Online Anomalous Trajectory Detection with Deep Generative Sequence Modeling

Yiding Liu† Kaiqi Zhao‡ Gao Cong† Zhifeng Bao§
†*Nanyang Technological University*, ‡*University of Auckland*, §*RMIT University*
† {liuy0130@e., gaocong@}ntu.edu.sg ‡ kaiqi.zhao@auckland.ac.nz
§ zhifeng.bao@rmit.edu.au

***Abstract*—Detecting anomalous trajectory has become an important and fundamental concern in many real-world applications. However, most of the existing studies 1) cannot handle the complexity and variety of trajectory data and 2) do not support efficient anomaly detection in an online manner. To this end, we propose a novel model, namely Gaussian Mixture Variational Sequence AutoEncoder (GM-VSAE), to tackle these challenges. Our GM-VSAE model is able to (1) capture complex sequential information enclosed in trajectories, (2) discover different types of normal routes from trajectories and represent them in a continuous latent space, and (3) support efficient online detection via trajectory generation. Our experiments on two real-world datasets demonstrate that GM-VSAE is more effective than the state-of-the-art baselines and is efficient for online anomalous trajectory detection.**

I. Introduction

With the proliferation of mobile devices and sensor technologies (e.g., GPS), a huge amount of location traces, i.e., *trajectories*, are being generated at an unprecedented speed [1]–[5]. For example, tens of thousand of taxis travel in a modern city everyday, generating massive volumes of trajectories. These trajectories contain rich information about the mobility of people, vehicles, goods and services. With the availability of large-scale trajectory data, automatically detecting anomalous trajectories has become a critical concern in many real-world scenarios. For example, many tourists are victims of taxi driving frauds, i.e., the taxi drivers take unnecessary detours to overcharge the passengers.

Intuitively, a trajectory is considered as anomalous if it does not follow the *normal routes* of a specific travel itinerary (i.e., from a source S to a destination D) [6], [7], where a *route* represents the path taken by a trajectory in the real world. Figure 1 shows two examples. Suppose we have a set of trajectories between a source S_1 and a destination D_1, and most of them follow two routes r_1 (purple line) and r_2 (blue line), respectively. We can see that the trajectory T_1 (red dashed line) is an anomaly because it does not follow the normal routes r_1 and r_2. Another case shown in Figure 1 is that the normal trajectories between S_2 and D_2 usually follow r_3 and r_4 (solid lines), while an anomalous trajectory T_2 (red dashed line) switches its route from r_3 to r_4, i.e., it does not follow r_3 or r_4 completely. This may indicate an unusual condition (e.g., road construction, car accident) on the sub-route r'_3.

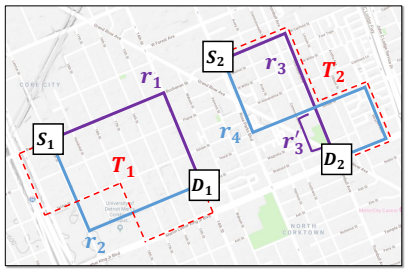


Fig. 1. Examples of anomalous trajectory detection.

However, detecting such anomalous trajectories is non-trivial due to the following two challenges:

- **Discovering normal routes**. The key of trajectory anomaly detection is to effectively discover and represent the normal routes of trajectories. This is challenging because the normal routes of trajectories may vary across different places due to the complexity of transportation systems, and thus are difficult to be predefined and represented. Also, the sequential correlation between different segments of a normal route should also be considered, as it is imporatnt for detecting route-switching anomalies as shown in Figure 1.
- **Efficient online detection**. To allow actions to be promptly taken when anomalies occur, it is more desirable to support efficient *online detection* of anomalous trajectories, i.e., updating the anomaly score of a trajectory while it is being generated sequentially. However, it is challenging because trajectories are often generated at a very high speed and in a massive scale. For example, 790 million GPS points of taxi trajectories were generated in Beijing in a period of 3 months [8], [9].

In recent years, anomalous trajectory detection has attracted extensive research attention and various methods have been proposed [6], [7], [10]–[14]. However, none of them solves the above two challenges simultaneously. In particular, most of the previous studies [6], [10]–[14] use hand-crafted features that usually fail at discovering normal routes of trajectories. Their heuristic definitions of "normal routes" cannot handle the *variety*, *complexity* and *sequential correlation* of the routes traveled by real-world trajectories. Worse still, there is very few research work on tackling the challenge of online detection. Chen *et al.* [10] are the first to study the problem of online anomalous trajectory detection. However, their proposed method needs to refer to the existing trajectories

when updating the anomaly score for a trajectory being generated, which is very inefficient. Wu *et al.* [7] also try to detect potential anomalies with partial observations. However, their method needs to complete a partial trajectory with the shortest path algorithm before the detection, which increases the computational cost significantly and cannot support "on-the-fly" online detection.

In this paper, we propose a novel solution for online anomaly detection of trajectories based on a deep generative model, namely Gaussian Mixture Variational Sequence AutoEncoder (GM-VSAE). GM-VSAE consists of a *route inference network* and a route-guided *generative network*, which are designed to handle the two aforementioned challenges, respectively.

To address the challenge of discovering normal routes, we propose an inference network that employs a recurrent neural network (RNN) to encode the complex sequential information underlying the trajectories, and represent the route of each trajectory as a vector in a continuous latent space. Meanwhile, a Gaussian mixture model is jointly learned to model the probability distribution of routes represented in the latent space. The distribution characterizes the probability of *being normal* of all the routes of trajectories. We also reveal that different Gaussian components can be interpreted as different types of routes, such as streets in downtown areas and highways that connect distant locations. As a result, in the latent space, routes that are close to the center of a Gaussian component are more likely to be normal routes, because they have higher probability to be traveled.

To address the challenge of efficient online detection, we design a novel *detection-via-generation* scheme using the generative network. The intuition is that the anomalous trajectories do not follow normal routes, and thus cannot be *well-generated* from normal routes using the generative network. Therefore, we detect a trajectory by computing its likelihood of being generated from normal routes, where the likelihood can be computed online and updated in $O(1)$ time. Nevertheless, the time cost of trajectory generation using GM-VSAE would increase by a factor of the number of components in the Gaussian mixture distribution. To this end, we further propose an approximate posterior inference method that selects only one component from the mixture distribution to generate and detect a trajectory, which significantly reduces the time cost of GM-VSAE. By doing this, the proposed detection-via-generation scheme enables efficient online anomaly detection.

Overall, our contributions can be summarized as follows:

- We develop a deep generative model GM-VSAE, which is powerful at capturing the sequential information of trajectories and represent their routes in a continuous latent space. We jointly use a Gaussian mixture distribution to model the route representations, which allow us to discover different types of normal routes to support effective anomalous trajectory detection.
- We propose a novel paradigm, *detection-via-generation*, for anomalous trajectory detection using GM-VSAE. To the best of our knowledge, we are the first to use the generation scheme to detect anomalous trajectories, which can be computed online and updated in $O(1)$ time. In addition, we propose an approximate posterior inference method that largely reduces the time cost of GM-VSAE and enables a more efficient online detection.
- We design an all-around evaluation procedure on two large-scale real-world trajectory datasets. We conduct extensive experiments on detecting anomalous trajectories with both partial and complete observations of the trajectories. The results show that our proposed GM-VSAE model is more effective than the state-of-the-art baselines and is efficient for online detection.

II. Related Work

A. Trajectory Anomaly Detection

We divide the existing studies of anomalous trajectory detection into two categories: metric-based methods and learning-based methods. The first category is usually based on hand-crafted features and contains two steps: 1) defining "normal routes" with representative trajectories and 2) comparing a target trajectory with representative trajectories based on distance or density metrics. For example, an early study [11] propose to detect anomalous trajectory segments via computing the distance of each segment of the target trajectory to other trajectories. Zhang *et al.* [6] propose to detect trajectory anomalies by checking how much a trajectory can be isolated from reference trajectories with the same itinerary. This method is further extended by considering the order of the locations in a trajectory [10]. Lv *et al.* [14] and Zhu *et al.* [12] both propose new edit distance metrics for trajectory comparison, which are applied for mining normal patterns. In addition, Ge *et al.* [15] try to combine distance and density metrics to detect taxi frauds. However, this category of methods have two drawbacks. Firstly, their heuristic definitions of representative trajectories are usually hand-crafted with many predefined parameters, such as the frequency or density thresholds of being "representative". As trajectories between different places are usually very different, those definitions might only be effective for several scenarios but fail for many other situations. Secondly, the trajectory comparison is usually time-consuming and cannot support efficient online detection.

As machine learning techniques, especially sequence modeling methods, are widely applied for modeling trajectories [16]–[22], some studies also propose learning-based methods for anomalous trajectory detection. Wu *et al.* [7] leveraged a probabilistic model that learns to detect anomalies for both the whole trajectories and partial trajectories. However, the linearity of the model makes it hard to capture complex information behind the data. Also, it does not take sequential information of trajectories into consideration. Another recent proposal [23] develops a supervised model with recurrent neural network (RNN) for anomalous trajectory detection. However, the model requires labeled data that is usually unavailable in real applications. Recently, Gray *et al.* [24] propose an adversarial learning method for anomalous trajectory detection. Nevertheless, their method cannot handle

trajectories with variable lengths. Moreover, it is not suitable for online detection of anomalous trajectories. These limit its application to real-world scenarios. Other studies [25]–[27] on other types of anomalous trajectories, such as detecting visually and temporally anomalous trajectories, which propose totally different solutions, will not be discussed further.

B. Anomaly Detection with Deep Representation Learning

Due to the powerful capacity of extracting complex information from large-scale data, many deep learning methods have been applied to the problem of anomaly detection [28]–[31]. Most of them leverage autoencoder (AE) to learn the latent features of data and compute anomaly scores using reconstruction (i.e., decoding) error. However, conventional autoencoders are both ineffective and inefficient for online anomalous trajectory detection. First, they lack the means of mining useful information about trajectory anomalousness in the latent embedding space, i.e., they may only embed trajectories in a low-dimensional space, without exploring which part in the latent space represents *normal* and which part stands for *anomalous*. Second, as a new GPS point of a trajectory comes, it needs to re-encode and re-decode the whole trajectory to update the anomaly score, and thus is inefficient for online detection. Our GM-VSAE model is different from conventional autoencoders in two-fold. First, GM-VSAE can characterize and memorize different types of normal routes using the latent Gaussian components in the embedding space. Second, we can detect a trajectory by generating it from the normal routes in an online manner, which saves the time cost of the encoding step (i.e., route inference step) and can support efficient online detection.

III. Problem Formulation

A. Problem Definition

A trajectory is defined as a sequence of chronologically ordered points $P = \{p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n\}$, where each point p_i is represented by GPS coordinates. Intuitively, a trajectory can be considered as a representative (i.e., sample) of a *route*, which represents the continuous path in the physical world the trajectory follows.

As illustrated in Figure 1, there usually exist *normal routes* within a specific travel itinerary, which have relatively high probability to be traveled by trajectories (e.g., r_1 and r_2 between S_1 and D_1 in Figure 1). Based on this, we can intuitively consider those trajectories that *do not* follow those normal routes as *anomalous trajectories*, such as T_1 and T_2 in Figure 1.

More formally, we formulate these intuitions using probabilities as follows:

- We define the probability of a route r being traveled by trajectories as $p(r)$. Higher $p(r)$ means r is more likely to be a normal route.
- We specify the probability of a trajectory T following a normal route (denoted as r_*) as $p(T|r_*)$. Lower $p(T|r_*)$ means T is more likely to be anomalous.

Next, we follow the previous studies [6], [17] to evenly partition a geographical space into discrete grids (denoted as $\mathcal{G}$) and map the GPS coordinates to the grids. By doing this, a trajectory can be represented by a sequence of grid tokens, i.e., $T = \{t_1 \rightarrow t_2 \rightarrow ... \rightarrow t_n\}$. Other methods such as map matching [32] to tokenize GPS coordinates is also applicable [7]. We represent the *source* and the *destination* of a trajectory T as $S_T = t_1$ and $D_T = t_n$. Note that S_T and D_T are both given *before* a trajectory T starts to be recorded [6]. Finally, we formulate the problem of online anomalous trajectory detection as:

*Definition 1 (**Online Anomalous Trajectory Detection**):* Given an ongoing trajectory T being generated, its source S_T and destination D_T, the purpose of Online Anomalous Trajectory Detection problem is to 1) discover the normal routes between S_T and D_T and, 2) compute and update the probability of T following the normal route, i.e., $p(T|r_*)$, while T is sequentially generated.

B. Overview of Our Solution

After formulating the problem, our main goals of designing an anomalous trajectory detection method become

- discovering normal routes by modeling $p(r)$, and
- detecting anomalous trajectories by modeling $p(T|r_*)$.

To achieve these goals, we develop a novel solution to solve the online anomalous trajectory detection problem based on a deep generative model, namely Gaussian mixture variational sequence autoencoder (GM-VSAE), which is inspired by previous studies on variational autoencoder (VAE) [33], [34].

In particular, we first learn a GM-VSAE model that can 1) infer and represent routes of trajectories as vectors in a latent embedding space, 2) model the probability distribution of the routes (i.e., modeling $p(r)$), and 3) generate trajectories given a specific route (i.e., modeling $p(T|r)$).

Based on the learned GM-VSAE model, we develop an efficient anomalous trajectory detection framework. The framework can 1) discover normal routes by exploiting the learned probability distribution of routes, and 2) detect a trajectory via computing its probability of being generated from normal routes. Furthermore, we propose an approximation module, namely SD-network (SDN), to further improve the detection efficiency.

Our solution enables an effective and efficient detection of anomalous trajectories. In the following, we introduce GM-VSAE, the detection framework and the SD-network module in Section IV, V and VI, respectively.

IV. Gaussian Mixture Variational Sequence Autoencoder

Figure 2 shows the architecture of the proposed GM-VSAE, which consists of the following components:

- **Inference network**. The inference network aims to infer the route of a given trajectory T and representing it as a vector $\mathbf{r}_T$ in a continuous latent space.

TABLE I
DESCRIPTION OF NOTATIONS.

Symbols	Descriptions
T	the token sequence of a trajectory
t_i	the ith grid token of a trajectory
n	the length of a trajectory
(S, D)	a pair of source and destination
$\mathcal{G}$	a set of all the grids on a geographical space
$\mathbf{r}$	the representation of a route in the latent space
k	the dimensionality of the RNN hidden vectors
d'	the dimensionality of the input grid embeddings
M	the dimensionality of the latent space of route
C	the number of route types
ϕ	the parameters of the inference network
γ	the parameters of the latent Gaussian mixture distribution
θ	the parameters of the generative network

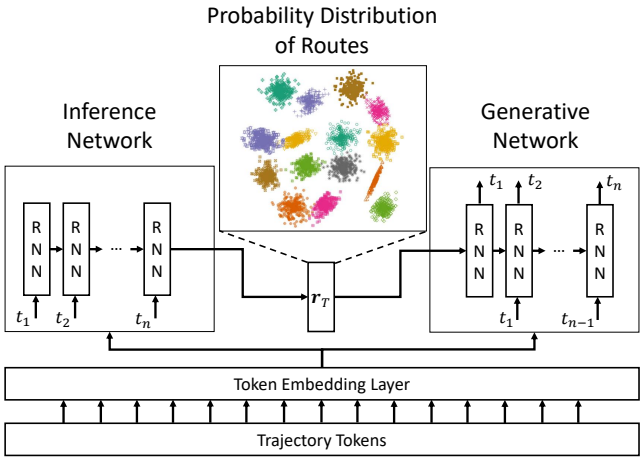


Fig. 2. The overall architecture of GM-VSAE.

- **Probability distribution of routes.** We use a probability distribution to model the routes and represent how likely a route can be considered as a normal route.
- **Generative network.** The generative network is designed with the ability of generating trajectories from a given route.

For GM-VSAE, our main goal is to jointly train these components, such that 1) the probability distribution can well characterize the routes of trajectories in the latent space, and 2) the generative network can generate trajectories from latent routes. The probability distribution of routes and the generative network will be subsequently used for online anomalous trajectory detection in Section V. In the rest of this paper, we use the terms "*latent route*", "*latent route space*" and "*latent route distribution*" to denote the vector representation of routes, the latent space and the probability distribution of routes, respectively. Table I lists the notations used in the model. In the next three subsections, we elaborate each of the three components.

A. Latent Route Inference.

Before we exploit routes and route distribution for anomalous trajectory detection, we need to infer and represent the routes of trajectories. To achieve this, we propose a route inference network $q_\phi(\mathbf{r}|T)$ to infer the latent route (i.e., the route representation vector) $\mathbf{r}_T \in \mathbb{R}^M$ for a given trajectory $T = (t_1, t_2, ..., t_n)$. Here, ϕ represents the parameters of $q_\phi(\mathbf{r}|T)$ and M represents the dimensionality of the latent route space.

In particular, we implement $q_\phi(\mathbf{r}|T)$ with a recurrent neural network (RNN), which accepts variable-length trajectories as inputs and can capture the sequential information behind trajectories. Specifically at each step i, the inference RNN reads t_i and produces a hidden state $\mathbf{h}_i \in \mathbb{R}^k$ by:

$$\mathbf{h}_i = f_1(t_i, \mathbf{h}_{i-1}) \quad i = 1, 2, ..., n, \tag{1}$$

where f_1 is a deterministic non-linear function to be learned, which can be implemented with Long Short-Term Memory (LSTM) [35] or Gated Recurrent Unit (GRU) [36]. By doing this, the sequential information of T is encoded in the latent representation $\mathbf{h}_n$. Note that the RNN accepts inputs in the form of real-valued vectors. Thus, we introduce a token embedding layer to embed the discrete token (i.e., t_i) in a vector.

To cope with the possible uncertainties and noise (e.g., GPS error) behind trajectory T, we infer $\mathbf{r}_T$ by drawing from a posterior distribution as

$$\mathbf{r}_T \sim q_\phi(\mathbf{r}|T) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\sigma}_T^2 \mathbf{I}) \tag{2}$$

where $\boldsymbol{\mu}_T \in \mathbb{R}^M$, $\boldsymbol{\sigma}_T \in \mathbb{R}^M$ and $\{\boldsymbol{\mu}_T, \boldsymbol{\sigma}_T\} = g_1(\mathbf{h}_n)$. The function $g_1(\cdot)$ is a non-linear function to be learned and $\mathbf{I}$ is an identity matrix. With the help of RNN, we are able to capture the sequential information of trajectories and represent their latent routes in the latent route space.

B. Modeling Latent Route with Gaussian Mixture Distribution.

With the ability of inferring and representing the routes of trajectories in the latent route space, we still lack knowledge about which routes in the space are *normal routes*, or which routes are more normal than the others. Motivated by this, we aim at modeling the latent routes with a probability distribution, which can be used to measure how likely a given route can be considered as a normal route. However, this is non-trivial as the routes in the real world are usually diverse and complicated. For example, vehicles may travel in different areas in an urban city (e.g., downtown, suburb) and on different types of roads (e.g., major/minor highway, ramp, street). It is non-trivial to model such complex route distribution.

To tackle this challenge, we propose to model the latent route distribution with a *Gaussian mixture distribution*, which is comprehensive for modeling various types of latent routes. In particular, we first assume there are C different *types* of routes underlying the trajectory data, where C is a hyper-parameter of the model. Different types of routes have different semantic meanings (e.g., road type, traveling area). To model different types of latent routes, we introduce two probability distributions:

- A multinomial distribution $p_\gamma(c) = Mult(\boldsymbol{\pi})$, where $\boldsymbol{\pi} \in \mathbb{R}^C$ are the parameters. It models the probability of a specific type (denoted as c) of routes being traveled by trajectories.
- A Gaussian distribution $p_\gamma(\mathbf{r}|c) = \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2\mathbf{I})$ for each type c, where $\boldsymbol{\mu}_c \in \mathbb{R}^M$ and $\boldsymbol{\sigma}_c \in \mathbb{R}^M$ are the mean and standard deviation vectors. It models the probability of a route $\mathbf{r}$ of type c being traveled by trajectories.

Here, we use $\gamma = \{\boldsymbol{\pi}, \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c\}$ to denote all the parameters in $p_\gamma(c)$ and $p_\gamma(\mathbf{r}|c)$. These two distributions together model the latent route space as $p_\gamma(\mathbf{r}) = p_\gamma(\mathbf{r}|c)p_\gamma(c)$.

After incorporating the latent route type c, we need to extend the inference network in Eq. (2) to $q_\phi(\mathbf{r}, c|T)$ to infer the route type c in addition to the latent route $\mathbf{r}$. Specifically, we apply the mean-field approximation to factorize $q_\phi(\mathbf{r}, c|T)$ as

$$q_\phi(\mathbf{r}, c|T) = q_\phi(\mathbf{r}|T)q_\phi(c|T), \tag{3}$$

where $q_\phi(\mathbf{r}|T)$ is defined in Eq. (2). For $q_\phi(c|T)$, it can be intuitively considered as the probability of route type c given the route of a trajectory T, i.e., $q_\phi(c|T) := p_\gamma(c|\mathbf{r}_T)$. Thus, $q_\phi(c|T)$ can be defined as

$$q_\phi(c|T) := p_\gamma(c|\mathbf{r}_T) = \frac{p_\gamma(c)p_\gamma(\mathbf{r}_T|c)}{\sum_{i=1}^{C} p_\gamma(c_i)p_\gamma(\mathbf{r}_T|c_i)}, \tag{4}$$

where $\mathbf{r}_T$ is drawn from $q_\phi(\mathbf{r}|T)$ as in Eq. (2). The intuition is that the probability of T traveling a route of type c (i.e., $p_\gamma(c|\mathbf{r}_T)$) is proportional to the probability $p_\gamma(\mathbf{r}_T|c)$.

C. Recurrent Trajectory Generation.

Next, we need to formalize the probability of a trajectory T following a given latent route $\mathbf{r}$, which enables the detection of anomalous trajectories in Section V. To achieve this, we design a novel generative network $p_\theta(T|\mathbf{r})$, where θ is the set of parameters. The objective is to optimize $p_\theta(T|\mathbf{r})$ such that a trajectory T can be well generated from its inferred latent route $\mathbf{r}_T$.

However, different from other generation tasks (e.g., image generation), the generation of a trajectory is naturally sequential, i.e., the mobility of an object at a specific step is correlated with its previous trace. Therefore, we resort to a recurrent generation process with the help from an RNN. Specifically at each step i, we consider that the generation of t_i is related to 1) the latent route $\mathbf{r}$, and 2) the previous sequence $t_{<i}$, where $t_{<i} = \{t_1 \to t_2 \to ... \to t_{i-1}\}$. Thus, we generate t_i via $t_i \sim p_\theta(t_i|t_{<i}, \mathbf{r})$. In particular, we first recurrently encode the information of $\mathbf{r}$ and the sequence $t_{\leq i} = \{t_1 \to t_2 \to ... \to t_i\}$ into a hidden vector $\mathbf{g_i} \in \mathbb{R}^k$ as:

$$\mathbf{g}_i = f_2(t_i, \mathbf{g}_{i-1}) \quad i = 1, 2, ..., n \quad \mathbf{g}_0 = \mathbf{r}, \tag{5}$$

where f_2 is a deterministic non-linear function to be learned (e.g., GRU cell, LSTM cell). Note that the same token embedding layer that is used in the inference network is also applied here before feeding inputs to the RNN. Based on these, at each step i, we can generate t_i from a multinomial distribution as

$$t_i \sim p_\theta(t|t_{<i}, \mathbf{r}) = p_\theta(t|\mathbf{g}_{i-1}) = Mult(\text{softmax}(g_2(\mathbf{g}_{i-1}))). \tag{6}$$

Here, the output function $g_2(\cdot)$ converts $\mathbf{g}_{i-1}$ to a $|\mathcal{G}|$-dimensional vector, where $|\mathcal{G}|$ denotes the total number of grid tokens. Then, we use a softmax function to get a probability vector to parameterize the multinomial distribution. By doing this at each time step, a trajectory can be recurrently generated.

D. Training GM-VSAE

Objective of joint optimization. The parameters to be learned in GM-VSAE include $\phi = \{f_1(\cdot), g_1(\cdot)\}$, $\gamma = \{\boldsymbol{\pi}, \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c\}$, $\theta = \{f_2(\cdot), g_2(\cdot)\}$ and the token embeddings. Intuitively, the objective of training GM-VSAE is to jointly optimize the inference and generative networks, and the latent route distribution, such that the generative network can properly generate a trajectory with its route inferred by the inference network. More formally, the objective is to maximize the marginal log-likelihood of generating the training data:

$$\log p_\theta(T^{(1)}, T^{(2)}, ..., T^{(N)}) = \sum_{j=1}^{N} \log p_\theta(T^{(j)}), \tag{7}$$

where $T^{(1)}, T^{(2)}, ..., T^{(N)}$ are the N trajectories in the training data. We derive the evidence lower bound on the marginal likelihood of each trajectory T (denoted as $\mathcal{L}(\theta, \gamma, \phi; T)$) as

$$\begin{aligned}\log p_\theta(T) \geq \mathcal{L}(\theta, \gamma, \phi; T) &= \mathbb{E}_{q_\phi(\mathbf{r}, c|T)}\left[\log \frac{p_{\theta,\gamma}(T, \mathbf{r}, c)}{q_\phi(\mathbf{r}, c|T)}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{r}|T)}\big[\log p_\theta(T|\mathbf{r})\big] - \mathbb{E}_{q_\phi(\mathbf{r}|T)}\big[D_{KL}(q_\phi(c|T)||p_\gamma(c))\big] \\ &\quad - \mathbb{E}_{q_\phi(c|T)}\big[D_{KL}(q_\phi(\mathbf{r}|T)||p_\gamma(\mathbf{r}|c))\big],\end{aligned} \tag{8}$$

where $D_{KL}(\cdot||\cdot)$ represents the KL-divergence between two distributions. For the log-probability $\log p_\theta(T|\mathbf{r})$, we can further decompose it as the sum of the log-likelihood of generating each $t_i \in T$, i.e., $\log p_\theta(T|\mathbf{r}) = \sum_{i=1}^{n} \log p_\theta(t_i|t_{<i}, \mathbf{r})$. Therefore, we can rewrite the first term in Eq. (8) as

$$\mathbb{E}_{q_\phi(\mathbf{r}|T)}[\log p_\theta(T|\mathbf{r})] = \mathbb{E}_{q_\phi(\mathbf{r}|T)}[\sum_{i=1}^{n} \log p_\theta(t_i|t_{<i}, \mathbf{r})]. \tag{9}$$

We train GM-VSAE via maximizing the lower bound.

Alternative optimization scheme. To maximize $\mathcal{L}(\theta, \gamma, \phi; T)$, we need to optimize three sets of parameters, i.e., ϕ, γ and θ, which parameterize the inference network, latent route distribution and the generative network, respectively. Jointly optimizing all the parameters would be difficult and unstable. Therefore, we adopt an alternative optimization scheme, which alternatively updates two sets of parameters $\{\phi, \theta\}$ and γ. We apply reparameterization trick and Stochastic Gradient Variational Bayes (SGVB) estimator [33] to train the model in an end-to-end manner with alternative gradient decent.

Complexity analysis. The average time complexity for training GM-VSAE is $O(Nk(k + d')\bar{n} + NkM)$, where N represents the number of trajectories in the training data, k represents the size of RNN hidden states, M is the dimensionality of the latent route space, d' represents the dimensionality of the input grid embeddings and $\bar{n}$ stands for the average length of the training trajectories. Intuitively, the first term

$O(Nk(k+d')\bar{n})$ represents the time cost of encoding and decoding trajectories, and the second term $O(NkM)$ is the time cost of mapping between RNN hidden states and latent route representations, e.g., mapping the last encoder state to a latent route representation. The value of M is usually set to be similar or equal to k. When $M = k$, the training complexity can be written as $O(Nk(k+d')\bar{n})$. We can see that the training time of GM-VSAE scales linearly w.r.t. the number of training trajectories.

V. Efficient Online Anomalous Trajectory Detection via Trajectory Generation

In this section, we design a novel *detection-via-generation* framework for online anomalous trajectory detection by exploiting the probability distribution $p_\gamma(\mathbf{r}|c)$ and the generative network $p_\theta(T|\mathbf{r})$ that are learned by GM-VSAE.

In particular, we first consider *normal routes* as those routes which are more likely to be traveled by trajectories. With the learned Gaussian probability distribution $p_\gamma(\mathbf{r}|c) = \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2\mathbf{I})$, the normal routes of a specific type c can be considered as the routes that are close to $\boldsymbol{\mu}_c$, i.e., $\mathbf{r}_*^c \simeq \boldsymbol{\mu}_c$. Based on this, we use $\boldsymbol{\mu}_c$ as normal routes to approximate $\mathbf{r}_*^c$. We will analyze the accuracy of this approximation later.

Next, we need to detect the probability of a given trajectory T following $\boldsymbol{\mu}_c$. By leveraging the generative network $p_\theta(T|\mathbf{r})$, we propose a *detection-via-generation* scheme, where the probability of T following $\boldsymbol{\mu}_c$ is implemented as the probability of T being generated from $\boldsymbol{\mu}_c$. The main advantage of this scheme is that the detection can be updated online while the trajectory is being sequentially generated, which is very efficiency. We will analyze the time complexity in the end of this section.

In particular, we define the anomaly score of a given trajectory T using $p_\theta(T|\boldsymbol{\mu}_c)$ as

$$s(T) = 1 - \arg\max_c \exp\Big[\frac{\log p_\theta(T|\boldsymbol{\mu}_c)}{n}\Big], \tag{10}$$

where $\exp(\cdot)$ is the exponential function and $\log p_\theta(T|\boldsymbol{\mu}_c) = \sum_{i=1}^{n} \log p_\theta(t_i|t_{<i}, \boldsymbol{\mu}_c)$. Figure 3 illustrates the detection procedure using GM-VSAE. Note that $\log p_\theta(T|\boldsymbol{\mu}_c)$ is divided by n, i.e., the length of a trajectory T, to normalize trajectories with different lengths.

For online detection, i.e., detecting an ongoing trajectory (denoted as $t_{\le i} = \{t_1 \to t_2 \to ... \to t_i \to\}$), we maintain the probability $p_\theta(t_{\le i}|\boldsymbol{\mu}_c)$ for all $\boldsymbol{\mu}_c$. When a new location point is recored, i.e., $t_{\le i}$ becomes $t_{\le i+1}$, we can derive the new anomaly score as

$$s(t_{\le i+1}) = 1 - \arg\max_c \exp\Big[\frac{\log p_\theta(t_{\le i}|\boldsymbol{\mu}_c) p_\theta(t_{i+1}|t_{\le i}, \boldsymbol{\mu}_c)}{i+1}\Big], \tag{11}$$

Here, we have $s(T) \in [0.0, 1.0]$ and high $s(T)$ indicates that T *cannot* be well generated from *any* type of normal route (i.e., has low $p_\theta(T|\boldsymbol{\mu}_c)$ values), and thus is more likely to be an anomaly. To have a deterministic definition of anomalous trajectories, one can set a threshold Δ on the anomaly score, i.e., T is detected to be anomalous if $s(T) > \Delta$.

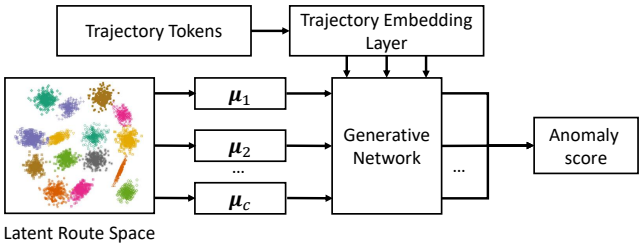


Fig. 3. The framework of anomalous trajectory detection using GM-VSAE.

Approximation ratio. We analyze the accuracy of using $\boldsymbol{\mu}_c$ to approximate $\mathbf{r}_*^c$. Based on Bayes' theorem, we have the following equation

$$\frac{p_\gamma(\mathbf{r}_*^c|c)p_\theta(T|\mathbf{r}_*^c)}{q_\phi(\mathbf{r}_*^c|T)} = p_{\gamma,\theta}(T|c) = \frac{p_\gamma(\boldsymbol{\mu}_c|c)p_\theta(T|\boldsymbol{\mu}_c)}{q_\phi(\boldsymbol{\mu}_c|T)}. \tag{12}$$

Thus, we derive the approximation ratio $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} = \frac{p_\theta(T|\boldsymbol{\mu}_c)}{p_\theta(T|\mathbf{r}_*^c)} = \frac{p_\gamma(\mathbf{r}_*^c|c)q_\phi(\boldsymbol{\mu}_c|T)}{p_\gamma(\boldsymbol{\mu}_c|c)q_\phi(\mathbf{r}_*^c|T)}. \tag{13}$$

Based on $\mathbf{r}_*^c \simeq \boldsymbol{\mu}_c$, we denote $\mathbf{r}_*^c = \boldsymbol{\mu}_c + \boldsymbol{\delta}_*$, where $\boldsymbol{\delta}_*$ is a small offset. We substitute the probability density functions of all the Gaussian distributions in Eq. (13) to finalize $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} = \exp\Big[\frac{\boldsymbol{\delta}_*^2(\boldsymbol{\sigma}_c^2 - \boldsymbol{\sigma}_T^2) + 2\boldsymbol{\delta}_*\boldsymbol{\sigma}_c^2(\boldsymbol{\mu}_c - \boldsymbol{\mu}_T)}{2\boldsymbol{\sigma}_c^2\boldsymbol{\sigma}_T^2}\Big]. \tag{14}$$

We can see from Eq. (14) that $\boldsymbol{\tau}$ is related to $\boldsymbol{\delta}_*$. If $\boldsymbol{\delta}_* \to \mathbf{0}$, we would have a good approximation ratio, i.e, $\boldsymbol{\tau} \to \mathbf{1}$. We consider a latent route $\mathbf{r}$ with a very high probability $p_\gamma(\mathbf{r}|c)$ as a normal route $\mathbf{r}_*^c$. This implies $\mathbf{r}_*^c \to \boldsymbol{\mu}_c$ and $\boldsymbol{\delta}_* \to \mathbf{0}$. Therefore, $\boldsymbol{\delta}$ has a very small value empirically, and can give us a good approximation ratio $\boldsymbol{\tau} \to \mathbf{1}$.

Complexity analysis. To compute $s(T)$, we need to generate T for C times, i.e., using different $\boldsymbol{\mu}_c$. According to Eq. (10), the time complexity of computing $s(T)$ is $O(CkM + Ck(k+d')n)$, where $O(kM)$ is the complexity of mapping between RNN states and latent route vectors, and $O(k(k+d'))$ is the complexity of updating the RNN hidden state $\mathbf{g_i} \in \mathbb{R}^k$ in each step $i = 1, 2, ..., n$. Because M, k, d' and C are constants, the time complexity of computing $s(T)$ is therefore $O(n)$. For online detection, the time complexity of updating $s(T)$ for an ongoing trajectory is $O(Ck(k+d'))$. It is because updating the log-likelihood $\log p_\theta(T|\boldsymbol{\mu}_c)$ only requires to compute the RNN hidden state $\mathbf{g_{i+1}}$ for the new step $i+1$ and then compute $p_\theta(t_{i+1}|\mathbf{g}_i)$. We omit the constants k, d' and C and finalize the time complexity of updating $s(T)$ as $O(1)$.

VI. Improving Efficiency with Approximate Inference of Route Type

Although GM-VSAE has a constant time complexity for online detection, the time cost is related to C, i.e., the number of Gaussian components in the latent route distribution. If C is large, the detection would still be slow. To address this,

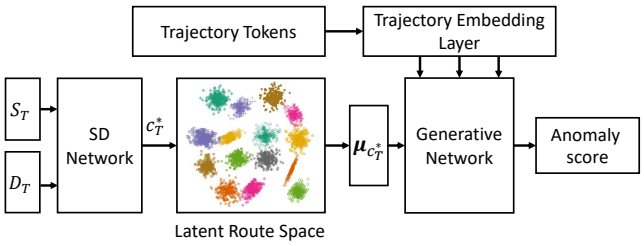


Fig. 4. The framework of anomalous trajectory detection using SD-VSAE.

we proposed to infer the route type c_T^* that is the most likely to generate a trajectory T. By doing this, we only need to generate a trajectory from only *one* route type $\boldsymbol{\mu}_{c_T^*}$.

A straightforward way is to use the route type c with the maximum posterior probability $q_\phi(c|T)$, i.e. $c_T^* = \arg\max_c q_\phi(c|T)$. However, this is not practical for online detection due to two reasons. First, the posterior inference $q_\phi(c|T)$ might be inaccurate with partial trajectory, i.e., while the trajectory is not complete. Second, $q_\phi(c|T)$ might be updated when a new location is recorded for the ongoing trajectory T. If $c_T^* = \arg\max_c q_\phi(c|T)$ is changed after updating $q_\phi(c|T)$, we need to re-generate T with the new c_T^* to update the anomaly score, which takes $O(n)$ time in the worst case for online detection.

To this end, we propose an approximation method to infer c_T^* for a trajectory T and achieve $O(1)$ time complexity for updating the anomaly score. We name the model SD-VSAE, which is extended from GM-VSAE. The intuition is that the normal routes between a specific source-destination pair (S, D) usually have the same or similar type. For example, vehicles usually drive through highway between two distant places. Motivated by this, we propose to infer c_T^* with S_T and D_T. As S_T and D_T are given *before* T is being generated, the inference of c_T^* can also be done beforehand.

SD-network. We propose a new module, namely SD-network (SDN), to distill the information of posterior $q_\phi(c|T)$ into a new probability distribution $q_\eta(c|S_T, D_T)$, which is parameterized by η. Specifically, we first introduce two extra embedding layers of grid tokens, i.e., S-embedding layer and D-embedding layer, respectively. Therefore, we can represent the S_T and D_T of a trajectory T as continuous vectors $\mathbf{s}_T$ and $\mathbf{d}_T$, respectively. Then, we concatenate $\mathbf{s}_T$ and $\mathbf{d}_T$ and feed it to a multi-layer perceptron as $MLP_L([\mathbf{s}_T, \mathbf{d}_T])$ where L denotes the number of layers in the MLP. We compute the probability values on the C types of routes as

$$q_\eta(c|S_T, D_T) = \text{softmax}(\mathbf{o}_L), \tag{15}$$

where $\mathbf{o}_L \in \mathbb{R}^C$ represents the final-layer output of the MLP. Since computing $q_\eta(c|S_T, D_T)$ can be done before T is being generated and does not need to be recurrently updated like $q_\phi(c|T)$, the time complexity of inferring c_T^* using $q_\eta(c|S_T, D_T)$ is $O(1)$, which is much more efficient than using $q_\phi(c|T)$, whose time complexity is $O(n)$.

Training SDN via knowledge distillation. We train the SDN module via knowledge distillation, i.e., using the original posterior of route type $q_\phi(c|T)$ as the supervision to train $q_\eta(c|S_T, D_T)$. In particular, we add one more step in the alternative optimization of $\mathcal{L}(\theta, \gamma, \phi; T)$. The objective of training SDN is to minimize the cross entropy between $q_\eta(c|S_T, D_T)$ and $q_\phi(c|T)$, i.e., $H(q_\phi, q_\eta) = -\sum_{c=1}^{C} q_\phi(c|T) \log q_\eta(c|S_T, D_T)$.

After training SD-VSAE, we can replace $q_\phi(c|T)$ with $q_\eta(c|S_T, D_T)$ to infer c_T^*. Subsequently, the online trajectory generation (i.e., detection) can be done with the inferred route type $\boldsymbol{\mu}_{c_T^*}$, where $c_T^* = \arg\max_c q_\eta(c|S_T, D_T)$. SD-VSAE is therefore C times faster than GM-VSAE. Figure 4 summarizes the detection procedure using SD-VSAE.

VII. Discussion

We discuss two extensions of our proposed methods (i.e., GM-VSAE and SD-VSAE) for different problem settings.

A. Handling Concept Drift of Anomalous Trajectories

As trajectories are usually recorded in a streaming manner, the concept of "normal route" and "anomalous trajectory" may change over time. For example, there are two places connected by a road r_{old}. When a more convenient (e.g., shorter distance) road r_{new} is constructed between the two places, drivers may gradually prefer to travel the new road than the old road. Thus, r_{new} should be considered as a normal route, while new trajectories traveling r_{old} will be considered as anomalous trajectories. To tackle this issue, we perform online update for the current model by feeding newly recorded trajectory data. Given a trajectory stream, we keep updating the model using stochastic gradient descent.

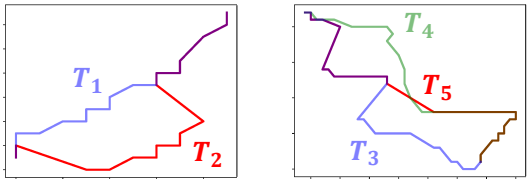
Moreover, we have conducted an experiment to validate this strategy, where the detailed results are not presented due to the space limitation. We observe from the experiments that the model with fine-tuning has a consistent performance when new trajectories are being continually recorded, while the performance of the model without fine-tuning keeps decreasing.

B. Detection of Other Types of Anomalous Trajectories

As introduced in Section III-A, this paper mainly focuses on detecting anomalous trajectories that do not follow normal routes. It is worth noting that there are also other anomalous trajectory detection problems with different settings. For example, Banerjee *et al.* [27] propose to detect *over-speeding* and *under-speeding* trajectories, i.e., the definition of anomalous trajectories is based on their *temporal* attributes. Our proposed methods can also be extended to solve those problems with different problem definitions. For detecting anomalous trajectories w.r.t. a particular sequential pattern, we just need to encode the patterns of trajectories into a low-dimensional vector (denoted by $\mathbf{x}$), and model $p_\gamma(\mathbf{x})$, $q_\phi(\mathbf{x}|T)$ and $p_\theta(T|\mathbf{x})$ using the inference network, the Gaussian mixture distribution and the generative network, respectively. Taking the over/under-speeding trajectory detection as an example, we can feed the speed features of a trajectory to the inference and generative network, instead of feeding grid token embeddings. Also, the generative network should be modified to generate the corresponding speed values of a given trajectory at different

TABLE II
TRAJECTORY DATASETS AFTER PREPROCESSING.

Dataset	#Points	#Trajectories	#(S, D)	Avg. n
Porto	11,635,104	262,574	4,567	44.31
Beijing	909,642	52,497	6,752	17.32



(a) Detour (D). (b) Route-switching (RS).

Fig. 5. Examples of generated anomalies.

time steps. By doing this, our proposed methods are able to capture the normal speed patterns in the low-dimensional latent space, and our detection-via-generation scheme can naturally be deployed to detect speed anomalies.

VIII. EXPERIMENTS

A. Experimental Setup

Datasets & Preprocessing. Our experiments are conducted on two real-world taxi trajectory datasets. In particular, the first dataset (Porto)[1] is generated from 442 taxis running in the city of Porto during Jan 07 2013 to Jun 30 2014. Each taxi reports its location at 15 second intervals. Following previous study [17], we partition the geographical space into 100m×100m grids and filter those (S, D) pairs with fewer than 25 trajectories. The filtered dataset contains 262,574 trajectories. The second dataset (Beijing) is a subset of T-Drive trajectory dataset[2] [8], [9], which contains a one-week trajectories of 10,357 taxis. We partition the geographical space into 300m×300m grids and select the grid cells with the highest trajectory density (e.g., top-200 popular cells) as S or D. We use the trips between the popular cells as the trajectories in the experiments. To choose trajectories with reasonable traveling distance and sampling rate, we further select the trajectories with a length of at least 10 and with time gaps between consecutive sample points being between 10 seconds and 10 minutes. We also filter those (S, D) pairs with fewer than 5 trajectories and finally obtain a dataset with 52,497 trajectories. Table II presents the statistics of the filtered datasets.

Ground truth. Since there is no labeled dataset available for anomalous trajectory detection, some existing studies [6], [10], [14] try to manually label anomalies. However, it is very time-consuming and often limits the evaluation on just a few (S, D) pairs. Following previous work on spatial outlier detection [37], we use *generated anomalies* for the evaluation.

In particular, we use two different perturbation schemes to generate two types of anomalous trajectories, namely detour anomalies (D) and route-switching anomalies (RS), respectively. The first perturbation scheme moves a random part of a trajectory along a direction that is roughly perpendicular to the moving direction of the trajectory. Figure 5(a) shows an example of a normal trajectory T_1 (blue line) and the generated anomaly T_2 (red line). We can see that it corresponds to the case where a moving object takes a long detour. The second perturbation scheme is exampled in Figure 5(b). In particular, we first randomly select two trajectories T_3 and T_4 between the same (S, D), each of which can be divided into two sub-trajectories, i.e., $T_3 = T_3^{(1)} \rightarrow T_3^{(2)}$ and $T_4 = T_4^{(1)} \rightarrow T_4^{(2)}$, where $T^{(1)}$ and $T^{(2)}$ represent the first and the second part of T. Then, we replace $T_3^{(2)}$ with $T_4^{(2)}$, i.e., injecting an anomaly $T_5 = T_3^{(1)} \rightarrow T_4^{(2)}$ to the dataset.

TABLE III
DESCRIPTION OF NOTATIONS USED IN EXPERIMENTS.

Symbol	Description
α	the proportion of the detour segment of detour anomalies
d	the moving distance (i.e., offset) of the detour segment
β	the split location of generating route-switching anomalies
ρ	the proportion of a trajectory being observed for detection

We parameterize the injection of detour anomalies with two parameters d and α, where d represents the moving distance of the detour segment and α defines the proportion of the detour part. For example, $\alpha = 0.3$ and $d = 5$ means 30% of a trajectory is moved with a distance of 5 grids. We parameterize the injection of route-switching anomalies with β, which stands for the split location of a trajectory. For example, $\beta = 0.3$ means $T^{(1)}$ contains the earliest 30% points of T and $T^{(2)}$ contains the remaining 70%. The notations of the injection parameters are listed in Table III. In the experiments, we vary d, α and β, and evaluate all the methods on detecting the two types of anomalies separately. Following previous studies [6], [7], [37], the proportion of the injected anomalies in each dataset is set to 5%. Note that the proportion of the injected anomalies in the training data would not largely affect the performance of our methods.

Baselines. We compare these methods in our experiments:

- **TRAOD** [11]. It first partitions all trajectories to segments. The anomaly score of a trajectory is based on the total length of its anomalous segments, which are distant to the other trajectories with the same (S, D).
- **T-DBSCAN** [14]. This method is a density-based method. It clusters trajectories with the same (S, D) using DBSCAN [38]. The anomaly score of a trajectory is computed as its distance to the closest "core trajectories" found in the clustering procedure.
- **iBAT** [6]. This method detects anomaly by checking how much the target trajectory can be isolated from reference trajectories with same (S, D) pair.
- **DBTOD-embed** [7]. This method considers driving speed, road levels and turning angle as features and learns a probabilistic model that can automatically detect anomalous trajectories. However, to make a fair comparison to other baselines that do not use any side information, we use a grid

[1]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data
[2]https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

token embedding vector to represent the features, which is jointly trained with the probabilistic model.

- **Sequence autoencoder (SAE)** [28]. This is a conventional RNN-based sequence to sequence model, which is learned via minimizing a reconstruction error. The anomaly score is defined based on the reconstruction error.
- **Variational sequence autoencoder (VSAE)**. This is a degenerated model of GM-VSAE, where the latent route distribution is a Gaussian distribution.
- **GM-VSAE**. This is our proposed model. We set $C = \{10, 20, 50\}$ on Porto data, and $C = 5, 10, 20$ on Beijing data. For example, GM-VSAE10 stands for the GM-VSAE model with 10 Gaussian components.
- **SD-VSAE**. This is extended from GM-VSAE model with SD-network module to improve the efficiency.

The implementation details and the source code of our models are available on `https://git.io/JelML`.

Evaluation metrics. The output of a detection method for each trajectory is an anomaly score between 0.0 and 1.0. We use Precision-Recall AUC (PR-AUC) as the metric [37]. PR-AUC is appropriate to evaluate anomaly detection methods on skewed datasets [37]. Note that the number of anomalies we inject into the trajectory datasets are only small portions of all the trajectories. We compute the PR-AUC value for each (S, D) and report the average values over all (S, D).

B. Effectiveness Evaluation

Varying observed ratios. To evaluate the performance of *online* detection, we use each method to detect the anomalousness of a target trajectory with different *observed ratios* (denoted as $\rho \in [0.0, 1.0]$). For example, $\rho = 0.1$ means to detect a trajectory when we only observe the earliest 10% locations of the trajectory. We use $\rho = 1.0$ to represent the detection on complete trajectories. Note that TRAOD and T-DBSCAN cannot detect anomalous trajectories in an online manner. Therefore, we only evaluate them on detecting complete trajectories (i.e., $\rho = 1.0$).

Detecting detour anomalies. We first compare all the methods on detecting detour anomalies. In particular, we apply different parameters to inject detour anomalies in both datasets, where $d = \{3, 5\}$ and $\alpha = \{0.1, 0.3\}$. We set $\rho = \{0.1, 0.5, 1.0\}$ in the experiments. The trajectories in Beijing dataset are usually very short, i.e., 10% of a trajectory may only contain one or two points. Therefore, we only use $\rho = \{0.5, 1.0\}$ on Beijing. Tables IV and V show the PR-AUC of detecting detour anomalies using different methods on Porto and Beijing datasets, respectively.

For detecting complete trajectories (i.e., $\rho = 1.0$), we can see that GM-VSAE is significantly better than all the baselines on both datasets. For example, when $d = 3$ and $\alpha = 0.1$, GM-VSAE20 outperforms the best baseline (i.e., SAE) by 13.1% on Porto, and GM-VSAE10 outperforms all the baselines by at least 8% on Beijing. We also find out that the improvements of VSAE over the distance and density based methods (e.g., TRAOD, T-DBSCAN) are more significant on Porto data than on Beijing data. For example, when $d = 5$ and $\alpha = 0.1$, VSAE improves TRAOD by 163.3% on Porto while only 56.3% on Beijing. This indicates that using RNN can more effectively capture useful information and reveal normal routes on large-scale dense data (i.e., Porto) than on sparse data (i.e., Beijing). Comparing with VSAE, GM-VSAE provides more accurate detection on both datasets. For example, GM-VSAE10 outperforms VSAE by 10.3% and 8.5%, on Porto and Beijing, respectively, when setting $d = 3$ and $\alpha = 0.1$. This is because GM-VSAE models latent routes using Gaussian mixture distribution, which can provide more useful insights (i.e., route types) to detect anomalous trajectories.

For detecting partial trajectories ($\rho = \{0.1, 0.5\}$), we can see that GM-VSAE is also the best among all the methods. For example, when $\rho = 0.5$, $d = 5$ and $\alpha = 0.1$, GM-VSAE10 outperforms iBAT and SAE by 21.5% and 7.1% on Beijing. Furthermore, at the very early stage of a trajectory (i.e., $\rho = 0.1$), the performance improvement of GM-VSAE is even better than the other models. For example, when $d = 5$ and $\alpha = 0.1$, GM-VSAE20 improves VSAE by 92.5% when $\rho = 0.1$, while only 6.0% when $\rho = 1.0$ on Porto data. This is because GM-VSAE mines the information about route types, which can provide extra confidence in anomaly detection even when the trajectory generation has just started. Hence, GM-VSAE would be more useful for launching early-warnings when detecting anomalous trajectories.

Detecting route switching anomalies. Tables IV and V also include the results of detecting route-switching anomalies using different methods on Porto and Beijing, respectively. In these experiments, we use $\beta = \{0.3, 0.5, 0.7\}$ to inject anomalies and report the results on detecting both partial and complete trajectories. For detecting partial trajectories, we set $\rho = \{0.5, 0.7, 0.9\}$ for different β values correspondingly. The results are qualitatively similar to the results of detecting detour anomalies. We observe that GM-VSAE outperforms all the baselines with respect to different β and ρ, in terms of PR-AUC. For example, GM-VSAE5 performs better than DBTOD-embed and T-DBSCAN by 46.8% and 51.7%, respectively, on Beijing when $\beta = 0.5$ and $\rho = 1.0$.

GM-VSAE vs. SD-VSAE. Next, we study the effectiveness of using SD-network (SDN) to distill the information of route types. We compare the effectiveness of GM-VSAE10 and SD-VSAE10 by detecting both detour anomalies (D) and route-switching anomalies (RS), where the injection parameters are set as $d = 3$, $\alpha = 0.3$ and $\beta = 0.3$. We can see from Figure 6 that SD-VSAE has comparable performance to GM-VSAE for detecting both types of anomalies on both datasets, in terms of PR-AUC. For example, when detecting detour anomalies (D) with complete trajectories ($\rho = 1.0$) on Porto data, the PR-AUC value of SD-VSAE only decreases by 0.94% compared to GM-VSAE. This indicates that our SDN module can effectively infer the route type for a given trajectory. It is worth noting that the effectiveness of SD-VSAE decreases less on Porto than Beijing. This is because each (S, D) in Porto has more trajectories, and thus can provide more information for inferring the route type between (S, D).

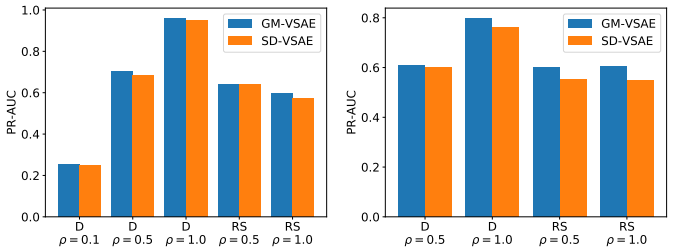
Influence of the number of Gaussian components. We

TABLE IV
PERFORMANCE COMPARISON FOR ANOMALOUS TRAJECTORY DETECTION IN TERMS OF PR-AUC ON PORTO DATASET.

	Detecting detour anomalies (D)									Detecting route-switching anomalies (RS)					
Perturb params	d=3; $\alpha = 0.1$			d=5; $\alpha = 0.1$			d=3; $\alpha = 0.3$			$\beta = 0.3$		$\beta = 0.5$		$\beta = 0.7$	
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
TRAOD	-	-	0.212	-	-	0.311	-	-	0.161	-	0.189	-	0.183	-	0.179
T-DBSCAN	-	-	0.231	-	-	0.305	-	-	0.253	-	0.240	-	0.245	-	0.201
iBAT	0.156	0.184	0.181	0.159	0.196	0.193	0.182	0.371	0.406	0.200	0.179	0.177	0.173	0.177	0.175
DBTOD-embed	0.148	0.146	0.277	0.148	0.141	0.279	0.159	0.297	0.384	0.138	0.285	0.171	0.292	0.240	0.292
SAE	0.152	0.46	0.717	0.154	0.502	0.824	0.176	0.666	0.921	0.469	0.396	0.459	0.424	0.440	0.426
VSAE	0.170	0.455	0.701	0.174	0.501	0.819	0.197	0.660	0.910	0.614	0.566	0.619	0.571	0.605	0.591
GM-VSAE10	0.230	0.473	0.773	0.234	0.513	0.842	0.253	0.702	0.961	0.640	0.597	0.636	0.606	0.634	0.618
GM-VSAE20	0.334	0.494	**0.811**	0.335	0.522	**0.868**	**0.361**	0.711	**0.969**	**0.717**	**0.662**	0.721	0.678	**0.720**	0.706
GM-VSAE50	**0.338**	**0.498**	**0.811**	**0.337**	**0.523**	**0.868**	0.358	**0.714**	0.963	0.715	0.660	**0.725**	**0.705**	0.718	**0.707**

TABLE V
PERFORMANCE COMPARISON FOR ANOMALOUS TRAJECTORY DETECTION IN TERMS OF PR-AUC ON BEIJING DATASET.

	Detecting detour anomalies (D)						Detecting route-switching anomalies (RS)					
Perturb params	d=3; $\alpha = 0.1$		d=5; $\alpha = 0.1$		d=3; $\alpha = 0.3$		$\beta = 0.3$		$\beta = 0.5$		$\beta = 0.7$	
Observed ratio (ρ)	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.7	1.0	0.9	1.0
TRAOD	-	0.333	-	0.343	-	0.265	-	0.297	-	0.283	-	0.288
T-DBSCAN	-	0.245	-	0.270	-	0.288	-	0.422	-	0.451	-	0.458
iBAT	0.383	0.352	0.396	0.364	0.413	0.367	0.500	0.477	0.472	0.448	0.480	0.468
DBTOD-embed	0.326	0.485	0.311	0.487	0.366	0.512	0.345	0.446	0.354	0.466	0.414	0.475
SAE	0.416	0.464	0.449	0.495	0.559	0.691	0.458	0.425	0.476	0.461	0.464	0.459
VSAE	0.421	0.506	0.461	0.536	0.588	0.741	0.454	0.445	0.469	0.467	0.465	0.472
GM-VSAE5	0.439	0.528	**0.481**	0.559	**0.612**	0.792	**0.665**	**0.650**	**0.667**	**0.684**	**0.659**	**0.663**
GM-VSAE10	**0.454**	**0.549**	**0.481**	**0.598**	0.608	**0.799**	0.608	0.599	0.606	0.621	0.618	0.622
GM-VSAE20	0.434	0.511	0.457	0.565	0.588	0.746	0.576	0.556	0.586	0.596	0.591	0.589



(a) Porto (b) Beijing.

Fig. 6. GM-VSAE10 vs. SD-VSAE10 (D: $d = 3$, $\alpha = 0.3$; RS: $\beta = 0.3$).

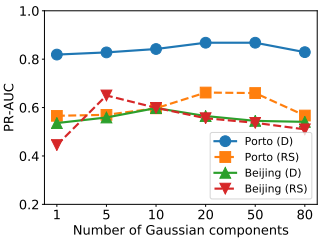


Fig. 7. Varying the number of Gaussian components of GM-VSAE (D: $d = 5$, $\alpha = 0.1$; RS: $\beta = 0.3$).

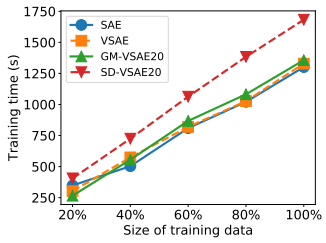


Fig. 8. Comparison of training scalability.

conduct experiments for GM-VSAE with different number of Gaussian components (i.e., the value of C). In particular, we vary the value of C from 1 to 80, and use the GM-VSAE models to detect detour and route-switching anomalies on both datasets. Figure 7 shows the experimental results. We can see that 1) the performance of our models depend on C, but even our least performed one beats the existing methods; 2) GM-VSAE achieves better results when $C = \{20, 50\}$ on Porto data, and $C = \{5, 10\}$ on Beijing data.

C. Efficiency Evaluation

Overall detection efficiency. Figure 9 presents the average runtime of detecting a trajectory using all the methods on both datasets. Note that the y-axis of Figure 9 is in a *logarithmic* scale. Firstly, we can see that all the methods are faster on Beijing data, where the trajectories are shorter. Furthermore, DBTOD-embed is the fastest method for detecting anomalous trajectories, because it is a linear model with low-dimensional grid embeddings as features, where the detection can be accomplished very quickly. Among the other (i.e., non-linear) models, we can see that VSAE and SD-VSAE10 both have the lowest time cost for anomalous trajectory detection. For example, SD-VSAE10 takes only 0.433ms to detect a trajectory on Porto, while the runtime of TRAOD and T-DBSCAN is around 40ms, which is over 100× slower. VSAE is shown to be approximately 2× faster than SAE, where their

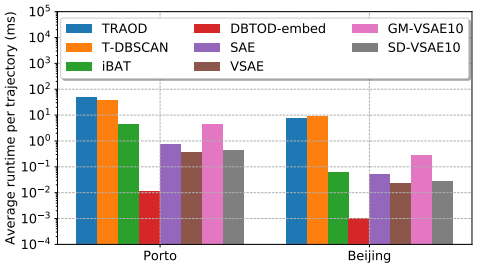
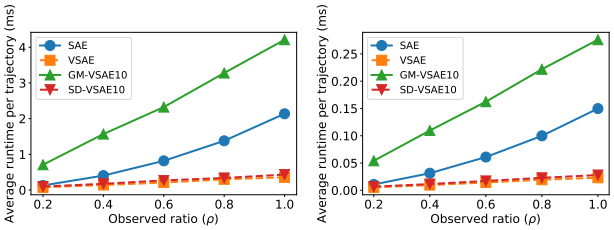


Fig. 9. Efficiency comparison for anomalous trajectory detection.



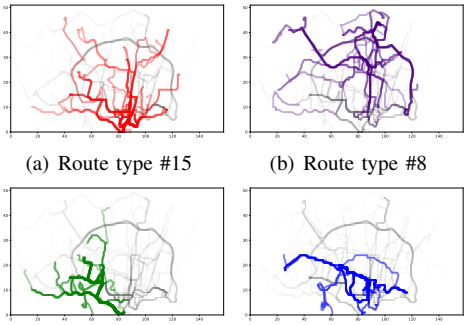
(a) Porto (b) Beijing.

Fig. 10. Accumulated time cost when a trajectory is generated sequentially.

time costs are 0.757ms and 0.357ms, respectively, on Porto data. This is because SAE detects anomalies via trajectory reconstruction, and thus needs to process a given trajectory twice (i.e., encoding and decoding). However, VSAE uses the *detection-via-generation* scheme, which is free of the encoding (i.e., inference) step, and thus is more efficient. Moreover, we find out that SD-VSAE10 is approximately $10\times$ faster than GM-VSAE10. This is because when detecting a target trajectory, GM-VSAE10 needs to generate trajectories from 10 Gaussian components in the latent route distribution, while in SD-VSAE10, only one component selected by the SDN module is needed.

Detection scalability. Furthermore, we compare the runtime of SAE, VSAE, GM-VSAE10 and SD-VSAE10 for detecting ongoing trajectories on both datasets. For detecting each trajectory, we record the time cost accumulated at different observed ratios (i.e., $\rho = \{0.2, 0.4, 0.6, 0.8, 1.0\}$). The reported runtime values are averaged on all the trajectories. Figure 10 shows that (1) the time cost of SAE increases quadratically with the generation of trajectories, while others increase linearly. This is because SAE needs to re-encode the ongoing trajectory when a new location is recorded, whose time complexity of updating the anomaly score is $O(n)$. (2) GM-VSAE10's accumulated time cost increases dramatically with more GPS points being observed for a trajectory; (3) As a trajectory is sequentially generated, the time costs for updating its anomaly score of SD-VSAE and VSAE are the lowest among all these methods. This shows the remarkable efficiency of SD-VSAE for online detection. Along with the effectiveness comparison between GM-VSAE and SD-VSAE shown in Figure 6, we can conclude that SD-VSAE is able to largely reduce the time cost of GM-VSAE for online anomalous trajectory detection, while having similar effectiveness.

Training scalability. We also investigate the training scalability of SAE, VSAE, GM-VSAE20 and SD-VSAE20. We vary the size of Porto data for training these models from 20% to 100% of the whole dataset. All the trainings are conducted on a single NVIDIA Tesla 100 SXM2 GPU. Figure 8 shows the results. We can see from the figure that all the four methods scale *linearly* w.r.t. the size of the training data. Thus, they are promising to handle large-scale trajectory data. Among these methods, the time costs of SAE, VSAE and GM-VSAE20 are very similar, varying from around 300s (20%) to over 1300s (100%). This is because they have similar architectures, where most of the computation can be attributed to the recurrent



(a) Route type #15 (b) Route type #8

(c) Route type #19 (d) Route type #4

Fig. 11. Visualization of trajectories sampled from different route types.

encoding and decoding using RNNs. For SD-VSAE20, it costs more training time than the other three models, as it includes an extra SD-network module, which needs to be trained together with GM-VSAE20. However, we can see that the extra cost is not very high. Compared to GM-VSAE20, SD-VSAE20 only costs an extra 325.6s (i.e., around 5 min) for training on the whole dataset (100%), which is acceptable in real-world applications.

D. Visualization

Different Route Types. We demonstrate the different route types learned by GM-VSAE by visualizing trajectories that belong to a specific route type. Figure 11 visualizes four randomly selected route types from GM-VSAE20 on Porto data. We can see that the trajectories in different route types are different from each other. In particular, most of the trajectories from route type #15 are crowded in the downtown area, while the trajectories from route type #8 cover the highways outside of the central district. Moreover, the trajectories from route type #4 usually travel across the city and trajectories from route type #19 usually commute within the west side of Porto. Therefore, GM-VSAE is able to reveal route types with different semantic meanings, which is helpful for detecting anomalous trajectories.

Case Study. We visualize the trajectories between 16 randomly selected (S, D) pairs on Porto data. Among the trajectories between each (S, D) pair, we highlight the trajectory with the highest anomaly score detected by SD-VSAE20 as shown in Figure 12. Note that in the case study, we do not inject generated anomalies into the dataset, i.e., all the trajectories shown are real-world trajectories. We can see that the anomalous trajectories (i.e., orange dashed lines) are intuitively very different from the majorities of other trajectories (i.e., blue solid lines that overlap) traveling from S to D. This demonstrate the superior capability of our method.

IX. Conclusion

In this paper, we propose a novel deep generative model, namely GM-VSAE, to solve the problem of online anomalous trajectory detection. GM-VSAE can 1) effectively capture complex sequential information of trajectories and model normal route in a latent embedding space, and 2) efficiently detect anomalous trajectories via a novel detection-via-generation

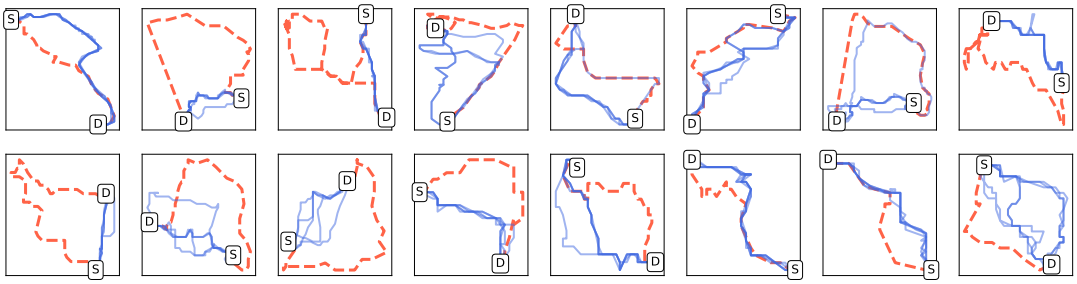


Fig. 12. Case study on Porto data (**the most anomalous trajectory**: orange dashed lines; **other trajectories**: blue solid lines).

scheme. We also develop SD-VSAE that significantly improves the efficiency of GM-VSAE. Experiments on two large-scale datasets demonstrate the superiority of our methods over baseline methods in detecting anomalous trajectories.

Acknowledgment This research is supported by a MOE Tier-2 grant MOE2016-T2-1-137, and a MOE Tier-1 grant RG31/17. Zhifeng Bao was partially supported by ARC DP170102726, DP180102050, and NSFC 61728204, 91646204. We thank the anonymous reviewers for providing useful comments.

REFERENCES

[1] Y. Zheng, "Trajectory data mining: an overview," *ACM TIST*, vol. 6, no. 3, p. 29, 2015.
[2] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and X. Qin, "Fast large-scale trajectory clustering," *PVLDB*, vol. 13, no. 1, pp. 29–42, 2019.
[3] L. Chen, Y. Gao, Z. Fang, X. Miao, C. S. Jensen, and C. Guo, "Real-time distributed co-movement pattern detection on streaming trajectories," *PVLDB*, vol. 12, no. 10, pp. 1208–1220, 2019.
[4] S. Wang, Z. Bao, J. S. Culpepper, Z. Xie, Q. Liu, and X. Qin, "Torch: A search engine for trajectory data," in *SIGIR*. ACM, 2018, pp. 535–544.
[5] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and G. Cong, "Reverse k nearest neighbor search over trajectories," *IEEE TKDE*, vol. 30, no. 4, pp. 757–771, 2017.
[6] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "ibat: detecting anomalous taxi trajectories from gps traces," in *UbiComp*. ACM, 2011, pp. 99–108.
[7] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *CIKM*. ACM, 2017, pp. 837–846.
[8] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *SIGKDD*. ACM, 2011, pp. 316–324.
[9] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *SIGSPATIAL*. ACM, 2010, pp. 99–108.
[10] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iboat: Isolation-based online anomalous trajectory detection," *IEEE TITS*, vol. 14, no. 2, pp. 806–818, 2013.
[11] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *ICDE*. IEEE, 2008, pp. 140–149.
[12] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Time-dependent popular routes based trajectory outlier detection," in *WISE*. Springer, 2015, pp. 16–30.
[13] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Effective and efficient trajectory outlier detection based on time-dependent popular route," *WWWJ*, vol. 20, no. 1, pp. 111–134, 2017.
[14] Z. Lv, J. Xu, P. Zhao, G. Liu, L. Zhao, and X. Zhou, "Outlier trajectory detection: A trajectory analytics based approach," in *DASFAA*. Springer, 2017, pp. 231–246.
[15] Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou, "A taxi driving fraud detection system," in *ICDM*. IEEE, 2011, pp. 181–190.
[16] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *WWW*. ACM, 2019, pp. 1017–1027.
[17] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *ICDE*. IEEE, 2018, pp. 617–628.
[18] C. Ju, Z. Wang, and X. Zhang, "Socially aware kalman neural networks for trajectory prediction," *arXiv preprint arXiv:1809.05408*, 2018.
[19] C. Ju, Z. Wang, C. Long, X. Zhang, G. Cong, and D. E. Chang, "Interaction-aware kalman neural networks for trajectory prediction," *arXiv preprint arXiv:1902.10928*, 2019.
[20] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *ICDE*. IEEE, 2019, pp. 1358–1369.
[21] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *IJCNN*. IEEE, 2017, pp. 3880–3887.
[22] D. Yao, C. Zhang, J. Huang, and J. Bi, "Serm: A recurrent model for next location prediction in semantic trajectories," in *CIKM*. ACM, 2017, pp. 2411–2414.
[23] L. Song, R. Wang, D. Xiao, X. Han, Y. Cai, and C. Shi, "Anomalous trajectory detection using recurrent neural network," in *ADMA*. Springer, 2018, pp. 263–277.
[24] K. Gray, D. Smolyak, S. Badirli, and G. Mohler, "Coupled igmm-gans for deep multimodal anomaly detection in human mobility data," *arXiv preprint arXiv:1809.02728*, 2018.
[25] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee, "Top-eye: Top-k evolving trajectory outlier detection," in *CIKM*. ACM, 2010, pp. 1733–1736.
[26] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *SIGKDD*. ACM, 2014, pp. 422–431.
[27] P. Banerjee, P. Yawalkar, and S. Ranu, "Mantra: a scalable approach to mining temporally anomalous sub-trajectories," in *SIGKDD*. ACM, 2016, pp. 1415–1424.
[28] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.
[29] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *SIGKDD*. ACM, 2017, pp. 665–674.
[30] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *SDM*. SIAM, 2017, pp. 90–98.
[31] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.
[32] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *SIGSPATIAL*. ACM, 2009, pp. 336–343.
[33] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[34] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: an unsupervised and generative approach to clustering," in *IJCAI*. AAAI Press, 2017, pp. 1965–1972.
[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[36] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
[37] G. Zheng, S. L. Brantley, T. Lauvaux, and Z. Li, "Contextual spatial outlier detection with metric learning," in *SIGKDD*. ACM, 2017, pp. 2161–2170.
[38] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *SIGKDD*, vol. 96, no. 34, 1996, pp. 226–231.