Driving Maneuver Anomaly Detection Based on Deep Auto-Encoder and Geographical Partitioning

MIAOMIAO LIU and KANG YANG, Dept. of Computer Science and Engineering, University of California, Merced, USA
YANJIE FU, Dept. of Computer Science, University of Central Florida, USA
DAPENG WU, Dept. of Computer Science, City University of Hong Kong, China
WAN DU, Dept. of Computer Science and Engineering, University of California, Merced, USA

This paper presents *GeoDMA*, which processes the GPS data from multiple vehicles to detect anomalous driving maneuvers, such as rapid acceleration, sudden braking, and rapid swerving. First, an unsupervised deep auto-encoder is designed to learn a set of unique features from the normal historical GPS data of all drivers. We consider the temporal dependency of the driving data for individual drivers and the spatial correlation among different drivers. Second, to incorporate the peer dependency of drivers in local regions, we develop a geographical partitioning algorithm to partition a city into several sub-regions to do the driving anomaly detection. Specifically, we extend the vehicle-vehicle dependency to road-road dependency and formulate the geographical partitioning problem into an optimization problem. The objective of the optimization problem is to maximize the dependency of roads within each sub-region and minimize the dependency of roads between any two different sub-regions. Finally, we train a specific driving anomaly detection model for each sub-region and perform in-situ updating of these models by incremental training. We implement *GeoDMA* in Pytorch and evaluate its performance using a large real-world GPS trajectories. The experiment results demonstrate that *GeoDMA* achieves up to 8.5% higher detection accuracy than the baseline methods.

CCS Concepts: • **Computing methodologies → Model development and analysis**; **Artificial intelligence**; • **Applied computing → Transportation;**

Additional Key Words and Phrases: Anomaly detection, deep auto-encoder, peer dependency, geographical partitioning

ACM Reference format:
Miaomiao Liu, Kang Yang, Yanjie Fu, Dapeng Wu, and Wan Du. 2023. Driving Maneuver Anomaly Detection Based on Deep Auto-Encoder and Geographical Partitioning. *ACM Trans. Sen. Netw.* 19, 2, Article 37 (April 2023), 22 pages.
https://doi.org/10.1145/3563217

This research was partially supported by the National Science Foundation (NSF) via the grant numbers: 2008837, 2007210, 1755946, I2040950 and 2006889. Some preliminary results have been published as a poster in "The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys 2020)" [1]. The work was done when Dr. Dapeng Oliver Wu was in the department of Electrical & Computer Engineering at the University of Florida.
Authors' addresses: M. Liu, K. Yang, and W. Du (corresponding author), Dept. of Computer Science and Engineering, University of California, Merced, Merced, CA, USA; emails: {mliu71, kyang73, wdu3}@ucmerced.edu; Y. Fu, Dept. of Computer Science, University of Central Florida, Florida, USA; email: yanjie.fu@ucf.edu; D. Wu, Dept. of Computer Science, City University of Hong Kong, Hong Kong, China; email: dapengwu@cityu.edu.hk.

1550-4859/2023/04-ART37 $15.00
https://doi.org/10.1145/3563217

1 INTRODUCTION

Although annual traffic fatalities have decreased from 42,702 in 2006 to 36,560 in 2018 in the United States, it is far from the "Toward Zero Deaths National Strategy" goal [2]. Most of the fatal crashes are the results of human error or negligence, such as speeding, distracted driving, or driving under the influence of drugs and alcohol [3].

Smartphone apps [4, 5] have been developed by companies like Google and Uber to detect anomalous driving maneuvers. They read instant driving sensing data (e.g., vehicle velocity and orientation) from smartphone sensors for driving anomaly detection. A recent work, pBEAM [6], applies conditional adversarial recurrent neural network to develop a personalized model for each driver. It detects anomalies from instant sensor readings and captures the temporal dependency of driving data. However, these existing solutions are focused on individual drivers, the peer dependency of drivers are not considered, i.e., the correlation of driving maneuvers across vehicles, which is normally caused by local road structures, traffic conditions, and driving habits. Peer dependency depicts the similarity of driving maneuvers across vehicles, which is an important factor that needs to be considered in driving anomaly detection. For example, if most of the drivers in proximity show similar driving maneuvers during the same period, they should not be related to anomalous drivings.

In this paper, we develop *GeoDMA*, which detects driving maneuver anomaly in real time by analyzing the instant GPS samples of drivers and taking the peer dependency of drivers into account. Driving maneuver anomalies, like aggressive driving, distracted driving, drowsy driving, and driving under the influence, can be analyzed from vehicle-based features including the pressure exerted on the brake, the fluctuation on vehicle speed, the angle of wheels, and the steering wheel movement [7–11]. We summarize the anomalous driving maneuvers that can be detected by *GeoDMA* in Table 1, including rapid acceleration, sudden braking, rapid swerving, frequent speed, and direction changing. To effectively detect anomalous driving maneuvers in real time, *GeoDMA* takes instant GPS samples as input, and processes them by a deep auto-encoder model for driving maneuver anomaly detection. The deep auto-encoder model is designed by considering both temporal dependency of each individual vehicle and peer dependency across different vehicles. In addition, to capture stronger vehicle-vehicle peer dependency, a geographical region partitioning algorithm is developed to divide a city into different sub-regions, in each of which we train an auto-encoder-based driving anomaly detector, further improving the detection accuracy.

Our deep auto-encoder model is composed of an encoder and a decoder. The encoder takes the driving state transition feature vector calculated from the instant GPS samples of vehicles as input. A driving state is a combination of a speed-related operation and a direction-related operation. The driving speed and direction of a vehicle can be calculated from its GPS samples. The encoder is designed as a fully-connect layer and a **Recurrent Neural Network (RNN)** layer, which transforms an original driving state transition feature into a representation feature in a lower-dimensional latent space. The decoder is a fully-connected neural network with two fully-connected layers, which recovers the lower-dimensional representation feature to a reconstruction feature. It takes the representation feature as input and outputs the reconstructed feature. The reconstructed feature has the same dimension as the input of the encoder (the original driving state transition feature). The temporal dependency can be captured by the RNN, and peer dependency is incorporated as a regularizer of the deep auto-encoder model. *GeoDMA* uses the reconstruction error between the original feature vector and the reconstructed feature vector of the auto-encoder for anomaly detection, since anomalous driving data cannot be represented and reconstructed well by a model that was trained only by normal driving data. With unsupervised learning, the

Table 1. Anomalous Driving Maneuvers that can be Detected by *GeoDMA*

Anomalous Driving Maneuvers	**Corresponding Anomaly in Driving Feature**
Rapid Acceleration	The transition duration from a driving state to an acceleration-related driving state is small
Sudden Braking	The transition duration from a driving state to a deceleration-related driving state is small
Rapid Swerving	The transition duration from a driving state to a bearing-related driving state is small
Frequent Acceleration or Deceleration	The transition probability from a driving state to an acceleration-related or deceleration-related state is high
Frequent Turns	The transition probability from a driving state to a bearing-related driving state is high

labeling of anomalous driving data is not needed when training the model. The deep auto-encoder is trained to learn what the normal driving maneuvers look like. Any driving maneuvers that do not follow the distribution of normal driving maneuvers will be considered as anomalous driving maneuvers.

To maximize the effectiveness of spatial peer dependency in driving maneuver anomaly detection, *GeoDMA* incorporates the First Law of Geography [12], *"Everything is related to everything else, but near things are more related than distant things"*. [13] also proposes that locality preservation of spatial data leads to better service. We consider the locality of peer dependency in our system. Since the road layouts and traffic conditions vary across a city, the driving behaviors exist in common patterns in small regions with similar contextual features (i.e., traffic conditions and road structures). The peer dependency in a local area is stronger than that in the entire city. We develop a geographical partitioning scheme to perform driving anomaly detection in a geo-distributed way. Specifically, we divide a city into several sub-regions. The data of the vehicles from the same sub-region will be collected together to develop the anomaly detection model for this sub-region. The detection accuracy of the model in each sub-region is expected to achieve higher accuracy than the centralized model that is trained by the data from the whole city.

The objective of geographical region partitioning is to maximize the spatial dependency within the same sub-region and minimize the spatial dependency among different sub-regions. The road segments that have stronger dependency are supposed to be divided into the same sub-region and road segments that have weaker dependency are supposed to be divided into different sub-regions. We first construct the road network of a city as an undirected and weighted graph. We treat the road segments of the city as vertices of the graph. There is an edge between two vertices if the roads they represent are geographically connected. We then formulate the geographical region partitioning problem as an optimization problem and solve the optimization problem by **Normalized Cut (NCut)** algorithm. In NCut algorithm, we extend the vehicle-vehicle spatial dependency to calculate the road-road spatial dependency. The weight of an edge is specifically designed to depict the spatial dependency between two vertices (road segments), which is measured by the similarity of the driving maneuvers that have happened on these road segments and similarity of the representation features of those driving maneuvers learned from the auto-encoder.

After obtaining the partitioning result, we train a specific anomaly detection model for each sub-region and perform in-situ updating of these models by incremental training to further improve the detection accuracy. We implement *GeoDMA* in Pytorch platform [14] and conduct extensive experiments using the T-Drive dataset [15, 16], which contains vehicle GPS trajectories collected from a big city. Results from extensive experiments demonstrate that *GeoDMA* achieves up to 8.5% and 2.2% higher accuracy than the single-user approach and the centralized approach.

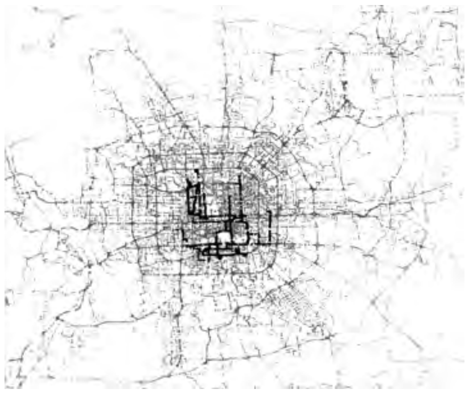


Fig. 1. The trajectories of vehicles that receive low detection accuracy with single-user models.

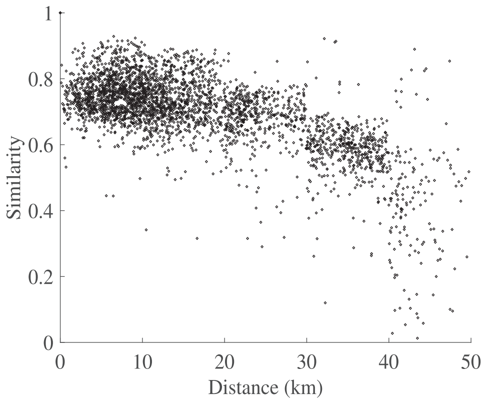


Fig. 2. The relationship between vehicles' distance and vehicles' driving maneuver similarity.

2 MOTIVATION

In this section, we process the T-Drive dataset and demonstrate the motivation of two key components in *GeoDMA*. The detailed descriptions of the T-Drive dataset is in Section 5.2.

Peer dependency of multiple drivers. Peer dependency is considered in driving behavior analysis [17], which processes the historical GPS trajectories of vehicles to evaluate their driving history. We believe peer dependency is also useful for online driving anomaly detection. For example, if a driver exhibits stop-and-go (deceleration and acceleration) frequently, it is difficult to validate whether this kind of behavior is caused by the driver's anomalous driving maneuver, the road structure, or the local traffic conditions. If a driver is observed to consistently show different driving maneuvers from the other drivers around him/her, it's reasonable to suspect that the driving maneuvers of that driver are abnormal. Here we do experiments to investigate the importance of peer dependency in anomaly detection. We use auto-encoder (details in Section 3.2) to detect anomalous driving maneuvers and test the performance of single-user models. We use the driving data of each driver to train an auto-encoder detector independently and analyze the detection result of each single-user model. We find that the accuracy of some models can be lower than 0.7. Figure 1 highlights the trajectories of those vehicles that receive inaccurate driving anomaly detection at the same time slot. They are geographically close and driving in the downtown of the city. The driving maneuvers of these vehicles can be the reference to one another when developing the driving anomaly detection models. Based on this observation, we propose to develop a centralized model by incorporating peer dependency in the single-user model to improve its accuracy.

Locality of peer dependency. The centralized model is able to consider the vehicle-vehicle peer dependency across a city. However, the peer dependency should be considered in a fine-grained way. If a vehicle is driving far away from the other vehicle, they will have less peer dependency than some vehicles that are close to one another because it is hard for a centralized model developed for a big city to take all local features of the city into consideration. Figure 2 depicts the relationship between the similarity of driving features and their distances between any two drivers. The driving feature similarity is measured by Equation (5), which will be introduced in details in Section 3.2.2. We use 504 drivers to do the experiments and show the relationship of the driving feature similarities and the distances of all driver pairs. As shown in Figure 2, as the distance increases, the similarity of driving features decreases. The experiment result confirms with the First Law of Geography, which motivates us to partition the road segments in proximity into the same region for driving maneuver anomaly detection.

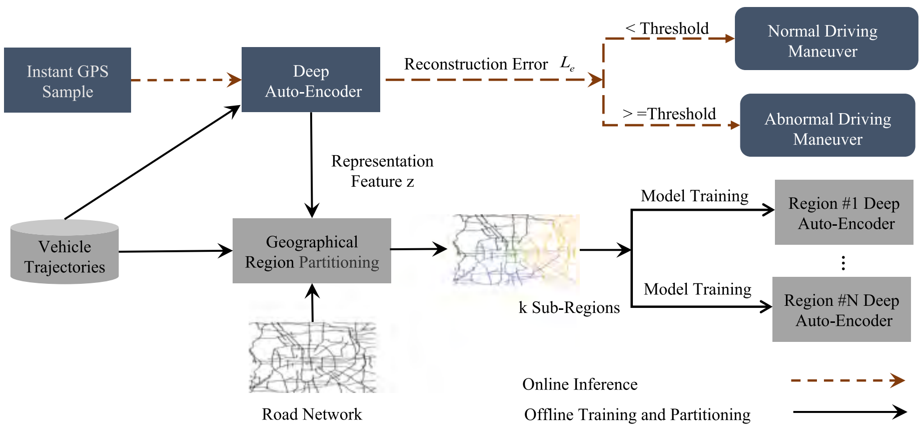


Fig. 3. System architecture of *GeoDMA*.

3 SYSTEM DESIGN OF *GEODMA*

In this section, we introduce the design of GeoDMA, including a brief overview, deep auto-encoder for driving anomaly detection, geographical partitioning, sub-region model, and in-situ updating.

3.1 System Overview of *GeoDMA*

Figure 3 depicts the architecture of *GeoDMA*, which is mainly composed of two modules, i.e., a deep auto-encoder for driving anomaly detection and a geographical region partitioning algorithm.

The driving data from each driver will be firstly converted into the driving state transition vectors (Section 3.2.1). They are the input of our deep auto-encoder network. In the deep auto-encoder, we consider both individual vehicle temporal dependency and the vehicle-vehicle spatial dependency (Section 3.2.2). The deep auto-encoder model is trained by normal driving maneuvers; it cannot reconstruct the anomalous driving data accurately. We use the reconstruction error of the model to perform driving maneuver anomaly detection (Section 3.2.3).

To further improve detection accuracy, we partition a city into multiple sub-regions by a geographical region partitioning algorithm. We formulate the region partitioning problem as a graph partitioning problem (Section 3.3.1). As shown in Figure 3, the inputs of region partitioning include the road network of a city, the vehicle trajectories and the representation features generated by the centralized auto-encoder. We develop an NCut algorithm to solve the optimization problem (Section 3.3.2). Finally, we train a specific anomaly detection model for each sub-region and update these models in-situ by incremental training (Section 3.4).

3.2 Auto-Encoder for Driving Maneuver Anomaly Detection

Figure 4 presents the inference workflow of our driving anomaly detection algorithm. During each time window, the vehicle GPS data is fed into the vector calculator to generate the original driving feature vector x (state transition vector). Then the auto-encoder verifies x is normal or anomalous. Next, we will introduce the vector calculator, the deep auto-encoder, and the driving anomaly detection in detail.

3.2.1 Vector Calculator. In the context of driving maneuver anomaly detection, the driving state of a vehicle can be described by its moving speed and direction. The speed-related driving operations include acceleration, deceleration, and driving in a constant speed. The direction-related operations include turning left, turning right, and going straight. A speed-related operation

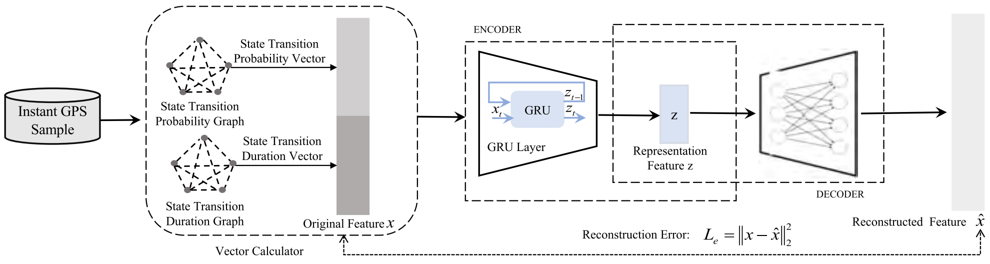

Instant GPS Sample
State Transition Probability Vector
State Transition Probability Graph
State Transition Duration Vector
State Transition Duration Graph
Original Feature x
Vector Calculator
ENCODER
GRU
x_t
z_{t-1}
z_t
GRU Layer
z
Representation Feature z
DECODER
Reconstructed Feature $\hat{x}$
Reconstruction Error: $L_e = \|x - \hat{x}\|_2^2$

Fig. 4. The workflow of deep auto-encoder. The vehicle GPS data is preprocessed by vector calculator. It generates the original feature vector x. The vector x is represented by combining the state transition probability vector and the state transition duration vector. This feature x is mapped to a representation feature z by the encoder. Then z is reconstructed as $\hat{x}$ by the decoder. Finally, we leverage the reconstruction error L_e as the criterion to detect anomaly.

and a direction-related operation constitute a driving state. There can be nine driving states: S_1(acceleration, turning left), S_2(acceleration, turning right), S_3(acceleration, going straight), S_4(deceleration, turning left), S_5(deceleration, turning right), S_6(deceleration, going straight), S_7(constant speed, turning left), S_8(constant speed, turning right) and S_9(constant speed, going straight). The details about speed and direction calculation from GPS data are introduced by Equations (14) and (15).

When a vehicle is moving, the driving state of a vehicle usually changes over time. A sequence of driving states of a vehicle during a time window can be obtained. For instance, [S_2, S_2, S_5, ..., S_8]. Such a time-varying sequence can be summarized as a state transition graph. As shown in the vector calculator of Figure 4, the nodes of the graph are driving states, there are nine nodes in the graph and they are fully connected, we use five nodes in the figure for simplicity. The weights of edges depict the relations between any two states [18]. In this paper, the weights are constructed from two aspects, the state transition probability and state transition duration between two driving states [17]. The value of the weights are normalized between 0 and 1. The driving state transition probability is the frequency a driver drives from a state to another state during a time window. The driving state transition duration is how long a driver takes to respond from one driving state to another. For example, if the transition from <constant speed, going straight> to <acceleration, going straight> is small, it indicates this is a rapid acceleration anomaly.

During each time window, we can get a state transition probability graph and a a state transition duration graph. The feature of the driving maneuvers are captured from these two graphs. The adjacent matrices of these two graphs can be flattened into two vectors, the state transition probability vector and the state transition duration vector. As shown in Figure 4, the combination of these two vectors is the original feature x, which is the output of the vector calculator and the input of the deep auto-encoder. Since there are nine states, there can be 81 state transitions (including self to self) for both probability graph and duration graph, the dimension of x is 162. The state transition vectors that differ significantly from the normal state transition vectors will be detected as anomalies.

3.2.2 Deep Auto-Encoder. As shown in Figure 4, the encoder projects the original feature vector x into a lower-dimensional feature, *i.e.*, representation feature z. The decoder reconstructs z to $\hat{x}$. The more similar the x and the $\hat{x}$, the more accurate the model is. In this work, the encoder is a fully-connected layer and a **Gated Recurrent Unit (GRU)** [19] layer. GRU can capture the temporal dependency of the input features. The decoder is a fully-connected neural network with two fully-connected layers. The auto-encoder is updated as:

$$\mathbf{z}_i^t = GRU\left(\mathbf{z}_i^{t-1}, \mathbf{x}_i^t\right) \tag{1}$$

$$\hat{\mathbf{x}}_i^t = \mathbf{D}_\theta\left(\mathbf{z}_i^t\right) \tag{2}$$

where the $\mathbf{D}_\theta$ in Equation (2) represents the decoder network and θ is the weight of it.

The standard method to train an auto-encoder is to minimize the reconstruction error L_e between x_i^t and $\hat{\mathbf{x}}_i^t$.

$$L_e = \|\mathbf{x}_i^t - \hat{\mathbf{x}}_i^t\|_2^2 \tag{3}$$

If the trajectories of two drivers exhibit similar original driving features x, the representation features z of them that are generated from the encoder should be similar. Therefore, the vehicle-vehicle peer dependency can be integrated into the loss function. The peer dependency can be modeled as a regularizer L_r.

$$L_r = s_{i,j}^t \cdot \|\mathbf{z}_i^t - \mathbf{z}_j^t\|_2^2 \tag{4}$$

$$s_{i,j}^t = cos\left(\mathbf{x}_i^t, \mathbf{x}_j^t\right) \tag{5}$$

where $s_{i,j}^t$ in Equation (4) is the cosine similarity between the original driving feature vectors x_i^t and x_j^t that are generated from driver d_i and driver d_j during time window t, which is calculated by Equation (5). The z_i^t and z_j^t are the representation features of x_i^t and x_j^t during time window t. To incorporate the peer dependency, the loss function L is defined as:

$$L = \arg\min \sum_{t\in\mathcal{T}}\left(\sum_{d_i\in\mathcal{D}} L_e + \alpha \cdot \sum_{d_j\in\mathcal{D}, d_i\neq d_j} L_r\right) \tag{6}$$

where α is the hyperparameter to control the regularizer L_r. The d_i and d_j are any two drivers in the driver set D. The regularizer L_r is used to guide the training of the auto-encoder.

3.2.3 Driving Maneuver Anomaly Detection. Instead of training a multiclass classifier using labeled normal and anomalous data by supervised learning, we leverage deep auto-encoder to train a one class classifier. One class classifier is an unsupervised learning process and has been widely used in the state of the arts for anomaly detection [20, 21]. We use one class classifier because it is not easy to collect, predefine and manually label various types of anomalies to train a supervised muilticlass classifier to detect different types of anomalies. More importantly, if any new anomalies occur in the future, the anomalies need to be redefined and the classifier needs to be retained. By using unsupervised one class classifier, we address the above problems. During the training phase, the deep auto-encoder model is trained only on the normal driving data. During the inference phase, the anomalies can be detected as long as the model memorized the feature of normal data well. That is because anomalies cannot be reconstructed accurately by the model trained only on normal data [22–24]. As shown in Figure 4, we leverage the reconstruction error L_e as the criterion to detect the anomalies.

During the inference phase, the model is expected to produce higher reconstruction error for the driving anomalies than the normal ones. The detection principle is shown as follows:

$$x = \begin{cases} Normal\ Input, & if\ L_e(x, \hat{x}) < \tau \\ Anomalies, & Otherwise \end{cases} \tag{7}$$

where τ is the largest reconstruction error of normal data. It is predefined for online inference [20, 21]. During inference, given a sample input x, if the reconstruction error L_e between input vector x and its reconstructed vector $\hat{x}$ is lower than the threshold, it indicates that the well-trained model can reconstruct the input accurately. The input x will be considered as a normal driving maneuver. Otherwise, the input x will be detected as an anomalous driving maneuver.

3.3 Geographical Partitioning for Anomaly Detection

In this work, we propose to develop the driving anomaly detection model in a geo-distributed way. We first partition a city into several sub-regions to get stronger spatial peer dependency, and then develop a specific model for each sub-region. The challenge here is how to partition the region to the full extent of spatial peer dependency. To tackle the challenge, we develop a geographical region partition algorithm. The design objective is to improve the accuracy of driving maneuver anomaly detection in each sub-region. We extend the vehicle-vehicle dependency to the road-road dependency. Three principles are considered in the partition algorithm. (1) The road segments with stronger correlations should be partitioned into the same sub-region. (2) The road segments with weaker correlations should be partitioned into different sub-regions. (3) The road segments that are partitioned into the same sub-region should be geographically connected.

3.3.1 Problem Formulation. We represent the road network of a city as a graph $G = (V, E)$, where $V = (v_1, v_2, \ldots, v_n)$ represents all road segments, and $E = \{e(v_m, v_n)\}_{v_m, v_n \in V}$ describes the relationships among these road segments. If two road segments are geographically connected, there will be an edge connecting these two vertices. The weight of an edge, $w(v_m, v_n)$ of $e(v_m, v_n)$, qualifies the correlation between two vertices v_m and v_n. We extend the vehicle-vehicle dependency to road-road dependency to get the correlation between two road segments. The similarity of driving maneuvers that happened on two road segments is used to approximate the correlation between them. We formulate this similarity by Equation (8).

$$w(v_m, v_n) = \sum_{t \in \mathcal{T}} \frac{1}{K_m^t K_n^t} \sum_{d_i \in \mathcal{D}_m, d_j \in \mathcal{D}_n} \frac{s_{d_i, d_j}^t}{\|\mathbf{z}_{d_i}^t - \mathbf{z}_{d_j}^t\|_2^2} \tag{8}$$

where $\mathcal{T}$ represents all time windows, v_m and v_n denote two road segments, K_m^t and K_n^t represent the number of drivers on these two road segments at time window t, $\mathcal{D}_m$ and $\mathcal{D}_n$ are two sets of drivers on these two road segments at time t, s_{d_i,d_j}^t is the similarity of original driving maneuvers between driver d_i and driver d_j at time t. s_{d_i,d_j}^t is acquired by Equation (5), its value range is between 0 to 1. The higher the similarity, the higher the s_{d_i,d_j}^t. The d_i is the driver on road segment v_m and the d_j is the driver on road segment v_n. The $z_{d_i}^t$ and $z_{d_j}^t$ are the representation features of the driving maneuvers that are generated from the encoder for driver d_i and d_j at time t. If the original driving maneuvers (x) of two drivers are similar, the s_{d_i,d_j}^t is higher, and the $\|z_{d_i}^t - z_{d_j}^t\|_2^2$ is lower, the ratio of them is higher. At each time window, we calculate this ratio from all of driver pairs on the two road segments v_m and v_n and get the average ratio of all pairs. We use multiple time windows to do the experiments and use the average value from these time windows as the correlation of these two road segments. The underlying rationale for the Equation (8) is that two road segments have strong correlation when the original driving maneuvers (x) that happened on these two road segments are similar and their representation features (z) are also similar. A higher $w(v_m, v_n)$ means a higher correlation between the road segments v_m and v_n.

Based on Equation (8), we can obtain the weight of each edge in G. The objective of graph partitioning is to divide vertices $v_x \in V$ into k subsets V_1, V_2, …, V_k. Each subset V_i corresponds to a set of the road segments that are partitioned into the same sub-region. According to the above partition principles, the spatial correlation within one subset V_i should be strong, and the spatial correlation among different sub-regions (V_i and V_k) should be weak. We formulate it as an optimization problem. Its objective is shown as follows:

$$max \ \frac{1}{M}\sum_{v_m, v_n \in V_i} w(v_m, v_n) - \frac{1}{N}\sum_{\substack{v_m \in V_i, v_l \in V_k \\ v_m \neq v_n \neq v_l, V_i \neq V_k}} w(v_m, v_l), \tag{9}$$

where V_i and V_k are two vertex subsets in the graph $G = (V, E)$, vertex v_m and v_n belong to the same subset V_i, v_l is a vertex in subset V_k. $w(v_m, v_n)$ is the weight of edge $e(v_m, v_n)$, $w(v_m, v_l)$ is the weight of edge $e(v_m, v_l)$. The edge $e(v_m, v_l)$ connects subset V_i and V_k. M is the total number of edges within the same subset V_i and N is the total number of edges that connect subset V_i and V_k. The first part of the objective is the average weight of edges within a subset V_i, the second part is the average weight of edges that connect two subsets V_i and V_k. The objective is to maximize the average weight difference within a subset and among different subsets.

3.3.2 *Normalized Cut Algorithm.*

Graph partitioning is proved to be an NP-hard problem [25–27]. Partitioning the graph into $k = 2$ parts is already an NP-hard problem. Solutions are generally derived using heuristics algorithms. Spectral clustering is one of the most common graph partitioning methods by grouping graph vertices. MinCut, RatioCut, and Ncut are three common spectral clustering algorithms. MinCut tries to find the minimum weight of edges that connect different sub-graphs. But in many cases, MinCut simply separates a vertex in the graph from the rest of the vertices, which is not what we want. A reasonable solution should consider that there are as many vertices as possible in each sub-graph. RatioCut and Ncut were proposed to solve the limitation of MinCut. In RatioCut, the size of a sub-graph is measured by the number of vertices in this sub-graph, while in Ncut the size of a sub-graph is measured by the weights of edges in this sub-graph. Since maximizing the number of vertices in each sub-graph does not necessarily mean the total weights of this sub-graph is large, cutting the graph based on the weights is more in line with our goal. Ncut is a normalized spectral clustering algorithm, while RatioCut is an unnormalized one. In general, Ncut is better than RatioCut [28]. Hence, we leverage Ncut [29] to solve our optimization problem. The normalized cut criterion measures the total similarity within each subset of the graph and the total dissimilarity among different subsets of the graph.

It is important to construct a similarity function among different vertices when using Ncut. The vertices which are defined as "similar" by the similarity function should be closely related in the application. In our scenario, the correlation between any two vertices does not only depend on how similar these two vertices are. It first depends on the physical geographical connection. Because vertices are the road segments. The two road segments that are supposed to be divided into the same sub-region should be at first connected with each other. We use the correlation function (Equation (8)) as the similarity function of vertices. Suppose the vertex set V and edge set E in the graph $G = (V, E)$ can be partitioned into two subsets V_i and V_k, $V_i \cup V_k = V$, $V_i \cap V_k = \varnothing$ by removing the edges that connect these two subsets. The degree of similarity between these two parts is defined as $cut(V_i, V_k)$. It is the total weights of the edges that connect these two subsets.

$$cut(V_i, V_k) = \sum_{v_i \in V_i, v_k \in V_k} w(v_m, v_l) \tag{10}$$

where v_m is the vertex in subset V_i, v_l is the vertex in subset V_k. Finding the minimum cuts is the objective of graph partitioning. It is designed by considering both the total disassociation (Ncut) between two subsets and the total association (Nassoc) within each subset. The Ncut and Nassoc are defined as follows:

$$Ncut(V_i, V_k) = \frac{cut(V_i, V_k)}{cut(V_i, V)} + \frac{cut(V_i, V_k)}{cut(V_k, V)} \tag{11}$$

$$Nassoc(V_i, V_k) = \frac{cut(V_i, V_i)}{cut(V_i, V)} + \frac{cut(V_k, V_k)}{cut(V_k, V)} \tag{12}$$

Importantly, the Ncut and Nassoc are closely related to each other, it can be inferred that:

$$\begin{aligned} Ncut(V_i, V_k) &= \frac{cut(V_i, V_k)}{cut(V_i, V)} + \frac{cut(V_i, V_k)}{cut(V_k, V)} \\ &= \frac{cut(V_i, V) - cut(V_i, V_i)}{cut(V_i, V)} + \frac{cut(V_k, V) - cut(V_k, V_k)}{cut(V_i, V)} \\ &= 2 - \frac{cut(V_i, V_i)}{cut(V_i, V)} + \frac{cut(V_k, V_k)}{cut(V_k, V)} \\ &= 2 - Nassoc(V_i, V_k) \end{aligned} \tag{13}$$

So minimizing the dissociation (Ncut) between the sub-graphs and maximizing the association (Nassoc) within each sub-graph can be reached simultaneously.

The solution can be approximated as follows:

(1) Build a weighted graph $G = (V, E, w)$ from the road network, compute the weight of each edge according to Equation (8).
(2) Get the diagonal matrix D of the graph. The diagonal value is $d_i = \sum_j w(i, j)$. Then get the symmetrical matrix W of the graph with $W(i, j) = w_{ij}$.
(3) Solve the equivalent eigenvalue system $(D - W)y = \lambda Dy$ for eigenvectors, obtain the second smallest eigenvalue.
(4) Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
(5) Recursively partition the subgraphs until the desired number of clusters is reached.

3.4 Sub-Region Model and In-Situ Updating

The subsets partitioned by our Ncut are the sub-regions in the road network. We then train a specific detection model for each sub-region by using the vehicle trajectories that are just collected from this sub-region and perform in-situ updating of these models in each sub-region periodically.

3.4.1 Model Training and Inference for Sub-Regions. The models for the sub-regions need to be trained by using the driving data collected from each sub-region. First, we map the data in the dataset into different sub-regions. To implement this, we add the <RoadID> attribute to every sample of the GPS data. By grouping the GPS data with the same RoadID value, the data will be mapped into different sub-regions. Besides, we want to find the drivers with driving behaviors in a set of consecutive time windows to train the model, so we filter the drivers that do not meet the requirements. The number of sub-regions k to be partitioned is a parameter of Ncut algorithm. We try different values of k and find the one with the best performance. Since we divide the whole city into several sub-regions, the driving maneuvers that happened within each sub-region become the peer dependencies when training the models. The models trained using the driving data within each sub-region learn the distribution of normal driving maneuver in corresponding sub-regions. The workflow of models for sub-regions is the same as the base model we introduced in 3.2. After training, we use the threshold of reconstruction error of these models to classify the driving maneuvers. The sub-region models are trained with different subsets of the data, the reconstruction error threshold for these models are not exactly the same.

3.4.2 In-Situ Updating of the Sub-Region Models. To improve the trustworthiness of the geo-distributed driving maneuver anomaly detection system, we will perform incremental learning to update all the driving maneuver detection models in-situ. The models will be updated offline after each server collects a reasonable amount of data or simply after a fixed time period T. In our current implementation, T is set to 25 time windows, corresponding to 54 minutes. We update the models twice in our implementation. Each server uses the data which detects as the normal driving data by itself to update its driving maneuver detection model.

4 IMPLEMENTATION

Deep Learning Models. *GeoDMA* is implemented on Pytorch framework [14]. The encoder contains one fully-connected layer and one GRU layer. The fully-connected layer has 40 neurons. Each GRU cell has 20 neurons. The decoder is consisted of two fully-connected layers, which has 40 and 162 neurons in the first and second layer. The activation function is sigmoid function. The performance comparison of different representation sizes can be found in Section 5.5. Besides, we use SGD optimizer to optimize our models, as it has shown good performance in modeling sequential data.

The learning rate is set to be 0.1. The batch size is 128. The number of epochs is 500. The α in Equation (6) is 0.01. The models are trained on Alianware Aurora R7, which contains one Intel Core i5 8400 CPU (6-core) and one NVIDIA GeForce GTX 1070 GPU (8GB memory). Sklearn python package is used to implement the evaluation metrics.

Graph Partitioning. It is non-trivial to construct a graph based on the road network for a big city. To implement it, the road network information is needed. We get the *GEOJSON* file of road network from OpenStreetMap [30], which is a kind of file format for representing geodata. After parsing the file, there are 43 types of road, including 138,155 road segments. We visualize the road information and filter the road segments that do not contain any driving data, such as pedestrian way, cycle way, and so on. There are many road segments that only contain several pair of coordinates, which indicates these roads are short. We first combine these short road segments into longer ones and then construct the graph using the longer road segments. We assign a *RoadID* to each road and find out the connected roads by comparing coordinates. After getting the geographical connection of each road, the graph $G = (V, E)$ can be constructed and the weight of each edge can be computed. Finally, we implement Ncut by using Sklearn package to partition the graph.

5 EVALUATION

We conduct a variety of experiments to evaluate the performance of GeoDMA, including overall performance, performance under different scenarios, effectiveness of region partitioning, in-situ model updating, and execution efficiency.

5.1 Experiment Setting

We compare the performance of *GeoDMA* with two baselines over a large vehicle GPS trajectories dataset. We use 80% of the data to do training, and 20% to do testing [31–33]. We set the number of sub-regions to 4, the size of the representation feature to 20. We use these settings by default in the following experiments. In Section 5.5, we also conduct experiments to set these parameters.

5.1.1 *Performance Metrics.* We use F1 Score [34] and AUC to do the performance evaluation.

5.1.2 *Baselines.* We compare the performance of *GeoDMA* with two baselines.

- **Centralized Model.** The centralized model is a simple version of *GeoDMA*. It uses all training data to train a general model, without geographical partitioning. The general model is based on an auto-encoder that takes both temporal dependency and peer dependency into account.
- **Single-User Model.** We implement a similar version of the latest personalized driving anomaly system pBEAM [6] based on deep auto-encoder architecture. The single-user model

is trained using the driving data that is collected from each individual driver without considering peer dependency and geographical partitioning.

5.2 Dataset and Data Processing

We conduct data-driven evaluation on a real-world dataset, T-Drive [15, 16]. The data format in the dataset is (Driver ID d, Date Time t, Longitude λ, Latitude φ). We first use a map matching algorithm [35] to map the GPS locations to corresponding locations on the road network. We then interpolate the missing data after map matching. We next pre-process the data, calculate the driving speed and direction of the vehicles and simulate anomalous data for evaluation.

Map Matching. The T-Drive dataset includes the GPS trajectories collected from 10,357 drivers. In the dataset, its sampling rate of trajectories is uneven, the average sampling interval is about 177 seconds with a distance of about 623 meters. When the sampling interval is large, there are only a few data samples during their corresponding time windows. That is not good to derive the driving features. Therefore, it may not fully show the driving maneuvers of drivers during these time windows. In this paper, we leverage a map matching algorithm to solve this drawback. The map matching algorithm maps the GPS trajectories to the road network [35, 36]. After map matching, the data can be augmented and is much more dense than the original data. The average sampling rate is about 9 seconds. Practically, we apply a standard map matching algorithm based on Hidden Markov Model [35] to map the driving trajectories in T-Drive dataset onto the road network. Map matching algorithm is implemented on Java platform.

Data Interpolation. After map matching, we find some pieces of data do not have the timestamp attribute. To solve this problem, We use an interpolation method to fill up the lost timestamps. It assumes that all the drivers are driving with a constant speed between two consecutive timestamps. This assumption is in line with normal driving maneuvers. For example, point A, B, C, D, E are five samples on a route, only A and E have time values. We first calculate the average speed v_{AE} of the vehicle from A to E, it is $v_{AE} = \frac{d_{AE}}{t_E - t_A}$, where d_{AE} is the driving distance from A to E. We assume the driver drives in a constant speed, so $t_B = t_A + \frac{d_{AB}}{v_{AE}}$. Thus, the time information of all the samples can be got. And all the drivers are driving in a constant speed most of the time. Therefore, we assume all the driving maneuvers derived from the dataset after map matching and data interpolation are normal.

Data Pre-processing. To make the data in the dataset closer to the normal driving data, we filter the bad data from the dataset. After filtering, there are 504 drivers left in the dataset. The driving time of each driver is 404 minutes. We then use a sliding time window to construct the two driving state transition graphs. The default size of sliding time window is 30 minutes. For example, the first time window is 0-30 minutes, the second one is 1-31 minutes, the third one is 2-32 minutes, and so on. The driving state transition graph of drivers during each time window are the inputs to anomaly detection models. Since the total driving time of each driver is 404 minutes, we divide it into 375 sliding time windows. Each driver generates one driving feature vector during each window. The total sample size of normal data is 189,000.

Driving Speed and Direction. In the T-Drive dataset, given two data samples $(d_1,\ t_1,\ \lambda_1,\ \varphi_1)$ and $(d_1,\ t_2,\ \lambda_2,\ \varphi_2)$, the driving distance $D_{1,2}$ of driver d_1 from t_1 to t_2 can be calculated by [37]:

$$\begin{aligned} D_{1,2} = atan2\Bigg(&\sqrt{sin^2\left(\frac{\Delta\varphi}{2}\right) + cos\varphi_1 \cdot cos\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)}, \\ &\sqrt{1 - sin^2\left(\frac{\Delta\lambda}{2}\right) - cos\lambda_1 \cdot cos\lambda_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)}\Bigg) \cdot 2R \end{aligned} \tag{14}$$

Table 2. Data Distribution of Normal Data and Anomalous Data

	Value	Normal	Anomaly 1 acc (1, 1)	Anomaly 2 angle (1, 1)	Anomaly 3 acc (0.2, 0.04) angle (0.2, 0.04)
Acceleration (m/s^2)	<−0.5	21.5%	17.0%	21.5%	18.8%
	[−0.5, 0.5]	57.3%	19.5%	57.3%	52.3%
	>0.5	21.2%	63.5%	21.2%	28.9%
Bearing Angle (radian)	<−1	8.1%	8.1%	7.1%	13.5%
	[−1, 1]	84.3%	84.3%	43.4%	68.9%
	>1	7.6%	7.6%	49.5%	17.6%

where $\Delta\varphi = \varphi_2 - \varphi_1$ is the difference between latitudes, $\Delta\lambda = \lambda_2 - \lambda_1$ is the difference between longitudes, and R is the radius of the earth. Then we can get the average driving speed from t_1 to t_2 by $v_{1,2} = \frac{D_{1,2}}{t_2 - t_1}$. Similarly, the driving speed $v_{2,3}$ from t_2 to t_3 can be acquired. By comparing the value of $v_{1,2}$ and $v_{2,3}$, we can know the driver is accelerating, decelerating or driving in a constant speed. The bearing radian $\theta_{1,2}$ of driver d_1 from t_1 to t_2 can be calculated by [37]:

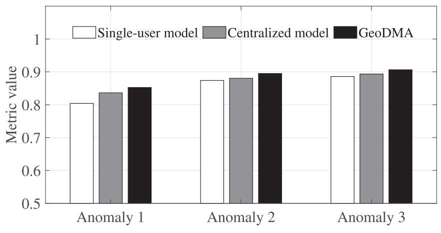
$$\theta_{1,2} = atan2(sin\Delta\lambda \cdot cos\varphi_2, cos\varphi_1 \cdot sin\varphi_2 - sin\varphi_1 \cdot cos\varphi_2 \cdot cos\Delta\lambda) \tag{15}$$

By comparing $\theta_{1,2}$ and $\theta_{2,3}$ between two consecutive timestamps, we can obtain the bearing angle of the driver, i.e., bearing towards left, bearing towards right, or driving straight.

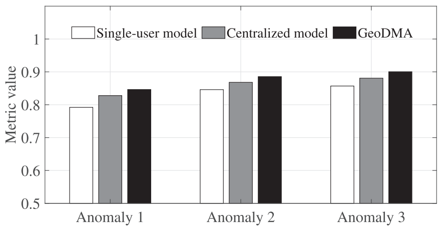
Driving Maneuver Anomaly Data. We need anomalous driving maneuvers to test the performance of our system. Since it is dangerous to collect anomalous driving data from real world, we borrow the idea from pBEAM [6] to simulate the anomalies based on the real-world data to evaluate the system. Three kinds of anomalies are simulated as follows:

- Anomaly 1: For aggressive drivers, they usually have irregular acceleration or deceleration [8]. This is one of the typical anomalies in real world. To simulate this scenario, we add Gaussian noise (mean 1.0, variance 1.0) to the acceleration of our normal test data. The unit of acceleration is m/s^2.
- Anomaly 2: When the driver is sleepy or distracted, the driving trajectories may show unexpected bearing angles [7]. We add Gaussian noise (mean 1.0, variance 1.0) to the bearing angle of our normal test data to simulate this scenario. The unit of bearing angle is radian.
- Anomaly 3: For **DUI (driving under the influence)** or **DWI (driving while intoxicated/driving while impaired)**, both the acceleration and the bearing angle can become abnormal [38]. To simulate this scenario, we add Gaussian noise (mean 0.2, variance 0.04) to both acceleration and bearing angle. The mean and variance are set to a smaller value than the previous two cases because there are changes in both acceleration and bearing angle.

Table 2 shows the data distribution of normal data and the generated anomalous data. We use 0.5 m/s^2 as the threshold to define the acceleration of a vehicle. If the acceleration of the vehicle is larger than 0.5 m/s^2, we define the vehicle is accelerating. If it is smaller than −0.5 m/s^2, we define it is decelerating. Otherwise, it is going at a constant speed. Similarly, we use 1 radian and −1 radian as the threshold to define if a vehicle is turning right, turning left, or going straight. In most of the experiments, we set the ratio of normal and abnormal test data as 1:1. Table 4 shows the experimental result of changing this ratio.



(a) AUC (area under the ROC curve)



(b) F1 Score

Fig. 5. Overall performance comparison.

5.3 Overall Performance

For the centralized model, we use the driving data of all 504 drivers to train it. In *GeoDMA*, we divide drivers into four sub-regions based on their GPS trajectories. But most of the drivers do not drive in the same sub-region all the time. To be consistent with the centralized model, we filter the drivers in each sub-region to find the drivers that drive in the same sub-region during all the 375 sliding windows. After filtering, there are 33, 24, 10, and 9 drivers left in each sub-region, respectively. We train a model for each sub-region by using the filtered data. We calculate the average AUC and F1 score of four region-based models. For the single-user model, we train 76 different models for these 76 drivers and calculate the average AUC and F1 score of these models. Figure 5 depicts the performance of *GeoDMA* and two baselines.

The experiment results show that the AUC of centralized model achieves 0.836, 0.881, and 0.894 for the three generated anomalies. The corresponding F1 score is 0.828, 0.868, and 0.881. It achieves up to 8.5% higher accuracy than the single-user model. The reason for such an improvement is that we consider the vehicle-vehicle peer dependency in the centralized model. Moreover, the AUC of *GeoDMA* achieves 0.853, 0.895, and 0.907 for the three generated anomalies, respectively. The corresponding F1 score is 0.846, 0.886, and 0.901. It further improves the accuracy of the centralized model by up to 2.2%.

In all anomalous scenarios, *GeoDMA* outperforms the centralized model. This is because *GeoDMA* considers the region partitioning. The driving features exist common pattern in a small region across the vehicles due to similar contextual environments. Whereas the centralized model is a general model and is trained for the whole city, it cannot take the local contextual features into account. We also found basically the performance of all the systems on anomaly 2 is better than anomaly 1. The performance on anomaly 3 is almost the same as anomaly 2. This demonstrates the model is more sensitive to anomalous bearing angles than anomalous speeds.

5.4 Performance under Different Scenarios

We evaluate the performance of *GeoDMA* in different sub-regions, with different amounts of anomalous data. We also analyze the performance of *GeoDMA* along with the time and among different drivers.

Table 3 depicts the performance comparison of *GeoDMA* with two baselines. Among these four regions, the model for region 4 performs the best; the performance of the other three region models under the the same accuracy metric do not show much difference. The model for region 4 achieves up to 10.2% and 6.0% higher accuracy than the single-user model and the centralized model on average under F1 score. It also achieves up to 11.5% and 6.6% higher accuracy than the single-user model and centralized model on average under AUC. More importantly, almost all of these four models achieve better performance than the single-user model and centralized model under the

Table 3. Performance Comparison of Different Models on Real-World Data with Simulated Anomalies

	Model	Anomaly 1				Anomaly 2				Anomaly 3			
AUC	Single-User	0.804				0.874				0.886			
	Centralized	0.836				0.881				0.894			
	GeoDMA	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
		0.842	0.841	0.841	**0.886**	0.9	0.902	0.857	**0.921**	0.906	0.902	0.888	**0.93**
F1	Single-User	0.792				0.846				0.857			
	Centralized	0.828				0.868				0.881			
	GeoDMA	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
		0.837	0.835	0.83	**0.883**	0.891	0.892	0.844	**0.916**	0.911	0.892	0.874	**0.925**

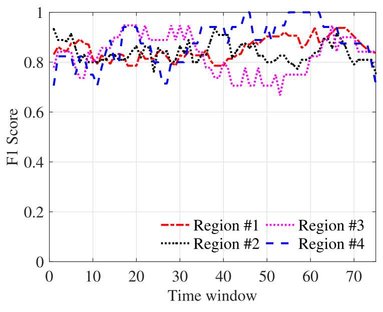
Table 4. Performance Comparison of Different Ratios of the Normal Data to the Anomalous Data in Test Data

	Ratio	Anomaly 1	Anomaly 2	Anomaly 3
AUC	1:1	0.853	0.895	0.907
	3:2	0.854	0.894	0.908
	3:1	0.852	0.89	0.908
F1 Score	1:1	0.846	0.886	0.901
	3:2	0.865	0.89	0.899
	3:1	0.881	0.893	0.899

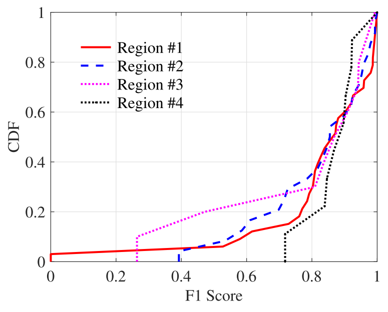
same evaluation metric on the three anomalies except the model for region 3. But the average performance of these models is still better than the single-user model and the centralized model as we see from Figure 5. The experiment results demonstrate that it is reasonable to divide a big city into multiple sub-regions and develop multiple region models.

5.4.1 Different Ratio of Normal Data to the Anomalous Data. Table 4 shows the performance of *GeoDMA* when we change the ratio of normal and anomalous test data. For all of the three anomalies under AUC, the performance is basically the same when we change the ratio. The performance under F1 score on anomaly 2 and anomaly 3 do not change much. The performance under F1 score on anomaly 1 becomes a little higher when the anomalous test data becomes smaller. This experiment shows that the performance of our system is robust and suitable to the real-world scenarios.

5.4.2 Performance Analysis from Time Perspective. We further investigate the performance of *GeoDMA* from a time perspective. For the construction of state transition vectors, we set the length of a sliding window as 30 minutes. The driving maneuvers of the drivers may change over different time windows. We evaluate the performance of drivers over different time windows. In test data, there are 75 time windows. Figure 6(a) presents the anomaly detection performance of all drivers from a time perspective under four sub-regions. As shown in the figure, the F1 score of anomaly detection is higher than 0.8 during most of the time windows. And we can see that the F1 score of the model fluctuates slightly but is still stable during different time windows. For the third region, from time window 35 to time window 60, its F1 score is lower than 0.8 but still higher than 0.7. This experiment demonstrates that *GeoDMA* has good performance along with time.

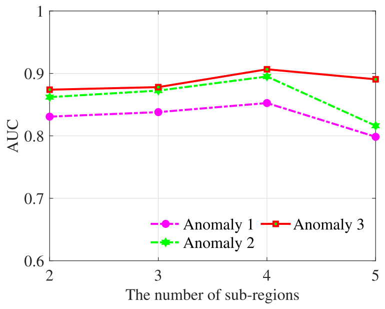


(a) Accuracy along with time.

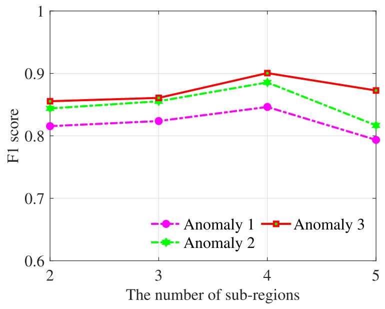


(b) Accuracy of all drivers.

Fig. 6. The performance of *GeoDMA* along with time and for different drivers.



(a) AUC of different numbers of sub-regions



(b) F1 score of different numbers of sub-regions

Fig. 7. Performance of the different number of sub-regions.

5.4.3 Performance Analysis from Driver Perspective. We analyze the performance of drivers in each sub-region from a space perspective. Each sub-region has a different driver set. Figure 6(b) depicts the cumulative probability of F1 score for the drivers in the four sub-regions. The cumulative probability of drivers achieving a certain F1 score in the four sub-regions has a similar trend. On average, only 15.2% of the drivers have an F1 score lower than 0.73. The F1 score of 66.7% drivers is higher than 0.79, the F1 score of 44.4% drivers is higher than 0.90. It proves that *GeoDMA* can provide high accuracy for almost all drivers. Moreover, the region models are robust because they show a similar trend in all sub-regions.

5.5 Parameters Settings of *GeoDMA*

We further conduct experiments to set two important parameters in *GeoDMA*, *i.e.*, the number of sub-regions, the dimension size of hidden representation feature vector.

5.5.1 The Number of Sub-Regions. As introduced in Section 3, we use our customized Ncut algorithm to partition a city into multiple sub-regions. However, the Ncut algorithm cannot decide the optimal number of sub-regions automatically. The number needs to be tuned based on different applications. We explore this number from 2 to 5 to find the optimal one. For each number, we conduct experiments over the whole dataset to evaluate driving anomaly detection accuracy.

Figure 7 presents the performance comparison under different numbers of sub-regions. For each case, we use the average AUC and F1 score of all sub-regions as its performance. Figure 7 shows that both AUC and F1 score increase first and then degrade on all the three anomalies. The number of sub-region parameter has a great influence on the system. For example, if the number is 2, the partition result has a little difference with original road network. If *GeoDMA* applies such a partition, it may have similar performance with the centralized model, *i.e.*, 0.831 *v.s.* 0.836 in F1 score.

When the number changes from 2 to 4, the performance increases. The reason is that the more sub-regions, the better the sub-region models extract peer dependency. However, based on our current dataset, if the number of sub-regions is too big, there will be few data belonging to the same sub-region, which may cause the model overfitting because of insufficient data. When the number is 5, we notice that the performance degrades much. We dig into this case and find that two of these five region models do not perform very well and influence the average performance of the system. Under the setting of our current dataset, *GeoDMA* performs the best when we partition the city into four regions. However, if we can get enough effective training data from this city, the optimal number of divisions may be different. In addition, if the dataset is collected from another city, the road network is different, the optimal partition result will be different. The larger the dataset, the more parts the system will divide a city into.

5.5.2 The Size of the Representation Feature Vector. We also do experiments to find out the optimal size of the representation feature. We set the representation feature size to 10, 20, 30, and 40 to train the region models. For each feature size, we show the average performance of 4 region models under AUC and F1 score. Here we use the Anomaly 1 to do the experiments. Figure 8 shows that when the feature size is 20, the models perform the best in terms of both evaluation metrics. Therefore, we set the representation feature size to 20 to train all the driving anomaly detection models, including the single-user model and the centralized model.

5.6 In-Situ Model Updating of *GeoDMA*

The prior experiments of *GeoDMA* are done without model updating. *GeoDMA* updates the model for each sub-region periodically in order to improve its performance. Each region model is updated by using the data collected from the vehicles within its coverage. Due to the limited available data, we update the models twice. Each time we use the data from 25 sliding windows to do incremental training. The data for model updating is different from the initial training data in the dataset. Figure 9 depicts the performance of *GeoDMA* under the settings without model updating ($U0$), updating once ($U1$) and updating twice ($U2$). It presents the average performance of the four sub-region models. We find that after each update, the performance of the updated models is improved on all anomalies under both AUC and F1 score.

5.7 Execution Efficiency of *GeoDMA*

A model for driving maneuver anomaly detection is required to be lightweight since the detected anomalies should be sent to the nearby drivers or passengers as fast as possible to avoid possible traffic. *GeoDMA* is lightweight and efficient. The size of *GeoDMA* auto-encoder is only 72.2 KB. The model provides an inference result in each sliding window (1-minute step). It takes about 8.37 ms to execute one inference in one time window for one driver. The inference time is measured on Alianware Aurora R7 (one Intel Core i5 8400 CPU (6-core) and one NVIDIA GTX 1070 GPU) without using its GPU. The driving data from other vehicles is no longer needed when doing anomaly detection for a specific vehicle with a well-trained model.

6 RELATED WORK

Autoencoder-based Anomaly Detection. With the development of deep learning, some recent solutions apply deep autoencoders for anomaly detection [22, 39–41]. RDA [22] demonstrates the

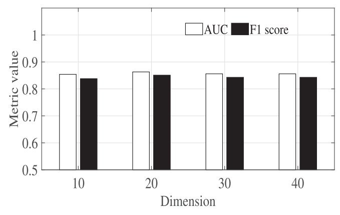
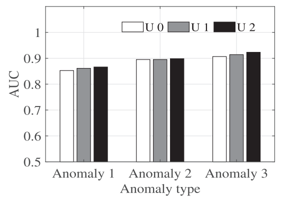
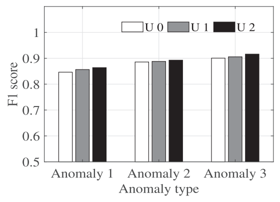


Fig. 8. Different representation sizes.



(a) AUC



(b) F1 score

Fig. 9. *GeoDMA* with or without model updating.

effectiveness of deep autoencoder-based anomaly detection on image dataset. MemAE [40] uses deep autoencoder for both image and video anomaly detection. [41] leverage deep autoencoder for video anomaly detection. DAGMM [39] shows the effectiveness of deep autoencoder-based anomaly detection model on network intrusion detection, thyroid cancer analysis and arrhythmia analysis. In this paper, we apply deep autoencoder for anomaly driving maneuver detection.

Driving Anomaly Detection. Smartphone-based methods have been developed by commercial companies, such as Google and Uber, to detect anomalous driving maneuvers [4, 5, 42–44]. CarSafe [4] detects drowsy and distracted drivers using cameras on smartphones. DriveSafe [5] uses the rear camera, the microphone, the inertial sensors, and the GPS of the smartphones to assess if a driver is drowsy or distracted. The camera-based methods are hard to achieve real-time performance. SenSpeed [44] utilizes the accelerometer and gyroscope of smartphones to acquire the instant vehicle speed. However, this solution assumes the alignment between the vehicle's coordinate system and the smartphone's coordinate system is fixed, which is hard to guarantee when the vehicle is driving. [42, 43] use GPS data, and SafeDrive [18] leverages the data from On-Board Diagnostics to identify driving anomalies. However, they rely on the sensor readings of a single driver, and the temporal and spatial correlations of the driving maneuvers from multiple drivers are ignored. Deep learning shows its efficiency in anomaly detection [45–48]. In pBEAM [6], it leverages unsupervised conditional adversarial RNN to train a single-user driving anomaly detection model. However, it ignores the peer dependency across vehicles. In this paper, we consider the temporal correlation of driving maneuvers and peer dependency of driving maneuvers in our driving anomaly detection model.

Driving Behavior Modeling. In the literature, there are also some crowdsensing works on collecting data from multiple users for driving behavior modeling [11, 49–51]. Some of these studies process the data of individual vehicles independently, but the statistical correlations across vehicles in those studies were ignored. Some studies group behaviors of drivers, but do not detect driving maneuvers of each vehicle. Most of the driving behavior modeling methods are offline analysis and assessment. PTARL [17] learns the representation of driving state transitions and uses the learned representation features to assess the historical driving score of each driver. It focuses on representation learning but not driving maneuver anomaly detection. However, *GeoDMA* is focused on real-time driving anomaly detection, which takes both individual driving maneuvers and vehicle-vehicle peer dependency into account. Besides, the contextual information like weather, traffic, and road condition in a small region tends to be similar.

Geographic Region Partition. The First Law of Geography proposed in 1970 [12] has been applied in many applications, like city management, urban traffic control, and transportation simulations [25, 52–58]. Graph-based road network partition methods are commonly used for region partitioning [25, 52, 59]. Following those solutions, we also leverage the road network graph to perform geographical partitioning. However, the graph we defined is closely related to the driving maneuver anomaly detection, e.g., we define the weight of edges based on the spatial correlation of road segments. Our optimization objective is to maximize the spatial correlation in a sub-region and minimize the spatial correlation between any two sub-regions.

7 DISCUSSION

Privacy Issues. During online driving anomaly detection, *GeoDMA* collects the GPS locations from all the vehicles and uploads the locations to the corresponding edge servers. To protect user privacy, we anonymize the user identification information by a code. We will not collect any private and sensitive information from the users.

Static Geographical Partitioning. Due to the limited size of our current dataset, we perform the geographical partition statically. To provide better services for users, it is more reasonable to adjust the geographical partition dynamically as more new data is collected from the city. If the geographical partition is formed dynamically, new models for new local regions will be trained based on the new partition, using the data collected under new local regions. All historical data can be used for training, as long as it follows the new geographical partitioning. The new models will be trained offline. We still use the previous models to do online detection, while we are training the new models. Once the new models are well trained, they will be deployed for detection. Transfer learning that leverages the old models to train the new models will be explored to shorten the training time of new models in future work.

Model Generalization. Our models can be used to do anomaly detection for different scenarios (i.e., sparse, or dense traffic flows) if the models are trained with sufficient data collected from those scenarios. Moreover, we believe the detection accuracy can be further improved if the models are developed depending on different scenarios. As more and more GPS locations are collected from the users, the models in the same area can be trained in a fine-grained way. We can train different models for workdays and weekends. To be more specific, during the same day, we can train different models for different time periods, e.g., a model for peak hours, a model for off-peak hours in the daytime, and a model for the night.

8 CONCLUSION

This paper presents *GeoDMA*, which leverages unsupervised deep auto-encoders and a geo-distributed partitioning algorithm to develop a driving maneuver anomaly detection system. The auto-encoder learns normal driving features from historical data by considering both temporal and peer dependency. Our geo-distributed partitioning algorithm further divides a city into several sub-regions. We then train a specific model for each sub-region to improve the detection accuracy in each sub-region. Extensive experiments in a large-scale real-world vehicle GPS trajectory dataset show that *GeoDMA* outperforms the baseline methods.

REFERENCES

[1] Miaomiao Liu and Wan Du. 2020. Geo-distributed driving maneuver anomaly detection. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 310–311.

[2] NHTSA. ([n.d.]). 2018 Fatal Motor Vehicle Crashes: Overview. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812826.

[3] WHO. ([n.d.]). Global Status Report on Road Safety 2018. https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/.

[4] Chuang-Wen You, Nicholas D. Lane, Fanglin Chen, Rui Wang, Zhenyu Chen, Thomas J. Bao, Martha Montes-de Oca, Yuting Cheng, Mu Lin, Lorenzo Torresani, et al. 2013. CarSafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *ACM MobiSys*.

[5] Luis Bergasa and Roberto Arroyo. 2014. DriveSafe: An app for alerting inattentive drivers and scoring driving behaviors. In *IEEE IV*.

[6] Xingzhou Zhang, Mu Qiao, Liangkai Liu, Yunfei Xu, and Weisong Shi. 2019. Collaborative cloud-edge computation for personalized driving behavior modeling. In *ACM/IEEE SEC*.

[7] Sinan Kaplan, Mehmet Amac Guvensan, Ali Gokhan Yavuz, and Yasin Karalurt. 2015. Driver behavior analysis for safe driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 16, 6 (2015), 3017–3032.

[8] Cian Ryan, Finbarr Murphy, and Martin Mullins. 2020. End-to-end autonomous driving risk analysis: A behavioural anomaly detection approach. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[9] Zhongyang Chen, Jiadi Yu, Yanmin Zhu, Yingying Chen, and Minglu Li. 2015. D 3: Abnormal driving behaviors detection and identification using smartphone sensors. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON'15)*. IEEE, 524–532.

[10] Lex Fridman, Daniel E. Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsekin, Julia Kindelsberger, Li Ding, et al. 2019. MIT advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation. *IEEE Access* 7 (2019), 102021–102038.

[11] Jin-Hyuk Hong, Ben Margines, and Anind K. Dey. 2014. A smartphone-based sensing platform to model aggressive driving behaviors. In *ACM SIGCHI*.

[12] Waldo R. Tobler. 1970. A computer movie simulating urban growth in the Detroit region. *Economic Geography* 46 (1970), 234–240.

[13] Isam Mashhour Al Jawarneh, Paolo Bellavista, Antonio Corradi, Luca Foschini, and Rebecca Montanari. 2020. Locality-preserving spatial partitioning for geo big data analytics in main memory frameworks. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 1–6.

[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NeurIPS-W*.

[15] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: Driving directions based on taxi trajectories. In *ACM SIGSPATIAL*.

[16] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *ACM SIGKDD*.

[17] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Pengfei Wang, Yu Zheng, and Charu Aggarwal. 2018. You are how you drive: Peer and temporal-aware representation learning for driving behavior analysis. In *ACM SIGKDD*.

[18] Mingming Zhang, Chao Chen, Tianyu Wo, Tao Xie, Md Zakirul Alam Bhuiyan, and Xuelian Lin. 2017. SafeDrive: Online driving anomaly detection from large-scale vehicle data. *IEEE Transactions on Industrial Informatics* 13, 4 (2017), 2087–2096.

[19] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[20] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. 2018. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3379–3388.

[21] Muhammad Zaigham Zaheer, Jin-ha Lee, Marcella Astrid, and Seung-Ik Lee. 2020. Old is gold: Redefining the adversarially learned one-class classifier training paradigm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14183–14193.

[22] Chong Zhou and Randy C. Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *ACM SIGKDD*.

[23] Hongyong Wang, Xinjian Zhang, Su Yang, and Weishan Zhang. 2021. Video anomaly detection by the duality of normality-granted optical flow. *arXiv preprint arXiv:2105.04302* (2021).

[24] Mujtaba Asad, Jie Yang, Enmei Tu, Liming Chen, and Xiangjian He. 2021. Anomaly3D: Video anomaly detection based on 3D-normality clusters. *Journal of Visual Communication and Image Representation* 75 (2021), 103047.

[25] Ying-Ying Ma, Yi-Chang Chiu, and Xiao-Guang Yang. 2009. Urban traffic signal control network automatic partitioning using Laplacian eigenvectors. In *IEEE ITSC*.

[26] Andreas Emil Feldmann and Luca Foschini. 2015. Balanced partitions of trees and applications. *Algorithmica* 71, 2 (2015), 354–376.

[27] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Vol. 174. Freeman San Francisco.

[28] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (2007), 395–416.

[29] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.

[30] ([n.d.]). OpenStreetMap. https://www.openstreetmap.org.

[31] Xianzhong Ding, Wan Du, and Alberto Cerpa. 2019. Octopus: Deep reinforcement learning for holistic smart building control. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 326–335.

[32] Xianzhong Ding, Wan Du, and Alberto E. Cerpa. 2020. MB2C: Model-based deep reinforcement learning for multi-zone building control. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 50–59.

[33] Xianzhong Ding and Wan Du. 2022. DRLIC: Deep reinforcement learning for irrigation control. In *ACM/IEEE IPSN*.
[34] Miaomiao Liu, Xianzhong Ding, and Wan Du. 2020. Continuous, real-time object detection on mobile devices without offloading. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS'20)*. IEEE, 976–986.
[35] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *ACM SIGSPATIAL*.
[36] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. 2020. DMM: Fast map matching for cellular data. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.
[37] ([n.d.]). Geosphere. http://www.edwilliams.org/avform.htm#Dist.
[38] Jiangpeng Dai, Jin Teng, Xiaole Bai, Zhaohui Shen, and Dong Xuan. 2010. Mobile phone based drunk driving detection. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, 1–8.
[39] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In *ICLR*.
[40] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1705–1714.
[41] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis. 2016. Learning temporal regularity in video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 733–742.
[42] Apichon Witayangkurn, Teerayut Horanont, Yoshihide Sekimoto, and Ryosuke Shibasaki. 2013. Anomalous event detection on large-scale GPS data from mobile phones using hidden Markov model and cloud platform. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. 1219–1228.
[43] Siqian Yang, Cheng Wang, Hongzi Zhu, and Changjun Jiang. 2019. APP: Augmented proactive perception for driving hazards with sparse GPS trace. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 21–30.
[44] Jiadi Yu, Hongzi Zhu, Haofu Han, Yingying Jennifer Chen, Jie Yang, Yanmin Zhu, Zhongyang Chen, Guangtao Xue, and Minglu Li. 2015. Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments. *IEEE Transactions on Mobile Computing* 15, 1 (2015), 202–216.
[45] Vidyasagar Sadhu, Teruhisa Misu, and Dario Pompili. 2019. Deep multi-task learning for anomalous driving detection using CAN bus scalar sensor data. *arXiv preprint arXiv:1907.00749* (2019).
[46] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).
[47] Sarah M. Erfani, Mahsa Baktashmotlagh, Masud Moshtaghi, Vinh Nguyen, Christopher Leckie, James Bailey, and Kotagiri Ramamohanarao. 2017. From shared subspaces to shared landmarks: A robust multi-source classification approach. In *AAAI*.
[48] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *ACCV*.
[49] Abd-Elhamid M. Taha and Nidal Nasser. 2015. Utilizing CAN-bus and smartphones to enforce safe and responsible driving. In *IEEE ISCC*.
[50] Tanushree Banerjee, Arijit Chowdhury, and Tapas Chakravarty. 2016. MyDrive: Drive behavior analytics method and platform. In *ACM WPA*.
[51] Xiaoyu Zhu, Yifei Yuan, Xianbiao Hu, Yi-Chang Chiu, and Yu-Luen Ma. 2017. A Bayesian network model for contextual versus non-contextual driving behavior assessment. *Transportation Research Part C: Emerging Technologies* 81 (2017), 172–187.
[52] Yan Xu and Gary Tan. 2012. An offline road network partitioning solution in distributed transportation simulation. In *IEEE/ACM DS-RT*.
[53] Yuxuan Ji and Nikolas Geroliminis. 2012. On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological* 46, 10 (2012), 1639–1656.
[54] Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *ACM SIGKDD*.
[55] Nicholas Jing Yuan, Yu Zheng, and Xing Xie. 2012. Segmentation of urban areas using road networks. *MSR-TR-2012–65, Tech. Rep.* (2012).
[56] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. 2011. Discovering spatio-temporal causal interactions in traffic data streams. In *ACM SIGKDD*.
[57] Yanjie Fu, Pengyang Wang, Jiadi Du, Le Wu, and Xiaolin Li. 2019. Efficient region embedding with multi-view spatial networks: A perspective of locality-constrained spatial autocorrelations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 906–913.

[58] Yunchao Zhang, Yanjie Fu, Pengyang Wang, Xiaolin Li, and Yu Zheng. 2019. Unifying inter-region autocorrelation and intra-region structures for spatial embedding via collective adversarial learning. In *ACM SIGKDD*.
[59] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. 2007. Adaptive fastest path computation on a road network: A traffic mining approach. In *VLDB*.

Received 27 December 2021; revised 27 June 2022; accepted 24 August 2022