Sybil Attack Identification for Crowdsourced Navigation: A Self-supervised Deep Learning Approach

James J.Q. Yu, *Senior Member, IEEE*

***Abstract*—Crowdsourced navigation is becoming the prevalent automobile navigation solution with the widespread adoption of smartphones over the past decade, which supports a plethora of intelligent transportation system services. However, it is subjected to Sybil attacks that inject carefully designed adversarial GPS trajectories to compromise the data aggregation system and cause false traffic jams. Successful Sybil attacks have been launched against real crowdsourced navigation systems, yet defending such critical threats has seldom been studied. In this work, a novel deep generative model based on Bayesian deep learning is devised for Sybil attack identification. The proposed model exploits time-series features to embed trajectories in a latent distribution space, which serves as a basis for identifying ones generated by Sybil attacks. Case studies on three real-world vehicular trajectory datasets reveal that the proposed model improves the performance of state-of-the-art baselines by at least 76.6%. Additionally, a hyper-parameter test develops guidelines for parameter selection, and a fast training scheme is proposed and assessed to boost the model training efficiency.**

***Index Terms*—Sybil attack detection, crowdsourced navigation, intelligent transportation systems, Bayesian deep learning, deep generative model, cybersecurity.**

I. Introduction

With the ever-increasing penetration of mobile devices equipped with location-sensing technologies such as GPS, crowdsourced navigation is rapidly becoming the prevalent approach for vehicular navigation [1]. Crowdsourced navigation systems, such as Google Maps and Waze, collect real-time user navigation data to guide the routing of others. Used by over two billion users worldwide, these systems produce a massive scale of navigation data as a reliable source of real-time traffic information, which is trusted and adopted by a much wider community for services like trip planning and congestion control [2]. Therefore, crowdsourced navigation is among the most critical and fundamental services of modern intelligent transportation systems, whose cybersecurity is of the utmost importance for smart city operation [3]–[6].

Contemporary crowdsourced navigation systems, however, are subjected to various attacks aiming to dramatically influence routing decisions. Among them, the Sybil attack is one of the most straightforward yet effective approaches, where the adversaries compromise the data aggregation system by creating a large number of pseudonymous identities and use them to report forged navigation trajectories [7]. Carefully designed adversarial trajectories are capable of circumventing the bad data detection mechanism in navigation systems and either creating false alarms on traffic jams or preventing real ones from being recognized [8]. Several studies have been carried out on launching Sybil attacks to real-world crowdsourced navigation systems with success, see [8]–[10] for some examples. Successful Sybil attacks can notably impair the stability of modern transportation systems. For instance, a congested main road may attract more road users to drive through if a Sybil attack compromised crowdsourced navigation systems to consider the road under free-flow. Consequently, the congestion is further aggrevated. Futhermore, as the traffic is highly dynamic and correlated, one road compromised may lead to potential breakdowns of the whole transportation network. Properly identifying Sybil attacks is a significant issue to be addressed in ITS.

Despite being a critical threat to intelligent transportation systems, Sybil attacks on crowdsourced navigation systems are not well countered in the literature. Limited results have been published on defending such attacks, and they address the problem from the identity validation [11], traffic network design [12], or crowdsourcing reward [13] viewpoints. As far as we are concerned, there is no comprehensive investigation in identifying such attacks by analyzing the characteristics of user-reported trajectories, so that the navigation is still credible even if the identity or reward systems are compromised. One may note a similar problem that also exists when routing vehicles, namely, anomalous trajectory detection [14]–[16]. Nonetheless, it aims to distinguish authentic vehicular trajectories of fixed source and destination locations that are generated by real sensing devices but follow abnormal routes [16]. Another related research problem in the crowdsourcing context is the truth inference problem where methods are proposed to discover the correct observation to an event or correct answer to a question [17], [18]. A number of studies have been carried out based on voting, optimization, probabilistic graph, or neural networks for the problem [19], [20]. However, their application rely on the crowdsourcing users to answer the same event or question [17]–[20]. In the crowdsourced navigation context, this means that multiple vehicles shall report their respective speeds at exactly the same geographical location and time, which is not practical. Therefore, the respective so-

This work was supported by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032 and by the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation No. 2020B121201001.

James J.Q. Yu is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (email:yujq3@sustech.edu.cn).

lutions cannot be directly adopted to identify adversarial Sybil attack trajectories. Furthermore, as crowdsourced navigation systems aim to provide real-time traffic status to users and ITS services, real-time vehicular trajectories are largely adopted in developing such information. This imposes another challenge to defend Sybil attacks: indentification in real time. If the identifier cannot process incoming trajectories at a fast pace, no real-time traffic information can be developed, rendering crowdsourced navigation invalid.

To bridge this research gap, we present a self-supervised deep learning approach for Sybil attack identification for crowdsourced navigation in this work. The proposed model is capable of learning the latent representation of trajectories as multivariate random variables in latent trajectory space. When the representation is adopted to reconstruct trajectories, adversaries can be distinguished from authentic ones with a credibility scoring scheme. To achieve this, we adopt the recent advances of Bayesian deep learning [21], [22] and deep generative model theories to construct an intelligent identification deep learning model. In the model, a Long Short-Term Memory (LSTM)-like architecture is employed to learn the temporal data correlation within raw data, and the incorporated Bayesian inference principle helps the model to capture the stochastic uncertainties in trajectories. The main contribution of this work is summarized as follows:

- We devise a novel deep generative model for capturing the sequential characteristics of vehicular trajectories with measurement and displacement uncertainties, and representing them in a latent trajectory space as multivariate random variables.
- We propose a novel Sybil attack identification algorithm based on the proposed deep learning model, which is among the pioneer efforts of detecting Sybil attacks in crowdsourced navigation from the trajectory viewpoint.
- We create a novel fast training scheme for the proposed deep learning model based on trajectory clustering, which follows the nature of aggregating regional trajectories and demonstrates sub-linear speedup with parallel training.
- We design a comprehensive evaluation procedure on three large-scale real-world vehicular trajectory datasets to show the efficacy of the proposed model.

The remainder of this paper is organized as follows. Section II presents the definition of the Sybil attack trajectory identification problem in crowdsourced navigation. Section III elaborates on the architecture of the proposed deep generative model with a comprehensive analysis of its training and inference. Section IV introduces the attack identification algorithm based on the proposed model and the fast training scheme. Section V demonstrates the numerical results of case studies with detailed discussions. Finally, this paper is concluded in Section VI.

II. Sybil Attack Identification

In crowdsourced navigation, traffic states are estimated using GPS trajectories generated by users [1]. Sybil attack adversaries compromise the data aggregation system by creating pseudonymous identities and provide forged trajectories to the service providers [7]. By employing a large volume of these identities, a disproportionately large influence can be attained, enabling the adversaries to manipulate the traffic state at will by submitting adversarial trajectories. The Sybil attack identification problem aims to precisely identify these trajectories so that the system integrity does not tamper.

Formally, a GPS trajectory is defined as a sequence of chronologically ordered points $R = \{r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n\}$, where r_t is the t-th GPS record within the trajectory composed of latitude and longitude measurements lat_t, lng_t as well as a sampling timestamp ts_t. From the start location $l_1 = \langle \mathrm{lat}_1, \mathrm{lng}_1 \rangle$ to the end location $l_n = \langle \mathrm{lat}_n, \mathrm{lng}_n \rangle$, there are typically multiple authentic or rational trajectories with particular displacement dynamics that are generated by normal driving behaviors under the respective transient traffic conditions. Others that do not follow the dynamics can be considered adversarial trajectories. Given a trajectory $R \in \mathcal{R}$ where $\mathcal{R}$ is the set of all trajectories in the given dataset, its probability of being traveled and subsequently contributing to crowdsourced navigation is defined as $\Pr(R)$[1]. Let the set of rational trajectories be $\mathcal{R}^* \subset \mathcal{R}$ in which the trajectories follow the "rational" definition given above, the probability of an arbitrary trajectory R following the authentic dynamics is given by $\Pr(R|\mathcal{R}^*)$. Consequently, whether a trajectory is adversarial can be developed from it: a lower $\Pr(R|\mathcal{R}^*)$ indicates a higher probability of an adversarial R.

With the above definitions and notations, Sybil attack identification discovers the rational displacement dynamics and calculates the probability of R following it, i.e., $\Pr(R|\mathcal{R}^*)$. The primary challenge of this problem is twofold. First and foremost, massive stochastic uncertainty exists in GPS trajectories even authentically generated. Developing proper and precise displacement dynamics out of the noisy raw data is a challenging task to be addressed. Second, the historical trajectories are typically of heterogeneous lengths, and ones with the same origin and destination coordinates can be rare. This requires the identifier to capture the latent displacement dynamics within a region instead of learning the travel pattern between fixed origin-destination pairs as did in previous anomalous trajectory detection research.

III. Self-supervised Bayesian Recurrent Autoencoder

To overcome these challenges while identifying adversarial trajectories with efficacy, we propose a novel solution for online Sybil attack identification based on deep learning, named Bayesian Recurrent Autoencoder (BRAE). This model is inspired by recent studies on Bayesian deep learning and deep generative model and tries to combine the merits of both for solving the problem. By representing trajectories as multivariate random variables in a latent trajectory space, the model is capable of modeling the probability distribution of authentic trajectories. Therefore, the model does not require explicit knowledge on the model of Sybil attacks. BRAE can

[1]By abuse of notation, we use R to denote all locations l_i of the trajectory in the sequel.

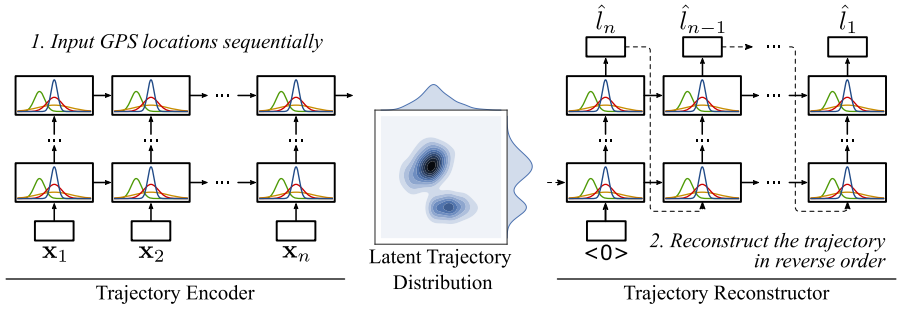


Fig. 1. Overview of BRAE.

be trained by self-supervised learning with authentic and unlabeled trajectories $\mathcal{R}^*$. Based on the model, we further devise a highly-efficient adversarial trajectory detection algorithm for Sybil attack identification.

In this section, we first introduce the architecture of the proposed BRAE model. We then elaborate on the training scheme of this model in detail. Subsequently, we present the attack identification approach based on BRAE in Section IV.

A. Model Architecture

Fig. 1 shows the architecture of the proposed deep learning-based BRAE model. Two major components comprise the neural network, namely, a *trajectory encoder* and a *trajectory reconstructor*. The main objective of BRAE is to train the network model so that the probability distribution of trajectories within the investigating region can be characterized in the latent space as a *latent trajectory distribution*. This latent distribution is capable of reconstructing new trajectories based on the input one, which can be utilized to identify whether an arbitrary trajectory is authentic or adversarial.

1) Trajectory Encoder and Bayesian Neural Network: To exploit the latent distribution, we first propose a trajectory encoder network to embed an arbitrary trajectory R into a latent feature vector. Intuitively, each location l_t within R shall demonstrate a strong temporal correlation — next locations are closely related to the previous ones following an intrinsic stochastic displacement dynamics. To reflect this physical relationship, we implement the encoder with a Bayesian recurrent neural network that integrates Bayesian deep learning with LSTM to handle variable-length trajectory data and capture the stochastic sequential latent data characteristics.

The proposed trajectory encoder is a variant of the recurrent neural network, which takes time-series data sequentially for classification and regression. During the data processing, the encoder holds a latent state $\mathbf{c}_t$ that distills data points up to time step t in the time-series. Together with the new data point input $\mathbf{x}_t$ and generated output $\mathbf{h}_{t-1}$ in the previous time step, the three pieces of data are regulated by three gates that describe how much respective information should be reflected in the network output, namely, forget $\mathbf{f}_t$, input $\mathbf{i}_t$, and output $\mathbf{o}_t$ gates. Specifically, the forget gate determines how much previous information in terms of $\mathbf{h}_t$ will be retained; the input gate controls the acceptance of new input $\mathbf{x}_t$; and the output gate decides what shall be output at this time step. Following this design principle of LSTM [23], the data propagation rule is formulated using perceptrons:

$$[\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t] = \sigma\big(\mathbf{w} \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + b\big), \tag{1a}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\big(\mathbf{w}_c \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + b_c\big), \tag{1b}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{1c}$$

where $\mathbf{w}$, $\mathbf{w}_c$, b, and b_c are trainable parameters, and $\odot$ is the element-wise multiplication operation. We use $\boldsymbol{\phi}$ to denote the collection of the neural network parameters above [24]. When embedding an arbitrary trajectory R, each location l_t is sequentially inputted into the network, i.e., $\mathbf{x}_t = l_t$. Subsequently, the input data is aggregated with the previous output $\mathbf{h}_{t-1}$ and the latent state $\mathbf{c}_{t-1}$ to calculate the current output $\mathbf{h}_t$.

With (1), the encoder is capable of predicting next step trajectory locations. However, such a feature exploitation scheme cannot fully interpret the stochastic uncertainties of GPS trajectories, rendering the primary challenge of Sybil attack identification unresolved. From the probability theory viewpoint, the deterministic trajectory encoder is a probabilistic model $\Pr(\mathbf{h}_t|\mathbf{x}, \boldsymbol{\phi})$. With a fix-valued $\boldsymbol{\phi}$, its training follows the maximum likelihood estimation with the training set $\mathcal{D} = \{\mathbf{y}^i|\mathbf{x}^i\}_i$ where $\mathbf{x}^i$ is the i-th sample in the set and $\mathbf{y}^i$ is the embedding of $\mathbf{x}^i$, i.e., $\mathbf{h}_t$ above:

$$\boldsymbol{\phi}^{\text{MLE}} = \arg\max_{\boldsymbol{\phi}} \Pr(\mathcal{D}; \boldsymbol{\phi}) = \arg\max_{\boldsymbol{\phi}} \sum_i \log \Pr(\mathbf{x}^i; \boldsymbol{\phi}). \tag{2a}$$

When regularization is adopted for overfitting prevention, the a priori is included the model and the parameters follows the maximum a posteriori:

$$\boldsymbol{\phi}^{\text{MAP}} = \arg\max_{\boldsymbol{\phi}} \log \Pr(\boldsymbol{\phi}|\mathcal{D}) \tag{3a}$$

$$= \arg\max_{\boldsymbol{\phi}} \log \Pr(\mathcal{D}|\boldsymbol{\phi}) + \log \Pr(\boldsymbol{\phi}), \tag{3b}$$

where the parameters are fixed values learnt by gradient descent algorithm. If we consider the parameters follow the posterior probability distributions given the training set, i.e., $\Pr(\boldsymbol{\phi}|\mathcal{D})$, the model is capable of exploiting data uncertainties with Bayesian inference [25].

Following this design principle, a Bayesian neural network-based trajectory encoder is constructed, whose output is defined by $q_{\boldsymbol{\phi}}(\mathbf{x}) \equiv \mathbf{h}_t$ with a prior distribution over the parameter space $\Pr(\boldsymbol{\phi})$. The likelihood function for Bayesian regression is therefore defined by $\prod_i l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)$ where the likelihood $l(\cdot)$ can be formulated by a Gaussian distribution.

We subsequently have the learnt posterior distribution over parameters as

$$\Pr(\boldsymbol{\phi}|\mathcal{D}) = \frac{\Pr(\boldsymbol{\phi}) \prod_i l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)}{\int \Pr(\boldsymbol{\phi}) \prod_i l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big) \mathrm{d}\boldsymbol{\phi}}. \tag{4}$$

With $\Pr(\boldsymbol{\phi}|\mathcal{D})$, $q_{\boldsymbol{\phi}}(\mathbf{x})$ is capable of subsequent inference with uncertainty quantification captured by the learnt parameter distributions. Given a newly observed trajectory R', its embedded latent distribution in the latent space is given by

$$\Pr(\hat{\mathbf{y}}'|\mathbf{x}', \mathcal{D}) = \int l\big(\hat{\mathbf{y}}'|q_{\boldsymbol{\phi}}(\mathbf{x}')\big) \Pr(\boldsymbol{\phi}|\mathcal{D}) \mathrm{d}\boldsymbol{\phi}, \tag{5}$$

where $\mathbf{x}' = R'$ and $\hat{\mathbf{y}}'$ is the generated embedding of R'. By (1), (4), and (5), a stacked L-layer Bayesian LSTM encoder can be constructed [24]. Nonetheless, there are still two major challenges in training and using this model, i.e., how to describe the prior distribution in the parameters space and how to perform the Bayesian inference via $\Pr(\boldsymbol{\phi}|\mathcal{D})$ and $\Pr(\hat{\mathbf{y}}'|\mathbf{x}', \mathcal{D})$. We will address these challenges in Section III-B.

2) Trajectory Reconstructor: The fundamental design principle of BRAE is to learn the authentic trajectory distribution in a latent space, so that adversarial ones are embedded as "out-of-distribution" latent (random) vectors. However, deriving the latent distribution from historical data is non-trivial since such latent space is typically non-interpretable, rendering it intractable to construct the training objective manually for supervised learning. Therefore, we resort to self-supervised learning on the historical trajectory data for learning their latent representation.

In particular, we employ the autoencoder (encoder-decoder) framework to embed trajectories in a latent space in this work. The key inductive bias in this self-supervised learning model is that a stochastic operation must be applied at each time step of a time-series (trajectory) to develop the next step, which resembles the intuition of entity movement on a road: given a partial past trajectory, the next location will be within a defined random region in the area. Combining with the previous trajectory encoder, a trajectory reconstructor that decodes embedded latent trajectory vectors back to their respective original trajectory locations is devised to construct the encoder-decoder architecture. We leverage the same data processing technique in the encoder, i.e., Bayesian LSTM model, to reconstruct the trajectory. Specifically, L layers of Bayesian LSTM are stacked with the k-th layer's output $\mathbf{h}_t^{(k)}$ be the input of the $(k+1)$-th one, namely, $\mathbf{x}_t^{(k+1)}$, where the superscripts denote the layer indices.

A major difference between the encoder and the reconstructor lies in their outputs. While the encoder develops the embedded latent vector of length N^{enc} by adopting the same number of neurons in its last neural network layer, the reconstructor only produces a sequence of locations $\hat{l}_n, \hat{l}_{n-1}, \cdots, \hat{l}_1$ corresponding to the original trajectory in a reversed order, where the hat above symbols indicates a prediction. Given an arbitrary length-n trajectory denoted by $\{l_1, l_2, \cdots, l_n\}$, the encoder exploits its latent information and develops the final cell state vector $\mathbf{c}_n^{(L)}$, which corresponds to the embedded trajectory vector. This vector is then passed to the trajectory reconstructor in which it serves as the initial cell state, i.e., $\mathbf{c}_0^{(1)}$. Using a zero-vector starting token $\langle 0 \rangle$ as the first input $\mathbf{x}_1$, the reconstructor recovers the trajectory from the input cell state starting from the n-th trajectory point according to (1), (4), and (5). Then in the next time step of the neural network, the recovered $\hat{l}_n$ is taken as the input of the next timestep to reconstruct l_{n-1}. This process repeats until the last location l_1 is reconstructed. When training this model, the reconstruction loss $\mathcal{L}(R, \hat{R})$ between the input authentic trajectory R and the output reconstructed one $\hat{R}$ can be employed as the parameter optimization objective. Fig. 2 demonstrates an example of this trajectory reconstruction process. In this figure, the blue dots on the left are the input trajectory points, and the green dots on the right are the reconstructed trajectory points. The reconstruction starts from the ending location and iteratively infers the previous one until the trajectory is reproduced. The dashed box indicates copying from the previous step output of the reconstructor.

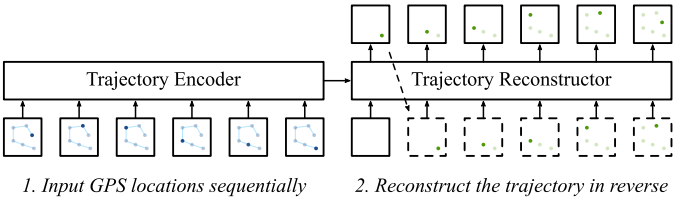


Fig. 2. An example of the trajectory reconstruction process.

With this configuration, a question arises: *why should this model learn a good embedded latent trajectory vector* $\mathbf{c}_n^{(L^{\text{enc}})}$*?* This is due to the representation learning capability of autoencoder neural networks. The state of the trajectory encoder after the last input is the representation of the whole input trajectory, with which the reconstructor is being asked to reconstruct the input sequence [26]. To achieve this objective, the representation must retain information on the critical characteristics of the trajectory during parameter training. Furthermore, two facts on this architecture prevent the model from collapsing to an identity mapping that effectively copies the input of encoder to the output of the reconstructor. First, the fixed and limited number of neurons in the hidden layers of this neural network render it highly unlikely to learn trivial mappings from arbitrary length input trajectories. Second, the same Bayesian LSTM recurrent operations are employed in the reconstructor recursively so that the same displacement dynamics must be employed at any stage of representation decoding, which further prevents the model from becoming an identity mapping.

B. Variational Inference

In Section III-A, there remains two issues that hinder the model construction and training based on (1), (4), and (5): prior distribution determination and intractable Bayesian inference. In a Bayesian neural network, the prior distribution of a network parameter represents the prior belief about the true distribution of the parameter. However, since the physical meaning of neural network parameters mostly remains unclear, it is difficult to determine their prior distribution with empirical domain knowledge. Despite this, prior studies on Bayesian

methods [27], [28] indicate that adopting standard parametric distributions for these parameters is among the most effective solutions in such cases. We utilize this hypothesis and set the prior distributions of all network parameters to be zero-mean Gaussian distributions, which also bring about the merit of regularization [27].

With the prior parameter distributions set, the remaining challenge of the proposed neural network is on the calculation of $\Pr(\boldsymbol{\phi}|\mathcal{D})$, which is critical in determining the optimal parameters. As this true posterior is typically intractable especially for complex learning models like the proposed one, we employ the variational inference technique to approximate it through a variational distribution [29], [30]. In particular, we construct a new Bayesian neural network based trajectory reconstructor parameterized by $\boldsymbol{\omega}$ — whose output is denoted by $q_{\boldsymbol{\omega}}(\boldsymbol{\phi})$ — to approximate the intractable true posterior Bayesian inference $\Pr(\boldsymbol{\phi}|\mathcal{D})$. As all neural network parameters in the reconstructor are probability distributions, $q_{\boldsymbol{\omega}}(\boldsymbol{\phi})$ is the approximated posterior distribution. The approximation can be achieved by minimizing the Kullback-Leibler (KL) divergence between the two distributions:

$$\begin{aligned}
&\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi}|\mathcal{D})\right)\\
&=\int q_{\boldsymbol{\omega}}(\boldsymbol{\phi})\log\frac{q_{\boldsymbol{\omega}}(\boldsymbol{\phi})}{\Pr(\boldsymbol{\phi}|\mathcal{D})}\mathrm{d}\boldsymbol{\phi} &\text{(6a)}\\
&=\int q_{\boldsymbol{\omega}}(\boldsymbol{\phi})\log\frac{q_{\boldsymbol{\omega}}(\boldsymbol{\phi})}{\Pr(\boldsymbol{\phi})}\mathrm{d}\boldsymbol{\phi}-\sum_i\int q_{\boldsymbol{\omega}}(\boldsymbol{\phi})\log\left(l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)\right)\mathrm{d}\boldsymbol{\phi}\\
&\quad+\log\Big(\int\Pr(\boldsymbol{\phi})\prod_i l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)\mathrm{d}\boldsymbol{\phi}\Big), &\text{(6b)}
\end{aligned}$$

where (6b) is derived by substituting (4) into (6a). In (6b), the first term is the KL divergence between $q_{\boldsymbol{\omega}}(\boldsymbol{\phi})$ and $\Pr(\boldsymbol{\phi})$ that describes the "closeness" of these two distributions. The second term is the expected log-likelihood on $q_{\boldsymbol{\omega}}(\boldsymbol{\phi})$, by which the variational distribution is encouraged to resemble the correlation within $\mathcal{D}$ via $q_{\boldsymbol{\phi}}(\mathbf{x})$. The last term can be safely omitted when minimizing $\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi}|\mathcal{D})\right)$ as it is not dependent on $\boldsymbol{\omega}$.

When minimizing the KL divergence, two factors render it so computationally expensive that alternative solutions need to be developed. Initially, the training dataset $\mathcal{D}$ typically comprises a massive volume of samples, rendering it time-consuming to evaluate the second term of (6b). Besides, the highly complex $q_{\boldsymbol{\phi}}(\mathbf{x})$ requires high-dimensional integration over the nondeterministic non-differentiable $\boldsymbol{\phi}$. To overcome the computation burden, data subsampling [31] and reparameterization [22] strategies are adopted [27]. In the first step, a random subset $\mathcal{S}$ of $\mathcal{D}$ is constructed to approximate $\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi}|\mathcal{D})\right)$ based on (6b):

$$\begin{aligned}
\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi}|\mathcal{D})\right)\propto\mathbb{E}_{\mathcal{S}}\Big[&\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi})\right)\\
&-\frac{|\mathcal{D}|}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}^{\mathcal{S}}\int q_{\boldsymbol{\omega}}(\boldsymbol{\phi})\log\left(l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)\right)\mathrm{d}\boldsymbol{\phi}\Big]. \quad (7)
\end{aligned}$$

Subsequently, $\boldsymbol{\phi}$ is reparameterized by a deterministic differentiable transformation $\boldsymbol{\phi}=g(\boldsymbol{\phi},\boldsymbol{\epsilon})$ with $q(\boldsymbol{\epsilon})$ being a nonparametric distribution [27]. Given an arbitrary Gaussian variational distribution $q_{\boldsymbol{\omega}}=\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathsf{T}})$, it can be considered as a transformation of $\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})$ by $g(\boldsymbol{\phi},\boldsymbol{\epsilon})=\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon}$. Therefore, the integral in the second term of (7) can be approximated by

$$\begin{aligned}
&\int q_{\boldsymbol{\omega}}(\boldsymbol{\phi})\log\left(l\big(\mathbf{y}^i|q_{\boldsymbol{\phi}}(\mathbf{x}^i)\big)\right)\mathrm{d}\boldsymbol{\phi}\\
&\approx\int q(\boldsymbol{\epsilon})\log\left(l\big(\mathbf{y}^i|q_{g(\boldsymbol{\phi},\boldsymbol{\epsilon})}(\mathbf{x}^i)\big)\right)\mathrm{d}\boldsymbol{\epsilon} &\text{(8a)}\\
&=\mathbb{E}_{\boldsymbol{\epsilon}}\Big[\log\left(l\big(\mathbf{y}^i|q_{g(\boldsymbol{\phi},\boldsymbol{\epsilon})}(\mathbf{x}^i)\big)\right)\Big]. &\text{(8b)}
\end{aligned}$$

The fitness objective of the KL divergence minimization can be accordingly constructed by substituting (8b) into (7), which can be optimized by stochastic optimizers. The obtained optimal $\boldsymbol{\omega}^*=\{\boldsymbol{\mu}^*,\boldsymbol{\sigma}^*\}$ is also an optimum to the original $\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\phi})||\Pr(\boldsymbol{\phi}|\mathcal{D})\right)$ [21], [32], and $q_{\boldsymbol{\omega}^*}(\boldsymbol{\phi})$ is an approximation to the true posterior $\Pr(\boldsymbol{\phi}|\mathcal{D})$. Consequently, (5) can be accordingly rewritten as

$$\Pr(\hat{\mathbf{y}}'|\mathbf{x}',\mathcal{D})=\frac{1}{|\mathcal{S}|}\sum_k^{|\mathcal{S}|}l\big(\hat{\mathbf{y}}'|q_{\boldsymbol{\phi}_k}(\mathbf{x}')\big),\ \boldsymbol{\phi}_k\sim q_{\boldsymbol{\omega}^*}(\boldsymbol{\phi}), \quad (9)$$

which can be incorporated in the loss function of the proposed autoencoder as follows, whose output aims at reconstructing the input, i.e., $\mathbf{y}^i=\mathbf{x}^i$:

$$\mathcal{L}(\mathcal{R},\hat{\mathcal{R}})=\lambda||\boldsymbol{\phi}||^2-\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\log\left(l(\mathbf{x}^i|q_{\boldsymbol{\phi}_k}(\mathbf{x}^i))\right), \quad (10)$$

where the first term is a network parameter regularization term conditioned by weight decay λ.

With (10), the neural network parameter training can be outlined by Bayes by Backprop [33] as in Algorithm 1. In practice, we employ Adam [34] to slightly modify lines 5–6 of Algorithm 1 based on adaptive estimates of lower-order moments for training acceleration.

Algorithm 1: Parameter Training of BRAE

Data: $\mathcal{R}=\{\mathbf{x}^i\}_{i=1}^N$, learning rate η.
Result: $\boldsymbol{\phi}^*$
1 Randomly initialize $\boldsymbol{\phi}$.
2 **while** $\boldsymbol{\phi}$ *not converged* **do**
3 Sample N random variables $\boldsymbol{\epsilon}_i\sim q(\epsilon)=\mathcal{N}(0,\mathbf{I})$.
4 Sample $\mathcal{S}$ as a random subset of $\{1,2,\cdots,N\}$.
5 Calculate derivative of $\boldsymbol{\phi}$ as $\Delta_{\boldsymbol{\phi}}=-\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\frac{\partial}{\partial\boldsymbol{\phi}}\log\left(l(\mathbf{x}^i|q_{g(\boldsymbol{\phi},\boldsymbol{\epsilon}_i)}(\mathbf{x}^i))\right)+\frac{\partial}{\partial\boldsymbol{\phi}}\lambda||\boldsymbol{\phi}||^2$.
6 $\boldsymbol{\phi}\leftarrow\boldsymbol{\phi}+\eta\Delta_{\boldsymbol{\phi}}$.
7 **end**
8 Save current $\boldsymbol{\phi}$ as $\boldsymbol{\phi}^*$.

IV. Attack Identification by Trajectory Inference and Clustering

With the previously proposed BRAE model, trajectories can be embedded in a latent trajectory space as multivariate random variables. Because the model is constructed according to Bayesian and recurrent neural network theories, it can

intrinsically address the two challenges proposed in Section II, namely, capturing trajectory uncertainties and handling heterogeneous trajectory lengths. In this section, we propose a novel trajectory credibility scoring algorithm based on BRAE inference for Sybil attack trajectory identification. Additionally, we further devise a trajectory clustering mechanism to reduce the training complexity of BRAE, which will also be elaborated later in this section.

A. Sybil Attack Identification

Contributed by the autoencoder architecture of BRAE as shown in Fig. 1, the model is capable of reconstructing a trajectory based on the latent information in the input trajectory and the historical authentic ones in the form of trained neural network parameters. Interpreting this capability with the attack identification context, the model can generate new trajectories which 1) are considered authentic according to historical knowledge, and 2) are as approximate to the given trajectory as possible. Consequently, if the generated ones deviate from the input significantly, a Sybil attack is likely to be launched. The proposed trajectory credibility scoring algorithm for Sybil attack identification is formulated based on this insight.

In particular, we consider an incoming trajectory R' of locations $\mathbf{x}' = \{l_1, l_2, \cdots, l_n\}$ to be identified by the algorithm. With the learnt optimal BRAE parameters $\boldsymbol{\phi}^*$, K synthesized trajectories that closely resembles R' can be generated by

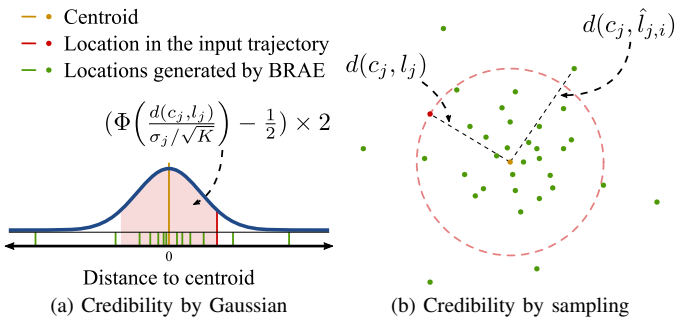
$$\hat{\mathcal{R}}' = \{q_{g(\boldsymbol{\phi}^*, \boldsymbol{\epsilon}_i)}(\mathbf{x}') | \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \mathbf{I}), 1 \leq i \leq K\}. \tag{11}$$

For each of the n locations in R', e.g., l_j, the corresponding ones in all reconstructed trajectories $\hat{R}' \in \hat{\mathcal{R}}'$ are aggregated to formulate a respective location family $\mathcal{O}_j$ with the centroid location denoted by c_j. Subsequently, the distance from c_j to each location $\hat{l}_{j,i} \in \mathcal{O}_j$ can be calculated and denoted by $d(c_j, \hat{l}_{j,i})$ where $d(\cdot, \cdot)$ computes the Euclidean space distance of two locations. At this point, we propose two schemes to compute the credibility of R' according to different presumptions as follows.

1) Credibility by Gaussian: In this scheme, we hypothesize that the reconstructed locations in $\mathcal{O}_j$ follow a Gaussian distribution, which is a strong yet effective assumption as will be shown by empirical results in Section V-D. Accordingly, the credibility of R' is given by

$$c^{\mathrm{G}}(R') = \prod_i^n \left(1 - \left(\Phi\left(\frac{d(c_j, l_j)}{\sigma_j / \sqrt{K}}\right) - \frac{1}{2}\right) \times 2\right) \in (0, 1], \tag{12}$$

where $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} \mathrm{d}t$ is the cumulative of the standard Gaussian distribution, and σ_j is the standard deviation of $\{d(c_j, \hat{l}_{j,i}) | \hat{l}_{j,i} \in \mathcal{O}_j\}$. In (12), the term inside the product denotes the probability of synthesized locations more distant than l_j from the centroid c_j. The larger this term is, the more credible l_j is as the j-th location in R'. Fig. 3a gives a conceptualized illustration of this term. Consequently, multiplying this credibility score for each location in R' produces the credibility of R'.



(a) Credibility by Gaussian (b) Credibility by sampling

Fig. 3. A conceptualized illustration to the two credibility scoring schemes.

2) Credibility by sampling: In this scheme, we employ the frequentist inference to derive the probability of a location being close enough to the centroid or not. Particularly, the credibility of R' is given by

$$c^{\mathrm{P}}(R') = \prod_i^n \left|\{\hat{l}_{j,i} | d(c_j, \hat{l}_{j,i}) \geq d(c_j, l_j), 1 \leq i \leq K\}\right| / K. \tag{13}$$

For $c^{\mathrm{P}}(R')$, the term inside the product represents the percentage of respective generated locations more distant than l_j from c_j. Similar to (12), larger $c^{\mathrm{P}}(R')$ values indicate more credible trajectories.

Either credibility has its own merits and drawbacks. While $c^{\mathrm{P}}(R')$ relaxes the assumption on posterior distribution $\Pr(\hat{\mathbf{y}}' | \mathbf{x}', \mathcal{D})$, generally it requires a large K to achieve a good approximation by $\hat{l}_{j,i}$ samples. In the meantime, the calculation of $c^{\mathrm{G}}(R')$ is stable even with small K values, which can significantly reduce the computation effort during the online attack identification process. However, the assumption of posterior being a Gaussian distribution is not a priori grounded on theoretical analyses. Therefore, a general rule of thumb of selecting the credibility scoring scheme is based on the available computation resource. Credibility $c^{\mathrm{P}}(R')$ is preferred if K allowance is large, otherwise $c^{\mathrm{G}}(R')$ yields better results. The empirical comparison of the two schemes will be carried out in Section V-D. Employing either of the scoring schemes, we can calculate the credibility $c(R')$ of an arbitrary new trajectory. To have a deterministic and binary identification of Sybil attack trajectories, a credibility threshold Δ can be utilized, i.e., R' is identified as a trajectory adversarially generated by a Sybil attack if $c(R') < \Delta$.

B. Fast Training with Trajectory Clustering

In Section III-A we propose the architecture of BRAE, which is powered by a recurrent autoencoder neural network. With the possible massive volume of historical authentic trajectories, obtaining the optimal network parameters can be highly computationally expensive, especially with the multi-sample backpropagation as shown in Algorithm 1. To accelerate the training process of BRAE, we propose a trajectory clustering scheme for fast BRAE parameter training. The design principle is simple yet effective: within an investigated area, the trajectories can be classified into multiple categories

(a) GPS locations in Porto dataset (b) GPS locations in Beijing dataset (c) GPS locations in Cologne dataset

Fig. 4. Real-world datasets investigated in case studies.

considering their routes. In the previous training algorithm, one BRAE model is required to learn the characteristics of all these categories with the same set of neural network parameters, which requires a significant model capacity. Considering this, we can adopt multiple parallel and independent BRAE models, each of which handles one or a few categories to reduce the model capacity requirement. This, in turn, leads to faster convergence of parameter optimization.

Following the analysis, the proposed training with the trajectory clustering scheme is as concise as the following two steps. Given a trajectory clustering algorithm, e.g., k-medoids clustering on Hausdorf distance between trajectory pairs for its simplicity, the authentic trajectory set $\mathcal{R}$ is split into M clusters denoted by $\mathcal{R}^m$, $1 \leq m \leq M$. Subsequently, M BRAE models with the same architecture are constructed and initialized, each of which employs a cluster $\mathcal{R}^m$ to construct its training dataset for parameter tuning using Algorithm 1. Because each set consists of much less number of trajectories whose route variance is also reduced during the clustering, the whole training process is expected to be completed with less time. If multiple computing devices are employed, the training can be accelerated with proper parallelization.

With this trajectory clustering scheme, the Sybil attack identification needs to be accordingly modified. Adopting the M trained BRAE models $q_{g(\boldsymbol{\phi}^*_m,\boldsymbol{\epsilon})}$ parameterized by $\boldsymbol{\phi}^*_m$ for the m-th one, M groups of synthesized trajectories $\hat{\mathcal{R}}'_m$ can be calculated referring to (11). Using credibility by either Gaussian or sampling, each group develops a score $c_m(R')$. Finally, R' is considered adversarial if $\max_m\{c_m(R')\} < \Delta$.

V. Case Studies

In this section, we carry out a series of comprehensive case studies on three real-world datasets to evaluate the efficacy of the proposed Sybil attack identification model. In particular, we first investigate the size of the region that BRAE can process with one standalone model through a scalability test. Then the adversarial trajectory identification accuracy of BRAE is evaluated and compared with existing related models. Subsequently, a sensitivity test is conducted to evaluate the impact of hyperparameter selection on the proposed model. Finally, the efficiency of the proposed fast training scheme with trajectory clustering is demonstrated.

A. Experimental Configurations

In this work, we employ three real-world vehicular trajectory datasets, namely, Porto[2], Beijing[3], and Cologne[4] data. Specifically, the Porto dataset is generated from 442 taxis traversing the city of Porto from Jan. 07, 2013 to Jun. 30, 2014. The sampling interval of GPS records is 15 s. We filter out short trajectories of less than 5 locations, which accords with the data aggregation pattern of crowdsourced navigation where only persistent user reports are considered. The filtered Porto dataset has 1 628 269 trajectories. The Beijing dataset is generated from the T-Drive data [35] of 10 357 taxis from Feb. 2, 2008 to Feb. 8, 2008 within Beijing. The average sampling interval is approximately 177 s, and GPS records with more than 10 min intervals are regarded as trajectory breakpoints [16]. De-duplication is performed over the raw data to remove redundant GPS locations with the same timestamp and taxi identifier, and less-than-five-record trajectories are removed. As a result, 57 203 trajectories are developed. We note that the 177 s sampling interval of GPS records in this dataset are not as practical for real-time crowdsourced navigation. Nonetheless, we still employ this dataset in the hope of demonstrating the robustness of BRAE on handling trajectories with a wide range of sampling intervals. Finally, the Cologne dataset is based on the data of the TAPASCologne project, which is an initiative of the Institute of Transportation Systems at the German Aerospace Center aiming at reproducing the car traffic of the city of Cologne [36], [37]. The original sampling interval is 1 s and we downsample all trajectories to 5 s intervals. The main incentive for the downsampling scheme

[2]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

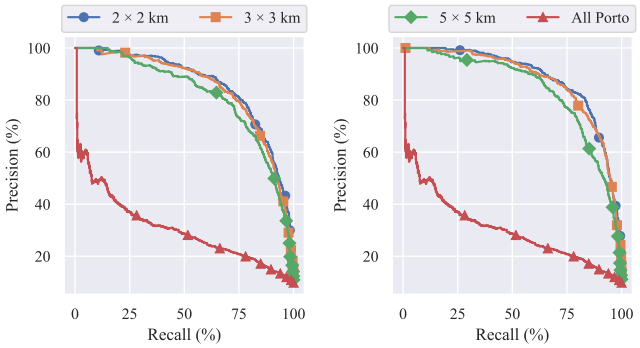
[3]https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[4]http://kolntrace.project.citi-lab.fr/

is to notably reduce the computational burden of training BRAE with high density data. By shrinking the dataset to one fifth of its original size, a better and faster convergence of neural network training can be expected. Additionally, this data modification is also valid in practice, as it is trivial for crowdsourced navigation applications to remove redundant real-time GPS records within a $5\,\mathrm{s}$ window. After removing short ones with less than five records, the dataset has $662\,053$ trajectories. Fig. 4 depicts the GPS locations of the three real-world datasets.

As there is no labeled dataset available for Sybil attack detection, we follow the previous work on Sybil attack [8], spatial outlier detection [38], and anomalous trajectory detection [16] to generate adversarial trajectories as Sybil attacks for the assessment. In particular, four trajectory perturbation and synthesis schemes are employed, namely, detour (DT), route-switching (RS), down-sampling (DS), and slowdowns (SD). The detour scheme moves a random part of a trajectory along a direction roughly perpendicular to the head direction. The route-switching scheme "grafts" two randomly selected geographically adjacent trajectories in the dataset [16]. Both of these two schemes aim at compromising the crowdsourced navigation by migrating the traffic from one road to another, and we strictly follow [8] to implement these two schemes. Additionally, we have the down-sampling scheme which uniformly removes approximately 67% of all GPS records from the trajectory to emulate a faster traffic speed than the real one. Finally, the simulated slowdown algorithm proposed in [8] is employed as the fourth adversarial trajectory generation scheme where we follow the proposed attack generation algorithm in the respective literature to produce adversarial trajectories. The four attack schemes were tested on a real-world crowdsourced navigation system and all schemes can compromise the reported transient traffic state. The detailed attack results are not included though due to legal concerns. Following the previous work [14], [16], [38], we inject 5% synthesized adversarial trajectories in each dataset, in which all trajectories are grouped into a training and a testing set by the ratio of $3:1$ for cross-validation consideration. We note that if the training dataset is accumulated or has small changes over time, incremental learning techniques can be employed for online machine learning [39].

According to the design in Section IV-A, an arbitrary trajectory is assessed by a credibility score between zero (exclusive) and one (inclusive). In both the simulated environment and real-world scenarios, authentic and adversarial trajectories are highly imbalanced in terms of occurrence and we are more concerned about the detection of adversarial ones. Therefore, the precision-recall area under curve (PR-AUC) is adopted as the main performance metric which is highly capable of evaluating skewed/biased datasets. Since this performance metric is directly related to both the precision and recall of the identification, the ratio of positive samples in the dataset does not have significant impact on the PR-AUC value. Unless otherwise stated, hyperparameters $L = 3$ in which each layer has 128 neurons except for $N^{\text{enc}} = 32$. Additionally, data subsample size $|\mathcal{S}|$ and synthesized trajectory size K are both set to be 32. The credibility of trajectories is evaluated by (12).



(a) Downtown trajectories. (b) Residential trajectories.

Fig. 5. Precision-recall curve of the Porto dataset.

Z-score standardization is employed to normalize the datasets. All case studies are conducted on computing servers with two Intel Xeon E5 CPU and 128 GB RAM. The simulation is developed in Python and BRAE is modeled with PyTorch. Eight nVidia GTX 2080 Ti GPUs are employed on each server for algebraic and neural network computing acceleration.

B. Identification Accuracy

By inspecting Fig. 4, it is clear that the size of the covered region by the real-world datasets is massive and using one BRAE model to exploit all vehicular displacement dynamics can be challenging given the fixed model capacity. Therefore, it is more computationally efficient to divide the whole urban city region into small grids for feature extraction than feed all trajectories into one model and hope for finding the parameter space optimum within finite time. In this subsection, we investigate the scalability of BRAE and propose guidelines for its training.

To illustrate BRAE scalability, we first construct sub-datasets of different sizes based on the three real-world datasets. Particularly, two parameters are adopted to control the construction, namely, the size of the sub-region ($2\,\mathrm{km}\times2\,\mathrm{km}$, $3\,\mathrm{km}\times3\,\mathrm{km}$, or $5\,\mathrm{km}\times5\,\mathrm{km}$) and whether it is in the downtown area or not. Then for each of the six possible parameter configurations, four non-overlapping different sub-regions are created, and all trajectories in the original dataset that intersect with the sub-regions are completely included in the respective sub-datasets. As a result, 3 (cities)$\times$3 (sizes)$\times$2 (downtown)$\times$ $4 = 72$ BRAE models are trained on the constructed sub-datasets, whose identification accuracy and training time are summarized in Table I. Three additional BRAE models are also included in the table for reference (labeled by "All") which are directly trained with the original datasets.

From the comparative results, several conclusions can be developed. First, the PR-AUC performance slightly degrades with the increase in the size of sub-regions from $2\,\mathrm{km}\times2\,\mathrm{km}$ to $5\,\mathrm{km}\times5\,\mathrm{km}$. This observation is contributed by two factors. On the one hand, the model capacity of BRAE is sufficient to exploit the vehicular displacement dynamics within $5\,\mathrm{km}\times5\,\mathrm{km}$ regions with the neural network parameters, rendering comparable identification accuracy. On the other

TABLE I
PERFORMANCE COMPARISON OF BRAE TRAINED WITH SUB-DATASETS OF REAL-WORLD VEHICULAR TRAJECTORIES

	PR-AUC	Porto				Beijing				Cologne			
		2×2 km	3×3 km	5×5 km	All	2×2 km	3×3 km	5×5 km	All	2×2 km	3×3 km	5×5 km	All
Downtown	Overall	0.857	0.847	0.817	0.306	0.813	0.759	0.754	0.267	0.863	0.845	0.834	0.395
	DT	0.863	0.844	0.805	0.332	0.767	0.739	0.739	0.300	0.853	0.837	0.836	0.435
	RS	0.858	0.800	0.824	0.281	0.753	0.751	0.744	0.259	0.852	0.832	0.820	0.359
	DS	0.763	0.790	0.688	0.266	0.787	0.649	0.615	0.246	0.789	0.755	0.751	0.329
	SD	0.987	0.983	0.985	0.448	0.988	0.960	0.956	0.315	0.988	0.981	0.977	0.606
	Training	8.5 h	9.6 h	11.5 h	>1 d	9.3 h	10.3 h	12.4 h	>1 d	9.1 h	10.3 h	12.7 h	>1 d
Residential	Overall	0.886	0.877	0.838	-	0.821	0.763	0.758	-	0.885	0.853	0.839	-
	DT	0.885	0.846	0.832	-	0.782	0.721	0.700	-	0.884	0.818	0.844	-
	RS	0.866	0.882	0.835	-	0.852	0.728	0.754	-	0.876	0.850	0.823	-
	DS	0.815	0.794	0.745	-	0.708	0.703	0.683	-	0.824	0.770	0.752	-
	SD	0.993	0.984	0.977	-	0.977	0.956	0.953	-	0.985	0.984	0.979	-
	Training	4.2 h	7.1 h	10.0 h	-	4.9 h	7.9 h	10.9 h	-	4.5 h	7.7 h	10.8 h	-

hand, the notable increase in the number of trajectories to be processed with the region size imposes difficulty on the parameter optimizer to locate the optimal solutions in the parameter space. As a side effect, longer training time is required for larger regions. Second, the performance deviation of BRAE for downtown and residential regions is insignificant (± 0.03 PR-AUC for Porto, ± 0.008 for Beijing, and ± 0.022 for Cologne). This observation demonstrates the generality of BRAE on handling regions with different densities and displacement patterns of vehicular trajectories. Third, the comparison suggests that inputting all trajectories of a city into BRAE yields bad identification performance. This is due to the massively increased number of trajectories, which require BRAE to, therefore, capture the significantly increased patterns of trajectory dynamics. As a result, the training process takes longer than one day and the pre-mature models after training for $24\,\mathrm{h}$ generate inferior PR-AUC performance.

Considering both the identification accuracy and training time factors, a guideline for using the BRAE model can be derived. While dividing an arbitrary investigated region into small grids yields the best accuracy performance and shortest single model training time, seven to nine independently trained BRAEs handling $2\,\mathrm{km}\times 2\,\mathrm{km}$ grids are required to cover one $5\,\mathrm{km}\times 5\,\mathrm{km}$ region. If models are to be trained sequentially, the total training time of small grids far exceeds that of a large one despite the slight accuracy improvement. In the meantime, too large regions, e.g., a whole city, can overwhelm the model capacity of BRAE, rendering inferior performance. Therefore, properly selecting the size of regions covered by BRAE is crucial. Either $3\,\mathrm{km}\times 3\,\mathrm{km}$ or $5\,\mathrm{km}\times 5\,\mathrm{km}$ is preferred, and we adopt $5\,\mathrm{km}\times 5\,\mathrm{km}$ in the following case studies.

One may note that collusion of adversaries is a significant issue of many crowdsourcing systems. Such a scenario is indeed captured by the case study where the adversarial trajectory generation process emulates that only one Sybil attacker is generating a massive amount of adversarial trajectories. This is in principle equivalent to the case where multiple adversaries cooperate to launch the attack (collusion) with lossless information exchange. Since the proposed BRAE handles real-time trajectories without considering possible inter-trajectory spatial or temporal correlations, the performance is not compromised under collusion scenarios.

C. Comparison with Baseline Algorithms

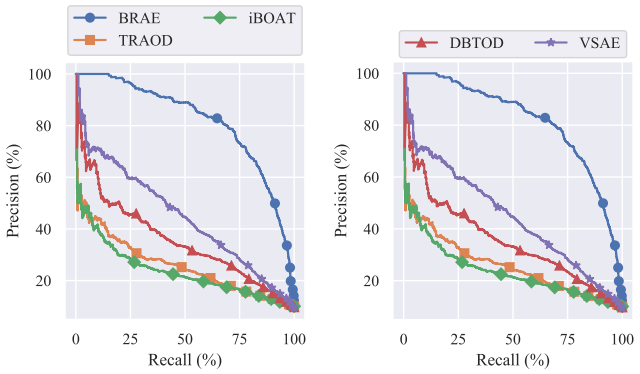
To comprehensively evaluate the performance of the proposed model, we conduct a comparative study to compare BRAE with state-of-the-art anomaly detection algorithms. As introduced in Section I, there is no existing solution developed for Sybil attack trajectory identification to the best of our knowledge. Therefore, four existing trajectory outlier and anomalous trajectory detection algorithms are compared, which aim at addressing similar trajectory data mining problems. In particular, the following baseline algorithms are employed:

- TRAOD [40] detects outlier trajectory from others by first partitioning a trajectory into segments and perform the detection based on them. The anomaly is evaluated on the total length of outlying segments.
- iBOAT [15] maintains an adaptive working window on GPS trajectories and compares the covered locations against historical trajectories to develop a "support" fitness value. This value is thresholded to detect anomalies.
- DBTOD [41] considers the driving statistics of trajectories, e.g., driving speed and bearing, and constructs a probabilistic model for anomalous trajectory detection.
- VSAE [16] adopts a Gaussian mixture variational autoencoder to capture the sequential latent trajectory features so that normal trajectory routes can be developed. The subsequent anomaly detection is based on the routes. The GM-VSAE variant is employed in the comparison.

For all baseline approaches, their respective publicly available source code is employed in this case study with non-critical changes to adapt to the adversarial trajectory detection problem. In particular, all baselines focus on investigating trajectories with one or a few source-destination pairs, where the source and destination are artificial grids in the geographical space. Additionally, VSAE encodes all trajectory locations into grids. To accommodate this restriction, the investigated geographical space of all datasets are partitioned into $50\,\mathrm{m}\times 50\,\mathrm{m}$ grids. Please note that this change does not apply to the proposed BRAE, and it does grant unfair advantages to either BRAE or the baselines. All other case study configurations

TABLE II
PERFORMANCE COMPARISON OF BRAE AND BASELINE ALGORITHMS

	PR-AUC	Porto 5 km×5 km					Beijing 5 km×5 km					Cologne 5 km×5 km				
		BRAE	TRAOD	iBOAT	DBTOD	VSAE	BRAE	TRAOD	iBOAT	DBTOD	VSAE	BRAE	TRAOD	iBOAT	DBTOD	VSAE
Downtown	Overall	**0.817**	0.260	0.240	0.357	0.448	**0.754**	0.238	0.231	0.351	0.427	**0.834**	0.269	0.260	0.373	0.466
	DT	**0.805**	0.234	0.240	0.351	0.420	**0.739**	0.222	0.210	0.328	0.438	**0.836**	0.237	0.262	0.381	0.450
	RS	**0.824**	0.254	0.231	0.385	0.418	**0.744**	0.228	0.236	0.314	0.436	**0.820**	0.276	0.229	0.436	0.460
	DS	**0.688**	0.258	0.214	0.284	0.419	**0.615**	0.253	0.224	0.332	0.343	**0.751**	0.264	0.259	0.289	0.398
	SD	**0.985**	0.346	0.335	0.549	0.674	**0.956**	0.277	0.304	0.540	0.626	**0.977**	0.370	0.326	0.477	0.715
	TPR	**0.902**	0.683	0.669	0.753	0.781	**0.873**	0.649	0.696	0.749	0.772	**0.897**	0.687	0.675	0.759	0.791
	TNR	**0.911**	0.673	0.655	0.747	0.777	**0.890**	0.674	0.671	0.735	0.779	**0.911**	0.674	0.675	0.741	0.783
Residential	Overall	**0.838**	0.242	0.211	0.332	0.422	**0.758**	0.229	0.226	0.324	0.413	**0.839**	0.261	0.251	0.362	0.445
	DT	**0.832**	0.238	0.207	0.295	0.429	**0.700**	0.181	0.202	0.290	0.397	**0.844**	0.237	0.269	0.326	0.405
	RS	**0.835**	0.205	0.177	0.361	0.401	**0.754**	0.236	0.226	0.323	0.390	**0.823**	0.218	0.243	0.365	0.406
	DS	**0.745**	0.263	0.197	0.260	0.351	**0.683**	0.243	0.209	0.289	0.394	**0.752**	0.289	0.213	0.324	0.405
	SD	**0.977**	0.312	0.309	0.566	0.656	**0.953**	0.294	0.323	0.503	0.609	**0.979**	0.350	0.346	0.541	0.700
	TPR	**0.908**	0.680	0.654	0.748	0.790	**0.880**	0.679	0.670	0.700	0.785	**0.925**	0.669	0.676	0.753	0.774
	TNR	**0.915**	0.671	0.654	0.740	0.772	**0.886**	0.669	0.668	0.731	0.776	**0.909**	0.676	0.676	0.748	0.782



(a) Downtown trajectories. (b) Residential trajectories.

Fig. 6. Precision-recall curve of BRAE and baselines on the Porto 5 km×5 km sub-dataset.

are kept identical across the board. Table II presents the PR-AUC performance, true positive rate (TPR), and true negative rate (TNR) of Sybil attack identification using BRAE and baseline algorithms on Porto, Beijing, and Cologne datasets, respectively.

For detecting adversarial trajectories launched by Sybil attacks, BRAE significantly outperforms all baseline algorithms on all datasets. In particular, 82.4% (downtown) to 98.6% (residential) improvements over the best-performing baseline, i.e., VSAE, are developed on the Porto dataset considering 5 km×5 km regions. This advantage is maintained in the other two datasets, where the percentage improvements are 76.6% to 83.5% in Beijing and 79.0% to 88.5% in Cologne, respectively. This indicates that a tailor-made adversarial trajectory detection model like BRAE is required for Sybil attack identification, and related outlier/anomalous trajectory algorithms cannot fully exploit the trivial differences between authentic and adversarial trajectories. The comparison also implies that distance and density-based algorithms (TRAOD and iBOAT) are generally not suitable for the investigated task. This is due to their non-compatible design principle that mainly considers the macroscopic route variance among trajectories for outlier detection. Additionally, the introduction of time-series analysis in BRAE and VSAE helps the respective models to capture time-dependent and latent displacement dynamics. Comparing with VSAE, the proposed BRAE relaxes the geographical grid assumption that imposes a dilemma during the algorithm design process: while fine-grained small grids yield better data granularity which means better feature exploitation, the corresponding training time and model capacity requirements increase exponentially. Therefore, it is non-trivial for such grid-based algorithms to find the optimal grid size for detecting adversarial trajectories in meso- or even microscopic traffic flow. The design of BRAE does not have this concern and are accordingly better performing. For readers' reference, the precision-recall curves of all testing trajectories in Porto 5 km×5 km regions are plotted in Fig. 6.

In Table II, the PR-AUC scores of the four adversarial trajectory schemes are also presented to give insight into the characteristics of Sybil attack identification. Specifically, we can conclude that the down-sampled accelerating trajectories are the most difficult scheme to be accurately detected and the slowdown scheme proposed in [8] that compromised Waze is the simplest one. This accords with intuition. As the slowdown scheme only regulates the instantaneous vehicular speed of adversarial trajectories, the unnaturally straight traces make them suspicious to all detectors. In the meantime, both the detour and the route-switching scheme introduces sharp turns at the points of perturbation. This is an easy characteristic that can be seldomly observed in authentic trajectories and thus captured by learning models. Finally, as removing locations from a trajectory is in general equivalent to lower the data granularity with instantaneous speed compromised, it is relatively hard to distinguish them from sampling noise. A possible solution for detecting such down-sampling trajectories is to include the acceleration attribute in the identification process, which can be a future extension to the feature-based BRAE or DBTOD.

Last but not least, we also summarize the TPR (recall) and TNR (specificity) of BRAE and baselines in Table II, where the former reflects the Type II error performance

TABLE III
BRAE VARIANTS AND PERFORMANCE ON PORTO $5\,\mathrm{km}\times5\,\mathrm{km}$ DATASET

Model	Neurons	Downtown		Residential	
		PR-AUC	Training	PR-AUC	Training
BRAE	$128\times2+32$	0.817	8.5 h	0.838	4.2 h
A	$128\times1+32$	0.742	5.3 h	0.763	2.9 h
B	$128\times3+32$	0.823	19.5 h	0.836	11.0 h
C	$256\times1+32$	0.791	6.2 h	0.803	3.9 h
D	$256\times2+32$	0.825	15.0 h	0.841	9.9 h
E	$128\times2+16$	0.633	7.8 h	0.636	4.0 h
F	$128\times2+64$	0.815	10.6 h	0.846	4.8 h

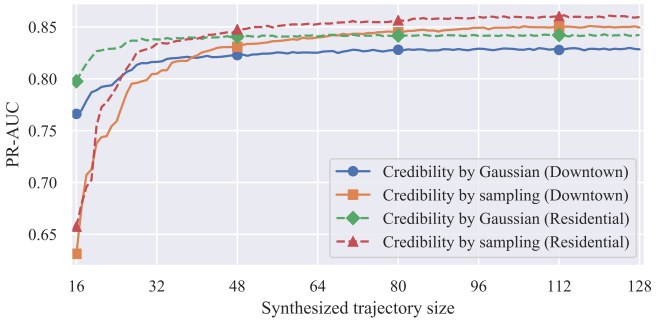


Fig. 7. PR-AUC of credibility hyper-parameters.

and the latter corresponds to Type I error. The simulation results clearly indicate that BRAE notably outperforms baseline approaches in terms of accurately identifying adversarial trajectories (TPR) while retaining authentic ones (TNR). In general, the ratio of false positives can be reduced by half based on the best-performing baseline, i.e., VSAE, in all three datasets. Furthermore, an approximately 90% TPR indicates that the proposed BRAE is capable of defending practical Sybil attacks. As the key to these attacks is to create a large number of pseudonymous identities to compromise crowdsourcing, 90% TPR means that the number of such forged identities has to be increased to ten times its original value. This imposes great difficulty in launching successful Sybil attacks as contemporary crowdsourced navigation systems usually consider the "reputation" of pseudonymous identities, rendering the attacks significantly more costly [8]. Despite significantly outperforming existing models, we acknowledge that the approximately 10% false positives potentially lead to notable information loss at the control center of crowdsourced navigation systems. The challenge is less severe when the crowdsourced trajectories are abundant so that the remaining 90% authentic ones resemble unbiased samples from the ground truth trajectory population. Meanwhile, small-scale systems may observe degenerated performance over large ones. In light of this, we call for further research on pushing the TPR to near-perfect.

D. Hyper-parameter Sensitivity

In the design of BRAE, multiple hyper-parameters are introduced to control the data flow and training of the model. They play a crucial rule in determining the performance of Sybil attack identification. To illustrate the sensitivity of BRAE on them, we perform a hyperparameter sweep test in this subsection. In particular, we first propose a series of BRAE variants with different neural network hyper-parameters for an architecture test. Then we examine the influence of different credibility hyper-parameters, i.e., K value and credibility scoring scheme. The Porto $5\,\mathrm{km}\times5\,\mathrm{km}$ dataset is employed in this test, and offline simulations show that the others demonstrate similar results.

In particular, Table III presents a summary of the BRAE variants and their respective identification and training performance on the Porto $5\,\mathrm{km}\times5\,\mathrm{km}$ data, in which each value is the average of the results from the four different sub-regions as introduced in Section V-B. The labels under the "Neurons" column refer to the number of layers and neurons in the trajectory encoder and reconstructor. For example, "$128\times2+32$" stands for two hidden layers with 128 neurons and a last layer with $N^{\mathrm{enc}}=32$ neurons in the encoder (i.e., $L=3$), and the decoder replaces the last one with a length-2 layer for location reconstruction. From the summary, it can be concluded that the Sybil attack identification performance of BRAE is in general not sensitive to the neural network hyper-parameters as long as model capacity is saturated. This can be derived by comparing BRAE with variant models B, D, and F, where the number of neural layers L, the number of neurons, and the embedded latent vector length N^{enc} are increased based on BRAE, respectively. While there is no notable improvement on the PR-AUC metric, the corresponding training time is extended due to the significantly larger parameter searching space caused by the increasing number of learnable neural network parameters. In the meantime, model C removes one layer but doubles the number of per-layer neurons. This change reduces the model capacity to the under-saturated region, rendering slightly worse PR-AUC and shorter training time. Model A drastically shrinks the model capacity and model E constructs a latent trajectory space of limited expressibility. Neither of the variants is preferred despite their respective reduced training time.

Besides, Fig. 7 presents the performance sensitivity of credibility hyper-parameters. From the results, several conclusions can be developed. First, when the K value is small, credibility by Gaussian yields better PR-AUC than that of the credibility by sampling scheme. This is because a large sampling size is generally required to have a good sampling distribution to approximate the population, whereas the credibility by Gaussian assume the latter to be Gaussian. However, this assumption may deviate from the ground truth, rendering inferior performance when the sampling distribution is accurate enough. Second, both schemes converge with sufficiently large K, which accords with the intuition. When converged, the performance difference of the two scheme originates from the Gaussian distribution assumption error. Third, a general rule of thumb is to use credibility by Gaussian with $K\leq40$ and credibility by sampling otherwise. The selection of K depends on the computation burden of real-time attack identification. For reference, the proposed BRAE takes $73.1\,\mathrm{ms}$ on average to develop the credibility of a trajectory at $K=32$, and an

TABLE IV
TRAJECTORY CLUSTERING TRAINING ON PORTO $5\,\mathrm{km}\times5\,\mathrm{km}$ DATASET

M	Downtown		Residential	
	PR-AUC	Training (Par./Seq.)	PR-AUC	Training (Par./Seq.)
1	0.817	8.5 h / 8.5 h	0.838	4.2 h / 4.2 h
2	0.811	3.9 h / 7.5 h	0.825	2.2 h / 4.3 h
4	0.809	2.2 h / 8.3 h	0.833	1.3 h / 4.8 h
8	0.813	1.5 h / 11.0 h	0.829	0.9 h / 6.1 h

extra $1.7\,\mathrm{ms}$ is required to increase K by one.

E. Efficiency of Trajectory Clustering

In Section IV-B, we propose a trajectory clustering scheme for fast BRAE training. Table IV presents the summary of a case study on its efficiency in reducing the training time. In this table, the values under the "Training (Par./Seq.)" column present the respective parallel and sequential training time. From the results, it can be concluded that the proposed fast training scheme can effectively reduce the training time if multiple models can be trained in parallel. Additionally, this efficiency improvement does not undermine the Sybil attack identification performance with notable PR-AUC degradation. Nonetheless, this scheme is only recommended in case that multiple training and inference devices are available. The total sequential training time may unfavorably increase due to the introduced computational overhead, e.g., k-medoids clustering.

VI. CONCLUSIONS

In this paper, we propose a novel self-supervised Bayesian Recurrent Autoencoder for detecting adversarial trajectory in Sybil attacks targeting crowdsourced navigation systems. The proposed deep learning model is capable of exploiting the time-series features of vehicular trajectories with significant sampling and displacement uncertainties. By adopting an encoder-reconstructor architecture, trajectories are embedded in a latent trajectory distribution space as multivariate random variables, which are later employed to reconstruct resembling trajectories that are considered authentic by the model. Subsequently, by comparing the input trajectory with them, a credibility score can be calculated statistically. Lastly, a fast training scheme is devised based on trajectory clustering for accelerated training in parallel. To evaluate the efficacy of the proposed model, a series of comprehensive case studies are conducted on three real-world vehicular trajectory datasets. Results demonstrate that the proposed model outperforms state-of-the-art baselines with at least 76.6% PR-AUC metric improvement. Furthermore, a hyper-parameter sensitivity test is carried out to illustrate the impact of and develop guidelines for hyper-parameter selection. Finally, simulation results indicate that the proposed fast training scheme achieves sub-linear speedup with a parallel model training system.

In the future, we want to explore extensions to other deep generative or auto-regressive models for adversarial trajectory identification tasks. We look forward to follow-up research on developing further methods and datasets for the critical problem of defending Sybil attacks. The proposed BRAE and its credibility scoring scheme also demonstrate potential of being applied to GPS sensor and inertial navigation spoofing attacks [42], [43] with proper enhancements. This is another promising future research direction.

REFERENCES

[1] X. Fan, J. Liu, Z. Wang, Y. Jiang, and X. Liu, "Crowdsourced road navigation: concept, design, and implementation," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 126–128, 2017.

[2] X. Fan, J. Liu, Z. Wang, Y. Jiang, and X. S. Liu, "CrowdNavi: demystifying last mile navigation with crowdsourced driving information," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 771–781, 2017.

[3] A. Perallos, U. Hernandez-Jayo, I. J. G. Zuazola, and E. Onieva, *Intelligent transport systems: technologies and applications*. John Wiley & Sons, 2015.

[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[5] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, pp. 1–1, 2020.

[6] J. J. Q. Yu, "Semi-supervised deep ensemble learning for travel mode identification," *Transp. Res. Part C Emerg. Technol.*, vol. 112, pp. 120–135, 2020.

[7] J. R. Douceur, "The Sybil attack," in *Proc. International Workshop on Peer-to-Peer Systems*, Berlin, Heidelberg, 2002, p. 251–260.

[8] M. B. Sinai, N. Partush, S. Yadid, and E. Yahav, "Exploiting social navigation," 2014, `arXiv:1410.0151 [cs.CR]`.

[9] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Ghost riders: Sybil attacks on crowdsourced mobile mapping services," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1123–1136, 2018.

[10] Y. Cao, Q. Luo, and J. Liu, "Road navigation system attacks: a case on GPS navigation map," in *Proc. IEEE International Conference on Communications*, 2019, pp. 1–5.

[11] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Defending against Sybil devices in crowdsourced mapping services," in *Proc. Annual International Conference on Mobile Systems, Applications, and Services*, 2016, p. 179–191.

[12] Y. Shoukry, S. Mishra, Z. Luo, and S. Diggavi, "Sybil attack resilient traffic networks: a physics-based trust propagation approach," in *Proc. ACM/IEEE International Conference on Cyber-Physical Systems*, 2018, pp. 43–54.

[13] J. Lin, M. Li, D. Yang, G. Xue, and J. Tang, "Sybil-proof incentive mechanisms for crowdsensing," in *Proc. IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[14] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "IBAT: detecting anomalous taxi trajectories from GPS traces," in *Proc. International Conference on Ubiquitous Computing*, 2011, p. 99–108.

[15] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iBOAT: isolation-based online anomalous trajectory detection," *IEEE Trans. Intell. Transport. Sys.*, vol. 14, no. 2, p. 806–818, Jun. 2013.

[16] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *Proc. International Conference on Data Engineering (ICDE)*, Dallas, TX, Apr. 2020.

[17] D. A. Garcia-Ulloa, L. Xiong, and V. Sunderam, "Truth discovery for spatio-temporal events from crowdsourced data," *Proc VLDB Endow*, vol. 10, no. 11, pp. 1562–1573, Aug. 2017.

[18] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, and J. Gao, "Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing," in *Proc. International World Wide Web Conference*, Lyon, France, Apr. 2018, pp. 13–22.

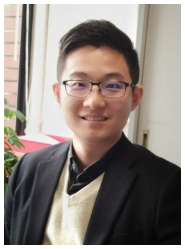
[19] F. Tahmasebian, L. Xiong, M. Sotoodeh, and V. Sunderam, "Crowdsourcing under data poisoning attacks: A comparative study," in *Proc. Data and Applications Security and Privacy*, New Orleans, LA, Mar. 2020, pp. 310–332.

[20] K. Atarashi, S. Oyama, and M. Kurihara, "Semi-supervised learning from crowds using deep generative models," in *Proc. AAAI Conference on Artificial Intelligence*, New Orleans, LA, Feb. 2018, pp. 1555–1562.

[21] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations*, Banff, Canada, Apr. 2014.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[25] H. Wang and D.-Y. Yeung, "Towards bayesian deep learning: a framework and some existing methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, pp. 3395–3408, 2016.
[26] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3940–3951, Oct. 2019.
[27] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Cambridge, 2016.
[28] D. J. C. MacKay, "Bayesian methods for adaptive models," Ph.D. dissertation, California Institute of Technology, 1992.
[29] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, pp. 183–233, 1999.
[30] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1–2, p. 1–305, Jan. 2008.
[31] A. Graves, "Practical variational inference for neural networks," in *Proc. International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2011, p. 2348–2356.
[32] G. Duarte, C. Rolim, and P. Baptista, "How battery electric vehicles can contribute to sustainable urban logistics: a real-world application in Lisbon, Portugal," *Sustainable Energy Technologies and Assessments*, vol. 15, pp. 71–78, 2016.
[33] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. International Conference on Machine Learning*, 2015, p. 1613–1622.
[34] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. International Conference on Learning Representations*, San Diego, CA, Dec. 2015.
[35] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, p. 316–324.
[36] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Trans. Mobile Comput.*, vol. 13, no. 5, pp. 1061–1075, May 2014.
[37] J. J. Q. Yu and A. Y. S. Lam, "Autonomous vehicle logistic system: Joint routing and charging strategy," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2175–2187, 2018.
[38] G. Zheng, S. L. Brantley, T. Lauvaux, and Z. Li, "Contextual spatial outlier detection with metric learning," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, p. 2161–2170.
[39] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online deep learning: Learning deep neural networks on the fly," Nov. 2017, arXiv:1902.10162 [cs.AI].
[40] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: a partition-and-detect framework," in *Proc. IEEE International Conference on Data Engineering*, Cancun, Mexico, 2008, p. 140–149.
[41] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *Proc. ACM Conference on Information and Knowledge Management*, Singapore, 2017, p. 837–846.
[42] K. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," in *Proc. USENIX Conference on Security Symposium*, Baltimore, MD, Aug. 2018, pp. 1527–1544.
[43] S. Narain, A. Ranganathan, and G. Noubir, "Security of GPS/INS based on-road location tracking systems," in *Proc. IEEE Symposium on Security and Privacy*, San Francisco, CA, May 2019, pp. 587–601.



James J.Q. Yu (S'11–M'15–SM'20) received the B.Eng. and Ph.D. degree in electrical and electronic engineering from the University of Hong Kong, Pokfulam, Hong Kong, in 2011 and 2015, respectively. He was a post-doctoral fellow at the University of Hong Kong from 2015 to 2018. He is currently an assistant professor at the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and an honorary assistant professor at the Department of Electrical and Electronic Engineering, the University of Hong Kong. He is also the chief research consultant of GWGrid Inc., Zhuhai, and Fano Labs, Hong Kong. His research interests include smart city technologies, deep learning and big data, intelligent transportation systems, and energy systems.