DeepTEA: Effective and Efficient Online Time-dependent Trajectory Outlier Detection

Xiaolin Han
The University of Hong Kong
Hong Kong SAR, China
xlhan@cs.hku.hk

Reynold Cheng
The University of Hong Kong
Hong Kong SAR, China
ckcheng@cs.hku.hk

Chenhao Ma*
The University of Hong Kong
Hong Kong SAR, China
chma2@cs.hku.hk

Tobias Grubenmann
University of Bonn
Bonn, Germany
grubenmann@cs.uni-bonn.de

ABSTRACT

In this paper, we study anomalous trajectory detection, which aims to extract abnormal movements of vehicles on the roads. This important problem, which facilitates understanding of traffic behavior and detection of taxi fraud, is challenging due to the varying traffic conditions at different times and locations. To tackle this problem, we propose the **deep**-probabilistic-based **t**ime-d**e**pendent **a**nomaly detection algorithm (**DeepTEA**). This method, which employs deep-learning methods to obtain time-dependent outliners from a huge volume of trajectories, can handle complex traffic conditions and detect outliners accurately. We further develop a fast and approximation version of DeepTEA, in order to capture abnormal behaviors in real-time. Compared with state-of-the-art solutions, our method is 17.52% more accurate than seven competitors on average, and can handle millions of trajectories.

PVLDB Reference Format:
Xiaolin Han, Reynold Cheng, Chenhao Ma, and Tobias Grubenmann. DeepTEA: Effective and Efficient Online Time-dependent Trajectory Outlier Detection. PVLDB, 15(7): 1493 - 1505, 2022.
doi:10.14778/3523210.3523225

1 INTRODUCTION

Due to the popularity of location sensing devices (e.g., GPS) used in mobile phones and vehicles, it is now possible to collect a huge amount of *trajectories*, or the movement information of the vehicles in their trips. In this paper, we study how to use *trajectory data* to detect *trajectory outliers* (or *outliers* in short). Given two points S and D, an outlier is a trajectory that deviates significantly from commonly-used routes. For example, in Figure 1(a), trajectory r_1 is frequently employed by drivers. Trajectory r_2, which deviates significantly from r_1, is an outlier.

The extraction of outliers from trajectories is an important problem in urban computing [1, 23, 25, 37, 42]. This problem is often used to discover abnormal driving behaviors. For example, taxi and ride-sharing companies may wish to find out drivers who are taking anomalous routes while they are carrying passengers, and examine whether they have committed taxi driving fraud, i.e., overcharging passengers through unnecessary detours [1, 5, 37, 42].

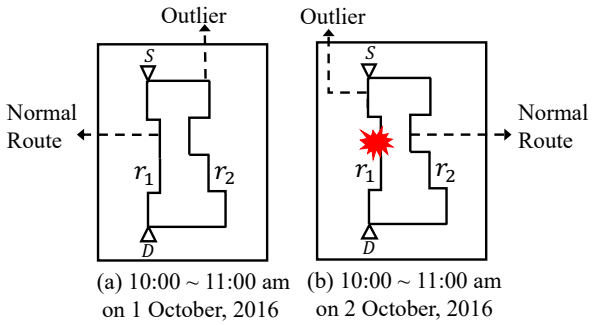

Figure 1: Time-dependent outlier : during (a) 10:00 ~ 11:00 am on 1 October, 2016; (b) 10:00 ~ 11:00 am on 2 October, 2016, when an incident (marked red) happens at route r_1.

In the literature, most trajectory outlier solutions are *shape-based* [1, 18, 23, 25, 30, 37, 42]. Essentially, the shape of the trajectories in the road network is used to judge whether a trajectory is an outlier. In Figure 1(a), because the shape of trajectory r_2 deviates from the majority of trajectories between points S and D, which has the "shape" of r_1, those solutions consider r_2 as an outlier. The problem of this approach is that it does not consider the fact that outliers can change over time [44, 45]. This happens, for instance, when normal routes (i.e., non-outliers) become congested, forcing vehicles to switch to another usually-less-popular route. In Figure 1(b), r_1 is congested between 6:00 and 7:00 pm, due to an incident at a road traversed by r_1. Hence, vehicles took another path, r_2, which is less congested. Note that during 6:00 and 7:00 pm, r_1 is an outlier, while r_2 is not. In this sense, trajectory outliers are *time-dependent* [44, 45].

Compared with shape-based outlier detection, the problem of finding *time-dependent* trajectories is relatively new, and is only addressed by few works [44, 45]. These solutions consider different factors that affect the time-dependent normal routes, e.g., incidents, roadblocks, and rush hours. Then, distance-based measures

*Chenhao Ma is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 15, No. 7 ISSN 2150-8097.
doi:10.14778/3523210.3523225

were employed to differentiate anomalous trajectories from normal routes – intuitively, an outlier is not similar to normal trajectories if its "distance" from normal trajectories is larger than a certain threshold. This method has two shortcomings. First, it is crucial to set a good value for the threshold, in order to yield accurate results. However, finding such a value of threshold can be difficult, because it can depend on the traveling source and destinations. Second, these solutions, which compute normal routes and identify outliers based on distance measures, have a high time complexity. This hinders their use in online outlier detection [1, 8, 17, 23, 37], which requests anomalous behaviors to be detected in a real-time manner. These call for a more *effective* and *efficient* time-dependent trajectory outlier detection solution.

Our contributions. Our first goal is to develop an effective time-dependent outlier detection. This is not easy, because the traffic conditions are dynamic and diverse among different locations at different times. Even for the same source and destination points of a trajectory, the "normal" trajectories may vary across different times due to the evolving traffic conditions. We tackle this challenge by developing a deep probabilistic neural network, called DeepTEA. The novelty of our paper is that we can capture the dynamic traffic conditions by deriving the latent traffic pattern from the real-time traffic condition during the traveling time, which is non-trivial. We maximize the likelihood of observing the trajectories under the dynamic traffic conditions by optimizing the evidence lower bound (ELBO). The latent traffic pattern is derived to capture the dynamic traffic conditions, e.g., {smooth → congested}. Then, the time-dependent normal trajectories are obtained based on the derived latent traffic pattern.

Second, we study how to improve the efficiency of the time-dependent outlier detection model in order to allow companies or passengers to be alerted online by abnormal driving behaviors during a journey [1, 8, 17, 23, 37]. Because time-dependent outliers can be affected by traffic conditions, which are changing with time, we need to recompute time-dependent outliers quickly in response to the changes of traffic. However, this is challenging, because trajectories are typically updated frequently, e.g., 2 - 4 seconds [1], and so the outliers need to be computed within a short interval. Here we make use of a generative network based on the latent traffic pattern learnt, and maximize the generation of the observed trajectories under the latent traffic pattern. The trajectories which cannot be generated under the latent traffic pattern are marked as time-dependent outliers. We further propose an approximate online detection algorithm (DeepTEA-A), where the latent representation of the time-dependent route is approximated by a co-training neural network given the source, destination locations, and traffic condition at the departure time. This method avoids frequent updates of latent representation, and only requires constant time cost for detecting time-dependent outliers in an online mode.

We have performed substantial experiments on two real trajectory datasets against seven existing methods. Compared with the best time-dependent state-of-the-art method, our model is 19% more effective and 32 times faster. Our solution is also scalable on large trajectory datasets.

[1] https://outreach.didichuxing.com/appEn-vue/XiAnOct2016?id=8

The rest of this paper is organized as follows. In Section 2, we present the formal problem definition. In Section 3, we propose DeepTEA. Section 4 discusses how to detect anomalies in an online manner. We further propose an approximate algorithm to speed up the online detection process in Section 5. Section 6 shows the evaluation results. Section 7 discusses related work. We conclude in Section 8.

2 PROBLEM DEFINITION

In this section, we first present definitions of the basic concepts. Then, we give a formal problem definition.

Definition 1 (Trajectory). *A point p_{t_i} is a three-tuple (t_i, x, y) that contains a timestamp t_i, the latitude x and longitude y of its current position. A trajectory T is an ordered sequence of points $\langle p_{t_1}, \cdots, p_{t_i}, \cdots, p_{t_n} \rangle$ where $t_1 < \ldots < t_i < \ldots < t_n$.* □

Based on whether the time-dependent sense is considered, the trajectory outliers can be categorized into two groups: 1) the non-time-dependent trajectory outlier [1, 18, 23, 25, 37, 42] is the anomalous trajectory that only considers divergence from normal routes; and 2) the time-dependent trajectory outlier [44, 45] is the anomalous trajectory that diverges from time-dependent normal routes.

Definition 2 (Time-dependent Trajectory Outlier [44, 45]). *Given a trajectory T, the source S_T, and destination D_T of the trajectory, and its travel time, a time-dependent trajectory outlier is the one that rarely occurs and is different from other trajectories during its travel time, i.e., starting at S_T at the same departure time and arriving at destination D_T at the same time.* □

For example, if one trajectory starts at 10:00 am and arrives at 11:00 am on 1 October 2016 in Figure 1 (a), then it is defined as an outlier if it rarely occurs and is different from other trajectories during its travel time, i.e., 10:00 ~ 11:00 am on 1 October 2016, with the same source S_T and destination D_T.

In this work, we focus on detecting time-dependent trajectory outliers in an online manner.

Problem 1 (Online Time-dependent Trajectory Outlier Detection). *Given an ongoing trajectory T being generated, the online detection computes and updates the probability of T being a time-dependent trajectory outlier in real-time.* □

3 THE DEEPTEA MODEL

In this section, we first introduce the framework of DeepTEA. After that, we explain each component of the model in detail. The key idea is to learn the latent patterns from both real-time traffic conditions and the trajectory transition. Then, we show the optimization function which is utilized to maximize the likelihood of trajectory observations. Finally, we discuss the time complexity of DeepTEA.

3.1 Framework

Figure 2 shows the framework of DeepTEA. It contains an inference network and a generation network. Given the trajectory T, we infer the latent traffic pattern $q(z|T)$ during the travel time (Section 3.2). The trajectory observation τ, which reflects time-dependent trajectory transition, is utilized to model the latent time-dependent

route r in the inference network (Section 3.3). After that, the time-dependent route r is used to generate the trajectory observation τ (Section 3.4).

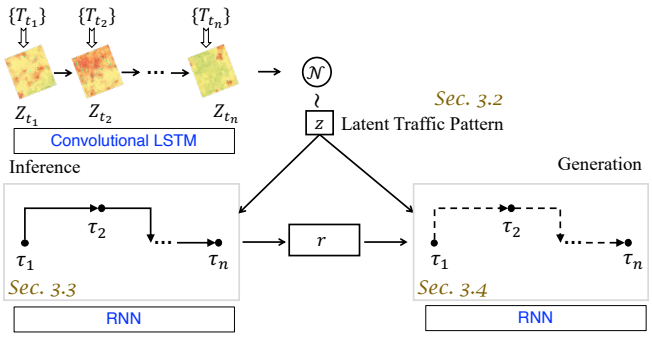

Figure 2: Overview of DeepTEA. The notation $\mathcal{N}$ denotes the Gaussian distribution. And the dashed line represents the process of trajectory generation.

3.2 Latent Traffic Pattern Inference

Before we derive the latent traffic pattern, we first explain the meaning of the latent traffic pattern z. It indicates the dynamic traffic conditions, e.g., {smooth → congested → smooth}, or {congested → smooth}, during the trip. Then we introduce the challenges and the algorithm design.

3.2.1 ***Challenges.*** Given the trajectory T, we intend to infer the latent traffic pattern $q(z|T)$ based on the spatial transition of T, e.g., a large step forward transition means the traffic condition is smooth at the travel time. However, a single trajectory T may express some random behaviors, e.g., stop for a relaxation, which could not represent the real traffic pattern. This is the first challenge about how to represent the real traffic pattern. The second challenge is that the traffic condition varies across different locations, and updates dynamically as time changes. Especially, traffic conditions for different source and destination pairs are very different at a certain time. Moreover, the traffic condition changes dynamically during the whole trip. It can be smooth at the beginning and then turns congested at the end of the trip. It is essential to capture the traffic condition since it affects time-dependent normal routes significantly.

3.2.2 ***Design.*** To address the first challenge, instead of learning the latent traffic pattern z from a single trajectory T, we propose to learn it from the set of trajectories $\{T_{t_i}\}$ at the time t_i. Here we use the time point series to represent the travel time. To organize the traffic information from $\{T_{t_i}\}$ well, we propose to use a map grid matrix Z_{t_i} with the average speed in each grid cell to model the traffic condition at t_i. As we can see from Figure 2, the red color in the grid matrix means low average speed, and the traffic is congested at these locations. The green color denotes high average speed, and the traffic is smooth. While the yellow color means that the traffic condition is going to be congested, and the average speed is lower than the green cells but still higher than the red cells. To address the problem of traffic diversity across different locations at a certain time, we propose to capture it by a CNN model. For the locations without vehicles, CNN model can learn missing values from cells with vehicles for them, instead of representing them as missing cells. As traffic condition changes frequently in real-time, we use the RNN model to capture the ever-changing traffic dynamics in order to fix the second challenge. In this way, the transition behind the traffic condition can be well captured by the RNN model. Then we use the hidden state of the RNN model to infer the latent traffic pattern z with a Gaussian distribution. The combination of CNN and LSTM we used is Convolutional LSTM [39].

Specifically, we use the dynamic traffic conditions that change over time to infer the latent traffic pattern z. The intuition is that traffic condition changes rapidly due to complex real-time factors, e.g., traffic light, incidents, and peak hours. Therefore, we update Z as time changes in DeepTEA, and denote it as Z_{t_i}, which represents the traffic condition that the trajectory point p_{t_i} is facing at time t_i. Particularly, we infer the latent traffic pattern z from the real traffic conditions $Z = \{Z_{t_i}, Z_{t_{i+1}}, \cdots, Z_{t_{i+n}}\}$ at the time that the trajectory points $\{p_{t_i}, p_{t_{i+1}}, \cdots, p_{t_{i+n}}\}$ face. For real traffic condition Z, we obtain the average speed at the time that the trajectory T traversed. In other words, real traffic condition Z_{t_i} is an average speed matrix covering the moving condition of the whole city at t_i. Note that Z_{t_i} and $Z_{t_{i+1}}$ may be very similar if the time gap of two trajectory points is very short. In this way, we gather the average speed over time intervals, e.g., 10 minutes, instead of time points. To alleviate the traffic sparsity issue, we propose to apply a CNN to propagate traffic conditions from locations with vehicles to locations with missing data. To capture the dynamic traffic conditions with different traveling times, we utilize the Recurrent Neural Network (RNN) to model the traffic transition in the temporal dimension. In this way, both spatial traffic correlation and temporal transition of traffic dynamics can be captured by $f_1(Z)$ as follows:

$$f_1(Z) = \text{RNN}(\text{CNN}(Z)), \tag{1}$$

where function $f_1(\cdot)$ is modeled by a CNN and an RNN. The CNN model is applied for each Z_{t_i}, and then the RNN model is utilized to learn the traffic transition as time changes.

To maintain the capability of generation and model the uncertainty of traffic condition, we model the latent traffic pattern z given the real traffic condition Z by following the Gaussian distribution, which can be utilized to approximate the distribution of latent traffic pattern z given the trajectory T as Equation 2. We denote the parameters for learning the latent traffic pattern as the set ϕ.

$$q_\phi(z|T) \;:=\; q_\phi(z|Z) = \mathcal{N}(\mu_Z, \text{diag}(\sigma_Z^2)), \tag{2}$$

where the mean μ_Z and the standard deviation σ_Z are learned by a Multilayer Perceptron (MLP) function $g_1(f_1(Z))$ during the training phase. Here the parameter set $\phi = \{f_1(\cdot), g_1(\cdot)\}$.

In this way, the latent traffic pattern z can be well inferred from the trajectory T. In the training phase, the parameters in ϕ can be learned to capture the latent traffic pattern z, which represents the traffic diversity and dynamics given the trajectory T.

3.3 Latent Time-dependent Route Inference

3.3.1 ***Challenges***. The trajectory T can represent not only the location information, but also the latent traffic pattern z between the transition of two trajectory points. Instead of only maximizing the likelihood of location information, the combination of location and latent traffic pattern z is more informative as it can reflect the trajectory transition under time-dependent traffic conditions.

3.3.2 ***Design***. A trajectory T can not only reflect its location p_{t_i}, but also convey the latent traffic pattern z based on the transition between two consecutive trajectory points $p_{t_{i-1}}$ and p_{t_i}. Here we use $o\left(p_{t_i}, z\right)$ to denote the observation τ_i behind the trajectory T. It is challenging to model the combination of the observation p_{t_i} and z from the trajectory T. We seek the help of a neural network to handle the combination of the observation as:

$$\tau_i = o\left(p_{t_i}, z\right) = f_2(p_{t_i}, z) = \mathrm{NN}(p_{t_i}, z)\,, \tag{3}$$

We apply a neural network to learn the observation of p_{t_i} at the latent traffic pattern z as:

$$\mathrm{NN}(p_{t_i}, z) = W p_{t_i} + Q z\,, \tag{4}$$

where $\mathrm{NN}(\cdot)$ is a function with W and Q as parameters to be learned during the training phase.

Next, we learn the latent time-dependent route r that a trajectory T traverses. As we explained, the trajectory T can not only represent the location information p_{t_i}, but also indicate the latent traffic pattern z from the transition between two trajectory points $p_{t_{i-1}}$ and p_{t_i}. For the meaning of the latent time-dependent route r, it could be that drivers usually turn to the *highway*, which is *smooth* in rush hours, when the traffic condition is *congested* in the urban roads.

The representation of the latent time-dependent route r that trajectory T traversed can be represented as :

$$r_T \sim q_\gamma(r|T)\,, \tag{5}$$

where γ represents the parameters for inferring the latent time-dependent route r.

Based on previous trajectory points and the latent traffic pattern z, we obtain the transition of the trajectory observation by an RNN neural network denoted as f_3. Here the transition of the trajectory observation is modeled by the RNN model, i.e., Gated Recurrent Unit GRU [2].

$$h_i = f_3(h_{i-1},\ \tau_i)\,, \tag{6}$$

where h_{i-1} is the hidden state of previous observation τ_{i-1}, i.e., trajectory point $p_{t_{i-1}}$ with the latent traffic pattern z.

For the uncertainty of trajectory observations, we model $q_\gamma(r|T)$ by following Gaussian distribution with $\{\mu_T, \sigma_T\}$ as mean and standard deviation.

$$q_\gamma(r|T) = \mathcal{N}(\mu_T, \mathrm{diag}(\sigma_T^2))\,, \tag{7}$$

where we use a neural network $g_3(h_n)$ to learn the mean and standard deviation $\{\mu_T, \sigma_T\}$.

To distinguish the normal transition of trajectories under latent traffic pattern from abnormal ones, we should design a component to model the latent time-dependent normal route from trajectories, in which the latent traffic pattern z provides time-dependent traffic information. Inspired by [23], we utilize Gaussian distribution to model it as:

$$p_\gamma(r|k, z) = \mathcal{N}(\mu_r, \mathrm{diag}(\sigma_r^2))\,, \tag{8}$$

where k denotes the latent time-dependent route type, and it follows a multinomial distribution as:

$$p_\gamma(k) = \mathrm{Mult}(\pi)\,, \tag{9}$$

where π is the parameter of the multinomial distribution. Then, the latent time-dependent routes that are close to the means of $q_\gamma(r|T)$ are time-dependent normal routes.

Then the inference network can infer the latent time-dependent route r, the latent time-dependent route type k, and the latent traffic pattern z from the trajectory T as $q_{\gamma,\phi}(r, k, z|T)$. By applying mean-field approximation, it can be factorized as:

$$q_{\gamma,\phi}(r, k, z|T) = q_\gamma(r|T)\ q_\phi(z|T)\ q_\gamma(k|T)\,, \tag{10}$$

where $q_\gamma(k|T)$ can be transformed as the distribution of route type k under the condition of the latent time-dependent route r that trajectory T traversed:

$$q_\gamma(k|T) := p_\gamma(k|r_T) = \frac{p_\gamma(k)\ p_\gamma(r_T|k)}{\sum_{i=1}^{K} p_\gamma(k_i)\ p_\gamma(r_T|k_i)}\,, \tag{11}$$

where K is a hyperparameter that indicates the number of route types.

Hence, the inference network can infer the latent time-dependent route r from the observation $o\left(p_{t_i}, z\right)$ of the trajectory T. The parameters $\gamma = \{f_2(\cdot), f_3(\cdot), g_3(\cdot), \pi, \mu_r, \sigma_r\}$ for inferring the latent time-dependent route can be learned in the training phase.

3.4 Trajectory Observation Generation

The goal of trajectory observation generation is to maximize the probability of generating trajectory observation τ_i, i.e., $o\left(p_{t_i}, z\right)$, based on its inferred latent time-dependent route r, the time-dependent route type k, and the latent traffic pattern z. We formalize the probability as $p_\theta(T|r, z, k)$, where θ represents the parameters for generation. Symmetrically, we design an RNN neural network to generate the trajectory observation τ_i, i.e., $o\left(p_{t_i}, z\right)$, of T as :

$$\begin{aligned} \eta_i &= f_4(\tau_i,\ \eta_{i-1}) = f_4(o\left(p_{t_i}, z\right),\ \eta_{i-1}) \\ &= \mathrm{RNN}(o\left(p_{t_i}, z\right),\ \eta_{i-1})\,, i = 1, 2, \cdots, n,\ \textit{and}\ \eta_0 = r, \end{aligned} \tag{12}$$

where the initial input of the RNN is η_0, which follows $q_\gamma(r|T)$. Starting from η_1, the inputs become the last hidden state η_{i-1} and the trajectory observation $o\left(p_{t_i}, z\right)$. Therefore, the observation τ_i, i.e., p_{t_i} at the latent traffic pattern z, can be generated as:

$$\begin{aligned} \tau_i &= o\left(p_{t_i}, z\right) \sim p_\theta(o\left(p_{t_i}, z\right)|\ o\left(p_{t_{1:i-1}}, z\right), r) \\ &= p_\theta(\tau|\eta_{i-1}) = \mathrm{Mult}(\mathrm{softmax}(g_4(\eta_{i-1}))), \end{aligned} \tag{13}$$

where $g_4(\cdot)$ is a function to map the output to the size of the map grid. The softmax function is used to make the sum of the probability to one. After that, the trajectory observation τ_i can be generated from the multinomial distribution.

Therefore, the trajectory observation τ_i, i.e., $o\left(p_{t_i}, z\right)$, can be well generated based on the latent time-dependent route r, route type k, and latent traffic pattern z. We mark the parameters for generation as the set $\theta = \{f_4(\cdot), g_4(\cdot)\}$. These parameters can be learned in the training step.

Algorithm 1: Training of the DeepTEA model

Input : The trajectory T
Output: model parameters $\phi = \{\{f_1(\cdot), g_1(\cdot)\}$, $\gamma = \{f_2(\cdot), f_3(\cdot), g_3(\cdot), \pi, \mu_r, \sigma_r\}$, $\theta = \{\{f_4(\cdot), g_4(\cdot)\}$

Construct the real traffic condition Z from trajectory set $\{T\}$
Construct traffic condition transition $f_1(Z)$ by Equation 1
Draw latent traffic pattern z by Equation 2 and $g_1(\cdot)$
for *Every* t_i *in the time that trajectory* T *traversed* **do**
  Obtain τ_i by $f_2(p_{t_i}, z)$ as Equation 4
  Calculate h_i by $f_3(h_{i-1}, \tau_i))$ in Equation 6
end
Draw latent time-dependent route r from $q_\gamma(r|T)$ by $g_3(h_n)$
Construct $p_\gamma(r|k, z)$ with μ_r, σ_r by Equation 8
Obtain $p_\gamma(k)$ with π by Equation 9
Construct $q_\gamma(k|T)$ by Equation 11
Calculate trajectory transition based on latent traffic pattern by $\eta_i = f_4(\tau_i,\ \eta_{i-1})$
Draw τ_i from a Multinominal distribution with the probability based on softmax($g_4(\eta_{i-1})$)
Construct ELBO = $\mathcal{L}(\phi, \gamma, \theta; T)$ of the marginal log-likelihood by Equation 16
Optimize $\mathcal{L}(\phi, \gamma, \theta; T)$ to update ϕ, γ, θ
return ϕ, γ, θ

3.5 Optimization

As we discussed, the trajectory observation can not only reflect the location information, but also convey the latent traffic pattern based on the transition between two consecutive trajectory points. Therefore, the objective is to maximize the marginal log-likelihood of the observed trajectory as:

$$\log p_\theta(T^{(1)}, T^{(2)}, \cdots, T^{(N)}) := \log p_\theta(\tau^{(1)}, \tau^{(2)}, \cdots, \tau^{(N)}) \,. \quad (14)$$

We optimize the marginal log-likelihood by maximizing the evidence lower bound (ELBO) [15] of the marginal log-likelihood.

$$\log p_\theta(T) \geq \text{ ELBO } = \mathcal{L}(\phi, \gamma, \theta; T) \,. \quad (15)$$

The ELBO of the marginal log-likelihood of trajectory T is calculated as:

$$\begin{aligned}\mathcal{L}(\phi, \gamma, \theta; T) &= \mathbb{E}_{q_{\gamma,\phi}(r,k,z|T)}\left[\log \frac{p_{(\phi,\gamma,\theta}(r,k,z,T)}{q_{\gamma,\phi}(r,k,z|T)}\right] \\ &= -\mathbb{E}_{q_\gamma(r|T)} D_{KL}\left(q_\gamma(k|T) \,||\, p_\gamma(k)\right) \\ &- \mathbb{E}_{q_\gamma(k|T)} D_{KL}\left(q_\gamma(r|T) \,||\, p_\gamma(r|k,z)\right) \\ &- D_{KL}\left(q_\phi(z|T) \,||\, p_\phi(z)\right) + \mathbb{E}_{q_{\gamma,\phi}(r,k,z|T)} \log p_\theta(T|r,z,k) \,,\end{aligned} \quad (16)$$

where $p_\phi(z)$ is the prior probability of the latent traffic pattern z. The generative network $\log p_\theta(T|r, z, k)$ can be calculated as:

$$\log p_\theta(T|r,z,k) = \sum_{i=1}^{n} \log p_\theta(\tau_i|\tau_{1:i-1}, r, z, k) \quad (17)$$

The overall process of training is shown in Algorithm 1. During training, the model parameters are learned by optimizing the ELBO from the trajectory T. Then these learned parameters can be utilized for online anomaly detection in the following sections.

3.6 Complexity Analysis

The overall complexity of training DeepTEA is dominated by $O(N \cdot (d_{Z_1} d_{Z_2} \overline{V} + \overline{n}))$, where N is the number of trajectories, d_{Z_1} and d_{Z_2} are the sizes of Z, $\overline{V}$ is the average number of time intervals, and $\overline{n}$ is the average length of trajectories.

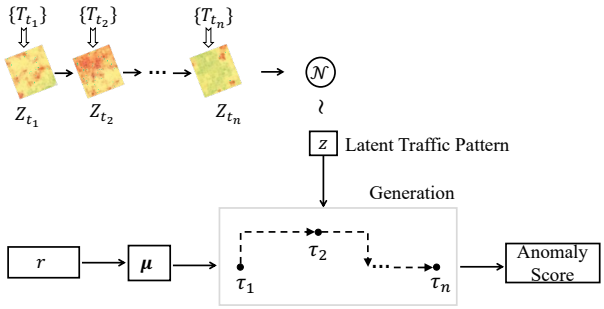

Figure 3: Overview of online detection by DeepTEA.

4 ONLINE TRAJECTORY OUTLIER DETECTION BY DEEPTEA

Based on the learned parameters from Algorithm 1, the online detection updates the anomaly score when the next trajectory observation τ_{i+1} comes in real-time. The process should be able to calculate rapidly once the trajectory updates on the fly. In this case, the anomaly score can be updated even the trip has not been finished. This is beneficial for both drivers and passengers. On the one hand, passengers can take early action once they are detoured by dishonest drivers. On the other hand, drivers can show evidence from our model to passengers who have complaints about the route choices during the trip. The earlier the anomaly can be detected, the less the loss is.

4.1 Online Detection by Generation

Figure 3 shows the process of online anomalous trajectory detection. We detect the outlier by generating the observed trajectories based on learned parameters ϕ, γ, θ. Specifically, the distribution $q_\gamma(r|T)$ of the latent time-dependent route can be calculated based on parameters in γ. And the latent traffic pattern z can be obtained based on parameters in ϕ with Z, which is aggregated from the trajectory set $\{T\}$. Given the k-th mean u_k of components in $q_\gamma(r|T)$, we apply an RNN neural network to generate the trajectory observation as:

$$\eta_i = f_4(\tau_i,\ \eta_{i-1}) = \text{RNN}(\tau_i,\ \eta_{i-1}) \,, i = 1, 2, \cdots, n, \ \textit{and}\ \eta_0 = u_k, \quad (18)$$

where the initial input of the RNN is η_0, which is set as u_k. Starting from η_1, the inputs become the last hidden state η_{i-1} and the trajectory observation τ_i, i.e., $o\,(p_{t_i}, z)$. Therefore, τ_{i+1} can be generated as:

$$p_\theta(\tau_{i+1}|\ \tau_{1:i},\ u_k) = \text{softmax}(g_4(\eta_{i-1})), \quad (19)$$

where $g_4(\cdot)$ is the function to map the output to the size of the map grid. The softmax function is used to make the sum of the probability to one.

Algorithm 2: Online detection by the DeepTEA

Input : The trajectory T,
learned model parameters $\phi = \{\{f_1(\cdot), g_1(\cdot)\}$,
$\gamma = \{f_2(\cdot), f_3(\cdot), g_3(\cdot), \pi, \mu_r, \sigma_r\}$,
$\theta = \{\{f_4(\cdot), g_4(\cdot)\}$
Output : The anomaly score of trajectory T

Construct the real traffic condition Z from trajectory set $\{T\}$
for *Every trajectory observation* τ_{i+1} *that comes in real-time* **do**
  if *the time interval of* τ_{i+1} *is different from the time interval of* τ_i **then**
    Update the latent traffic pattern z from Z based on learned parameters ϕ
    Obtain $q_\gamma(r|T)$ based on learned parameters γ
  end
  largest_likelihood = 0.0
  for *the k-th mean* u_k *of the components in* $q_\gamma(r|T)$ **do**
    Calculate η_i based on Equation 18
    Obtain $p_\theta(\tau_{i+1}|\ \tau_{1:i},\ u_k)$ based on $g_4(\cdot)$
    Obtain likelihood of previous trajectory observations $p_\theta(\tau_{1:i}|u_k)$
    Update likelihood l of trajectory containing the new observation τ_{i+1} by $\exp\left[\frac{\log p_\theta(\tau_{1:i}|u_k) p_\theta(\tau_{i+1}|\tau_{1:i}, u_k)}{i+1}\right]$
    if $l >$ *largest_likelihood* **then**
      largest_likelihood = l
    end
  end
  $s_a(\tau_{1:i+1})$ = 1 − largest_likelihood
  return $s_a(\tau_{1:i+1})$ *of the trajectory observation* $\tau_{1:i+1}$
end

Given the learned $q_\gamma(r|T)$ and latent traffic pattern z, the anomaly score $s_a(\tau_{1:i})$ of the trajectory T given in the real-time can be calculated as one minus the generation likelihood of the trajectory observation $\tau_{1:i}$, i.e., $\{\tau_1 \rightarrow \tau_2 \rightarrow \cdots \rightarrow \tau_i\}$, as:

$$s_a(\tau_{1:i}) = 1 - \arg\max_k \exp\left[\frac{\sum_{i=1}^{n} \log p_\theta(\tau_i|\tau_{1:i-1}, u_k)}{n}\right] \quad (20)$$

where u_k is the mean of the k-th component in the distribution $q_\gamma(r|T)$ of the latent time-dependent route r.

In the online setting, given the previous trajectory observation $\tau_{1:i}$, the anomaly score of the next incoming trajectory observation $\tau_{1:i+1}$ can be updated based on the previous trajectory $\tau_{1:i}$ as:

$$s_a(\tau_{1:i+1}) = 1 - \arg\max_k \exp\left[\frac{\log p_\theta(\tau_{1:i}|u_k) p_\theta(\tau_{i+1}|\tau_{1:i}, u_k)}{i+1}\right] \quad (21)$$

The overall process of online detection is shown in Algorithm 2. The inputs are the trajectory T, and the learned model parameters ϕ, γ and θ from the training in Algorithm 1. For the newly coming trajectory observation τ_{i+1}, we update the latent traffic pattern z if it changed. Then we calculate the likelihood of observing the trajectory observation $\tau_{1:i+1}$ based on $\tau_{1:i}$. Finally, the anomaly score is returned.

4.2 Complexity Analysis

The overall complexity of detection is dominated by $O(d_{Z_1} d_{Z_2})$, where d_{Z_1} and d_{Z_2} are the sizes of Z.

5 THE DEEPTEA-A MODEL: APPROXIMATE ONLINE DETECTION

5.1 Approximation Algorithm

5.1.1 ***Challenge.*** The online updating of anomaly score for τ_{i+1} based on $\tau_{1:i}$ replies on the calculation of convolution for traffic condition matrix Z at the time t_{i+1}, which could be time-consuming for the large road networks. It could make the online detection process slow. As a large volume of trajectories is generated (e.g., 2-4 seconds [1]) dramatically in real-time, the faster the detection process, the less the delay is.

5.1.2 ***Design.*** Inspired by [23], we further propose an approximate algorithm to speed up the online anomalous trajectory detection. We propose an algorithm to utilize the traffic condition matrix Z at the time interval of τ_1 as an approximation of traffic condition during the trip. In this way, the convolution of traffic condition matrix Z only needs to be calculated once for the first trajectory observation τ_1. For the online update of anomaly score for τ_{i+1}, it does not need convolution calculation anymore.

Given a source S_T and a destination D_T, we draw the best latent route type k from $q(k|S_T, D_T, z_{S_T})$ to approximate the best latent route pattern u_k, which needs to be found from k means of $q_\gamma(r|T)$, where Z_{S_T} is the traffic condition when the trip starts. In this way, the best latent route type k can be obtained once the S_T, D_T and Z_{S_T} are given when the trip starts. Starting from the second trajectory observation, the best latent route pattern does not need to be recalculated anymore for updating the anomaly score.

For the source location S_T, the hidden state of the traffic condition Z_{S_T} can be calculated as:

$$f_1(Z_{S_T}) = \text{CNN}(Z_{S_T}), \quad (22)$$

Then z_{S_T} can be drawn from the probability $q_\phi(z_{S_T}|Z_{S_T})$ as:

$$q_\phi(z_{S_T}|Z_{S_T}) = \mathcal{N}(\mu_{Z_{S_T}}, diag(\sigma^2_{Z_{S_T}}))\,, \quad (23)$$

After that, τ_{S_T} can be obtained by $f_2(\cdot)$ as:

$$\tau_{S_T} = f_2(S_T, z_{S_T}) = \text{NN}(S_T, z_{S_T}) = WS_T + Qz_{S_T}\,, \quad (24)$$

where f_2 is the function with W and Q as parameters. Similarly, τ_{D_T} can be calculated with the same process.

After that, $q(k|S_T, D_T, z_{S_T})$ can be modeled by an MLP neural network as:

$$q(k|S_T, D_T, z_{S_T}) = \text{softmax}(f_5(\tau_{S_T}, \tau_{D_T}))\,, \quad (25)$$

The softmax function is used to normalize the probability, and f_5 is the MLP neural network. We mark the parameters for approximation as $\delta = \{f_5(\cdot)\}$.

Instead of finding the largest likelihood of observing τ by k times computation to obtain the best route type k as Equation 20, one straightforward way is to find the best route type k directly from the trajectory T by $q_\gamma(k|T)$. Therefore, the distribution of $q(k|S_T, D_T, z_{S_T})$ and $q_\gamma(k|T)$ in Equation 11 should be as close as possible. Here we apply the cross entropy to minimize the difference between these two distributions as:

$$l_k = -\sum_{k=1}^{K} q_\gamma(k|T) \log q(k|S_T, D_T, z_{S_T})\,, \quad (26)$$

Algorithm 3: DeepTEA-A: approximate online detection

Input : The trajectory T,
the source location S_T,
the destination location D_T,
learned model parameters $\phi = \{\{f_1(\cdot), g_1(\cdot)\}$,
$\gamma = \{f_2(\cdot), f_3(\cdot), g_3(\cdot), \pi, \mu_r, \sigma_r\}$,
$\theta = \{\{f_4(\cdot), g_4(\cdot)\}$,
$\delta = \{f_5(\cdot)\}$

Output : The anomaly score of trajectory T

Construct the real traffic condition Z_{S_T} from trajectory set $\{T\}$ when the trip starts at S_T
Calculate the hidden state of Z_{S_T} by a CNN based on Equation 22
Draw the latent traffic pattern z_{S_T} from a Gaussian distribution based on Equation 23
Construct τ_{S_T} and τ_{D_T} based on Equation 24
Obtain the best k with the highest probability in Equation 25
Obtain the best latent time-dependent route u_k by k
for *Every trajectory observation* τ_{i+1} *that comes in real-time* **do**
  Calculate η_i based on Equation 18
  Obtain $p_\theta(\tau_{i+1} | \tau_{1:i}, u_k)$ based on $g_4(\cdot)$
  Obtain likelihood of previous trajectory observations $p_\theta(\tau_{1:i}|u_k)$
  Update likelihood $l_{\tau_{1:i+1}}$ of trajectory containing the new observation τ_{i+1} by $\exp[\frac{\log p_\theta(\tau_{1:i}|u_k) p_\theta(\tau_{i+1}|\tau_{1:i}, u_k)}{i+1}]$
  $s_a(\tau_{1:i+1}) = 1 - l_{\tau_{1:i+1}}$
  return $s_a(\tau_{1:i+1})$ *of the trajectory observation* $\tau_{1:i+1}$
end

The cross entropy l_k is co-trained with ELBO in Equation 16 in the training phase. For the online detection phase, the best k is obtained from $q(k|S_T, D_T, z_{S_T})$ with the highest probability. Then the best latent time-dependent route u_k can be obtained accordingly. And it only needs to be calculated once for the first trajectory observation τ_1. Then u_k is utilized for the anomaly detection for the next trajectory observations starting from the second trajectory observation. In this way, the convolution operation does not need to be calculated for $\tau_{2:i}$, which saves computation costs for online detection.

Note that the training phase of the approximate algorithm is different from Algorithm 1. Firstly, the traffic conditions used are quite different, in which only the real traffic condition Z_{S_T} is considered for the approximation. Second, we add a co-training term, i.e., Equation 26, to approximate $q_\gamma(k|T)$ with $q(k|S_T, D_T, z_{S_T})$. After training, we obtain the learned model parameters ϕ, γ, θ and δ. The overall process of approximate online detection is shown in Algorithm 3. It first calculates the latent traffic pattern z_{S_T} from Z_{S_T}. Then it obtains the best latent time-dependent route u_k given the real traffic condition Z_{S_T}, the source location S_T, and the destination location D_T. The best latent time-dependent route u_k only needs to be calculated once when the trip starts. Then the online anomaly score is updated based on $p_\theta(\tau_{1:i}|u_k)$.

5.2 Complexity Analysis

The overall complexity of approximate online detection when new trajectory observation comes is $O(d_{h_t}(d_{h_t} + d_{\tau_i}))$. The term is the cost of RNN transition for new trajectory observation τ_{i+1}. Since d_{h_t} and d_{τ_i} are constant, the time complexity of approximation is $O(1)$, while the time complexity of DeepTEA is dominated by $O(d_{Z_1} d_{Z_2})$ in Section 4.

6 EXPERIMENTS

In this section, we evaluate our model with extensive experiments. We first show the experimental setup, which contains datasets, preprocessing, competitors, hyperparameter tuning, and performance metric. Then we evaluate the effectiveness and efficiency of our model and competitors.

6.1 Experimental Setup

6.1.1 Datasets. We evaluate our method on two real datasets.

• The first dataset (XN) is an open GPS data set[2], which contains GPS data in Xi'an around second ring road region from October 1 to October 7 in 2016 in China.

• The second dataset (CD) is an open GPS data set[2], which contains GPS data in Chengdu around second ring road region from October 1 to October 7 in 2016 in China.

Note that XN and CD contain roads with different road structures, not only the ring roads.

6.1.2 Preprocessing. Following previous work [1, 21, 23, 42, 44, 45], the geographical space is partitioned into grids. We use 100m × 100m grids on two datasets. We filter out (S, D) pairs that have fewer than 10 trajectories. The lengths of trajectories should be larger than 30. The time interval of traffic conditions is 20 minutes. The statistics of the filtered trajectories on two datasets are summarized in Table 1.

Table 1: Statistics of the filtered trajectories on two datasets.

Dataset	# Trajectory Points	# Trajectories
XN	1,446,470	13,515
CD	12,694,201	108,510

There is no available labeled open trajectory dataset for outlier detection. Some of the previous work manually labels outliers in their work [1, 25, 42, 44, 45]. Since manual efforts are limited and expensive, the total number of trajectories is limited to a few numbers, e.g., only 1,300 trajectories in the evaluation of [44, 45]. Therefore, we follow the anomaly generation strategy as [3, 23, 43] to generate ground truth with a large number of trajectories.

As [23], we apply two perturbation ways to generate two kinds of outliers, which are detour (**D**) and route-switching outliers (**RS**). Detour outliers are generated by two parameters, which are detour offset d and the proportion of detour segments α. For example, $d = 5$ and $\alpha = 0.1$ means that 10% of a trajectory is offset 5 grid cells. The route-switching outliers are generated by switching from one route to another with a parameter β to control the switching location . For example, $\beta = 0.2$ means the beginning 20% of the first route is concatenated with the latter 80% of the second route. We inject outliers in 5% of trajectories as [23]. Note that in order to generate time-dependent outliers, we randomly sample trajectories from the same (S, D) pairs with the same travel time, then inject two kinds of outliers into sampled trajectories. Since trajectories

[2] http://outreach.didichuxing.com/research/opendata/

with the exact same travel time are very few, we allow a relaxation ϵ in the travel time. In this evaluation, we set ϵ to 20 minutes.

6.1.3 Competitors. We compare our model with seven other methods, which can be categorized into two groups. More details can be found in Section 7.

Non-time-dependent trajectory outlier detection methods: In this category, there are two kinds of methods. The first kind of methods, which are **IBAT** [42] and **DBOTD** [25], utilizes anomaly score function to distinguish outliers from popular routes based on distance or density metrics. Another kind of methods applies learning-based approaches, which can capture features behind trajectories automatically by optimization. They are **LODA** [34], **DB-TOD** [37], **GM-VSAE** [23], and its extension **SD-VSAE** [23]. Among them, LODA [34], DB-TOD [37], GM-VSAE [23] and SD-VSAE [23] support online detection.

To make the comparison fair, we adapt the non-time-dependent normal routes based methods to time-dependent normal routes based ones. Particularly, instead of directly finding normal routes from (S, D) pairs ignoring the travel time, we extract time-dependent normal routes from trajectories with the same source and destination under the same travel time. Then outliers are detected by comparing testing trajectories with time-dependent normal routes.

Time-dependent trajectory outlier detection methods: The state-of-the-art model for time-dependent trajectory outlier detection is **TPRRO** [45], which is an extension for improving the efficiency of TPRO [44]. It utilizes time-dependent edit distance to distinguish outliers from time-dependent popular routes.

6.1.4 Performance Metric. We use a standard metric, named Precision-Recall AUC (PR-AUC), to evaluate the performance of models as in previous work [23, 43]. As outliers in trajectories are skewed, PR-AUC is a standard metric to evaluate the performance of these skewed datasets [23, 43]. We report the average PR-AUC among all source and destination pairs.

6.1.5 Hyperparameter Tuning. We construct training, validation, and testing sets by randomly partitioning two datasets with the ratio of 8:1:1. We run 5 times, and report the average PR-AUC results. We conduct hyper-parameter tuning by using a Bayesian optimizer for all the methods on all datasets. The scopes of hyper-parameters are learning rate from a range [0.0001, 0.1], regularization [0.0001, 0.1], decay rate [0.9, 1.0], embedding size of trajectory points from a set {32, 64, 128, 256, 512}, RNN hidden size {32, 64, 128, 256, 512}, negative sample size {32, 64, 128, 256, 512}, MLP hidden size {32, 64, 128, 256, 512}, the number of Gaussian components k {5, 10, 20}. For the methods with threshold, we use the range of threshold candidates in their original paper [25, 37, 42, 45] to find the best threshold.

6.2 Effectiveness Evaluation

We evaluate the effectiveness of all methods on two anomalous types, i.e., detour (D) and route-switching (RS) anomalies, w.r.t. the observed ratio ρ. Moreover, we discuss the effectiveness of our proposed approximate online detection method DeepTEA-A in Section 6.2.4.

We follow the setting of ρ in [23] as 0.1, 0.5, and 1.0 for detour anomalies on all methods. For route-switching anomalies, we set ρ = {0.5, 1.0} for β = 0.3, ρ = {0.7, 1.0} for β = 0.5, and ρ = {0.9, 1.0} for β = 0.7 as [23]. Note that we report average results over five runs for all methods.

6.2.1 Varying Observed Ratios. We evaluate the effectiveness of online detection w.r.t. the observed ratio ρ, where ρ = 0.1 denotes only 10 % of a trajectory is observed, and ρ = 1.0 means the completed trajectory is observed for anomaly detection. We evaluate the average PR-AUC of varying observation ratios for all methods on all datasets in Tables 2 and 3. We observe that:

(1) For GM-VSAE, SD-VSAE, and our model, the values of PR-AUC increase when the observation ratio ρ increases on the XN and CD datasets. It means that more observations are beneficial to encoder-decoder based approaches. And the values of PR-AUC become the largest ones when the full observations are complete, i.e., ρ=1.0.

(2) For other methods, most values of PR-AUC increase when the observation ratio ρ increases on the two datasets. We find that more observations could improve the effectiveness to some degree for these methods.

(3) Our model DeepTEA consistently outperforms other competitors w.r.t. all observation ratios ρ. For example, DeepTEA is 17.52% more accurate than other competitors on average on the XN and CD datasets. It validates the effectiveness of considering latent traffic patterns for time-dependent outlier detection.

6.2.2 Detection of Detour Anomalies. We evaluate the effectiveness of detecting detour outliers for all methods on two datasets. We follow the settings of the perturbation parameters d and α, and the observed ratio ρ as [23] in the evaluation. From Tables 2, 3, we make the following observations:

(1) For full observed trajectories, i.e., ρ = 1.0, the values of PR-AUC when d=3, α = 0.3 are the largest ones in detour anomalies detection on the XN and CD datasets. It means that compared with the detour offset d, the proportion of detour segment α plays a more important role in the detection performance. Second, given the same proportion of detour segment α, a larger detour offset d usually leads to higher PR-AUC values. It is consistent with the intuition that a larger degree of detour usually has a larger detour deviation from the normal routes. Third, learning-based methods usually perform better than metric-based methods. The reason is that instead of finding normal routes by density or distance metrics, the learning-based methods could learn the features behind trajectories, which are beneficial for outlier detection. Fourth, our model outperforms all other competitors when full trajectories are observed. It justifies the effectiveness of latent traffic patterns when detecting time-dependent outliers. For example, our model is 23.3% more effective than other methods on average on the CD dataset.

(2) For partially observed trajectories, i.e., ρ < 1.0, learning-based methods usually perform better than metric-based methods. Second, our model performs the best compared with other learning-based competitors when ρ = 0.1 and 0.5. It shows that with partially observed trajectories, borrowing traffic conditions during the travel time plays an important role for time-dependent outliers detection.

6.2.3 Detection of Route-switching Anomalies. We evaluate the effectiveness of detecting route switching outliers for all methods on two datasets. We follow the settings of the perturbation parameter

Table 2: Effectiveness on XN dataset. For each method, we report average results over five runs.

	Detour anomalies									Route-switching anomalies					
Perturb params	d=3; α=0.1			d=3; α=0.3			d=5; α=0.1			β=0.3		β=0.5		β=0.7	
Observed ratio ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0	0.5	1.0	0.7	1.0	0.9	1.0
IBAT	0.290	0.210	0.213	0.238	0.221	0.261	0.255	0.217	0.245	0.308	0.302	0.293	0.303	0.282	0.294
DBOTD	0.571	0.580	0.576	0.609	0.619	0.647	0.589	0.500	0.550	0.471	0.547	0.590	0.590	0.599	0.610
LODA	0.608	0.610	0.619	0.623	0.625	0.627	0.605	0.607	0.613	0.423	0.533	0.624	0.627	0.622	0.629
DB-TOD	0.526	0.568	0.618	0.568	0.585	0.640	0.544	0.578	0.626	0.470	0.530	0.629	0.638	0.623	0.631
GM-VSAE	0.627	0.854	0.878	0.816	0.929	0.929	0.667	0.863	0.874	0.472	0.549	0.626	0.641	0.726	0.739
SD-VSAE	0.644	0.853	0.889	0.810	0.916	0.915	0.666	0.878	0.863	0.451	0.518	0.622	0.641	0.725	0.738
TPRRO	0.559	0.573	0.573	0.630	0.630	0.632	0.586	0.593	0.594	0.472	0.542	0.610	0.617	0.589	0.591
DeepTEA	**0.676**	**0.855**	**0.901**	**0.874**	**0.946**	**0.954**	**0.672**	**0.889**	**0.895**	**0.473**	**0.552**	**0.639**	**0.665**	**0.727**	**0.742**

Table 3: Effectiveness on CD dataset. For each method, we report average results over five runs.

	Detour anomalies									Route-switching anomalies					
Perturb params	d=3; α=0.1			d=3; α=0.3			d=5; α=0.1			β=0.3		β=0.5		β=0.7	
Observed ratio ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0	0.5	1.0	0.7	1.0	0.9	1.0
IBAT	0.229	0.248	0.345	0.185	0.222	0.343	0.220	0.234	0.349	0.275	0.361	0.283	0.356	0.325	0.342
DBOTD	0.554	0.529	0.536	0.537	0.535	0.542	0.503	0.510	0.514	0.533	0.543	0.556	0.570	0.579	0.581
LODA	0.599	0.596	0.598	0.587	0.595	0.594	0.600	0.599	0.591	0.592	0.594	0.587	0.589	0.591	0.592
DB-TOD	0.526	0.568	0.618	0.568	0.585	0.640	0.544	0.578	0.626	0.580	0.630	0.629	0.638	0.623	0.631
GM-VSAE	0.652	0.834	0.853	0.851	0.882	0.873	0.660	0.837	0.842	0.601	0.703	0.753	0.745	0.767	0.752
SD-VSAE	0.651	0.830	0.845	0.852	0.883	0.880	0.660	0.834	0.850	0.590	0.700	0.743	0.731	0.773	0.760
TPRRO	0.579	0.592	0.593	0.585	0.600	0.602	0.583	0.585	0.591	0.571	0.585	0.571	0.586	0.579	0.585
DeepTEA	**0.664**	**0.842**	**0.879**	**0.853**	**0.911**	**0.922**	**0.675**	**0.858**	**0.881**	**0.606**	**0.754**	**0.791**	**0.801**	**0.826**	**0.827**

β and observed ratio ρ as [23] in the evaluation. From Tables 2, 3, we make the following observations:

(1) Detection effectiveness of route-switching anomalies is lower than detour anomalies on the XN and CD datasets. The reason is that the anomalies of route-switching are not that significant in the sense of shape, since they are formed by two existing routes in the trajectory data. On the other hand, detour anomalies do not exist in the trajectory data. Therefore, it is more difficult to detect route-switching anomalies.

(2) For complete trajectories, i.e., ρ = 1.0, there is a trend that larger split location β leads to higher values of PR-AUC for full observed trajectories in most methods, i.e., GM-VSAE, SD-VSAE, and our model on the XN and CD datasets. Second, learning-based methods usually have better PR-AUC values on the detection of route-switching anomalies for complete trajectories. It shows the benefits of learned complex features from trajectories for outlier detection. Third, our model performs the best for detecting route-switching anomalies given complete trajectories. It verifies the effectiveness of learning latten traffic patterns behind the route-switching anomalies during travel time. For example, our model is 17.4% more effective than other competitors on average on the CD dataset.

(3) For partially observed trajectories, i.e., ρ = 0.5, 0.7, 0.9, the performance increases when ρ increases from 0.5 to 0.9 in most methods on the XN and CD datasets. It shows that observing more trajectories is beneficial for route-switching anomalies detection. Second, learning-based methods work better than metric-based methods for partial trajectories observed in route-switching anomalies. Third, our model also performs better than other learning-based methods for route-switching anomaly detection with partial trajectories.

6.2.4 Effectiveness of Approximation. We evaluate the effectiveness of DeepTEA-A on two anomalous types, i.e., detour (D) and route-switching (RS) anomalies in Figure 6. We set the perturbation parameters as d = 3, α = 0.3 for detour (D), and β = 0.3 for route-switching (RS) outlier as in the evaluation of [23]. We observed that:

(1) There is a slight drop in the values of PR-AUC on the XN and CD datasets for DeepTEA-A compared with DeepTEA, i.e., 1.16% on average. It means that with the latent time-dependent route conditioned on only source, destination locations, and the traffic condition at the departure time, the performance slightly drops for time-dependent outlier detection in both two anomalous types D and RS.

(2) The performance gap on CD is less than the gap on XN for DeepTEA-A. For example, the average drop is 0.94% on CD, while 1.37% on XN. It may be the case that traffic condition on the CD dataset is more dense than it on XN dataset, e.g., around 50% of traffic condition is available on the CD dataset, while only 33% of it is available on the XN dataset. Since denser traffic conditions can reflect more accurate traffic flows, more accurate latent traffic patterns could be learned based on that. Therefore, the performance of time-dependent outlier detection based on denser traffic conditions could be better.

(3) The performance of DeepTEA-A increases when the observed ratio ρ increases. It confirms that more observed trajectories can give more accurate detection results for approximate method on both XN and CD datasets.

6.2.5 Effectiveness w.r.t. Time Interval Length. We evaluate the effectiveness w.r.t. time interval length. In Figure 8, we show the PR-AUC w.r.t. different time intervals on Detour anomalies with d=3 and α = 0.1 on CD dataset. Note that the trends on other parameter settings are similar as the trend in Figure 8. We can find that when the time interval is small, e.g., 5 minutes, the PR-AUC values are also small. When the time intervals equal 15 or 20 minutes, the PR-AUC values achieve the larger ones. Then PR-AUC values drop down when the time intervals are further enlarged. The reason is that smaller time intervals may lead to a very sparse speed matrix, which can not well capture the traffic condition. On the other hand, when the time intervals are very large, the corresponding speed matrix can not well reflect the changing of traffic conditions even though they are denser.

6.3 Efficiency Evaluation

We use GeForce GTX 1080 Ti 11 GB GPU for evaluation.

6.3.1 Overall Detection Efficiency. We evaluate the average detection time of one trajectory with the full observation, i.e., ρ=1.0, of all methods on all datasets in Figure 4 (d). Note that the y-axis is in a *logarithmic* scale. From Figure 4 (d), we make the following observations:

(1) DB-TOD detects outliers in the fastest speed on both XN and CD datasets. The reason is that DB-TOD learns the parameters with a linear function by maximizing the likelihood in the training phase. And the learned parameters only require a fast linear operation in the online detection phase, which only depends on a small number of parameters in their model.

(2) The methods LODA, SD-VSAE, and our approximation version DeepTEA-A perform faster than other methods except for only DB-TOD. The reason is that LODA ensembles some weak classifiers, which are fast linear functions. As DB-TOD, the linear functions are very efficient for online detection. And the number of linear functions is not large. For SD-VSAE, it reduces the computation cost by replacing k times most likely latent route computation with only one time calculation by source and destination locations. The computation cost of online detection for SD-VSAE is $O(1)$. Although our approximation method considers more information in traffic conditions during the travel time, we reduce its time cost by approximating the latent traffic pattern based on the traffic condition at the departure time. The time cost of online detection for our approximation method is also $O(1)$.

(3) Most metric-based methods work slower for online detection, e.g., DBOTD, IBAT, and TPRRO. They usually require finding the normal time-dependent normal routes from the dataset for given source, destination locations, and travel time based on density or distance metrics. However, these steps are usually time-consuming for online detection. They need to first extract time-dependent normal routes and then compare the testing trajectory with the time-dependent normal routes by the pre-defined threshold.

(4) The detection time of the approximation method DeepTEA-A is faster than DeepTEA. The reason is that we reduce the time cost of our standard method by learning the parameters for latent traffic patterns based on only the traffic condition at the departure time with a co-training network in the training phase. The learned parameters can be directly used to detect the outliers online without updating the latent traffic pattern when the trajectories move on the fly.

6.3.2 Detection Scalability. We evaluate the average detection time of one trajectory with different observed ratios ρ accumulated during detection for all methods on all datasets in Figure 4 (a-b). We set ρ = {0.2, 0.4, 0.6, 0.8, 1.0} as [23]. From Figure 4 (a-b), we make the following observations:

(1) DB-TOD runs the fastest among all methods for detection with all observation ratios ρ. The reason is that it only requires a fast linear computation, which only depends on the number of parameters in the linear function.

(2) The approximation method DeepTEA-A detects faster than our standard method DeepTEA in all observation ratios ρ. The reason is that the time cost of DeepTEA-A is $O(1)$, while DeepTEA requires $O(d_{Z_1} d_{Z_2})$ computation cost.

(3) The detection time increases when the observation ratio ρ increases on both XN and CD datasets. It is obvious that more trajectories observed require more time to calculate the anomaly score.

6.3.3 Training Scalability. As metric-based methods do not involve a training phase, we evaluate the training scalability on deep-learning-based methods: **GM-VSAE** [23], and its extension **SD-VSAE** [23] and our models as the setting of the training scalability evaluation in [23]. We evaluate their training scalability with respect to the number of trajectories in Figure 4 (c). We vary the size of training trajectories by sampling from the XN dataset with the range from 1,000 to one million. For the sizes that exceed the total training size in the XN dataset, we randomly sample multiple times until the sizes satisfy our required numbers. We report the average training time of one epoch for all methods. We make the following observations from Figure 4 (c):

(1) GM-VSAE, SD-VSAE, and our models are scalable on vast trajectories. It shows that these models are linear w.r.t. the size of training trajectory data. Therefore, it is feasible to train the large size trajectories with a reasonable time cost.

(2) The approximation method DeepTEA-A requires more time than DeepTEA to a slight degree. The reason is that the approximation method requires an additional co-training phase with an MLP neural network. The parameter learning requires extra time to train in the training step.

(3) GM-VSAE needs less training time than other methods. For GM-VSAE, it contains steps, e.g., the Gaussian mixture distribution for the latent pattern. For our model, we incorporate even more components, e.g., the latent pattern behind the traffic condition during the travel time. These more steps need more time to train.

6.3.4 Efficiency v.s. Accuracy. We show the tradeoff between PR-AUC and detection time on testing datasets on XN and CD datasets in Figure 7. For the detection time at the y-axis, the lower the place of the method is, the faster the detection time of the method is.

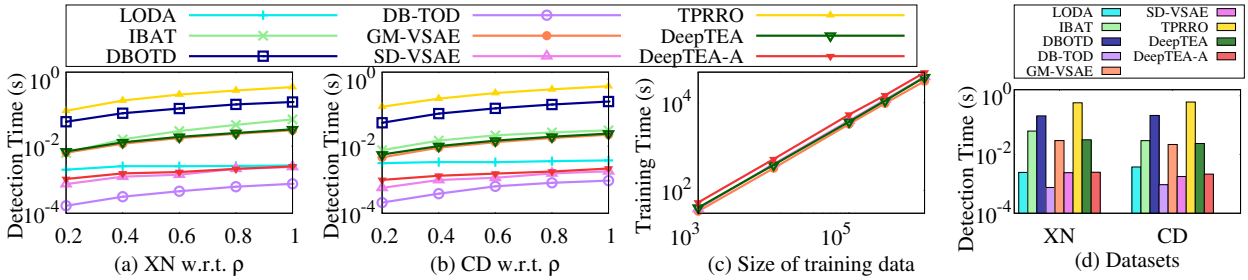

Figure 4: Scalability on (a-b) detection w.r.t. observed ratio ρ; (c) training w.r.t. the size of trajectories; (d) detection efficiency.

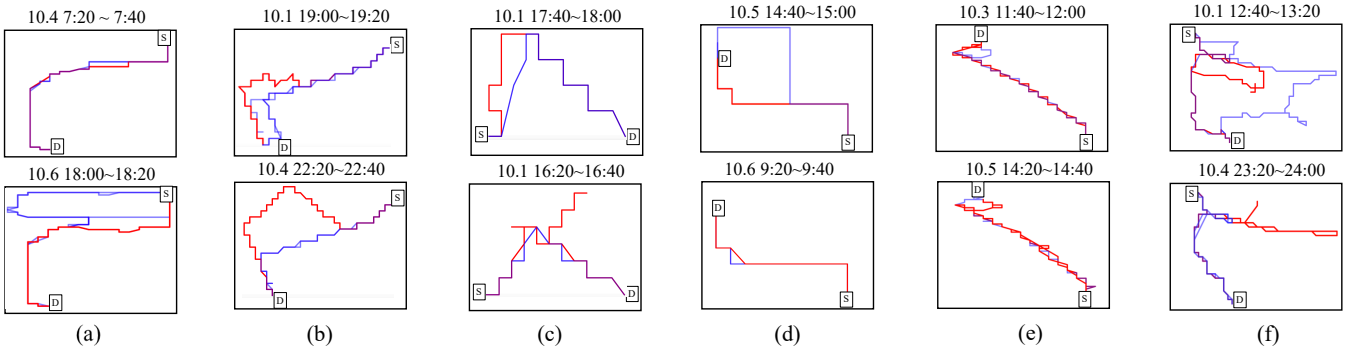

Figure 5: Case study. The red lines represent time-dependent outliers, and the blue lines denote normal trajectories.

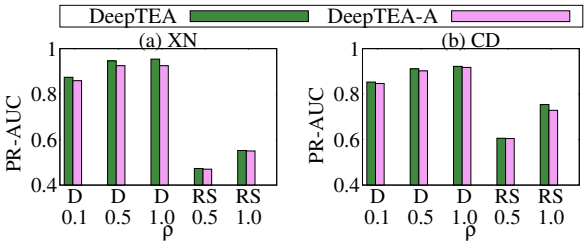

Figure 6: Effectiveness of DeepTEA-A on (a) XN; (b) CN.

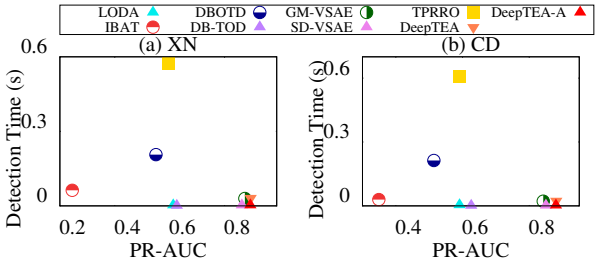

Figure 7: PR-AUC v.s. detection time on (a) XN; (b) CN.

For the PR-AUC at the x-axis, the closer to the right, the better the effectiveness of the method is. We found that our approximation method DeepTEA-A has comparable PR-AUC compared with

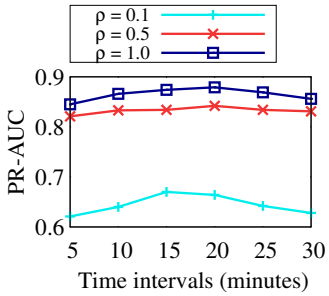

Figure 8: PR-AUC w.r.t. time intervals.

DeepTEA, which achieves the best PR-AUC values, while the detection time of DeepTEA-A is still fast. Compared with SD-VSAE, which has a similar detection time, DeepTEA-A has a better PR-AUC value than SD-VSAE.

6.4 Case Study

We show one case study to illustrate outlier behaviors during different travel times. In Figure 5 (a – f), all trajectories are real world trajectories, and we do not inject any outliers in the trajectories in this case study. We use the trained model to detect time-dependent outliers in these real world trajectories, and show the detected time-dependent outliers in red color, while the normal trajectories in blue color. In each subfigure a – f, the upper and bottom ones are from the same source S and destination D during different travel

times. For example, the upper place of Figure 5 (a) shows the trajectories during 4th October 7:20 ∼ 7:40 am, while the bottom place of Figure 5 (a) shows the trajectories during 6th October 18:00 ∼ 18:20 pm from the same source S and destination D. As Figure 5 (a – f) shows, given the same S and D, the normal trajectories may change during different travel times. And the time-dependent outliers may be different during different travel times for the same (S, D) pair. It verifies that outliers may change over time.

7 RELATED WORK

Graphs are prevalent to many data mining tasks [10, 11, 19, 20, 26–29]. The large volume of trajectory data generated from GPS-equipped vehicles enables extensive trajectory mining applications in recent years, e.g., trajectory pattern mining [12–14], routing [32, 33], traffic prediction [7, 46], trajectory similarity search [21, 38], and trajectory outlier detection [1, 42]. In particular, trajectory outlier detection has been extensively studied recently, which can be used to detect taxi driving fraud.

There are two categories of trajectory outlier detection methods: non-time-dependent and time-dependent outlier detection.

Non-time-dependent Trajectory Outlier Detection: There are two kinds of standard trajectory outlier detection methods, which are metric-based and learning-based methods. Metric-based methods summarize popular routes from normal trajectories, and define anomaly scores based on distance metrics, e.g., Euclidean distance, Hausdorff distance, Longest Common Sub-Sequence (LCSS) distance, and Dynamic Time Warping (DTW) distance, to find outliers [31]. Distance [35] and density based methods [5, 24] have been proposed to detect taxi fraud. A partition-and-detect framework named TRAOD [18] has been proposed to detect outliers from trajectories. They first partition the trajectories into sub-trajectories, then detect outliers based on distance and density metrics. An isolation-based approach IBAT [42] utilizes an isolation tree to find trajectories that are few and different as outliers. Another isolation-based method IBOAT [1] has been proposed to fix the issue of detecting incomplete trajectories. They utilize an adaptive window to update anomaly scores as trajectories come. A clustering-based approach [16] detects anomalies by dynamic time warping distance between trajectories. A Shannon entropy-based method [9] has been proposed to apply information theory for detection. Hausdorff distance-based methods [8, 17] detect outliers in an online mode. Structural similarity [41] is utilized to detect outliers. Common slices sub-sequences [40] based method utilizes distance to detect anomalies. DBOTD [25] extracts core routes from clusters of trajectories by a density-based cluster method DBSCAN [4].

However, metric-based methods heavily rely on distance thresholds based on popular routes in terms of distance or density metrics.

Learning-based methods utilize machine learning or deep learning to learn features behind normal behaviors, and use these features to judge outliers. A k classifier ensemble-based liner projection model LODA [34] is proposed to detect anomalies in an online mode. However, it cannot capture the spatial transition pattern behind trajectories. A probabilistic model DB-TOD [37] utilizes maximum entropy inverse reinforcement learning to optimize the likelihood of historical trajectories. And it also supports the detection of partial observation. However, it fails to consider sequential transition behind trajectories, which is essential for learning the behavior of trajectories. An RNN based model [36] has been proposed to learn sequential transition behind trajectories. A GAN [22] based model [6] has been proposed for anomaly detection in human mobility. However, these two methods are not designed to support the detection of incomplete trajectories. GM-VSAE [23] extends VAE [15] to learn hidden patterns behind trajectory transition, and use a generative framework to detect outliers. SD-VSAE [23] is further proposed to speed up the inference process of online detection.

Time-dependent Trajectory Outlier Detection: There are some works exploring time-dependent trajectory outlier detection. TPRO [44] was proposed to find popular routes from trajectories grouped by the same time intervals. Then a time-dependent edit distance is proposed to detect outliers. An extended model TPRRO [45] based on TPRO is further proposed to speed up the process of popular routes retrieval in real-time. They propose to cache the popular routes for the most frequently visited source and destination pair in the offline phase.

There are two open questions for time-dependent outlier detection. First, the time complexities of [44, 45] are quadratic to the number of trajectory points, which cannot satisfy the online detection requirement well. They propose to extract time-dependent normal routes and use time-dependent edit distance to detect outliers. However, the additional time-dependent edit distance computation increases the time cost dramatically, which requires $O(n^2)$ time cost , making it unsuitable for online detection scenarios. Second, traditional metric-based methods group trajectories by different time intervals in one day, and apply anomaly score function for trajectories at the same time intervals. In this way, they assume that trajectories happen at the same time intervals on different days follow exactly the same traffic condition. However, traffic conditions change dramatically when some factors happen, e.g., incidents or road construction.

8 CONCLUSIONS

In this paper, we study the problem of online time-dependent anomalous trajectory detection. We propose to learn the latent patterns from trajectories during the travel time, and use the learned latent patterns to detect anomalies from trajectories. For online detection, DeepTEA supports updating the anomaly score when the new trajectory point comes, in which the trip has not been finished. Further, we propose an approximate online detection algorithm DeepTEA-A to reduce the time cost by a co-training network. The experimental results show that our models are more effective when detecting outliers. Besides, DeepTEA-A can detect online time-dependent anomalous trajectory more efficiently. Further, both DeepTEA and DeepTEA-A are scalable with respect to the ratio of observed trajectories for online detection.

ACKNOWLEDGEMENT

Reynold Cheng, Chenhao Ma, and Xiaolin Han were supported by the University of Hong Kong (Projects 104005858 and 10400599). Tobias Grubenmann was partially supported by the Federal Ministry of Education and Research (BMBF), Germany under Simple-ML (01IS18054) and the European Commission under PLATOON (872592) and Cleopatra (812997).

REFERENCES

[1] 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.

[2] 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).

[3] Youcef Djenouri, Djamel Djenouri, and Jerry Chun-Wei Lin. 2021. Trajectory Outlier Detection: New Problems and Solutions for Smart Cities. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 2 (2021), 1–28.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *kdd*, Vol. 96. 226–231.

[5] Yong Ge, Hui Xiong, Chuanren Liu, and Zhi-Hua Zhou. 2011. A taxi driving fraud detection system. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 181–190.

[6] 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).

[7] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2021. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[8] Yuejun Guo and Anton Bardera. 2019. SHNN-CAD+: an improvement on shnn-cad for adaptive online trajectory anomaly detection. *Sensors* 19, 1 (2019), 84.

[9] Yuejun Guo, Qing Xu, Peng Li, Mateu Sbert, and Yu Yang. 2017. Trajectory shape analysis and anomaly detection utilizing information theory tools. *Entropy* 19, 7 (2017), 323.

[10] Xiaolin Han, Reynold Cheng, Tobias Grubenmann, Silviu Maniu, Chenhao Ma, and Xiaodong Li. 2022. Leveraging Contextual Graphs for Stochastic Weight Completion in Sparse Road Networks. In *SIAM International Conference on Data Mining*. SIAM.

[11] Xiaolin Han, Daniele Dell'Aglio, Tobias Grubenmann, Reynold Cheng, and Abraham Bernstein. 2022. A framework for differentially-private knowledge graph embeddings. *Journal of Web Semantics* 72 (2022), 100696.

[12] Xiaolin Han, Tobias Grubenmann, Reynold Cheng, Sze Chun Wong, Xiaodong Li, and Wenya Sun. 2020. Traffic incident detection: A trajectory-based approach. In *IEEE ICDE*. 1866–1869.

[13] Hoyoung Jeung, Man Lung Yiu, and Christian S Jensen. 2011. Trajectory pattern mining. In *Computing with spatial trajectories*. Springer, 143–177.

[14] Younghoon Kim, Jiawei Han, and Cangzhou Yuan. 2015. TOPTRAC: Topical trajectory pattern mining. In *SIGKDD*. 587–596.

[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR* (2013).

[16] Dheeraj Kumar, James C Bezdek, Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. 2017. A visual-numeric approach to clustering and anomaly detection for trajectory data. *The Visual Computer* 33, 3 (2017), 265–281.

[17] Rikard Laxhammar and Göran Falkman. 2013. Online learning and sequential anomaly detection in trajectories. *IEEE transactions on pattern analysis and machine intelligence* 36, 6 (2013), 1158–1173.

[18] 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 (ICDE)*. IEEE, 140–149.

[19] Xiaodong Li, Reynold Cheng, Kevin Chen-Chuan Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. 2021. On analyzing graphs with motif-paths. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1111–1123.

[20] Xiaodong Li, Reynold Cheng, Matin Najafi, Kevin Chang, Xiaolin Han, and Hongtai Cao. 2020. M-Cypher: A GQL Framework Supporting Motifs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3433–3436.

[21] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 617–628.

[22] Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. *Advances in neural information processing systems* 29 (2016), 469–477.

[23] 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.

[24] Zhipeng Liu, Dechang Pi, and Jinfeng Jiang. 2013. Density-based trajectory outlier detection algorithm. *Journal of Systems Engineering and Electronics* 24, 2 (2013), 335–340.

[25] 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 (DASFAA)*. Springer, 231–246.

[26] Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, Tobias Grubenmann, Yixiang Fang, and Xiaodong Li. 2019. LINC: a motif counting algorithm for uncertain graphs. *Proceedings of the VLDB Endowment* 13, 2 (2019), 155–168.

[27] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2022. A Convex-Programming Approach for Efficient Directed Densest Subgraph Discovery. In *SIGMOD*.

[28] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient algorithms for densest subgraph discovery on large directed graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1051–1066.

[29] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2021. On Directed Densest Subgraph Discovery. *TODS* 46, 4 (2021), 1–45.

[30] 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).

[31] Fanrong Meng, Guan Yuan, Shaoqian Lv, Zhixiao Wang, and Shixiong Xia. 2019. An overview on trajectory outlier detection. *Artificial Intelligence Review* 52, 4 (2019), 2437–2456.

[32] Simon Aagaard Pedersen, Bin Yang, and Christian S Jensen. 2020. Fast stochastic routing under time-varying uncertainty. *The VLDB Journal* 29, 4 (2020), 819–839.

[33] Simon Aagaard Pedersen, Bin Yang, and Christian S Jensen. 2020. A hybrid learning approach to stochastic routing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1910–1913.

[34] Tomáš Pevnỳ. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102, 2 (2016), 275–304.

[35] Ignacio San Román, Isaac Martín de Diego, Cristina Conde, and Enrique Cabello. 2019. Outlier trajectory detection through a context-aware distance. *Pattern Analysis and Applications* 22, 3 (2019), 831–839.

[36] 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.

[37] 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 (CIKM)*. 837–846.

[38] Dong Xie, Feifei Li, and Jeff M Phillips. 2017. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1478–1489.

[39] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*. 802–810.

[40] Qingying Yu, Yonglong Luo, Chuanming Chen, and Xiaohan Wang. 2018. Trajectory outlier detection approach based on common slices sub-sequence. *Applied Intelligence* 48, 9 (2018), 2661–2680.

[41] Guan Yuan, Shixiong Xia, Lei Zhang, Yong Zhou, and Cheng Ji. 2011. Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems* 7, 11 (2011), 4137–4144.

[42] 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 (UbiComp)*. 99–108.

[43] Guanjie Zheng, Susan L Brantley, Thomas Lauvaux, and Zhenhui Li. 2017. Contextual spatial outlier detection with metric learning. In *SIGKDD*. 2161–2170.

[44] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. 2015. Time-dependent popular routes based trajectory outlier detection. In *WISE*. 16–30.

[45] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. 2017. Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web* 20, 1 (2017), 111–134.

[46] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. 2018. Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns.. In *IJCAI*. 3732–3738.