the 10th International Conference on Ubiquitous Computing. ACM, pp. 312–321.