Online Anomalous Subtrajectory Detection on Road Networks with Deep Reinforcement Learning [Scalable Data Science]

Qianru Zhang[1,*], Zheng Wang[2,*], Cheng Long[2], Siu Ming Yiu[1], Yiding Liu[3], Gao Cong[2], Jieming Shi[4]
[1]Department of Computer Science, The University of Hong Kong, Hong Kong SAR
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore, [3]Baidu Inc, China
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR
{qrzhang,smyiu}@cs.hku.hk,{wang_zheng,c.long,gaocong}@ntu.edu.sg,liuyiding.tanh@gmail.com,jieming.shi@polyu.edu.hk

ABSTRACT

Detecting anomalous trajectories has become an important concern in many location-based applications. Most of the existing studies for this problem only give a vague judgement (e.g., an anomaly score) of whether a given trajectory is anomalous, but are not able to indicate which part of the trajectory is anomalous in a **timely manner**, i.e., detecting anomalous **sub**trajectory. Anomalous subtrajectory detection is important for a number of real-world applications, e.g. for ladies going home late, elderly care, and child custody. The problem is challenging due to three issues: 1) the lack of labeled data, 2) how to model anomalous subtrajectories more accurately, and 3) the efficiency of detection in a timely manner. To tackle these challenges, we propose a novel reinforcement learning based solution, which is able to expressively learn the characteristics of anomalous subtrajectories without labeled data, and efficiently detect subtrajectory anomalies in an online manner. Extensive experiments are conducted on two real-world trajectory datasets with manually labeled test data, and the results show that our solution can significantly outperform the state-of-the-art methods (with 20-30% improvement) and is efficient for online detection (it takes less than 0.1ms to process each newly generated data point).

PVLDB Reference Format:
Qianru Zhang[1,*], Zheng Wang[2,*], Cheng Long[2], Siu Ming Yiu[1], Yiding Liu[3], Gao Cong[2], Jieming Shi[4] . Online Anomalous Subtrajectory Detection on Road Networks with Deep Reinforcement Learning [Scalable Data Science]. PVLDB, 14(1): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

1 INTRODUCTION

With the proliferation of tracking facilities involving mobile phones and tracking technologies involving GPS, a large amount of *trajectory* data is being generated by moving objects. These trajectory datasets provide us an opportunity to identify important information for useful location-based applications. Existing studies on such datasets have attempted to solve many problems including: mining the traffic patterns [16], identifying co-movement objects [5, 25], conducting trajectory similarity search [20, 24, 25] and predicting the next location of a moving object [30].

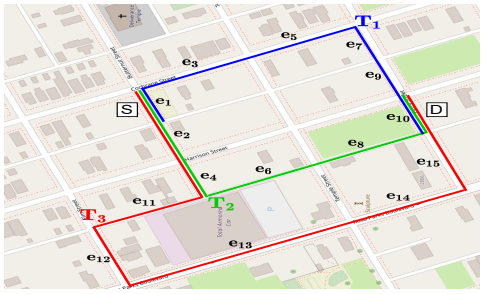


Figure 1: T_1, T_2 and T_3 are three trajectories from the same source e_1 to the same destination e_{10}.

An *anomaly* is an object that is different from or inconsistent with the remaining objects [8]. In our context, a trajectory anomaly or an anomalous trajectory refers to a trajectory that is different from the majority trajectories for the same source and destination [2, 3]. Recently, detecting anomalous trajectories has attracted much attention, and many approaches (e.g., [3, 17, 28, 34, 35]) have been proposed for solving it. In the following, we assume that each trajectory has a source and destination pair, referred as a SD pair. Consider the example in Figure 1. There are three trajectories from the source $S(e_1)$ to the destination $D(e_{10})$. The trajectory T_3 (the red one) is considered as anomalous with respect to the SD pair $< S, D >$ if the majority trajectories for the same SD pair follow either T_1 (the blue one) or T_2 (the green one). While most of the existing approaches can discriminate whether a target trajectory is anomalous or not, they are not able to provide information on **which part of the trajectory (*subtrajectory*) is anomalous in a timely manner**, which is critical for a number of real-world applications. For instance, a ride-haling company could immediately query the driver when the trajectory starts to deviate from a normal route, or provide a warning to the passenger before a possible tragedy occurs (e.g. a lady going home late at night). Other examples include elderly care, especially for those who suffer from Alzheimer or child custody in countries with high kidnapping rates.

Detecting anomalous subtrajectories effectively in a timely (online) manner is a non-trival task with the following challenging issues: (1) **Lack of labelled data.** Manually labeling large-scale subtrajectories costs unaffordable human effort. This obstructs some straightforward data-intensive methods to be applied for this problem. As far as we know, the only manually labelled dataset available (used in Chen et al. [3]) contains only 9 SD pairs with about 1,000

*Equal contribution.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

trajectories. Using a small dataset cannot provide accurate information on the performance of an approach in a real application. (2) **Subtrajectory modeling.** Existing approaches for modeling anomalous subtrajectories follow one of the following strategies. The first approach is to use a set of manually defined parameters to characterize an anomalous subtrajectory. Due to the limited amount of labelled data, a fixed set of parameters is not general enough to cover all possible situations and may not be adaptive to different road conditions. The second approach is to use a distance-based measure to differentiate anomalous trajectories from normal ones, i.e., when the deviation of distance of the current partial route exceeds a certain threshold from a normal route, it reports a detouring event. First, the threshold is hard to set appropriately. Second, this is an indirect measure for anomalous subtrajectories. It is not clear whether the detection can be done in a timely manner. The most promising approach is the learning-based method, however, all proposed learning-based methods do not focus on identifying anomalous subtrajectories and can only be applied to answering whether a given trajectory is anomalous or not. A possible reason is due to the lack of labeled data for anomalous subtrajectories. (3) **Efficient online detection.** Trajectories can be generated at a very high speed, and thus detection efficiency is a major concern. For example, taxi trajectories with 790 millions of GPS points, involving about 5 millions of trajectories, can be generated in Beijing in 3 months [32, 33]. The number of subtrajectories is usually several orders of magnitude larger than the number of trajectories. In our example, there are over 1 billion of subtrajectories corresponding to the 5 millions of trajectories. It is important to have a solution that can conduct *online detection*, where the anomalousness can be revealed as early as possible while a trajectory is being generated.

Existing studies cannot address the above challenges in an effective way (see the next section for more details). In this paper, we propose a novel reinforcement learning based solution RL4OASD to address the challenges. RL4OASD consists of three parts, namely the data preprocessing, RSRNet and ASDNet. First, we obtain map-matched trajectories via map-matching the raw trajectories (GPS points) on road networks. The map-matched trajectory is in the form of a sequence of road segments. We then design a *noisy labeling method* to label each road segment as 0 representing normal and 1 otherwise automatically, by counting each transition frequency from historical trajectories. This is the first step to address the issue of not enough labeled data. During the training process, the noisy labels are used to train the Road Segment Representation Network (RSRNet). In order to improve the effectiveness of RL4OASD, we model the detecting anomalous subtrajectories as a Markov decision process [21] named Anomalous Subtrajectory Detection Network (ASDNet). ASDNet is trained jointly with RSRNet, in a weakly supervised manner using noisy labeled data. The two networks can mutually refine each other, where RSRNet can provide expressive sequential information behind trajectories for ASDNet to detect, and ASDNet can offer supervision for RSRNet to better learn the characteristics of subtrajectories. During the testing period, RL4OASD perform testing on manually labelled test data. In this way, we address the first challenge effectively.

Second, RSRNet, which exploits road network representation learning and deep sequence modeling techniques that jointly model the *characteristics* underlying subtrajectories, which addresses the second challenge. Third, a novel detection algorithm, which leverages the learned policy and enables online detection of anomalous subtrajectories, which addresses the third challenge.

Our contributions can be summarized as follows. (1) We propose the first deep reinforcement learning based solution to detect anomalous subtrajectories. The proposed model 1) integrates the characteristics of subtrajectories and road network structure, and 2) can be trained without labeled data. (2) We propose a novel detection paradigm based on the learned policy, which allows the anomalous subtrajectory detection to be efficiently conducted in an online manner. (3) We conduct extensive experiments on two real-world trajectory datasets with manually-labelled test data. We compare our proposed solution with various baselines, and the results show that our solution can significantly outperform the state-of-the-art methods (with 20-30% improvement compared to the best existing approach) and is efficient for online anomalous subtrajectory detection (it takes less than 0.1ms to process each newly generated data point).

2 RELATED WORK

Online Anomalous Trajectory Detection. Online anomalous trajectory detection aims to detect the ongoing trajectory in an online manner. Existing studies propose many methods for online anomalous trajectory detection and involve two categories: heuristic-based methods [3, 6, 31, 34] and learning-based methods [17, 28].

For heuristic methods, Chen et al. [3] investigates the detecting anomalous trajectories in an online manner via the isolation-based (i.e., clustering) method, which aims to check which parts of trajectories are isolated from the reference (i.e., normal) trajectories with the same source and destination routes. This method aims to detect the anomalous subtrajectories via ongoing trajectories, which is similar to our paper. However, this method models reference trajectories based on many manually-set parameters. Besides, the performance of this method is evaluated on a small manually labelled dataset with 9 SD pairs only, which fail to reflect the generalization of this method. A recent work [34] proposes to calculate the trajectory similarity via discrete Frechet distance between a given reference route and the current partial route at each timestamp. If the deviation threshold exceeds a given threshold at any timestamp, and the system alerts that an anomaly event of detouring is detected. There are many predefined parameters involved in this method. Besides, the method of modelling a reference route is not given, which has a direct impact on the task. Our work differs from these heuristic-based studies in that it is based on a policy learned via reinforcement learning instead of the hand-crafted heuristic with many predefined parameters (e.g., the deviation threshold) for detecting anomalies. Besides, we evaluate the performance of our method on a large dataset.

For learning-based methods, a recent study Liu et al. [17] proposes to detect anomalous trajectories via a generation scheme, which utilizes the Gaussian mixture distribution to represent different kinds of normal routes and detects those anomalous trajectories that cannot be well-generated based on the given representations of normal routes. This method aims to detect whether the ongoing trajectory is anomalous or not. Another learning-based method [28] proposes a probabilistic model to detect trajectory anomalies via

modeling the distribution of driving behaviours from historical trajectories. The method involves many potential features that are associated with driving behaviour modeling, including road level and turning angle, etc. This method also only targets whether the online trajectory is anomalous or not. In our work, we target a more finer-grained setting, i.e, detecting which part of an anomalous trajectory, namely subtrajectory, is responsible for its anomalousness in an online manner. Nevertheless, subtrajectory detection is not the focus in these studies.

Offline Anomalous Trajectory Detection. Offline anomalous trajectory detection refers to detecting the target trajectory which is completed in an offline manner. Also there are two types of studies. One type is heuristic-based methods [1, 12, 18, 35, 37] and another type is learning-based methods [7, 23]. It is clear that these methods proposed in this line of research can not be utilized for the online detection scenario.

Other Types of Anomalous Detection Studies. Some works [1, 6, 15, 31] focus on other types of anomalous trajectory detection, which are related to ours. We review them as follows. Banerjee et al. [1] study temporal anomaly detection, which employs travel time estimation and traverses some potential subtrajectories in a detected trajectory, where the anomalies are identified if the travel time largely deviates from the expected time. Li et al. [15] detect temporal anomalies and a method utilizing historical similarity trends is studied. In addition, Ge et al. [6] investigates the Top-K evolving anomalies and Yu et al. [31] detects anomalous trajectories in large-scale trajectory streams.

3 PROBLEM DEFINITION

(1) Raw Trajectory. A raw trajectory T consists of a sequence of GPS points, i.e., $T =< p_1, p_2, ..., p_n >$, where a GPS point p_i is in the form of a triplet, i.e., $p_i = (x_i, y_i, t_i)$, to record a moving object is located on (x_i, y_i) at timestamp t_i, and n represents the length of the trajectory T.

(2) Road Network. A road network is represented as a directed graph $G(V, E)$, where V represents a vertex set referring to crossroads or intersections, and E represents an edge set referring to road segments, and for each edge e, it connects two vertexes u and v on the road network, denoted as $e = (u, v)$.

(3) Map-matched Trajectory. A map-matched trajectory refers to a trajectory generated by a moving object on road networks, which corresponds to a sequence of road segments after map-matching [29], i.e., $T =< e_1, e_2, ..., e_n >$.

For simplicity, we use T to denote a map-matched trajectory, and we refer to map-matched trajectories as trajectories or routes interchangeably in the rest of the paper.

(4) Subtrajectory and Transition. A subtrajectory $T[i, j]$ corresponds to a portion of the trajectory $T =< e_1, e_2, ..., e_n >$ from e_i to e_j, $1 \le i \le j \le n$. A transition is defined as a special case of a subtrajectory, which only consists of two adjacent road segments, i.e., $< e_{i-1}, e_i >$, where $1 < i \le n$.

(5) Online Anomalous Subtrajectory Detection (OASD). We study the problem of online anomalous subtrajectory detection. Given a source S and destination D pair (SD-pair) with a set of trajectories $\mathcal{T}$ between them. Intuitively, a trajectory T can be considered as *normal* if it follows the route traveled by the majority of trajectories in $\mathcal{T}$. Based on this, we denote an *anomalous* subtrajectory representing a part of a trajectory, which *does not* follow the normal routes within the SD-pair. We formulate the problem as follows.

Problem 1 (OASD). *Given an ongoing trajectory $T =< e_1, e_2, ...e_n >$ that is generated from its source S_T to destination D_T in an online fashion, where the points e_i are generated one by one, and the future points are not accessible in advance. The OASD problem is to identify which parts of T (i.e., subtrajectories) are anomalous if T involves some.*

4 METHODOLOGY

4.1 Overview of RL4OASD

Determining whether a subtrajectory of an ongoing trajectory is anomalous or not is a decision-making process, which could be modeled as a Markov decision process (MDP) [21]. We propose a weakly supervised framework called RL4OASD. The framework consists of three components, namely data preprocessing, RSRNet and ASDNet (see Figure 2).

In data preprocessing, we conduct map-matching on raw trajectories, mapping them onto road networks and obtain the map-matched trajectories. Based on the map-matched trajectories, we compute some labels of the road segments involved in trajectories using some heuristics based on the historical transition data among road segments. The labels, which are noisy, will be used to train RSRNet in a weakly supervised manner (Section 4.2). In RSRNet, we learn the representations of road segments based on both traffic context features and normal route features. These representations in the form of vectors will be fed into ASDNet to define the states of the MDP for labeling anomalous subtrajectories (Section 4.3). In ASDNet, we model the task of labeling anomalous subtrajectories as an MDP and learn the policy a policy gradient method [22] (Section 4.4). ASDNet outputs refined labels of road segments, which are further used to train RSRNet again, and then RSRNet provides better observations for ASDNet to train better policies. The process iterates and we call the resulting algorithm combining RSRNet and ASDNet as RL4OASD (Section 4.5).

4.2 Data Preprocessing for Noisy Labels

The process of obtaining noisy labels involves four steps.

Step-1: We group historical trajectories in a dataset with respect to different SD-Pairs and time slots. Here, we have 24 time slots if we partition one day with one hour granularity, and we say a trajectory falls into a time slot if its starting travel time is within the slot. For example, in Figure 1, we have three map-matched trajectories T_1, T_2 and T_3 with the source e_1 and destination e_{10}. Assuming the starting travel times of T_1, T_2 and T_3 are 9:00, 9:10, and 9:30, respectively. Then, all three trajectories are in the same group because they are within the same time slot with one hour granularity.

Step-2: We then compute the fraction of *transitions* each from a road segment to another with respect to all trajectories in each group. Suppose that there are 5 trajectories traveling along T_1, 4

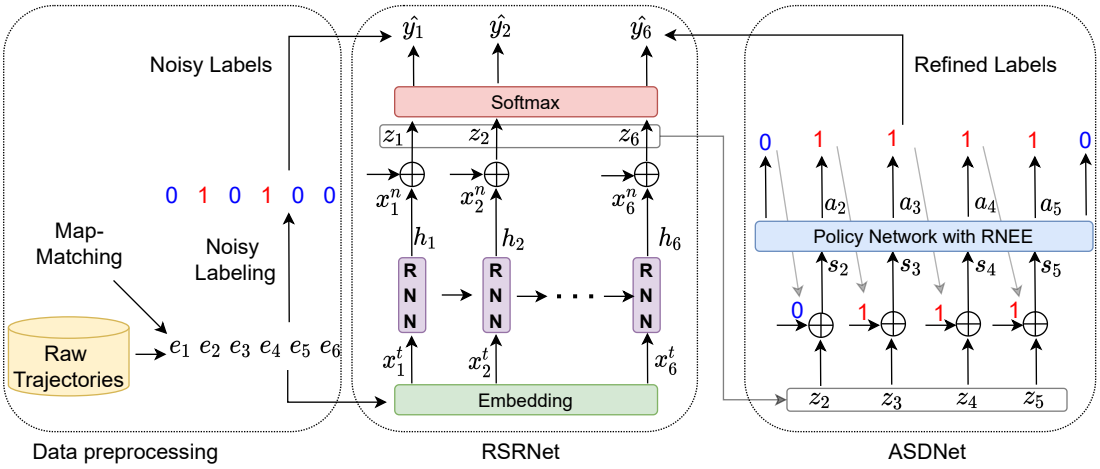


Figure 2: Overview of RL4OASD, where $\bigoplus$ denotes the concatenation operation.

along T_2, and only 1 along T_3, the fraction of *transition* $< e_1, e_2 >$ is calculated as $1/10 = 0.1$ since it appears only once (along T_3) in all 10 trajectories.

Step-3: For each trajectory, we refer to the group it belongs to and map it to a sequence of *transition* fractions with respect to each road segment. For example, for a trajectory traveling the route as T_3, its mapped *transition* sequence is $<< *, e_1 >, < e_1, e_2 >, < e_2, e_4 >, < e_4, e_{11} >, < e_{11}, e_{12} >, < e_{12}, e_{13} >, < e_{13}, e_{14} >, < e_{14}, e_{15} >, < e_{15}, e_{10} >>$, where we pad the initial transition as $< *, e_1 >$, and the corresponding fraction sequence is $< 1.0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 1.0 >$. Note that the fractions on the source e_1 (corresponding to $< *, e_1 >$) and the destination e_{10} (corresponding to $< e_{15}, e_{10} >$) are always set to 1.0, since the source and destination road segments are definitely travelled within its group.

Step-4: We obtain the noisy labels by using a threshold parameter α. For example, by using the threshold *alpha* = 0.1, we obtain the noisy labels of T_3 as $< 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 >$, where 0 denotes a normal road segment, whose fraction is larger than α meaning that the road segment is frequently traveled, and 1 otherwise. The noisy labels will be utilized in the training of RSRNet.

4.3 Road Segment Representation Network (RSRNet)

In RSRNet, we adopt the LSTM [9] structure, which accepts trajectories with different lengths and captures the sequential information behind trajectories. We embed two types of features into representations, namely the traffic context features on road networks and the normal route features for a given SD-Pair.

Traffic Context Feature (TCF). A map-matched trajectory corresponds to a sequence of road segments and each road segment is naturally in the form of the token (i.e., the road segment id). We pre-train each road segment in the embedding layer of RSRNet as a vector, which captures traffic context features (e.g., driving speed, trip duration, road type). To do this, we employ Toast [4], which is a recent road network representation learning model to support road segment based applications. The learned road segment representations will be used to initialize the embedding layer in RSRNet, and those embeddings can be further optimized with the model training. Other road network representation models [10, 26] are also applicable for the task.

Normal Route Feature (NRF). Given an SD-Pair in a time slot, we first infer the normal routes within it. Intuitively, the normal trajectories often follow the same route. If a trajectory contains some road segments that are rarely traveled by others, it probably contains anomalies. Therefore, we infer a route as the normal route by calculating the fraction of the trajectories passing through the route with respect to all trajectories within its SD-Pair. The inferred results are obtained via comparing a threshold (denoted by δ) with the fraction of each road segment, i.e., a route is inferred as the normal if the fraction is larger than the threshold, and vice versa. For example, in Figure 1, given $\delta = 0.4$, we infer T_1 (and resp. T_2) as the normal route, because its fraction $5/10 = 0.5$ (and resp. $4/10 = 0.4$) is larger than or equal to the threshold δ. Based on the inferred normal routes, we then extract the features of a trajectory. For example, given a trajectory following T_3, we extract the features as follows: a road segment in a detected trajectory is normal (i.e., 0) if the transition on that road segment occurs in the inferred normal routes; and 1 otherwise. For example, the extracted normal features of T_3 are $< 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 >$, where the feature on the road segment e_2 is 1 since the transition $< e_1, e_2 >$ does not occur in the normal route T_1. Note that the source and destination road segments always have the feature 0 (i.e., normal). After obtaining the normal route features that are in the form of tokens, we then obtain a vector for the feature by embedding the tokens as one-hot vectors. We call the obtained vectors embedded normal route features.

Training RSRNet. Figure 2 illustrates the architecture of RSRNet. In particular, given a sequence of embedded traffic context features $\mathbf{x_i}^t$ $(1 \le i \le n)$, where n denotes the trajectory length, the LSTM obtains the hidden state $\mathbf{h_i}$ at each road segment. We then concatenate $\mathbf{h_i}$ with the embedded normal route feature $\mathbf{x_i}^n$, denoted by $\mathbf{z}_i = [\mathbf{h}_i; \mathbf{x}_i^n]$. Note that the two parts $\mathbf{h}$ and $\mathbf{x}^n$ capture the sequential trajectory information and normal route information,

respectively. Note that we do not let $\mathbf{x}^n$ go through the LSTM since it preserves the normal route feature at each road segment. In addition, we adopt cross-entropy loss to train RSRNet between the predicted label $\hat{y}_i$ based on the $\mathbf{z}_i$ and the noisy label y_i, i.e.,

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{H}(y_i, \hat{y}_i), \tag{1}$$

where $\mathcal{H}$ denotes the cross-entropy operator.

4.4 Anomalous Subtrajectory Detection Network (ASDNet)

Consider the task of online anomalous subtrajectory detection is to sequentially scan an ongoing trajectory, and for each road segment, decide whether an anomaly happened at that position. This motivates us to model the process as an MDP, involving states, actions, and rewards.

States. We denote the state when scanning the road segment e_i as s_i. The state $\mathbf{s}_i$ ($1 < i \leq n$) is represented as the concatenation of $\mathbf{z}_i$ and $\mathbf{v}(e_{i-1}.l)$, i.e., $\mathbf{s}_i = [\mathbf{z}_i; \mathbf{v}(e_{i-1}.l)]$, where $\mathbf{z}_i$ is obtained from RSRNet and $\mathbf{v}(e_{i-1}.l)$ denotes a vector of the label on the previous road segment $e_{i-1}.l$ by embedding its token (i.e., 0 or 1). The rationale for the state design is to capture the features from three aspects, i.e., traffic context, normal routes and the previous label.

Actions. We denote an action of the MDP by a, which is to label each road segment as normal or not. Note that an anomalous subtrajectory boundary can be identified when the labels of two adjacent road segments are different.

Rewards. The reward involves two parts. One is called *local reward*, which aims to capture the local continuity of labels of road segments. The rationale is that normal road segments or anomalous road segments are usually continued for several times, and the labels would not change frequently. The second one is called *global reward*, which aims to indicate the quality of the refined labels (indicated by the cross-entropy loss of RSRNet). We take the loss as some feedback to guide the training of ASDNet.

The local reward is an intermediate reward, which encourages the continuity of the labels on road segments. Specifically, it is defined as

$$r_i^{local} = \text{sign}(e_{i-1}.l = e_i.l) \cdot \cos(\mathbf{z}_{i-1}, \mathbf{z}_i), \tag{2}$$

where the $\text{sign}(e_{i-1}.l = e_i.l)$ returns 1 if the condition $e_{i-1}.l = e_i.l$ is true and -1 otherwise; $\cos(\mathbf{z}_{i-1}, \mathbf{z}_i)$ denotes the consine similarity between $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$, which are obtained from RSRNet. We choose the consine similarity because it outputs a normalized value between 0 and 1, which corresponds to the same output range (i.e., between 0 and 1) of the global reward.

The global reward is designed to measure the quality of refined labels by ASDNet. To obtain the reward, we feed the refined labels into RSRNet, and compute the global reward as

$$r^{global} = \frac{1}{1+\mathcal{L}}, \tag{3}$$

where $\mathcal{L}$ denotes the cross-entropy loss in Equation 1. We notice this reward has the range between 0 and 1, and makes the objective of RSRNet well aligned with ASDNet (a smaller loss $\mathcal{L}$ means a larger global reward).

Policy Learning on the MDP. The core problem of an MDP is to learn a policy, which guides an agent to choose actions based on the constructed states such that the cumulative reward, denoted by R_n, is maximized. We learn the policy via a policy gradient method [22], called the REINFORCE algorithm [27]. To be specific, let $\pi_\theta(a|\mathbf{s})$ denote a stochastic policy, which is used to sample an action a for a given state $\mathbf{s}$ via a neural network, whose parameters are denoted by θ. Then, the gradients wrt the network parameters θ are estimated as

$$\nabla_\theta J(\theta) = \sum_{i=2}^{n} R_n \nabla_\theta \ln \pi_\theta(a_i|s_i), \tag{4}$$

which can be optimized using an optimizer (e.g., Adam stochastic gradient ascent). The expected cumulative reward R_n for a trajectory is defined as

$$R_n = \frac{1}{n-1}\sum_{i=2}^{n} r_i^{local} + r^{global}. \tag{5}$$

Joint Training of RSRNet and ASDNet. We jointly train the two networks (e.g., RSRNet and ASDNet). First, we map raw trajectories on road networks and generate the noisy labels as explained in Section 4.2. Then, we randomly sample 200 trajectories to pre-train the RSRNet and ASDNet, separately. In particular, for the RSRNet, we train the network in a supervised manner with the noisy labels. For the ASDNet, we specify its actions as the noisy labels, and train the policy network via a gradient ascent step as computed by Equation 4. The pre-training provides a warm-start for the two networks, where the parameters of the two networks have incorporated some information captured from the normal routes before joint training.

During the joint training, we randomly sample 10,000 trajectories, and for each trajectory, we generate 5 epochs to iteratively train the RSRNet and ASDNet. In particular, we apply the learned policy from ASDNet to refine the noisy labels, and the refined labels are used to train the RSRNet to obtain the better representation $\mathbf{z}_i$ on each road segment. Then, $\mathbf{z}_i$ is used to construct the state to learn better policy in ASDNet. Since the learned policy can further refine the labels, and the two networks are jointly trained with the best model is chosen during the process.

4.5 The RL4OASD Algorithm

Our RL4OASD algorithm is based on the learned policy for detecting anomalous subtrajectories. The process is presented in Algorithm 1. Specifically, RL4OASD accepts an ongoing trajectory in an online fashion for labeling each road segment as the normal (i.e., 0) or not (i.e., 1) (lines 1-9). It first labels the source or destination road segment as normal by definitions (lines 2-3). It then constructs a state via calling RSRNet (lines 5-6), and samples an action to label the road segment based on the learned policy in ASDNet (lines 7-8). Finally, it returns the anomalous subtrajectories which involve the abnormal labels (i.e., 1) on the road segments (line 10). We further develop two enhancements for boosting the effectiveness and efficiency of RL4OASD. One is called Road Network Enhanced Labeling (RNEL), and the other is called Delayed Labeling (DL).

Road Network Enhanced Labeling. For the RNEL, we utilize the graph structure of a road network to help labeling a road segment, where a label on a road segment is deterministic in one of three

Algorithm 1: The RL4OASD algorithm

Input: A map-matched trajectory $T =< e_1, e_2, ..., e_n >$ which is inputted in an online manner
Output: The anomalous subtrajectories

```
for i = 1, 2, ..., n do
    if i = 1 or i = n then
        e_i.l ← 0;
    else
        Call RSRNet for obtaining a representation z_i;
        Construct a state s_i = [z_i; v(e_{i-1}.l)];
        Sample an action (i.e., 0 or 1), a_i ~ π_θ(a|s);
        e_i.l ← a_i;
end
Return anomalous subtrajectories consisting of the road segments with the label 1;
```

cases: (1) If $e_{i-1}.out = 1$, $e_i.in = 1$, then $e_i.l = e_{i-1}.l$. (2) If $e_{i-1}.out = 1$, $e_i.in > 1$ and $e_{i-1}.l = 0$, then $e_i.l = 0$. (3) If $e_{i-1}.out > 1$, $e_i.in = 1$ and $e_{i-1}.l = 1$, then $e_i.l = 1$. Here, $e_i.out$ and $e_i.in$ denote the out degree and in degree of a road segment e_i, respectively, and $e_i.l$ denotes the label of the road segment e_i. The common intuition behind them is that: (a) any change of the anomalousness status from normal (0) at e_{i-1} to abnormal (1) at e_i means that there exist alternative transitions from e_{i-1} to other road segments (i.e., $e_{i-1}.out > 1$); and (b) any change of the anomalousness status from abnormal (1) at e_{i-1} to normal (0) at e_i means that there exist alternative transitions from other road segments to e_i (i.e., $e_i.in > 1$). Based on the rules, we only perform actions in the otherwise cases via the RL model, and some potential wrong decisions can therefore be avoided. In addition, the efficiency can be improved since the time of taking actions in some cases is saved via checking the rules instead of calling the RL model.

Delayed Labeling. For the DL, RL4OASD forms an anomalous subtrajectory whenever its boundary is identified, i.e., the boundary is identified at e_{i-1} if $e_{i-1}.l = 1$ and $e_i.l = 0$. Intuitively, it looks a bit rush to form an anomalous subtrajectory and may produce many short fragments from a detected trajectory. Therefore, we consider a delay technique as a post-processing step, it scans D more road segments that follow e_{i-1} when forming an anomalous subtrajectory. Among the D road segments, we choose the final position j ($i - 1 < j \leq i - 1 + D$) where the road segment is with label 1, and then convert some 0's to 1's between the position $i - 1$ and j. It could be verified the labeling with delay does not incur much time cost, and offers a better continuity to avoid forming too many fragments instead.

Time complexity. The time complexity of the RL4OASD algorithm is $O(n)$, where n denotes the length of a detected trajectory. The time is dominated by two networks, i.e., RSRNet and ASDNet. We analyze them as follows. In RSRNet, the time cost for one road segment consists of (1) that of obtaining embeddings of TCF and NRF, which are both $O(1)$ and (2) that of obtaining the $\mathbf{z}$ via a classic LSTM cell, which is $O(1)$. In ASDNet, the time cost for one road segment consists of (1) that of constructing a state, where the part $\mathbf{z}$ has been computed in RSRNet and the part $\mathbf{v}(e.l)$ is obtained via an embedding layer, which is $O(1)$ and (2) that of sampling an action via the learned policy, which is $O(1)$. As we can see in Algorithm 1, the two networks are called at most n times. Therefore, the time complexity of the RL4OASD is $O(n) \times O(1) = O(n)$. We note that the $O(n)$ time complexity maintains the current best time complexity as those of existing algorithms [3, 17, 28] for the trajectory or subtrajectory anomaly detection task, and can largely meet practical needs for online scenarios as shown in our experiments.

Table 1: Dataset statistics.

Dataset	Chengdu	Xi'an
# of trajectories	677,492	373,054
# of segments	4,885	5,052
# of intersections	12,446	13,660
# of labeled routes	1,688/558,098	1,057/163,027
# of anomalous routes	1,436/3,930	813/2,368
Anomalous ratio	0.7%	1.5%

5 EXPERIMENTS

5.1 Experimental Setup

Dataset. The experiments are conducted on two real-world taxi trajectory datasets from DiDi Chuxing [1], namely *Chengdu* and *Xi'an*. All raw trajectories are preprocessed to map-matched trajectories via a popular map-matching algorithm [29], and the road networks of the two cities are obtained from OpenStreetMap [2].

Ground Truth. By following the previous works [3, 18, 35], we manually label the anomalous subtrajectories, and take the labeled subtrajectories as the ground truth for the evaluation. Specifically, we randomly sample 200 SD-Pairs following the distribution of their traveled frequency in each dataset. For labeling subtrajectories, we illustrate all routes (i.e., map-matched trajectories) within each SD-Pair as shown in Figure 5. We invite 5 participants and let them identify whether the routes are anomalous or not based on the visualization. If one route is identified as anomalous, we ask the volunteer to label which parts are responsible for its anomalousness. For quality control, we randomly pick 10% trajectories, ask 5 other checkers to label these trajectories independently, adopt the majority voting to aggregate the labels, and compute the accuracy of the labels by the labelers against the aggregated ones by the checkers. The accuracy is 98.7% for Chengdu and 94.3% for Xi'an, which shows that our labeled datasets are with high accuracy.

Multiple trajectories may correspond to the same route and thus we have fewer routes than trajectories. We label 1,688 (resp. 1,057) routes, which correspond to 558,098 (resp. 163,027) raw trajectories before map-matching for Chengdu (resp. Xi'an). Among them, 1,436 (resp. 813) routes that correspond to 3,930 (resp. 2,368) raw trajectories are identified as the anomalous for Chengdu (resp. Xi'an), where the anomalous ratios for Chengdu and Xi'an are estimated as $3,930/558,098 = 0.7\%$ and $2,368/163,027 = 1.5\%$, respectively. The statistics of the two datasets are reported in Table 1.

Baseline. We review the literature thoroughly, and identify the following baselines for the online detection problem, including IBOAT [3], DBTOD [28], GM-VSAE [17], SD-VSAE [17], SAE [19], VSAE [11] and CTSS [34]. The detailed description of these algorithms is discussed in Section 2. For SAE and VSAE, they are adapted

[1] https://outreach.didichuxing.com/research/opendata/en/
[2] https://www.openstreetmap.org/

Table 2: Effectiveness comparison with existing baselines.

Mathods	Chengdu					Xi'an				
Trajectory Length	G1	G2	G3	G4	Overall	G1	G2	G3	G4	Overall
IBOAT [3]	0.426	0.431	0.406	0.590	0.431	0.424	0.467	0.487	0.503	0.472
DBTOD [28]	0.523	0.531	0.515	0.669	0.530	0.519	0.448	0.441	0.483	0.433
GM-VSAE [17]	0.375	0.493	0.504	0.658	0.451	0.371	0.464	0.498	0.521	0.467
SD-VSAE [17]	0.373	0.491	0.498	0.637	0.448	0.373	0.459	0.478	0.507	0.458
SAE [19]	0.375	0.492	0.499	0.658	0.450	0.372	0.460	0.479	0.517	0.461
VSAE [11]	0.374	0.492	0.504	0.656	0.450	0.369	0.461	0.487	0.520	0.463
CTSS [34]	0.730	0.708	0.625	0.741	0.706	0.657	0.637	0.636	0.672	0.658
RL4OASD	**0.888**	**0.892**	**0.725**	**0.774**	**0.854**	**0.964**	**0.864**	**0.809**	**0.844**	**0.857**

from GM-VSAE, where SAE replaces the encoder and decoder structure in GM-VSAE with a traditional Seq2Seq model, which aims to minimize a reconstruction error, and the reconstruction error is further used to define the anomaly score. VSAE replaces the Gaussian mixture distribution of the latent route representations in GM-VSAE with a Gaussian distribution.

Parameter Setting. In RSRNet, we embed the TCF and NRF features into the 128-dimensional vectors, and use the LSTM with 128 hidden units to implement the RSRNet. To train RSRNet, we construct noisy labels in 24 time slots with one hour granularity via empirical studies. The parameter α and δ are set to 0.5 and 0.4 by default. In ASDNet, the dimension of label vectors $\mathbf{v}(e_i.l)$ is 128. The policy network is implemented with a single-layer feedforward neural network, and the softmax function is adopted as the activation function. The delay parameter D is set to 8 by default. By empirical findings, the learning rates for RSRNet and ASDNet are set to 0.01 and 0.001, respectively. For baselines, we follow their strategies in the original papers.

Evaluation Metrics. To verify the results of subtrajectory detection, we adapt the evaluation metric F_1-score proposed for Named Entity Recognition (NER) [13, 14]. This is inspired by the fact that our task corresponds to one of tagging subsequences of a sequence, which is similar to NER, which tags phrases (i.e., subsequences) of a sentence (sequence). The intuition is that we take the anomalous subtrajectories as entities in NER task. Specifically, (1) let $C_{g,i}$ denote a manually labeled anomalous subtrajectory i in the ground truth set C_g, and $C_{o,i}$ denotes the corresponding subtrajectory i in the set C_o, which is returned by a detection method. We employ Jaccard to measure the ratio of intersection over union of the road segments between $C_{g,i}$ and $C_{o,i}$. (2) We then measure the similarity between C_g and C_o by aggregating the Jaccard $\mathcal{J}_i(C_{g,i}, C_{o,i})$.

$$\mathcal{J}_i(C_{g,i}, C_{o,i}) = \frac{|C_{g,i} \cap C_{o,i}|}{|C_{g,i} \cup C_{o,i}|}, \ \mathcal{J}(C_g, C_o) = \sum_{i=1}^{|C_g|} \mathcal{J}_i(C_{g,i}, C_{o,i}). \quad (6)$$

(3) Finally, we define the precision (P) and recall (R) by following [13, 14], and compute the F_1-score accordingly.

$$\mathrm{P} = \frac{\mathcal{J}(C_g, C_o)}{|C_o|}, \quad \mathrm{R} = \frac{\mathcal{J}(C_g, C_o)}{|C_g|}, \quad F_1 = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}. \quad (7)$$

We notice existing solutions [3, 17, 28, 34] output an anomaly score on each point in a detected trajectory. To support this for subtrajectory detection, we turn their thresholds of the anomaly scores in a development set (i.e, a set of 100 trajectories with manual labels), and for each turned threshold, the anomalous subtrajectories C_o is identified as those road segments, whose anomaly scores are larger than the threshold. and the threshold that is associated with the highest F_1-score is selected for experiments. For RL4OASD, it naturally outputs the anomalous subtrajectories, and we can compute its F_1-score based on the defined precision and recall.

Table 3: Ablation study for RL4OASD.

Effectiveness	F_1-score
RL4OASD	**0.854**
w/o noisy labels	0.626
w/o road segment embeddings	0.828
w/o RNEL	0.816
w/o DL	0.737
w/o local reward	0.850
w/o global reward	0.849

Evaluation Platform. We implement RL4OASD and other baselines in Python 3.6 and Tensorflow 1.8.0. The experiments are conducted on a server with 10-cores of Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz 64.0GB RAM and one Nvidia GeForce RTX 2080 GPU. The labeled datasets and codes can be downloaded via the link [3] to reproduce our work.

5.2 Experimental Results

(1) Effectiveness evaluation (comparison with existing baselines). We study the anomalous subtrajectory detection with our labeled datasets. In Table 2, we report the effectiveness in terms of different trajectory lengths, e.g., we manually partition the Chengdu dataset into four groups, i.e., G1, G2, G3 and G4, such that the lengths in a groups are $G1 < 15$, $15 \leq G2 < 30$, $30 \leq G3 < 45$ and $G4 \geq 45$. In addition, we also report the overall effectiveness in whole datasets. The results clearly show that RL4OASD consistently outperforms baselines in terms of different settings. For example, it outperforms the best baseline (i.e., CTSS) for around 20% (resp. 30%) in Chengdu (resp. Xi'an) in terms of the overall effectiveness, which verifies its data-driven nature.

(2-4) Effectiveness evaluation (varying parameter α, δ, D). We conduct the experiments to show the effects of parameter α for constructing noisy labels, parameter δ for the number of selected normal routes, and parameter D for the number of road segments in the delayed labeling. The detailed results and description can be found in the technical report [36] due to the page limit. We conclude the result of the parameters as: the effectiveness improves

[3] https://www.dropbox.com/sh/imix3otoep49v0g/AACQLPpgA0jMdTg7LV-GdL70a?dl=0

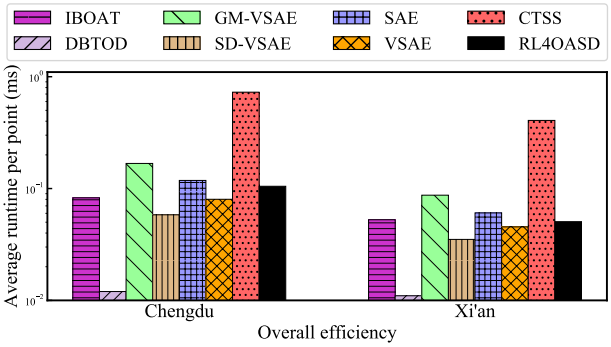
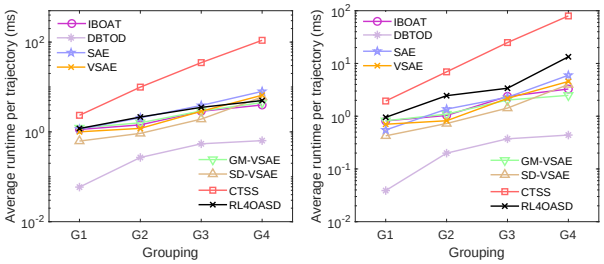


Figure 3: Overall detection efficiency.



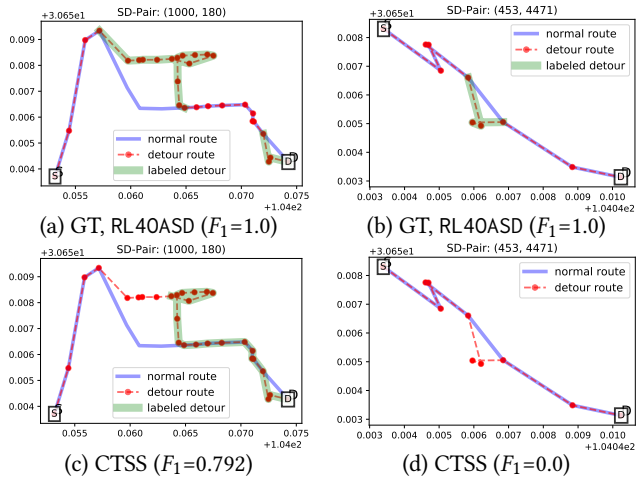
(a) Chengdu (b) Xi'an

Figure 4: Efficiency with varying trajectory lengths.

as the parameter increases and degrades as the parameter further increases. Overall, a moderate setting (i.e., $\alpha = 0.5$, $\delta = 0.4$ and $D = 8$) provides the best effectiveness.

(5) Effectiveness evaluation (ablation study). We conduct an ablation study to show the effects of some components in RL4OASD, including (1) noisy labels, (2) pre-trained road segment embeddings, (3) Road Network Enhanced Labeling (RNEL), (4) Delayed Labeling (DL), (5) local reward and (6) global reward. In particular, for (1), we replace the noisy labels with random labels; for (2), we randomly initialize the embeddings of road segments to replace the pre-trained embeddings provided by Toast [4]; for (3), we drop the RNEL, and the model needs to take the action at each road segment without guided by road networks; for (4), we drop the DL, and no delay mechanism is involved for labeling road segments; for (5), we drop the local reward, which encourages the continuity of refined labels; for (6), we drop the global reward, which provides some feedback indicating the quality of refined labels. In Table 3, we observe each component benefits the overall effectiveness, where the noisy labels contribute quite much, because it provides a necessary warm-start before the training. With random labels, the model converges to a local optimum and it is difficult to obtain further optimization. In addition, with the RNEL, we notice an effectiveness improvement of around 5%, since it simplifies the cases of making decisions and the model becomes easier to train. In addition, we notice an efficiency improvement of around 9% of RNEL, because it saves the time of taking actions via neural networks.

(6) Efficiency evaluation (overall detection). Figure 3 reports the online detection efficiency in terms of average running time per point. We observe DBTOD runs the fastest on both datasets, because it is a light model with cheaper feature engineering efforts for a binary classification problem on detecting whether an anomaly is contained in a trajectory. CTSS runs the slowest since it involves



(a) GT, RL4OASD (F_1=1.0) (b) GT, RL4OASD (F_1=1.0)

(c) CTSS (F_1=0.792) (d) CTSS (F_1=0.0)

Figure 5: Case study, blue lines: the normal routes traveled by most of trajectories; red dashed lines: the routes with anomalous subtrajectories; red points: the intersections on road networks; green lines: the detected anomalies.

discrete Frechet distance to compute the deviation between a detected trajectory and a given reference trajectory, which suffers from quadratic time complexity. Overall, RL4OASD runs reasonably fast and would meet the practical needs, e.g., it takes less than 1ms to process each point, which is much faster than the practical sampling rate of trajectory data.

(7) Efficiency evaluation (detection scalability). Further, we study the scalability for detecting ongoing trajectories in terms of different trajectory lengths. In Figure 4, we report the average running time per trajectory on both datasets. In general, we notice CTSS performs the worst, and its disparity between other baselines becomes larger as the trajectory length increases. RL4OASD scales well with the trajectory length grows.This is because it involves the graph structure of road networks for labeling road segments, and the time of taking actions is saved accordingly.

(8) Case study. In Figure 5, we investigate two representative detour cases in Chengdu, including the cases of two detours, and a single detour in a detected trajectory. We visualize the detours with the green lines labeled as Ground Truth (GT), CTSS and RL4OASD, where we choose CTSS for comparison, since it shows the best effectiveness among baselines. The results clearly show that RL4OASD can accurately identify detours, which are the same as GT and the returned detours are in line with human intuition. It demonstrates the capability of RL4OASD for real applications. More results are illustrated in the technical report [36] due to the page limit.

6 CONCLUSIONS

In this paper, we propose a novel deep reinforcement learning-based method called RL4OASD for online anomalous subtrajectory detection. RL4OASD is trained in a weakly supervised manner with noisy labels and powered with reinforcement learnng. Experiments on two real-world taxi trajectory datasets demonstrate that RL4OASD consistently outperforms existing algorithms, and runs comparably fast. In the future, we will explore the subtrajectory anomaly detection task in an offline scenario, where the whole trajectory is accessible during the detection process.

REFERENCES

[1] Prithu Banerjee, Pranali Yawalkar, and Sayan Ranu. 2016. Mantra: a scalable approach to mining temporally anomalous sub-trajectories. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1415–1424.

[2] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, and Shijian Li. 2011. Real-time detection of anomalous taxi trajectories from GPS traces. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 63–74.

[3] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, Shijian Li, and Zonghui Wang. 2013. iBOAT: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems* 14, 2 (2013), 806–818.

[4] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Chandran, and Richard Ellison. 2021. Robust Road Network Representation Learning: When Traffic Patterns Meet Traveling Semantics. (2021).

[5] Qi Fan, Dongxiang Zhang, Huayu Wu, and Kian-Lee Tan. 2016. A general and parallel platform for mining co-movement patterns over large-scale trajectories. *Proceedings of the VLDB Endowment* 10, 4 (2016), 313–324.

[6] Yong Ge, Hui Xiong, Zhi-hua Zhou, Hasan Ozdemir, Jannite Yu, and Kuo Chu Lee. 2010. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. 1733–1736.

[7] Kathryn Gray, Daniel Smolyak, Sarkhan Badirli, and George Mohler. 2018. Coupled igmm-gans for deep multimodal anomaly detection in human mobility data. *arXiv preprint arXiv:1809.02728* (2018).

[8] Jiawei Han, Micheline Kamber, and Data Mining. 2006. Concepts and techniques. *Morgan Kaufmann* 340 (2006), 94104–3205.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Tobias Skovgaard Jepsen, Christian S Jensen, and Thomas Dyhre Nielsen. 2019. Graph convolutional networks for road networks. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 460–463.

[11] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[12] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 140–149.

[13] Fei Li, Zheng Wang, Siu Cheung Hui, Lejian Liao, Dandan Song, and Jing Xu. 2021. Effective Named Entity Recognition with Boundary-aware Bidirectional Neural Networks. In *Proceedings of the Web Conference 2021*. 1695–1703.

[14] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[15] Xiaolei Li, Zhenhui Li, Jiawei Han, and Jae-Gil Lee. 2009. Temporal outlier detection in vehicle traffic data. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 1319–1322.

[16] Marco Lippi, Matteo Bertini, and Paolo Frasconi. 2010. Collective traffic forecasting. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 259–273.

[17] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online anomalous trajectory detection with deep generative sequence modeling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 949–960.

[18] Zhongjian Lv, Jiajie Xu, Pengpeng Zhao, Guanfeng Liu, Lei Zhao, and Xiaofang Zhou. 2017. Outlier trajectory detection: A trajectory analytics based approach. In *International Conference on Database Systems for Advanced Applications*. Springer, 231–246.

[19] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148* (2016).

[20] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. 2007. Similarity search in trajectory databases. In *14th International Symposium on Temporal Representation and Reasoning (TIME'07)*. IEEE, 129–140.

[21] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[22] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. PMLR, 387–395.

[23] Li Song, Ruijia Wang, Ding Xiao, Xiaotian Han, Yanan Cai, and Chuan Shi. 2018. Anomalous trajectory detection using recurrent neural network. In *International Conference on Advanced Data Mining and Applications*. Springer, 263–277.

[24] Eleftherios Tiakas, Apostolos N Papadopoulos, Alexandros Nanopoulos, Yannis Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. 2006. Trajectory similarity search in spatial networks. In *2006 10th International Database Engineering and Applications Symposium (IDEAS'06)*. IEEE, 185–192.

[25] Andreas Tritsarolis, George-Stylianos Theodoropoulos, and Yannis Theodoridis. 2021. Online discovery of co-movement patterns in mobility data. *International Journal of Geographical Information Science* 35, 4 (2021), 819–845.

[26] Meng-xiang Wang, Wang-Chien Lee, Tao-yang Fu, and Ge Yu. 2019. Learning embeddings of intersections on road networks. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 309–318.

[27] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[28] Hao Wu, Weiwei Sun, and Baihua Zheng. 2017. A fast trajectory outlier detection approach via driving behavior modeling. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 837–846.

[29] Can Yang and Gyozo Gidofalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science* 32, 3 (2018), 547–570.

[30] Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu, and Edward Y Chang. 2017. A neural network approach to jointly modeling social networks and mobile trajectories. *ACM Transactions on Information Systems (TOIS)* 35, 4 (2017), 1–28.

[31] Yanwei Yu, Lei Cao, Elke A Rundensteiner, and Qin Wang. 2014. Detecting moving object outliers in massive-scale trajectory streams. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 422–431.

[32] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 316–324.

[33] 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 *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*. 99–108.

[34] Dongxiang Zhang, Zhihao Chang, Sai Wu, Ye Yuan, Kian-Lee Tan, and Gang Chen. 2020. Continuous Trajectory Similarity Search for Online Outlier Detection. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[35] Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li. 2011. iBAT: detecting anomalous taxi trajectories from GPS traces. In *Proceedings of the 13th international conference on Ubiquitous computing*. 99–108.

[36] Qianru Zhang, Zheng Wang, Cheng Long, Siu Ming Yiu, Yiding Liu, Gao Cong, and Jieming Shi. 2022. Online Anomalous Subtrajectory Detection on Road Networks with Deep Reinforcement Learning (technical report). https://zhengwang125.github.io/paper/OASD-TR.pdf.

[37] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. 2015. Time-dependent popular routes based trajectory outlier detection. In *International Conference on Web Information Systems Engineering*. Springer, 16–30.