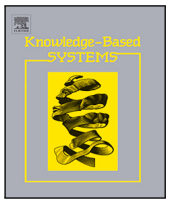
Knowledge-Based Systems 310 (2025) 112918



Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Learning to discover anomalous spatiotemporal trajectory via Open-world State Space model

Qiang Gao [a], Chaoran Liu [a], Li Huang [a,*], Goce Trajcevski [b], Qing Guo [c], Fan Zhou [d]

[a] School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, China
[b] Iowa State University, IA, USA
[c] Beijing Institute of Technology, Beijing, China
[d] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China

ARTICLE INFO

Keywords:
Anomalous trajectory discovery
Open-world learning
State space model
Spatiotemporal data

ABSTRACT

Identifying anomalous trajectories that deviate from usual driving patterns in an open-world context has recently become a critical and urgent task in location-aware systems. In contrast to the closed-world settings, its goal is to not only identify known but also detect the presence of *unknown* anomalous behaviors. Despite the achievements of recent self-motivated learning paradigms, existing solutions either do not have the ability to discover unknown anomalous behaviors or can only expose their presence. In this study, we first formally define our task as the *Anomalous Trajectory Discovery problem in an Open-world scenario* (ATDO), and introduce a novel fine-grained **O**pen-world **S**tate **S**pace **L**earning (**OSSL**) framework that has the ability to discover multiple unknown patterns in addition to classifying the existing known behavioral patterns of trajectories. Due to the inherent density behind massive trajectories, we are motivated by state space models and devise a Spatial–temporal State Space (S3) block to explore the long-term dependencies behind the lengthy trajectories. To enable open-world learning, we devise two adapters that operate interactively with three open-world objectives to discover multiple unknown patterns in addition to identifying existing behaviors. With the context setting of two progressive open-world tasks, the experimental results conducted on two large trajectory datasets demonstrate the superiority of OSSL for both known and unknown anomalous patterns.

1. Introduction

As a consequence of the pervasiveness of GPS-integrated mobile devices and location-aware pattern recognition systems, large volumes of location traces have been generated, enabling the collection of time-dependent trajectory datasets. These, in turn, provide unprecedented opportunities to explore (and extract) spatio-temporal knowledge and transfer it to a variety of downstream applications. As one of the critical tasks of trajectory mining, identifying anomalous trajectories has recently attracted a great deal of attention in various location-based systems (LBS) due to issues such as taxi driving fraud and noisy/fake trajectories [1,2].

Intuitively, trajectories are considered anomalous if they diverge from the usual driving patterns under the travel demand [3,4]. In practice, this means that if most vehicles traveling from a source S to destination D follow a certain route or set of routes, any trajectory that deviates from these established paths may be flagged as anomalous. These deviations can arise due to various factors such as unexpected traffic conditions, driver errors, or intentional route changes, thereby highlighting the critical as well as the uncertainty for detection. Traditionally, heuristic-based and machine learning-based paradigms were used to discover trajectories with unusual patterns. Heuristic-based methods (e.g., iBAT [5]) rely on density or distance measurements to detect anomalous trajectories through multiple handcrafted features which, however, lack the ability to explore the trajectory diversity. Although machine learning-based methods with functional approximation (e.g., [6,7]) can build a discriminative manner to classify unusual patterns, they cannot uncover the affluent spatial–temporal semantics behind massive trajectories such as dynamic dependencies. Due to the achievements in deep representation learning, learning with neural networks has become a widespread paradigm for exploring the implicit semantics behind trajectories. Most of the prior arts focus on devising recurrent neural networks (RNNs) and their variants to study intricate spatial or/and temporal dependencies to enhance the generality of trajectory representation [8,9]. Complementary, the

* Corresponding author.
E-mail addresses: qianggao@swufe.edu.cn (Q. Gao), liuchaoran@smail.swufe.edu.cn (C. Liu), lihuang@swufe.edu.cn (L. Huang), gocet25@iastate.edu (G. Trajcevski), 3120195496@bit.edu.cn (Q. Guo), fan.zhou@uestc.edu.cn (F. Zhou).

https://doi.org/10.1016/j.knosys.2024.112918
Received 25 July 2024; Received in revised form 25 September 2024; Accepted 21 December 2024
Available online 30 December 2024
0950-7051/© 2024 Published by Elsevier B.V.

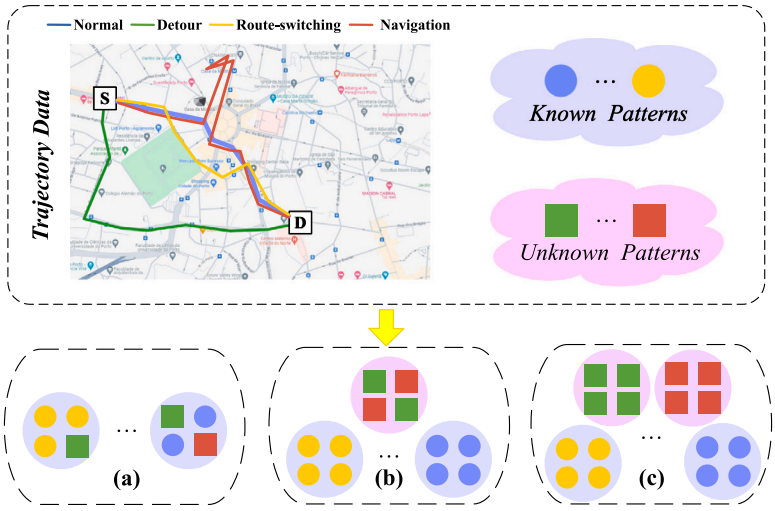


Fig. 1. A toy example in identifying anomalous trajectories. (a) represents a model trained under the closed-world assumption, which posits that all encountered patterns belong to predefined classes; (b) in contrast, an open-world setting (i.e., ATRO in previous study) is endowed with the capacity to discern unfamiliar patterns from those it recognizes upon encounter; (c) leveraging the proposed fine-grained open-world solution (ATDO in this study), when confronted with "unknown unknowns", not only differentiates these from known categories but also categorizes the unknowns into distinct sub-classes according to their intrinsic characteristics.

prevalent attention-based techniques (e.g., self-attention networks) become additional alternatives for exposing complicated time-varying interactions [10,11]. Despite the large number of advanced (deep) learning technologies devoted to identifying anomalous trajectories, we still consider two critical concerns that have not been investigated yet.

CC1: The top of Fig. 1 shows a scenario of a learning system created to tackle massive trajectory data. Typically, the behavior/pattern behind each trajectory can be tagged by domain experts. However, due to the dynamic nature of cognition, the current system may only acquire or be familiar with partial trajectory patterns (*Known Patterns* in Fig. 1). Thus, existing solutions either fail to detect unknown anomalous patterns or only expose the presence of unknown anomalous behavior. The current mainstream solutions, including: (i) combining LSTM-based autoencoder One-class Support Vector Machine (OC-SVM) [12]; (ii) extending DBSCAN with Hausdorff distance (and SVM) [13]; (iii) using generative models such as GM-VSAE [14] and DeepTEA [8] — rely heavily on the closed-world (CW) (sometimes referred to as "closed-set") assumption with self-motivated learning paradigms. As illustrated in Fig. 1(a), such a system considers that all of the behavioral collections have reported complete knowledge of the world, due in part to the abundance of labeled patterns. Hence, it can build either discriminative or generative-inspired models with a simple distribution assumption, such as independent and identically distributed (*i.i.d.*). However, such assumptions are problematic for many real-world scenarios that usually confront incomplete knowledge — called an open-world (OW) assumption. Inevitably, the existing approaches may not work in an OW context. To our knowledge, [2] was the first attempt to identify anomalous trajectories in the OW context with a metric rule, tackling the **ATRO** (Anomalous Trajectory Recognition problem in an Open-world scenario) variant. However, it only detects the *presence* of an unknown anomaly and cannot handle *different unknown patterns* in OW settings (cf. Fig. 1(b)).

CC2: Mining the time-varying dependencies is the core work of deep trajectory representation, which can significantly improve the accuracy or quality of downstream LBS tasks. Again, typically RNNs or attention-based models are the preferred option for time-varying dependency learning. For instance, ATD-RNN [15] leverages learned trajectory embeddings for characterizing trajectories, primarily striving to operate RNNs to uncover the sequential dynamics and depict the internal features between abnormal and normal trajectories. DiffTAD [16] is a probabilistic model that uses Transformer-based temporal and spatial encoders to distill temporal dependencies and spatial interactions, which formalizes the anomaly detection problem as a noisy-to-normal paradigm. Due to the inherent density behind lengthy trajectories and the vanishing or exploding gradients often encountered during training [17,18], these models are unable to effectively and efficiently solve the long-term dependency problems. Also, in practice, the presence of uncertainties (caused by e.g., road congestion or temporary closures) often leads to excessively long or varied trajectories. However, effectively and efficiently capturing the long-term dependencies behind trajectories allows to keep the driver's individual intents in mind, which further helps to detect unusual patterns.

These challenges lie in accurately modeling what constitutes a "normal" trajectory, which involves understanding the underlying travel demand and typical route choices between different origin–destination pairs. This requires robust paradigms capable of learning and adapting to the dynamic nature of driving patterns while effectively handling the "unknown unknowns" — factors or scenarios that were previously unobserved or unconsidered. To address this, with CC1 and CC2 in mind, we introduce a novel **O**pen-world **S**tate **S**pace **L**earning framework (**OSSL**), to identify anomalous trajectories in a more practical OW context. We formally define our task as the *Anomalous Trajectory Discovery problem in an Open-world scenario* (ATDO), significantly generalizing ATRO [2] with discovering multiple unknown or novel patterns in addition to characterizing the existing trajectories behaviors (cf. Fig. 1(c)). Inspired by the benefit of recent deep state space models in effectively capturing long-term dependencies, we devise a *S*patial-temporal *S*tate *S*pace block (*S3*) to capture dependency dynamics in a convolutional view, especially for exploring long-term dependencies in lengthy trajectories. To adapt S3 to an OW setting, we present two adapters, i.e., *Open Inference Net* and *Open Critic Net*, where the former maps each trajectory representation into a universe (OW) space while the latter aims to measure the pattern similarities. More importantly, due to the natural time-dependent learning adaptivity of state space in S3, we define three open-world goals along with subtrajectory augmentation, which can facilitate aligning trajectories that harbor the same unknown patterns while maintaining the ability to categorize known patterns. The main contributions are summarized as follows:

- We formulate a more fine-grained problem of anomalous trajectory discovery in a naturalistic open-world context that has not yet been investigated. In contrast to prior arts, it entails not merely classifying known trajectory patterns and precisely separating the unknown ones, but also advancing by further subdividing the unknown patterns.

- We introduce OSSL to identify anomalous trajectories. It mainly tackles the concern of long-term dependencies during trajectory learning with a flexible state matrix in our devised S3 block. Besides, we build two adapters with three OW objectives to guide the anomalous trajectory discovery in the OW context.
- Our experiments over two large-scale taxi datasets demonstrated the superiority of OSSL over the baselines, and OSSL successfully discovered multiple unknown patterns. Specifically, our proposed OSSL consistently achieves the most remarkable improvement in the setting of the ATRO and ATDO tasks.

The remainder of this paper is organized as follows: We review the most pertinent studies from two perspectives in Section 2. Then, we formalize the problem definition and present the details of our proposed OSSL in Section 3 and Section 4. Next, the experimental results are shown in Section 5. Finally, we conclude this study and outline our future work.

2. Related work

We now provide a review of two global categories of related works – deep trajectory learning and anomalous trajectory detection – and we position our results in that context.

2.1. Deep trajectory learning

In recent years, deep trajectory representation learning has become a critical foundation for capturing dependency dynamics and can facilitate learning for downstream tasks [19]. Various variants of RNNs, due to their proficiency in capturing contextual information and temporal dependencies, are extensively employed for trajectory representation [8,20,21]. For instance, GRLSTM [22] utilizes a multi-layer residual-connected LSTM to model trajectories, aiming to learn robust trajectory representations. TripSafe [9] operates the RNN variant (i.e., LSTM) in an end-to-end manner to explore the sequential dynamics of massive trajectories. However, existing RNN solutions usually confront gradient-vanishing issues [17], yielding poor performance in handling trajectories with diverse lengths. Due to their capacity to reveal the interactions between different elements of a trajectory, recent attention mechanisms have been explored by researchers [23–25]. For instance, Zhang et al. involve a self-attention network to explore diverse spatiotemporal preferences as well as historical sequential dynamics [26]. However, attention-based methods also cannot handle long-term dependencies well, which could fail to maintain individual intents underlying trajectories. Trajectory representation learning has extended beyond merely uncovering the sequential structure of trajectories. CNN-based trajectory representation learning models can capture the complex relationships between trajectories by mapping them into images and learning image representations. For example, TrjSR [27] captures the spatial patterns of trajectories and obtains trajectory embeddings by enhancing the quality of trajectory images. GNN-based models, by constructing a graph structure among trajectory points, can more effectively capture the complex relationships between these points [28,29]. For example, TrajGAT [28] designed a trajectory embedding model based on GAT, which integrates the hierarchical spatial structure and sequential relationships to represent the trajectory. START [30] employs GAT, using rich road features as input to model road access frequency. The travel semantics of road features and access frequency are integrated into the road representation, converting the road network into road segment representation vectors. In contrast, our proposed S3 block, conditioned on SSMs, employs two context-aware distillations for each region, enabling the learning of complex road network structures and inter-region relationships. Additionally, it effectively addresses long-term dependency issues in long-sequence trajectory modeling.

2.2. Anomalous trajectory detection

The majority of recent studies fall into two main categories of anomaly detection techniques [6,19], including heuristic (statistical)-based and learning-based, which primarily seek to discover popular or common routes from normal trajectories. In a nutshell, the heuristic (statistical)-based scheme usually uses the distance or density metrics to produce a deviation score between them and a target trajectory. And a variety of distance metrics can be employed, e.g., Euclidean distance, Hausdorff distance, Dynamic Time Warping (DTW) [6,31], etc. For example, [32] proposed a density-based trajectory outlier detection method, which takes into account the distribution of neighborhood objects. As anomalous trajectories are few and different, an isolation-based method, namely iBAT [5], was devised to discover outlier trajectories. Most heuristic-based solutions rely heavily on statistical distance measurement on trajectories while ignoring the potential transitional dependencies among the locations. The learning-based scheme provides a new perspective by employing machine learning to the feature context from intricate trajectories [33]. For instance, an adapted k-nearest neighbor (kNN) method for identifying the group outlier trajectories was presented in [7]. [34] introduced a novel time-series detection method based on one-class support vector machines. However, traditional learning-based methods are mostly based on simple feature extraction and cannot handle complex semantic dynamics or interactions well. Recent deep learning works have explored the rich context of underlying trajectories via representation learning and have achieved remarkable gains for trajectory-based applications, including anomalous trajectory detection [14,35,36]. For example, [15] presented a supervised learning approach based on recurrent neural networks (RNNs), aiming at exploring the temporal dynamics behind the varying trajectories. An LSTM-based autoencoder to learn the normal traffic pattern as well as the latent features was devised in [12], where the OC-SVM algorithm is combined to enhance the performance of anomaly detection. Due to the inherent uncertainty and diversity of trajectories, recent studies use deep generative models to capture the latent variability from complex and varying trajectories [8,14,35,37]. Most of them employed the seq2seq-based method with variational Bayes [38,39] to encode the trajectories into a low-dimensional space — e.g., GM-VSAE jointly uses a Gaussian Mixture model to learn the diversity underlying massive trajectories. DeepTEA, which considers complex traffic conditions, leverages the variational autoencoder to handle the time-dependent anomalies from a huge volume of trajectories [8]. In this domain, most of the existing studies focus on the CW (or, close-set) scenarios. Typically either supervised or unsupervised learning methods are employed to identify anomalous patterns. For instance, [3] proposed an anomalous trajectory detection and classification (ATDC) method to uncover anomalies in different anomalous patterns with a distance metric. RL4OASD [4] represented an unsupervised anomaly detection algorithm that utilizes network learning to discern features of trajectories and road networks for anomaly identification. ATROM [2] was the first OW solution that can detect the presence of unknown patterns (i.e., separate the trajectories with unknown patterns from the ones with already-known patterns). However, this solution can be regarded as an outlier detection task that is unable to cluster different unknown patterns and lacks versatility in the event of multiple unknown patterns in the future. In contrast, our proposed OSSL is a more general solution that can handle multiple unknown patterns, previously unseen by the learning system.

3. Preliminaries

We now present the basic definition and formalize the problem along with the fundamental background of the state space model.

3.1. Basic definition

Definition 1 (*Raw Trajectory*). A trajectory $T = \{p_1 \rightarrow p_2 \cdots \rightarrow p_n\}$ is a chronological sequence of GPS points. Each point p_i contains a geographical coordinate (e.g., longitude $p_i.lo$ and latitude $p_i.la$) and a timestamp $p_i.t$.

Following recent representative studies [8,14,35], we partition the geographical space into a grid of equal regions and map each GPS point into a region to relieve the modeling overhead problem. Given a city, we divide it into multiple non-overlapping regions by setting fixed physical dimensions. Assume that there are $m_c \times n_c$ equal-sized regions.

Definition 2 (*Mapped Trajectory*). For a given raw trajectory T, its corresponding *mapped trajectory* tr is obtained by mapping each p_i into the corresponding regions in which it is located. Assuming an enumeration (i.e., tokens) for the cells of the grid, tr can be represented as a sequence of region tokens, i.e., $tr = \{r_1 \rightarrow r_2 \cdots \rightarrow r_\tau\}$, $(\tau \leq n)$.

Herein, we treat the *source* and the *destination* of a mapped trajectory tr as $S_{tr} = r_1$ and $D_{tr} = r_\tau$.

Definition 3 (*Open Trajectory Set*). Given a source S and destination D pair (S, D), an open trajectory set $\mathcal{D} = \{tr_1, \ldots, tr_n, \ldots\}$ refers to a collection of trajectories corresponding to (S, D) that are not fixed or limited to a predefined number of routes. Instead, it is dynamic and continuously updated with new trajectories as they are observed. Under this context, open trajectory set $\mathcal{D}$ would include all trajectories of vehicles in this city, constantly updating with new trips including "unknown unknowns" trajectories.

Under the open set definition in our context, we formalize problems (including the previous and ours) as follows:

Anomalous Trajectory Recognition in an Open-world scenario (ATRO). According to the definition in [2], we again posit that the existence of massive collected trajectories in $\mathcal{D}$ can be classified into K known categories, named $\mathcal{C}_K = \{c_1, \ldots, c_K\}$, including the "normal" pattern (denoted as c_1) and $K-1$ kinds of anomalous categories (i.e., $\{c_2, \ldots, c_K\}$). Additionally, we consider that there exist n instances of unknown patterns in $\mathcal{D}$, i.e., these "unknown unknowns", and map them to $\mathcal{C}_K \cup \mathcal{C}_{unk} = \{c_1, \ldots, c_K, c_U\}$. Herein, $\mathcal{C}_{unk} = \{c_U\}$ indicates that all unknown patterns belong to that class (i.e., $U-K=1$). The goal of ATRO is twofold: 1) map the known patterns behind $\mathcal{D}$ to $\mathcal{C}_K$, and 2) recognize the unknown patterns not yet present from the n unknown instances and categorize them into $\mathcal{C}_{unk}$.

Anomalous Trajectory Discovery in an Open-world scenario (ATDO). In our context, based on the "unknown unknowns" of ATRO, we present the refined ATDO problem which is specifically formulated to achieve fine-grained separation of these patterns. That is to say, map the n unknown instances to fine-grained categories, named $\mathcal{C}_K \cup \mathcal{C}_{unk} = \{c_1, \ldots, c_K, c_{K+1}, \ldots, c_U\}$, where $\mathcal{C}_{unk} = \{c_{K+1}, \ldots, c_U\}$ (*i.e.*, $U-K>1$).

3.2. State Space Model (SSM)

We first briefly overview the basics of SSM, manifested in two foundations: *State Spaces* and *Discretization*.

(I) State Spaces. The primary goal of a basic SSM is to map an input $\boldsymbol{u}(t) \in \mathbb{R}^N$ to a latent state $\boldsymbol{x}(t) \in \mathbb{R}^d$ before projecting an output $\boldsymbol{y}(t) \in \mathbb{R}^M$, which can be broadly summarized as:

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \tag{1}$$

where $\boldsymbol{x}'(t) = \frac{d}{dt}\boldsymbol{x}(t)$. The above differential equation is parameterized by *state* matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, *input* matrix $\boldsymbol{B} \in \mathbb{R}^{d\times N}$, *output* matrix $\boldsymbol{C} \in \mathbb{R}^{M\times d}$, and *feedthrough* matrix $\boldsymbol{D} \in \mathbb{R}^{M\times N}$. The most significant *state* matrix $\boldsymbol{A}$, e.g., High-Order Polynomial Projection Operator (HiPPO) [40], is used to control information evolution of the control system, while the rest is considered as projection parameters.

(II) Discretization. Using Eq. (1), we can recursively operate the SSM to tackle each r_τ of a trajectory $\mathcal{T}$, where we treat r_τ as $\boldsymbol{u}(\tau)$ while $\boldsymbol{y}(\tau)$ is the corresponding output (e.g., the hidden state in RNNs). In our context, one major concern is that each trajectory is generated based on discrete timestamp sampling. To address this issue, the discrete-time SSMs [41] allow for a more flexible approach to handling sequential data in a nonlinear view, which primarily seeks to convert *state* matrix $\boldsymbol{A}$ into an approximation $\tilde{\boldsymbol{A}}$. Assuming that there exists a sample time η such that $\tau = (\tau-1) + \eta$, the recurrence form regarding Eq. (1) can be rewritten as follows:

$$\boldsymbol{x}_\tau = \tilde{\boldsymbol{A}}\boldsymbol{x}_{\tau-1} + \tilde{\boldsymbol{B}}\boldsymbol{u}_\tau, \quad \boldsymbol{y}_\tau = \tilde{\boldsymbol{C}}\boldsymbol{x}_\tau + \tilde{\boldsymbol{D}}\boldsymbol{u}_\tau. \tag{2}$$

For the sake of distinction, *state* matrix and *input* matrix are denoted in discrete SSMs by $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$, *output* matrix and *feedthrough* matrix are denoted by $\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{D}}$, respectively. Notably, there are multiple simple but widely used discretization methods to obtain the approximation matrices (e.g., $\tilde{\boldsymbol{A}}$), including Euler, *bilinear*, *zero-order hold* (ZOH), etc. [18,42]. Let $\boldsymbol{x}_{-1} = \boldsymbol{0}$, and Eq. (2) can be rewritten by:

$$\boldsymbol{y}_\tau = \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{\tau}\tilde{\boldsymbol{B}}\boldsymbol{u}_0 + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{\tau-1}\tilde{\boldsymbol{B}}\boldsymbol{u}_1 + \cdots + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{0}\tilde{\boldsymbol{B}}\boldsymbol{u}_\tau + \tilde{\boldsymbol{D}}\boldsymbol{u}_\tau. \tag{3}$$

4. Methodology

4.1. Framework overview

As Fig. 2 shows, OSSL contains three components, i.e., **(I) Spatial–temporal State Space (S3)** block, **(II) Open Inference Net (OIN)**, and **(III) Open Critic Net (OCN)**. In a nutshell, S3 follows the rule of recent state space models, which aims to encode time-dependent trajectories into a latent space by modeling dependency dynamics in addition to incorporating the two spatial semantics into each traveled region. The role of OIN is to map each trajectory representation into an OW pattern space, i.e., $\mathbb{R}^U$. To facilitate the discovery of different unknown patterns, OCN provides a pairwise similarity score between different trajectories, aiming at measuring whether they belong to the same pattern. Finally, we operate these components with three OW objectives, enabling precisely fine-grained cognition of open-set trajectories, i.e., mapping $tr \in \mathcal{D}$ to $\mathcal{C}_k \cup \mathcal{C}_{unk}$.

4.2. Spatial–temporal state space (S3)

We now formally introduce the details of S3, aiming to explore the dependency dynamics behind the trajectories.

4.2.1. Region encoding with semantics

Preparing a dense but meaningful representation for each region is an essential prerequisite for downstream dynamic dependency learning, especially for revealing the mutual interactions among different traveled regions within massive trajectories. To this end, we follow [2] and use two context-aware representations for each region with contrastive loss settings, including the *SD-centric distillation* (denoted as $\boldsymbol{v}_1(\cdot) \in \mathbb{R}^{d_v}$) and the *Region-centric distillation* (denoted as $\boldsymbol{v}_2(\cdot) \in \mathbb{R}^{d_v}$). Specifically, *SD-centric distillation* is derived by maximizing the mutual interaction between a travel need and its corresponding trajectory while *Region-centric distillation* is constructed by maximizing spatial proximity among traveled regions.

Given a trajectory $\mathcal{T} = \{r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_m\}$, the corresponding travel need can be denoted as $\rho_1 = (r_1, r_m)$ (a.k.a., the (S, D) pair), along with the spatial proximity set $\rho_2 = \{r_\tau | \forall r_\tau \in \mathcal{T}\}$. We utilize the metric learning with contrastive loss to maximize the closeness ρ_* with its relative (positive) while minimizing the correlation to its irrelevant (negative), which can be expressed as:

$$\mathcal{J}_{\rho_*} = \mathbb{E}_{r_\tau \in \mathcal{T} \setminus \rho_*}\left[\pi\left(\boldsymbol{v}_*(r_\tau), \boldsymbol{\rho}_*\right) - \log \sum_{j=1}^{J-1} \exp\left(\pi\left(\boldsymbol{v}_*(r'_j)\right), \boldsymbol{\rho}_*\right)\right]. \tag{4}$$

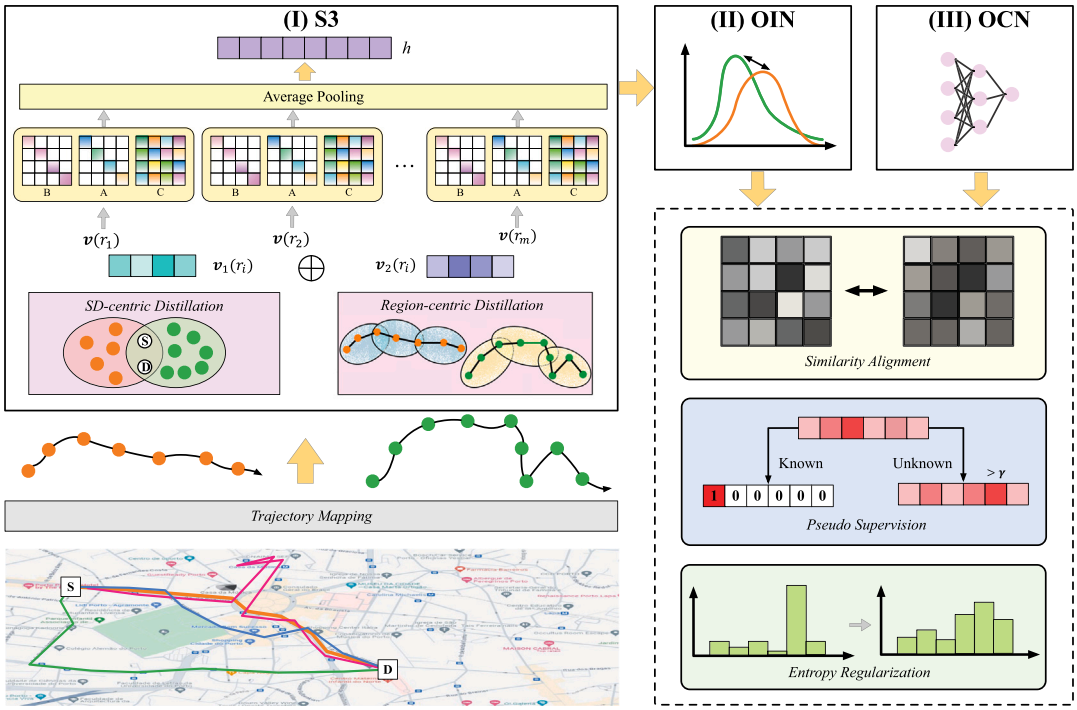


Fig. 2. The framework of OSSL.

Algorithm 1: SD-centric distillation.

Input: A map-matched trajectory set $\mathcal{D}$.
initialize region encodings $\boldsymbol{V}_1 = \{\boldsymbol{v}_1(r_i)\}_{i=1}^{n_c m_c}$;
repeat
  foreach $(S, D) \in \mathcal{D}$ **do**
    // $\mathcal{D}^{(S,D)}$ indicate that all trajectories in the set are from a source S to a destination D
    search for a trajectory subset $\mathcal{D}^{(S,D)}$;
    foreach $\mathcal{T}$ **do**
      remove the consecutive repeated regions;
    end
    // $\mathcal{PD}^{(S,D)}$ is the trajectory set compressed from $\mathcal{D}^{(S,D)}$
    foreach $\mathcal{T} \in \mathcal{PD}^{(S,D)}$ **do**
      set $\boldsymbol{\rho_1} = \boldsymbol{v}_1(r_1) + \boldsymbol{v}_1(r_m)$;
      Obtain positive sample $(\boldsymbol{v}_1(r_\tau))$;
      Obtain negative samples $(\{\boldsymbol{v}_1(r'_j)\}_{j=1}^{J-1})$;
      compute the loss by Eq. (4);
      backpropagate gradients;
      update $\boldsymbol{V}_1$;
    end
  end
until *Convergence*;
Output: Optimal $\boldsymbol{V}_1$.

Herein, ρ_* formulate an *anchor* in contrastive learning [43], '\' is the set-difference operation, excluding the inherent contrastive impact of the travel need itself. π denotes the similarity measurement such as *cosine* function and J represents the sample number. r_τ and r'_j refer to the positive and negative samples of contrastive learning, respectively. For *SD-centric distillation*, where $*= 1$, the concrete *anchor* $\boldsymbol{\rho}_1 = \boldsymbol{v}_1(r_1) + \boldsymbol{v}_1(r_m)$, r_τ refers regions of trajectory from r_1 to r_m is designated as *positive sample*, whereas all other samples are considered *negative sample* r'_j. In the case of *Region-centric distillation*, where $*= 2$, the *anchor* is defined by the context $\boldsymbol{\rho}_2 = \boldsymbol{v}_2(r_{\tau-1}) + \boldsymbol{v}_2(r_{\tau+1})$, associated with negative samples r'_j which are those not included in ρ_2 and positive samples r_τ are regions between anchor. After that, each traveled region r_τ in $\mathcal{T}$ can be formulated as:

$$\boldsymbol{v}(r_\tau) = [\boldsymbol{v}_1(r_\tau); \boldsymbol{v}_2(r_\tau)]\mathbf{W}_r + \mathbf{b}_r, \tag{5}$$

where $[;]$ refers to the concatenation operation. In addition, $\mathbf{W}_r \in \mathbb{R}^{2d_v \times d_v}$ and $\mathbf{b}_r \in \mathbb{R}^{d_v}$ are trainable parameters. For clarity, we detail the workflows of *SD-centric distillation* and *region-centric distillation* in Algorithm 1 and Algorithm 2, respectively.

Algorithm 2: Region-centric distillation.

Input: A map-matched trajectory set $\mathcal{D}$.
initialize region encodings $\boldsymbol{V}_2 = \{\boldsymbol{v}_2(r_i)\}_{i=1}^{n_c m_c}$;
repeat
  foreach $(S, D) \in \mathcal{D}$ **do**
    // $\mathcal{D}^{(S,D)}$ indicate that all trajectories in the set are from a source S to a destination D
    search for a trajectory subset $\mathcal{D}^{(S,D)}$;
    foreach $\mathcal{T}$ **do**
      remove the consecutive repeated regions;
    end
    // $\mathcal{PD}^{(S,D)}$ is the trajectory set compressed from $\mathcal{D}^{(S,D)}$
    foreach $\mathcal{T} \in \mathcal{PD}^{(S,D)}$ **do**
      set $\boldsymbol{\rho_2} = \boldsymbol{v}_2(r_{\tau-1}) + \boldsymbol{v}_2(r_{\tau+1})$;
      Obtain positive sample $(\boldsymbol{v}_2(r_\tau))$;
      Obtain negative samples $(\{\boldsymbol{v}_2(r'_j)\}_{j=1}^{J-1})$;
      compute the loss by Eq. (4);
      backpropagate gradients;
      update $\boldsymbol{V}_2$;
    end
  end
until *Convergence*;
Output: Optimal $\boldsymbol{V}_2$.

4.2.2. Dependency dynamic modeling

The core of the open anomalous trajectory discovery task is in effectively capturing the dependency dynamics, especially long-term ones. Its significance is in maintaining the individual intents behind a lengthy trajectory, which can further help to expose abnormal behavior that differs from normal ones. As each $\mathcal{T}$ is a sequence of traveled

regions, we take advantage of the recent SSMs and use Eq. (2) to explore the long-term dependencies underlying sequentially traveled regions, rewritten as:

$$\boldsymbol{x}_\tau = \tilde{\boldsymbol{A}}\boldsymbol{x}_{\tau-1} + \tilde{\boldsymbol{B}}\boldsymbol{v}(r_\tau), \quad \boldsymbol{h}_\tau = \tilde{\boldsymbol{C}}\boldsymbol{x}_\tau + \tilde{\boldsymbol{D}}\boldsymbol{v}(r_\tau), \tag{6}$$

where $\boldsymbol{h}_\tau$ indicates the output state regarding the input $\boldsymbol{v}(r_\tau)$. Specifically, we employ the commonly used zero-order hold (i.e., ZOH) [17,18] with nonlinear transformation to make discretization of state spaces. Hence, the above recursive procedure can be unrolled as:

$$\boldsymbol{h}_\tau = \sum_{j=0}^{\tau-1} \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^j\tilde{\boldsymbol{B}}\boldsymbol{v}(r_{\tau-j}) + \tilde{\boldsymbol{D}}\boldsymbol{v}(r_\tau), \text{ where } \tilde{\boldsymbol{A}} = \exp(\eta\boldsymbol{A}), \tag{7}$$

$$\tilde{\boldsymbol{B}} = (\eta\boldsymbol{A})^{-1}(\exp(\eta\boldsymbol{A}) - \boldsymbol{I})\eta\boldsymbol{B}, \ \tilde{\boldsymbol{C}} = \boldsymbol{C} \ \tilde{\boldsymbol{D}} = \boldsymbol{D}. \tag{8}$$

For simplicity, we can rewrite Eq. (7) to obtain the output at time step τ as follows:

$$\boldsymbol{h}_\tau = \sum_{j=0}^{\tau-1} \boldsymbol{\mathcal{K}}_j\boldsymbol{v}(r_{\tau-j}) + \tilde{\boldsymbol{D}}\boldsymbol{v}(r_\tau). \tag{9}$$

In recent autoregressive applications [42,44], including ours, Eq. (9) has a convolutional form, avoiding the gap between $\boldsymbol{h}_\tau$ and $\boldsymbol{x}_\tau$, i.e., eliminating the state variable $\boldsymbol{x}_\tau$. This form can be represented as $\boldsymbol{h} = \boldsymbol{\mathcal{K}}_m * \boldsymbol{v}$. As such, $\boldsymbol{\mathcal{K}}_m \in \mathbb{R}^m$ can be denoted as:

$$\boldsymbol{\mathcal{K}}_m = (\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^\tau\tilde{\boldsymbol{B}})_{\tau\in[m]} = (\tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}, \dots, \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{m-1}\tilde{\boldsymbol{B}}). \tag{10}$$

Modeling dependency dynamics can be regarded as a convolutional procedure in a non-autoregressive or parallel manner, where the total output (e.g., $\boldsymbol{h} \in \mathbb{R}^{d\times m}$) will be computed at once, i.e., $\boldsymbol{h} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_m]$. However, the above $\boldsymbol{\mathcal{K}}_m * \boldsymbol{v}$ would require $\mathcal{O}(m^2)$ multiplications, which could be prohibitively slow on lengthy trajectories. Hence, we attempt to solve two limits that slow up the trajectory modeling behind our S3. The first one is to compute the convolutional kernel $\boldsymbol{\mathcal{K}}_m$, and the second one is the initialization of state matrix $\boldsymbol{A}$ and other projections such as $\boldsymbol{B}$.

4.2.3. State settings

For the first limit, we are motivated by structured state space [17,18], and employ Vandermonde matrix multiplication to efficiently obtain the convolutional kernel $\boldsymbol{\mathcal{K}}_m$, which can be rewritten as:

$$\boldsymbol{\mathcal{K}}_m = (\boldsymbol{B}^T \odot \boldsymbol{C}) \cdot \mathcal{V}(\tilde{\boldsymbol{A}}), \tag{11}$$

where $\mathcal{V}(\tilde{\boldsymbol{A}})_{d,m} = \tilde{\boldsymbol{A}}_d^m$. $\mathcal{V}$ is a Vandermonde matrix, $\odot$ is Hadamard product, and $\cdot$ refers to matrix multiplication. $\mathcal{V}(\tilde{\boldsymbol{A}})$ is defined as:

$$\mathcal{V}(\tilde{\boldsymbol{A}}) = \begin{bmatrix} 1 & \tilde{\boldsymbol{A}}_0^1 & \tilde{\boldsymbol{A}}_0^2 & \cdots & \tilde{\boldsymbol{A}}_0^{m-1} \\ 1 & \tilde{\boldsymbol{A}}_1^1 & \tilde{\boldsymbol{A}}_1^2 & \cdots & \tilde{\boldsymbol{A}}_1^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \tilde{\boldsymbol{A}}_{d-1}^1 & \tilde{\boldsymbol{A}}_{d-1}^2 & \cdots & \tilde{\boldsymbol{A}}_{d-1}^{m-1} \end{bmatrix}. \tag{12}$$

For the second concern, due to the convolution being performed in the temporal domain, we initialize state matrix $\boldsymbol{A}$ with **S4D-Lin** scheme [18] to ensure that powers of $\tilde{\boldsymbol{A}}$ does not diverge and lead to exploding gradients. In **S4D-Lin**, $\boldsymbol{A}$ is defined as a complex diagonal matrix of $d \times d$, owning $2d$ trainable parameters and yielding $\mathcal{O}(m+d)$ computation. That is, $\boldsymbol{A}$ is composed of real and imaginary components of the diagonal entries, where any sth diagonal entry is initialized at $A_s = -\frac{1}{2} + i\pi s$ ($s = 0, 1, \dots, d-1$) and i is an imaginary unit. For $\boldsymbol{B}$ and $\boldsymbol{C}$, it is possible to use their product results ($\boldsymbol{B}^T\boldsymbol{C}$) directly instead of individually defining $\boldsymbol{B}$ and $\boldsymbol{C}$. For simplicity and efficiency, we fix $\boldsymbol{B}$ to an identity matrix $\boldsymbol{I}$ and $\boldsymbol{C}$ is randomly initialized with a standard Gaussian sampling, which is consistent with the HiPPO learning theory. As for $\boldsymbol{D}$, it can be treated as a residual network. Hence, we also randomly initialize it with a standard Gaussian sampling.

4.2.4. Trajectory representation

In general, a trajectory $\mathcal{T}$ can be represented as $\boldsymbol{\mathcal{T}} = \{\boldsymbol{v}(r_1), \boldsymbol{v}(r_2), \dots, \boldsymbol{v}(r_m)\}$ after the region encoding. Given a convolution kernel $\boldsymbol{\mathcal{K}}_m$, the outputs of state space regarding $\boldsymbol{\mathcal{T}}$ can be computed at once, denoted by:

$$\boldsymbol{h} = \boldsymbol{\mathcal{K}}_m * \boldsymbol{\mathcal{T}} + \boldsymbol{D}\boldsymbol{\mathcal{T}}, \tag{13}$$

where $*$ refers to convolution operation. In our implementation, we involve the Fast Fourier Transform (FFT) $\mathcal{F}$ to speed up the computation while using the GELU activation [45] to avoid kernel convolution blowing up if trajectories are extremely lengthy, summarized as follows:

$$\boldsymbol{h} = \text{GELU}(\mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{\mathcal{K}}_m) \cdot \mathcal{F}(\boldsymbol{\mathcal{T}})) + \mathbf{W}_D\boldsymbol{\mathcal{T}}), \tag{14}$$

where $\mathbf{W}_D \in \mathbb{R}^{d\times d_v}$ is a trainable parameter. As such, the generalization trajectory representation can be obtained after an average pooling operation:

$$\bar{\boldsymbol{h}} = \frac{1}{m}\sum_{\tau=1}^{m} \boldsymbol{h}_\tau. \tag{15}$$

Eventually, we can manage the outcome of the S3 block into the following open-world learning.

4.3. S3 with open-world learning

To adapt the S3 to an open-world context, we devise two open-world adapters with three optimization objectives.

4.3.1. Open-world adapters

Open Inference Net (OIN). We build OIN $\mathcal{S}_\theta$ with the *Softmax* function to map each trajectory representation $\bar{\boldsymbol{h}}$ to a U space, i.e., $\mathcal{S}_\theta(\bar{\boldsymbol{h}}) \in \mathbb{R}^U$, where the first K ($K < U$) logits indicate that the trajectory $\bar{\boldsymbol{h}}$ belongs to one of the known patterns and the last $U - K$ logits suggest that the trajectory $\mathcal{T}$ belongs to one of the unknown patterns. It should be noted that, due to the nature of the OW context, the learning system does not have prior knowledge about the trajectories with unknown patterns. That is, we do not know whether a trajectory belongs to a known pattern or an unknown pattern, unlike existing semi-supervised learning paradigms.

Open Critic Net (OCN). In contrast to the existing manners, our OSSL enables fine-grained categorization and enhanced data analysis capabilities, resulting in a precise recognition of open-set trajectories (i.e., $tr \mapsto C_k \cup C_{unk}$. Thus, we devise OCN $\mathcal{S}_\psi$ to produce a pairwise similarity score between different trajectories, aiming at measuring whether they belong to the same behavior. Intuitively, there only exist two possible relationships between two trajectories, i.e., either they belong to the same pattern or not. Therefore, the goal of OCN is to produce a similarity probability, i.e., $\mathcal{S}_\psi \mapsto [0, 1]$.

4.3.2. Open-world objectives

We now illustrate our optimization goals in the context of open-world learning.

(I) Similarity Alignment: To bunch the trajectories with different (both known and unknown) patterns, we present an alignment principle to formalize similarity measurement. As the output of the OIN $\hat{\boldsymbol{C}}$ shows the probability distribution of different patterns, given two different trajectories $\mathcal{T}_s$ and $\mathcal{T}_k$, we use the simple similarity measurement (e.g., cosine similarity) to estimate the pattern similarity while using the OCN to measure the trajectory similarity in the latent space. Then, we align these two similarity scores, summarized as follows:

$$\mathbf{a}_1 = ReLU(\mathbf{W}_1\bar{\boldsymbol{h}} + \mathbf{b}_1), \tag{16}$$

$$\mathcal{S}_\psi = \sigma(\mathbf{W}_2\mathbf{a}_1 + \mathbf{b}_2), \tag{17}$$

$$\mathcal{L}_s = (\pi(\hat{\boldsymbol{C}}^s, \hat{\boldsymbol{C}}^k) - \mathcal{S}_\psi(\bar{\boldsymbol{h}}^s, \bar{\boldsymbol{h}}^k))^2_{s\neq k}, \tag{18}$$

where π is the cosine function and σ is the *Sigmoid* activation. $\hat{\boldsymbol{C}}^*$ denotes the output probability from the OIN, e.g., $\hat{\boldsymbol{C}}^s = \mathcal{S}_\theta(\bar{\boldsymbol{h}}^s)$.

(II) Pseudo Supervision: OSSL needs to recognize the unknown patterns in addition to classifying the known patterns. However, OSSL in the training procedure is only informed that some trajectories tag known patterns. Hence, building a purely supervised learning paradigm on these tagged trajectories may produce unacceptable biases when confronted with trajectories of unknown patterns. Motivated by the consistency rule in decision boundaries [46,47], we generate pseudo-patterns for untagged trajectories from the OIN output. To mitigate the risk of unreliable pseudo-patterns, we exclusively generate pseudo-patterns for trajectories with satisfied confidence γ. In addition, to perturb the invariant output distribution in a manner that situates the decision boundary in a low-density area, we seek to produce an augmented version of a given trajectory to facilitate the generation of pseudo-patterns with high quality. Benefiting from the flexibility of S3 on discrete sequential dynamic modeling, we can generate a simplified version of a given trajectory by varying the sampling time η and use this version to produce the original trajectory's pseudo-pattern. For instance, given a trajectory $\mathcal{T}$, we first vary the sampling time $\eta \ast= \Delta \times \eta$ to produce the simple version $\mathcal{T}'$, whereafter we generate the pseudo-pattern by:

$$\tilde{c} = \mathbf{1}_{\max(\hat{\boldsymbol{C}}')}(\hat{\boldsymbol{C}}') \; s.t. \max(\hat{\boldsymbol{C}}') > \gamma. \tag{19}$$

$\tilde{c}$ is actually a pattern index in the implementation, $\hat{\boldsymbol{C}}'$ is the probability output of $\mathcal{T}'$ and $\gamma = 0.8$ leads to a binary pseudo-pattern [46]. After that, we use $\tilde{c}$ as the label of $\mathcal{T}$. In this way, any trajectory without tagging can have a pseudo-pattern. In training, we can use the standard cross-entropy loss to tackle all of the trajectories, where the objective regarding untagged $\mathcal{T}_k$ can be denoted as:

$$\mathcal{L}_p = -\sum_{j=1}^{U} \vec{\boldsymbol{C}}_j^k \log \hat{\boldsymbol{C}}_j^k. \tag{20}$$

Herein, $\vec{\boldsymbol{C}}^{\boldsymbol{k}}$ denotes one-hot embedding of a (pseudo-) pattern.

(III) Entropy Regularization: There is still a risk that the OIN could assign all of the trajectories with different unknown patterns to the same pattern. That is, OSSL could be overly confident in preferring a particular pattern when tackling trajectories with unknown patterns. To ensure a uniform distribution across various patterns, we employ entropy regularization [48,49] to enhance the diversity of pattern predictions, which can be denoted as:

$$\mathcal{L}_r = \sum_{j=1}^{U} \bar{\boldsymbol{C}}_j \log \bar{\boldsymbol{C}}_j, \tag{21}$$

where $\bar{\boldsymbol{C}}_j$ is the average probability of a batch of trajectories. In the implementation, we can use trajectories in each mini-batch to estimate the above objective.

4.3.3. Optimization

Lastly, we introduce the workflow of OSSL with the above three OW objectives. Due to the parallelizability of deep neural networks, we randomly choose a batch of trajectories $\mathcal{D}_b \in \mathcal{D}$ containing b trajectories, where some of them are tagged as known patterns while the rest are not marked at all. That is, there exist trajectories with both known and unknown patterns that are not known to the learning system. Then, we use our S3 to obtain a dense representation for each trajectory $\mathcal{T}^k \in \mathcal{D}_b$. According to Eq. (16), we use OIN to obtain the pattern probabilities and also use our OCN to output the similarity scores across the trajectories in $\mathcal{D}_b$. We use the above three objectives to compute the loss as:

$$\mathcal{L}_{\mathcal{D}_b} = \frac{1}{b}\sum_{j=1}^{b}(\mathcal{L}_s + \mathcal{L}_p + \mathcal{L}_r)_j, \tag{22}$$

Due to the extremely noisy similarity signal provided by OCN at the beginning of the training phase, simultaneously optimizing all parameters would seriously affect the ability of OIN to recognize already unknown patterns. In our implementation, we thus develop an interactive (bi-level) optimization strategy. Specifically, we employ a bi-level optimization which first optimizes S3 and OIN with Eq. (22) by:

$$(\phi^+, \theta^+) = (\phi, \theta) - \alpha_1 \nabla_{\phi,\theta} \mathcal{L}_{\mathcal{D}_b}(\phi, \theta, \psi), \tag{23}$$

where ϕ indicates the trainable parameters in S3. Next, OSSL optimizes the parameters in OCN by:

$$\psi^+ = \psi - \alpha_2 \nabla_{\psi} \mathcal{L}_{\mathcal{D}_b^+}(\phi^+, \theta^+). \tag{24}$$

Herein, $\mathcal{D}_b^+$ is a set of trajectories with the ground-truth known patterns. In this interactive manner, OSSL ensures that updating parameters does not compromise its capability to distinguish between established known patterns.

Algorithm 3 summarizes the workflows of our proposed OSSL. Initially, we acquire both SD-centric and region-centric encoding semantics for each region within the trajectory dataset $\mathcal{D}$ according to Eq. (4). As such, we can obtain two different encodings for each region. Then, we randomly choose a batch of trajectories $\mathcal{D}_b \in \mathcal{D}$ containing b trajectories, where some of them are tagged as known patterns while the rest are not marked at all. Next, we use our S3 to obtain a dense representation for each trajectory $\mathcal{T} \in \mathcal{D}_b$. According to Eq. (16), we use OIN to obtain the pattern probabilities and also use our OCN to output the similarity scores across the trajectories in $\mathcal{D}_b$. We use the above three objectives to compute the total loss. Now we use the interactive optimization strategy. That is, we first optimize the S3 and OIN, then optimize the parameters in OCN. In the end, we can obtain an optimal model that can identify the trajectories in the testing set well. During OSSL training, Algorithm 1 and Algorithm 2 provide an overview of the training process for region encoding with semantics. The computational complexity is $\mathcal{O}(m \cdot |\mathcal{T}| \cdot d_v)$, where m is the length of the trajectory sequence, $|\mathcal{T}|$ denotes the number of trajectories in datasets, and d_v indicates the embedding size. Algorithm 3 outlines the overall training process of our OSSL, where the computational complexities of S3 and open-world adapters are $\mathcal{O}(d + m + mlogm)$ and $\mathcal{O}(d_v^2)$, respectively. Here, d refers to the hidden size of the SSM. Consequently, the overall complexity of Algorithm 3 is $\mathcal{O}(d_v^2)$.

5. Evaluation

In this part, we conduct our experiments on two large-scale taxi trajectory datasets to show the effectiveness of OSSL solution. We first introduce data aspect including preprocessing and ground truth preparations. Then, we describe the experimental setup regarding the ATRO and ATDO tasks, followed by details of the performance analysis.

5.1. Data settings

Datasets & Preprocessing. We conduct our experiments on four widely used taxi trajectory datasets, comprising two large-scale and two small-scale datasets. The larger datasets include taxis trajectories from 442 taxis operating in Porto from Jan. 07, 2013 to Jun.30, 2014, as collected by Liu et al. [14], and taxi traces generated from Chengdu in Aug. 2014 by DiDi Chuxing.[1] The small-scale datasets are ride-hailing trajectories from Xi'an in Oct. 2016 by DiDi Chuxing and taxi mobility traces from approximately 320 taxis operating in Rome, Italy, collected in Feb. 2016 by IEEE DataPort.[2] We follow previous studies [2,14,35] and conduct similar trajectory pre-processing. Specifically, we partition the geographical space into a 100 m × 100 m cell grid and remove SD pairs with less than 25 trajectories to obtain rich trajectories to indicate normal routes. For Chengdu, we partition the geographical space into a grid of 300 m ×300 m cells and sample SD pairs with sufficient trajectories (e.g. almost 40 trajectories for each pair). For Xi'an, we

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

[2] https://ieee-dataport.org/open-access/crawdad-romataxi

Algorithm 3: The learning process of OSSL.

Input: Trajectories $\mathcal{D}$, K known and $U-K$ unknown patterns.

Initialize trainable parameters ϕ, θ, ψ and region encodings;
Obtain the optimal region encoding via Eq. (4) and Eq. (5);
for *sample a batch* $\mathcal{D}_b \in \mathcal{D}$ **do**
  Represent each trajectory $\mathcal{T} \in \mathcal{D}_b$ to $\bar{\boldsymbol{h}}$ as $\bar{\boldsymbol{h}} = \mathcal{S}_\phi(\mathcal{T})$;
  // Calculate pattern similarity and trajectory similarity by using OIN and OCN
  $(\pi(\hat{\boldsymbol{C}}^s, \hat{\boldsymbol{C}}^k), \mathcal{S}_\psi(\bar{\boldsymbol{h}}^s, \bar{\boldsymbol{h}}^k))_{s\neq k}, \hat{\boldsymbol{C}}^* = \mathcal{S}_\theta(\bar{\boldsymbol{h}}^*)$;
  // For each untagged trajectory, using simple version $\mathcal{T}'$ sampled by rate η to generate the pseudo-pattern with confidence threshold γ
  $\tilde{c} = \mathbf{1}_{\max(\hat{\boldsymbol{C}}')}(\hat{\boldsymbol{C}}')$ $s.t.\ \max(\hat{\boldsymbol{C}}') > \gamma$;
  // Align pattern similarity and trajectory similarity
  $\mathcal{L}_s = (\pi(\hat{\boldsymbol{C}}^s, \hat{\boldsymbol{C}}^k) - \mathcal{S}_\psi(\bar{\boldsymbol{h}}^s, \bar{\boldsymbol{h}}^k))^2_{s\neq k}$;
  // Obtain pseudo supervision and entropy regularization objective
  $\mathcal{L}_p = -\sum_{j=1}^{U} \tilde{\boldsymbol{C}}_j^k \log \hat{\boldsymbol{C}}_j^k$;
  $\mathcal{L}_r = \sum_{j=1}^{U} \bar{\boldsymbol{C}}_j \log \bar{\boldsymbol{C}}_j$;
  // Use the above three objectives to compute the loss
  $\mathcal{L}_{\mathcal{D}_b} = \frac{1}{b}\sum_{j=1}^{b}(\mathcal{L}_s + \mathcal{L}_p + \mathcal{L}_r)_j$;
  // Optimize parameters in S3 and OIN with $\mathcal{D}_b$
  update $\mathcal{S}_\phi(\cdot)$, $\mathcal{S}_\theta(\cdot)$ to minimize $\mathcal{L}_{\mathcal{D}_b}$ via Eq. (23);
  // Use trajectories set $\mathcal{D}_b^+$ with the ground-truth known patterns to optimize the parameters in OCN
  update $\mathcal{S}_\psi(\cdot)$ to minimize $\mathcal{L}_{\mathcal{D}_b^+}$ via Eq. (24);
end

Output: Obtain the optimal ϕ^+,θ^+,ψ^+.

Table 1
Trajectory datasets after preprocessing.

Dataset	#Points	#Trajectories	#(S, D)	Avg
Porto	11,635,104	262,574	4,567	44.31
Chengdu	24,266,393	414,414	11,860	58.56
Xi'an	3,256,610	87,571	1,596	37.19
Rome	3,652,570	15,576	1,282	234.50

partition the geographical space into a 100 m × 100 m cells and remove SD pairs with less than 25 trajectories. For Rome, we partition the geographical space into a 250 m × 250 m cells and remove SD pairs with less than 5 trajectories. Table 1 summarizes the statistics of datasets after preprocessing, where '#' and 'Avg' denote the total number of (e.g., trajectories) and the average length, respectively.

Ground Truth. We follow prior works [2,14] and manually label the anomalous trajectories. Then, we treat the labeled trajectories as the ground truth. Specifically, we adhere to the principles of [14] in generating two distinct types of anomalous trajectories, i.e., *detour anomalies* and *route-switching anomalies*. In addition, we follow [2] and generate the trajectories with GPS errors as the anomalies, i.e., *navigation anomalies*. As such, we will collect three types of anomalous trajectories, including *detour*, *route-switching*, and *navigation*. For a fair evaluation, we use 90% of the trajectories as the training set and the rest as the testing set.

5.2. Experimental settings

We conduct two groups of experiments in handling ATRO and ATDO tasks, respectively. Hence, our experimental settings are prepared as follows:

5.2.1. Under ATRO task scenario

Anomaly Settings. To address the ATRO task, we follow [2] to form our dataset by randomly injecting two kinds of labeled anomalous trajectories and one type of unknown anomaly into 70% trajectories, the rest as the normal routes.

Evaluation Protocols. On the ATRO task, we choose three common metrics for ATRO evaluations as previous studies [2,14]: F_1-score, Precision-Recall AUC (PR-AUC), and AUROC (Area Under ROC Curve). Notably, AUROC is typically used in out-of-distribution (OOD) tasks [50].

Baselines. We compare the proposed OSSL with several representative approaches in addressing ATRO task to evaluate its capacity. The details of these approaches are illustrated as follows:

- **SAE** [51] is a traditional end-to-end model using RNNs, aimed at unsupervised feature learning and dimensionality reduction to capture intricate patterns in numerous trajectories. Herein, we employ the GRUs as the basic kernel.
- **ATD-RNN** [15] employs RNNs to encode trajectories into lower-dimensional embeddings in a supervised manner, facilitating the detection of anomalies by identifying deviations from learned trajectory behaviors based on labeled trajectory data.
- **VSAE** [14] is a variational model using a sequence-to-sequence architecture where trajectories are embedded into a latent space with Gaussian assumptions, allowing for robust generative modeling and handling variability in spatiotemporal trajectories.
- **GM-VSAE** [14] extends the VSAE model by incorporating sequential learning and a Gaussian Mixture Model (GMM) to jointly model the probability distribution of trajectories, enhancing clustering and anomaly detection capabilities in diverse trajectories.
- **MainTUL** [23] leverages RNNs to explore sequential patterns underlying trajectories and includes a time-aware self-attentive module to learn temporal dependencies, making it ideal for understanding time-related patterns and anomalies in trajectory data.
- **VQRAEs** [52] use quasi-recurrent neural networks combined with vector quantization to learn discrete latent representations, enabling anomaly detection in time series data by capturing both continuous and discrete patterns.
- **ATROM** [2] is the first model to identify anomalous trajectories in an open-world learning setting, using a probabilistic metric rule to discover unknown patterns, making it suitable for detecting anomalies in scenarios with new and previously unseen patterns.

5.2.2. Under ATDO task scenario

Anomaly Settings. To meet the ATDO task, we inject three anomaly types into 70% trajectories in the training set, with the remainder designated as normal. Precisely, 50% of training data were labeled, including parts of normal type (i.e., the c_1) and a distinct anomaly (i.e., the c_2). The unlabeled half of instances mimic open-set situations, representing "unknown unknowns", meaning the model is aware that such instances exist but does not have any prior information to category.

Evaluation Protocols. On the ATDO task, we follow our baselines [53] to report standard accuracy for known patterns (tagged as *Known*), report clustering accuracy for unknown patterns (tagged as *Unknown*), and report joint accuracy on both known and unknown patterns (tagged as *All*).

Baselines. As all of the baselines described in Section 5.2.1 cannot handle the context of multiple unknown pattern discovery, we select popular benchmarks in the field of Novel Class Discovery (NCD) for comparison, where they use our S3 as the backbone:

- ***k*-means** [54] is a classical clustering method used to tackle ATDO in a purely unsupervised learning manner. It partitions the data into k clusters by minimizing the variance within each cluster, making it effective for identifying groupings in trajectory data without requiring labels.

- **UNO** [49] achieves the ability to discover new patterns of unlabeled instances through a pseudo-labeling method. This method addresses previous learning paradigms that required the addition of clustering objectives and auxiliary targets, enabling the identification of new categories within the data by iteratively refining pseudo-labels.
- **ORCA** [55] operates in an open-world semi-supervised learning setting. It uses a novel uncertainty adaptive margin mechanism to assign examples from unlabeled data to previously seen patterns or form new patterns by grouping similar instances. This mechanism helps in managing the uncertainty inherent in open-world scenarios, enhancing the model's flexibility and robustness.
- **TRSSL** [56] leverages prior knowledge about pattern distribution and employs the Sinkhorn–Knopp algorithm for generating pseudo-labels. This approach aims at estimating the emergence of new patterns by optimizing the assignment of instances to clusters based on their distribution, making it effective for semi-supervised learning tasks where some prior knowledge is available.
- **SORL** [53] proposes a spectral open-world representation learning solution that tailors graph decomposition for open-world settings. By decomposing the graph structure of the data, SORL can effectively capture the underlying relationships and detect new patterns, making it suitable for applications where the data distribution is not fully known and continuously evolving.

5.2.3. Implementation settings

In our implementation, all methods are implemented in Python using the PyTorch library, accelerated by the NVIDIA Tesla A100. For OSSL, we use Adam optimization. We set d_v to 128, J is 10, and the hidden size in S3 is 128. The Δ is 0.5. The initial learning rates α_1 and α_2 are 0.001 and 0.005, respectively. For reproducibility, the source code is available at https://github.com/swufe-NiceLab-GeoText/OSSL.

5.3. Performance in ATRO task

In the open-world context, ATRO primarily seeks to detect whether there is an unknown/unseen anomalous pattern of trajectories that has not been reported (cf. Section 1). Hence, we first tackle ATRO task which is considered as the simplest condition for ATDO — i.e., $U - K = 1$.

Table 2 reports the overall performance (best method in **bold**; second is underlined). To avoid pattern biases in model evaluation, we respectively select each anomalous pattern as the unknown — *Unseen: Navigation* indicates that it cannot be seen by a model. We observe that our proposed OSSL consistently outperforms baselines, achieving 11.7% F_1-score, 18.9% AUROC, and 7.2% PR-AUC averaged improvements compared to the best baseline on Porto, 14.1% F_1-score, 20.6% AUROC, and 6.8% PR-AUC averaged improvements on Chengdu, 42.3% F_1-score, 5.5% AUROC, and 4.4% PR-AUC averaged improvements on Xi'an, 48.7% F_1-score, 12.6% AUROC, and 8.2% PR-AUC averaged improvements on Rome, respectively. In detail, we have the following findings.

Among the baselines, the discriminative approaches such as MainTUL and ATD-RNN perform well on different scale datasets, suggesting that investing more effort in trajectory embedding to capture dependency dynamics can enhance the correlation between trajectories and their actual patterns. By leveraging RNNs and attention mechanisms, they can accurately capture the nuances of trajectory behaviors, highlighting their strength in effectively capturing the complex dependencies within trajectory data, especially smaller ones, where the patterns may be less complex but require precise modeling. The existing generative methods (e.g., GM-VSAE and VQRAE) perform poorly, worse than discriminative methods in most cases. We consider that the possible reason for this is that they are conditional on prior assumptions with latent space learning, which could degrade the model performance in tackling the unknown pattern due to the nature of CW prior settings. This issue, commonly referred to as *concept drift*, arises because generative models may struggle to adapt to new and unseen data distributions, resulting in less effective anomaly detection and pattern recognition. OSSL is superior to ATROM (the first solution of ATRO problem) which discovers unknown patterns using a metric rule conditioned on extreme value theory. As a result, there is significant uncertainty in the recognition of an unknown pattern. We conjecture that there are two reasons: (1) OSSL extends SSM to handle trajectories of different lengths, enabling the efficient capture of dependency dynamics. This extension allows OSSL to model a broader range of trajectory variations, improving its ability to recognize and classify different patterns accurately. (2) We involve three open-world objectives in OSSL optimization, yielding a clearer signal to disentangle the trajectories that have different patterns, including the unknown one. By optimizing for these multiple criteria, OSSL can more effectively separate and identify diverse trajectory patterns, leading to better recognition and classification of anomalies in an open-world context.

5.4. Performance in ATDO task

We now address ATDO in a complex unknown pattern discovery — i.e., $U - K > 1$, which is a more generalized version of ATRO.

As shown in Table 3, it reports the performance of all methods, where the known anomalous pattern is *Detour* (top of Table 3). We can find that our proposed OSSL still consistently outperforms these methods by a large margin in both large- and small-scale datasets, which demonstrates that OSSL has advantages in discovering unknown patterns in addition to known patterns. In addition to the top of Table 3 that reports the known anomalous pattern is *Detour*, we also conduct the experiments that either of the other two anomalous patterns is known for fairness. As shown in the middle and bottom of Table 3, we can observe that OSSL still outperforms the baselines by a large margin on all datasets. In detail, k-means as a simple clustering method performs better in some cases. This suggests that the trajectory representations we provide effectively boost the clustering of diverse patterns, including unknown ones. The success of k-means indicates that our feature engineering captures the essential characteristics of trajectories, facilitating their grouping into meaningful clusters even without complex modeling. From a general point of view, we find that UNO and ORCA perform more stable than other baselines. We consider that a possible reason for this is that they tackle trajectories with untagged patterns in open semi-supervised learning. UNO utilizes a pseudo-labeling approach that iteratively refines labels, which helps adapt to new and unseen patterns. ORCA employs an uncertainty adaptive margin mechanism, which allows it to dynamically adjust to the variability in handling trajectory data. Although TRSSL integrates an uncertainty-guided temperature scaling technique to enhance the reliability of pseudo-labels, it could lead to a biased preference for the known patterns. We can discover that it does not work well in unknown pattern recognition, especially on the Chengdu dataset. The technique meant to improve pseudo-labels might inadvertently overfit to the known patterns, reducing its effectiveness in generalizing to new, unseen data. In contrast, the entropy regularization we developed in the OSSL aims to address this problem effectively. SORL that identifies different patterns with graph decomposition also results in poor performance. The possible reason is that graph-based methods rely heavily on the structural assumptions about the data. In open-world settings, where the data distribution can be highly variable and include previously unseen patterns, such assumptions can lead to incorrect decompositions and poor pattern recognition. In contrast, OSSL with three OW objectives has a higher performance on new pattern discovery. OSSL's superior performance can be attributed to several key factors: (1) OSSL involves optimizing for three distinct open-world objectives. This multi-objective optimization helps in balancing the

Table 2
Performance comparison in addressing ATRO.

Dataset	Method	Unseen: Navigation			Unseen: Detour			Unseen: Route-switching		
		F_1-score	AUROC	PR-AUC	F_1-score	AUROC	PR-AUC	F_1-score	AUROC	PR-AUC
Porto	SAE	0.721	0.702	0.877	0.663	0.666	0.858	0.706	0.726	0.890
	ATD-RNN	0.717	0.704	0.884	0.681	0.678	0.856	0.741	0.755	0.904
	VSAE	0.243	0.503	0.750	0.309	0.499	0.750	0.242	0.504	0.753
	GM-VSAE	0.202	0.493	0.747	0.129	0.497	0.752	0.406	0.525	0.763
	MainTUL	0.709	0.721	0.888	0.667	0.651	0.849	0.709	0.736	0.896
	VQRAEs	0.367	0.468	0.744	0.317	0.466	0.750	0.421	0.577	0.797
	ATROM	0.768	0.801	0.911	0.722	0.763	0.906	0.757	0.779	0.912
	OSSL	**0.806**	**0.914**	**0.976**	**0.867**	**0.952**	**0.981**	**0.837**	**0.920**	**0.970**
Chengdu	SAE	0.638	0.634	0.840	0.639	0.616	0.848	0.127	0.490	0.870
	ATD-RNN	0.618	0.653	0.865	0.597	0.623	0.845	0.672	0.666	0.873
	VSAE	0.243	0.503	0.750	0.309	0.499	0.750	0.242	0.504	0.753
	GM-VSAE	0.216	0.503	0.750	0.434	0.640	0.863	0.139	0.517	0.758
	MainTUL	0.598	0.561	0.797	0.596	0.603	0.839	0.566	0.555	0.826
	VQRAEs	0.394	0.468	0.741	0.312	0.381	0.718	0.465	0.612	0.833
	ATROM	0.785	0.828	0.931	0.604	0.673	0.879	0.753	0.749	0.892
	OSSL	**0.804**	**0.916**	**0.976**	**0.893**	**0.941**	**0.984**	**0.787**	**0.858**	**0.928**
Xi'an	SAE	0.445	0.568	0.779	0.299	0.629	0.831	0.300	0.589	0.805
	ATD-RNN	0.432	0.584	0.800	0.367	0.642	0.828	0.496	0.619	0.819
	VSAE	0.394	0.543	0.784	0.399	0.704	0.872	0.538	0.578	0.796
	GM-VSAE	0.216	0.515	0.756	0.219	0.501	0.752	0.218	0.514	0.755
	MainTUL	0.393	0.572	0.775	0.456	0.614	0.796	0.493	0.553	0.777
	VQRAEs	0.289	0.543	0.787	0.282	0.563	0.798	0.374	0.509	0.757
	ATROM	0.518	0.753	0.903	0.448	0.806	0.928	0.512	0.626	0.814
	OSSL	**0.758**	**0.806**	**0.942**	**0.762**	**0.814**	**0.944**	**0.632**	**0.686**	**0.883**
Rome	SAE	0.239	0.518	0.759	0.248	0.517	0.757	0.206	0.503	0.746
	ATD-RNN	0.291	0.635	0.810	0.332	0.510	0.736	0.354	0.502	0.743
	VSAE	0.233	0.510	0.766	0.235	0.550	0.783	0.224	0.488	0.741
	GM-VSAE	0.213	0.502	0.753	0.218	0.516	0.756	0.210	0.529	0.769
	MainTUL	0.256	0.587	0.758	0.219	0.523	0.772	0.160	0.500	0.737
	VQRAEs	0.247	0.515	0.772	0.244	0.521	0.764	0.254	0.475	0.726
	ATROM	0.416	0.815	0.925	0.312	0.512	0.745	0.535	0.508	0.760
	OSSL	**0.712**	**0.874**	**0.958**	**0.639**	**0.653**	**0.849**	**0.558**	**0.564**	**0.834**

Table 3
Performance comparison in ATDO.

	Methods	Porto			Chengdu			Xi'an			Rome		
		Known	Unknown	All	Known	Unknown	All	Known	Unknown	All	Known	Unknown	All
Detour	k-means	27.9	21.8	46.4	30.0	48.6	55.6	21.8	36.4	51.8	41.5	38.1	48.2
	UNO	62.5	79.3	70.9	61.9	86.4	74.1	72.4	61.5	59.3	64.4	42.9	46.0
	ORCA	63.2	65.5	56.5	71.1	44.3	57.4	57.4	38.3	47.9	54.5	51.8	27.9
	TRSSL	60.7	39.0	49.8	83.2	3.7	43.5	44.9	6.9	25.9	56.6	3.1	29.9
	SORL	39.3	43.1	31.0	47.0	42.4	27.8	38.6	15.4	28.1	35.2	24.4	30.4
	OSSL	**90.4**	**90.1**	**90.3**	**86.6**	**89.2**	**87.9**	**84.3**	**62.5**	**73.4**	**84.1**	**52.8**	**68.5**
Route-switching	k-means	32.6	29.0	47.1	23.5	18.6	51.2	19.3	21.6	52.3	39.6	47.8	47.9
	UNO	81.1	48.1	64.6	79.5	51.4	65.4	63.2	57.5	60.4	55.6	40.5	48.1
	ORCA	75.0	53.3	64.1	72.4	42.5	57.5	36.4	47.4	36.0	49.1	49.9	26.6
	TRSSL	60.3	30.8	45.6	56.7	4.0	30.3	40.1	9.5	26.0	45.5	7.5	26.5
	SORL	61.5	26.3	43.9	43.3	41.4	27.2	38.6	15.4	28.1	39.3	35.5	30.5
	OSSL	**91.6**	**80.1**	**85.8**	**89.3**	**89.2**	**89.3**	**66.1**	**61.1**	**63.6**	**64.8**	**61.4**	**63.1**
Navigation	k-means	33.8	7.9	39.6	45.2	48.7	47.8	40.0	19.3	53.5	44.6	36.9	47.6
	UNO	84.3	49.2	66.7	82.4	49.6	66.0	83.8	50.0	66.9	63.3	30.0	45.2
	ORCA	77.7	46.6	62.2	82.6	44.3	63.5	89.1	1.1	45.1	52.9	7.4	29.4
	TRSSL	82.4	19.7	51.1	80.2	3.6	41.9	48.1	9.7	28.9	54.1	6.5	30.3
	SORL	63.8	45.5	36.0	48.9	45.5	27.6	34.7	22.9	22.8	41.2	22.6	28.3
	OSSL	**97.8**	**58.1**	**77.5**	**96.7**	**60.0**	**78.4**	**96.3**	**52.0**	**74.2**	**91.1**	**47.6**	**69.4**

trade-offs between various aspects of pattern recognition, such as sensitivity to new patterns and robustness to noise. The diverse objectives provide a comprehensive signal that aids in disentangling trajectories with different patterns, including unknown ones. (2) The trajectory embeddings used in OSSL are designed to capture both spatial and temporal dependencies. This robust representation ensures that the embeddings are rich enough to distinguish between different patterns, making it easier to identify both known and unknown anomalies. (3) The incorporation of open-world learning strategies allows OSSL to dynamically adapt to new and evolving data patterns. Unlike methods that rely on fixed assumptions or static models, OSSL can continuously learn and update its understanding of the trajectory dynamics, maintaining high performance even as new patterns emerge. In the following, we will make further investigations to specify the superiority of OSSL design regarding the ATDO task.

5.5. Superiority investigations

Ablation Study. To investigate the impact of our OW objectives, we modify OSSL by removing (i) Similarity Alignment (i.e., $w/o\ s$),

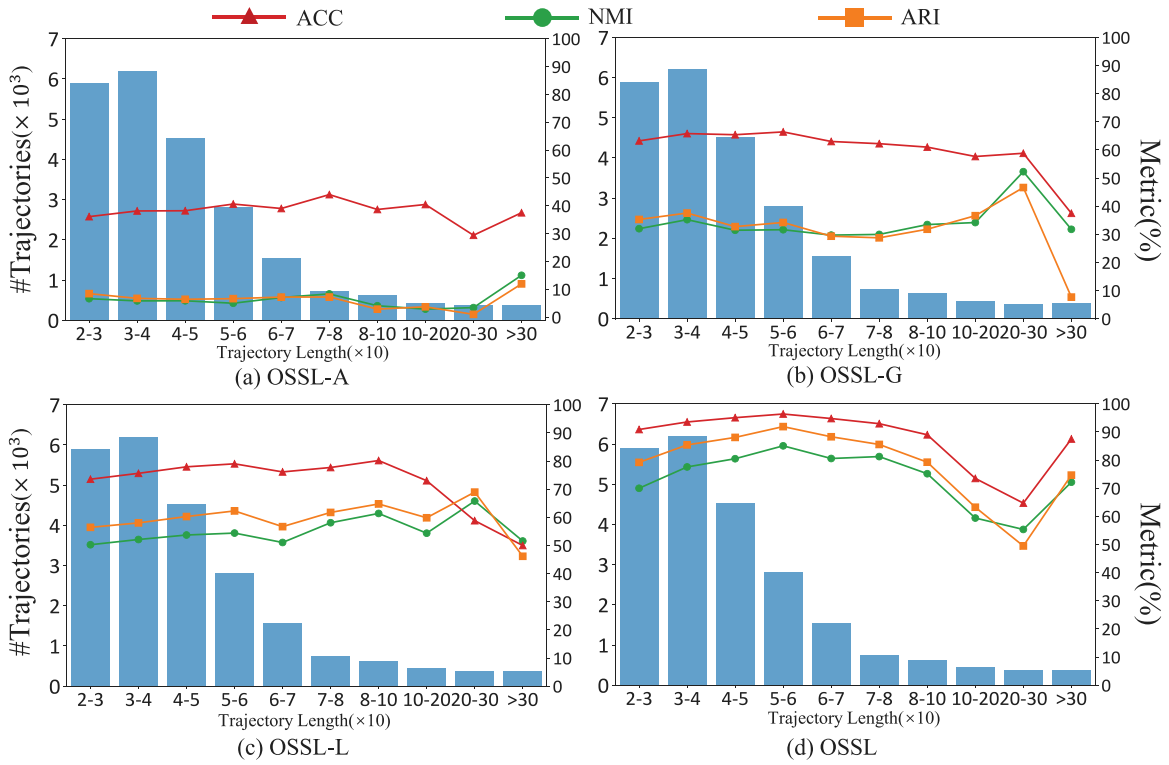


Fig. 3. Performance on trajectories of different lengths.

Table 4
Ablation study on Porto.

	Known	Unknown	All	NMI	ARI
OSSL	**90.4**	**90.1**	**90.3**	**69.6**	**79.3**
w/o s	87.7	70.3	79.0	46.4	46.0
w/o p	90.2	85.8	88.0	60.3	70.4
w/o r	90.3	42.1	66.2	38.9	44.2
w/o bl	88.3	89.1	88.7	67.5	72.3

(ii) Pseudo Supervision $\mathcal{L}_r$ (i.e., *w/o p*), (iii) Entropy Regularization (i.e. *w/o r*), and (iv) bi-level optimization (i.e., *w/o bl*) which is substituted by single-level optimization. Table 4 reports the results on Porto when the known anomalous pattern is *Detour*, including NMI [57] and ARI [58], where NMI (Normalized Mutual Information) and ARI (Adjusted Rand Index) are two additional metrics used to evaluate the effect of clustering. We observe that: (1) our OSSL gains the best results by a large margin, which indicates its effectiveness regarding the objective design; (2) We observe that removing any of the objectives significantly degrades the performance, especially on unknown patterns, further demonstrating their impact in discovering multiple unknown patterns; (3) Compared to OSSL, the single-level optimization variant exhibits a significant decrease in recognizing known patterns, demonstrating our statement that simultaneous optimization carries the risk of forgetting known patterns. Conversely, bi-level optimization strikes a balance between recognizing known patterns and uncovering novels, thereby avoiding performance fluctuation in OW conditions.

Dependency Learning. We build three variants of OSSL to investigate the contribution of S3: OSSL-A replaces S3 with a self-attention block; OSSL-G replaces S3 with the GRU block; and OSSL-L replaces S3 with the LSTM block. For fairness, the three variants above use the same regional encoding as our S3. Fig. 3 presents performance on trajectories of different lengths in Porto, where the blue bar indicates the distribution of trajectory length and red line indicates clustering accuracy for unknown patterns. In detail, when tackling trajectories with lengths $<$ 200, OSSL-G performs robustly, showing RNNs can handle the dependency dynamics well. When tackling trajectory lengths between 200 – 300. OSSL-G and OSSL exhibit a clear opposite turn. We believe the reason is that at this point OSSL-G starts to face a serious problem of vanishing or exploding gradients. OSSL, on the other hand, is more inclined to handle longer trajectories ($>$ 300). As shown in the right vertical axis of Fig. 3, when tackling trajectories with lengths between 200 – 300, OSSL still produces promising results against RNNs. In sum, the results uncover that OSSL can solve complex dependency learning problems, especially in tackling long-term dependencies.

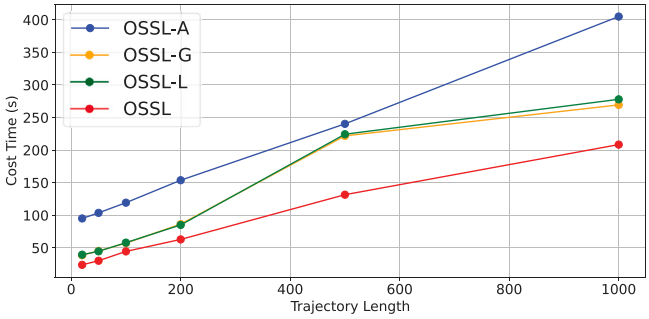


Fig. 4. Efficiency comparison in distinct trajectory lengths.

Efficiency Comparison. In addition to showing the superiority in dependency learning of our S3, we also compare OSSL with the above three variants regarding efficiency. In detail, we take the Porto dataset as an example and extract trajectories of different representative lengths, ranging from 20 to 1000. Since these trajectories of different lengths had different proportions, we replicated trajectories of different lengths to standardize the trajectory proportions (i.e., equal to the size of the training set). As shown in Fig. 4, we can observe that OSSL is the most efficient when compared to these variants, which further demonstrates its superiority in handling the trajectories in different lengths, especially for longer ones.

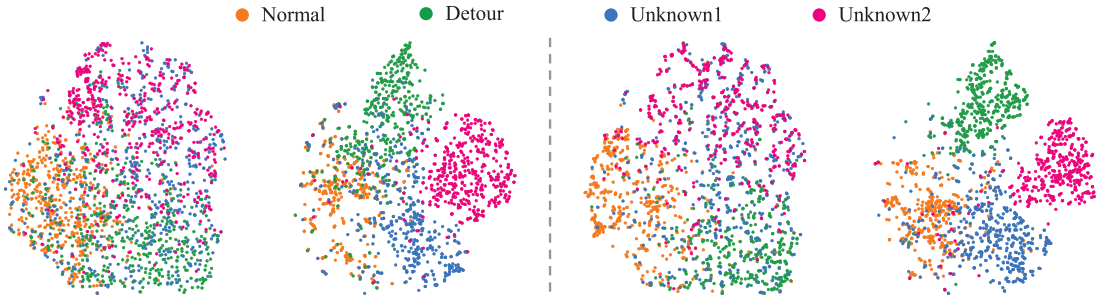


Fig. 5. Trajectory embeddings in Porto (self-att v.s. S3).

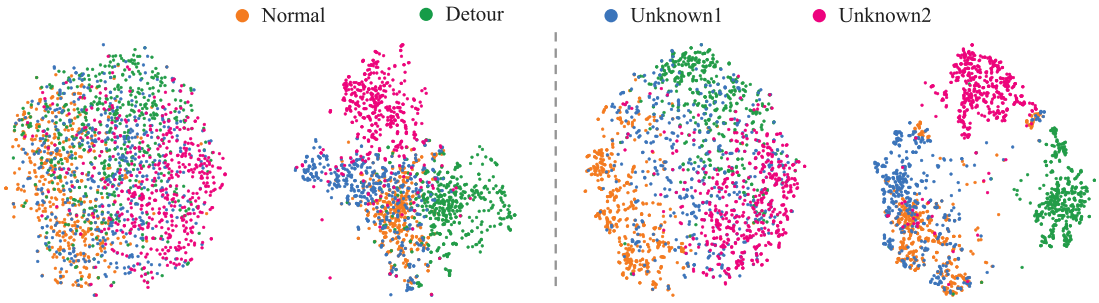


Fig. 6. Trajectory embeddings in Chengdu (self-att v.s. S3).

Interpretability Analysis. Finally, we use the popular t-SNE toolkit to visualize the distribution of trajectory embeddings in the testing set via our proposed OSSL. As shown in Fig. 5 and Fig. 6, we compare our S3 block with the standard self-attention network (denoted as *self-att*), where the left part shows the results of the first 2000 short trajectories while the right part is relative to 2000 long trajectories. We find that our OSSL can disentangle different behavioral patterns underlying trajectories well although the patterns of most trajectories are unknown. Moreover, the comparative results demonstrate that our S3 block exhibits superior capability in handling diverse (e.g., longer) trajectories compared to the self-attention network. This also indicates that the S3 block can more effectively capture the complex relationships and dynamics underlying trajectories, showcasing its powerful interpretability, particularly for long trajectories where behavioral patterns tend to be more intricate. In sum, these findings further underscore the advantages of our proposed solution in uncovering underlying trajectory patterns and its potential to surpass traditional self-attention network in handling complex real-world trajectory data.

Sensitivity Analysis. In this part, we turn to uncovering the impact of different parameter settings. Notably, we select Porto to conduct experiments on the ATDO task where the known anomalous pattern is *Detour*. In particular, the findings can be summarized as follows:

- *The impact of d_v:* We vary the value of d_v in SD- and region-centric embeddings that could affect the model performance. As shown in Fig. 7(a), We can observe that the larger d_v does not bring higher accuracy, indicating that a smaller dimension of embedding can produce better results in addition to the complexity concern.
- *The impact of d:* We use different hidden sizes (i.e., d) in our devised S3 block to investigate the impact of it. As shown in Fig. 7(b), we can find that the model can obtain the best results on three metrics when $d = 128$. When $d > 128$ could result in poor performance in unknown pattern discovery, e.g., $d = 128$.
- *The impact of Δ:* As we use Δ to adjust the sampling time ($\eta^* = \Delta \times \eta$), it can provide us with different simple versions of a given trajectory. Hence, we also vary the value of Δ. As shown in Fig. 7(c), we find that the model performs best if we sample 50% sequentially traveled regions in a given trajectory. Interestingly, we also find that using more traveled regions does not lead to higher performance. The plausible reason for this is that a suitably simple version could force the model to pay more attention to the association between trajectories and unseen patterns.
- *The impact of γ:* In Eq. (19), we set a threshold γ to determine the pseudo-pattern generation. Hence, we vary the value selection regarding γ. As shown in Fig. 7(d), we can discover that the γ significantly affects the model performance. We consider that γ strongly affects the generation of pseudo-patterns. Hence, we should carefully pick the value of γ. In this study, we set $\gamma = 0.8$.

Case Study. Ultimately, we conduct a case study to intuitively illustrate our OSSL's performance on the ATDO task under the open-world context. We randomly select an SD pair in the Porto dataset and visualize its corresponding mapped trajectories, as depicted in Fig. 8. The top is the ground truth, reflecting the trajectory with real patterns, along the bottom shows the recognition results from our OSSL. The four subfigures from left to right show the *Normal* pattern (c_1, known category), *Detour* (c_2, known category), *Unknown 1* (actually route-switch), and *Unknown 2* (actually navigation). First, we find that OSSL demonstrates a distinct impact on the differentiation of known and unknown categories, successfully distinguishing seven trajectories of all categories under our open-world settings. Besides, there are two misclassifications. The intriguing as #T1559, the *Unknown 1* (Ground Truth) vs. detour (recognition), which shares traits with detours from a human perspective, such as an increased distance from the S to D during transition. Another case, #T1798, is misclassified into *Unknown 1*, despite belonging to *Unknown 2* in the settings. We conjecture that this may stem from the reported navigation being close to another route, resulting in ambiguous decision-making, hence the misclassification. Nevertheless, OSSL demonstrates its proficiency in comprehending both spatial and travel semantics of trajectories. This capability is significant for the identification of "unknown unknowns" within the OW setting, yielding a critical requirement for an effective real-world traffic management system.

6. Conclusion

We tackled ATDO – a new general variant of the problem of discovering *and* categorizing patterns of anomalous trajectories in the OW

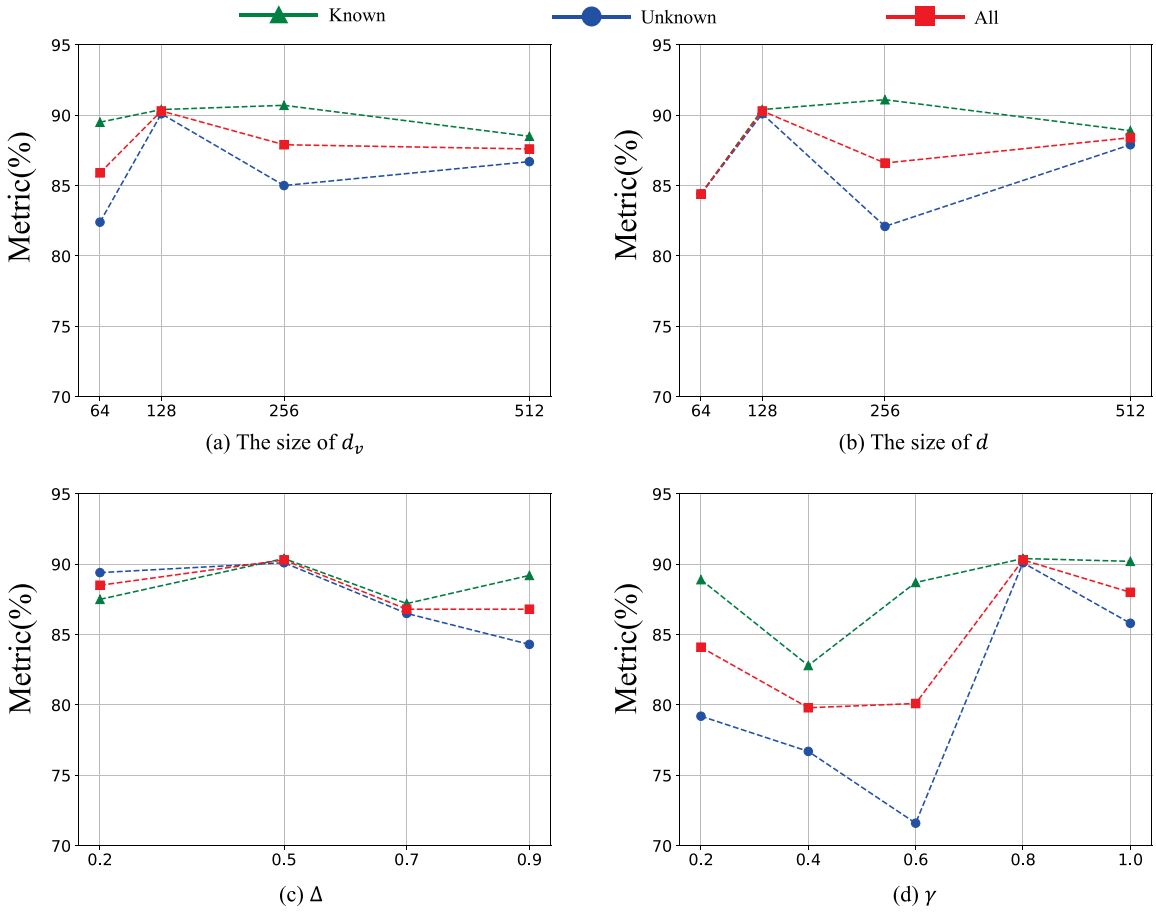


Fig. 7. The sensitivity analysis in Porto.

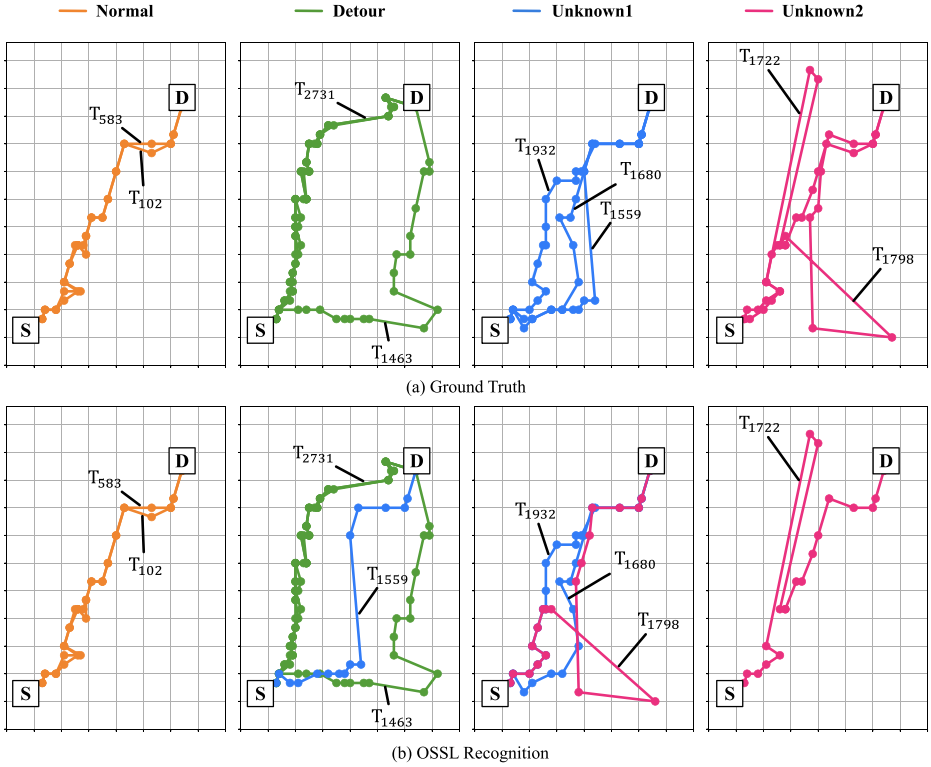


Fig. 8. A case study on ATDO task with OSSL.

context, and presented OSSL – a novel framework for addressing it. It consists of three main components: S3 (Spatial–temporal State Space) block to encode time-dependent trajectories with spatial semantics; OIN (Open Interface Net) to map the representations in OW space; and OCN (Open Critic Net) to assess similarity. We demonstrated that OSSL outperforms the competitors, even for the simpler (ATROM) version of detecting anomalous trajectories in OW. Notably, OSSL exhibits significant advantages in representing long trajectory, efficiently capturing complex patterns over extended periods. As part of our future work, we plan to investigate efficient and effective extensions to OSSL when richer (e.g., social-network) types of data are available. Additionally, we aim to enhance the framework's capability for continual learning of anomalous trajectory patterns and explore its potential for transfer learning across different city, ensuring robust and scalable anomaly detection in diverse urban environments.

CRediT authorship contribution statement

Qiang Gao: Writing – original draft, Methodology, Funding acquisition, Conceptualization. **Chaoran Liu:** Visualization, Validation, Software, Investigation. **Li Huang:** Writing – original draft, Validation, Methodology. **Goce Trajcevski:** Writing – review & editing, Methodology. **Qing Guo:** Writing – review & editing, Resources. **Fan Zhou:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62102326, the Sichuan Science and Technology Program under Grant No. 2023ZYD0145, the Natural Science Foundation of Sichuan Province under Grant No. 2023NSFSC1411 and No. 25QNJJ0627.

Data availability

Data will be made available on request.

References

[1] M. Zhang, T. Li, Y. Yu, Y. Li, P. Hui, Y. Zheng, Urban anomaly analytics: Description, detection, and prediction, IEEE Trans. Big Data 8 (3) (2020) 809–826.
[2] Q. Gao, X. Wang, C. Liu, G. Trajcevski, L. Huang, F. Zhou, Open anomalous trajectory recognition via probabilistic metric learning, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, 2023, pp. 2095–2103.
[3] J. Wang, Y. Yuan, T. Ni, Y. Ma, M. Liu, G. Xu, W. Shen, Anomalous trajectory detection and classification based on difference and intersection set distance, IEEE Trans. Veh. Technol. 69 (3) (2020) 2487–2500.
[4] Q. Zhang, Z. Wang, C. Long, C. Huang, S.-M. Yiu, Y. Liu, G. Cong, J. Shi, Online anomalous subtrajectory detection on road networks with deep reinforcement learning, in: 2023 IEEE 39th International Conference on Data Engineering, ICDE, IEEE, 2023, pp. 246–258.
[5] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, S. Li, iBAT: Detecting anomalous taxi trajectories from GPS traces, in: Proceedings of the 13th International Conference on Ubiquitous Computing, 2011, pp. 99–108.
[6] F. Meng, G. Yuan, S. Lv, Z. Wang, S. Xia, An overview on trajectory outlier detection, Artif. Intell. Rev. 52 (4) (2019) 2437–2456.
[7] Y. Djenouri, D. Djenouri, J.C.-W. Lin, Trajectory outlier detection: New problems and solutions for smart cities, ACM Trans. Knowl. Discov. Data (TKDD) 15 (2) (2021) 1–28.
[8] X. Han, R. Cheng, C. Ma, T. Grubenmann, DeepTEA: effective and efficient online time-dependent trajectory outlier detection, Proc. VLDB Endowm. 15 (7) (2022) 1493–1505.
[9] Y. Su, D. Yao, X. Zhou, Y. Zhang, Y. Fan, L. Bai, J. Bi, TripSafe: Retrieving safety-related abnormal trips in real-time with trajectory data, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 2446–2450.
[10] X. Chen, H. Zhang, F. Zhao, Y. Cai, H. Wang, Q. Ye, Vehicle trajectory prediction based on intention-aware non-autoregressive transformer with multi-attention learning for internet of vehicles, IEEE Trans. Instrum. Meas. 71 (2022) 1–12.
[11] Y. Lin, H. Wan, S. Guo, J. Hu, C.S. Jensen, Y. Lin, Pre-training general trajectory embeddings with maximum multi-view entropy coding, IEEE Trans. Knowl. Data Eng. (2023).
[12] M. Said Elsayed, N.-A. Le-Khac, S. Dev, A.D. Jurcut, Network anomaly detection using LSTM based autoencoder, in: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, 2020, pp. 37–45.
[13] A. Moavinis, A. Gounaris, I. Constantinou, Detection of anomalous trajectories for vehicle traffic data, in: G. Fletcher, V. Kantere (Eds.), Proceedings of the Workshops of the EDBT/ICDT 2023 Joint Conference, Ioannina, Greece, March, 28, 2023, in: CEUR Workshop Proceedings, vol. 3379, CEUR-WS.org, 2023, pp. 1–11.
[14] Y. Liu, K. Zhao, G. Cong, Z. Bao, Online anomalous trajectory detection with deep generative sequence modeling, in: 2020 IEEE 36th International Conference on Data Engineering, ICDE, IEEE, 2020, pp. 949–960.
[15] L. Song, R. Wang, D. Xiao, X. Han, Y. Cai, C. Shi, Anomalous trajectory detection using recurrent neural network, in: International Conference on Advanced Data Mining and Applications, Springer, 2018, pp. 263–277.
[16] C. Li, G. Feng, Y. Li, R. Liu, Q. Miao, L. Chang, DiffTAD: Denoising diffusion probabilistic models for vehicle trajectory anomaly detection, Knowl.-Based Syst. (2024) 111387.
[17] A. Gupta, A. Gu, J. Berant, Diagonal state spaces are as effective as structured state spaces, Adv. Neural Inf. Process. Syst. 35 (2022) 22982–22994.
[18] A. Gu, K. Goel, A. Gupta, C. Ré, On the parameterization and initialization of diagonal state space models, Adv. Neural Inf. Process. Syst. 35 (2022) 35971–35983.
[19] S. Wang, Z. Bao, J.S. Culpepper, G. Cong, A survey on trajectory data management, analytics, and learning, ACM Comput. Surv. 54 (2) (2021) 1–36.
[20] T.-Y. Fu, W.-C. Lee, Trembr: Exploring road networks for trajectory representation learning, ACM Trans. Intell. Syst. Technol. 11 (1) (2020) 1–25.
[21] Q. Gao, J. Hong, X. Xu, P. Kuang, F. Zhou, G. Trajcevski, Predicting human mobility via self-supervised disentanglement learning, IEEE Trans. Knowl. Data Eng. (2023).
[22] S. Zhou, J. Li, H. Wang, S. Shang, P. Han, GRLSTM: trajectory similarity computation with graph-based residual LSTM, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 4972–4980.
[23] W. Chen, S. Li, C. Huang, Y. Yu, Y. Jiang, J. Dong, Mutual distillation learning network for trajectory-user linking, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI, 2022, pp. 1973–1979.
[24] Y. Chen, H. Zhang, W. Sun, B. Zheng, Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer, in: 2023 IEEE 39th International Conference on Data Engineering, ICDE, IEEE, 2023, pp. 829–842.
[25] J. Zhu, X. Niu, F. Li, Y. Wang, P. Fournier-Viger, K. She, StTraj2Vec: A spatio-temporal trajectory representation learning approach, Knowl.-Based Syst. (2024) 112207.
[26] L. Zhang, Z. Sun, Z. Wu, J. Zhang, Y.S. Ong, X. Qu, Next point-of-interest recommendation with inferring multi-step future preferences, in: Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI, 2022, pp. 3751–3757.
[27] H. Cao, H. Tang, Y. Wu, F. Wang, Y. Xu, On accurate computation of trajectory similarity via single image super-resolution, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–9.
[28] D. Yao, H. Hu, L. Du, G. Cong, S. Han, J. Bi, TrajGAT: A graph-based long-term dependency modeling approach for trajectory similarity computation, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2275–2285.
[29] X. He, W. He, Y. Liu, X. Lu, Y. Xiao, Y. Liu, ImNext: Irregular interval attention and multi-task learning for next POI recommendation, Knowl.-Based Syst. 293 (2024) 111674.
[30] J. Jiang, D. Pan, H. Ren, X. Jiang, C. Li, J. Wang, Self-supervised trajectory representation learning with temporal regularities and travel semantics, in: 2023 IEEE 39th International Conference on Data Engineering, ICDE, IEEE, 2023, pp. 843–855.
[31] R. Laxhammar, G. Falkman, Online learning and sequential anomaly detection in trajectories, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2013) 1158–1173.
[32] Z. Liu, D. Pi, J. Jiang, Density-based trajectory outlier detection algorithm, J. Syst. Eng. Electron. 24 (2) (2013) 335–340.
[33] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, Proc. VLDB Endowm. 15 (9) (2022) 1779–1797.
[34] J. Ma, S. Perkins, Time-series novelty detection using one-class support vector machines, in: Proceedings of the International Joint Conference on Neural Networks, 2003, vol. 3, IEEE, 2003, pp. 1741–1745.

[35] X. Li, K. Zhao, G. Cong, C.S. Jensen, W. Wei, Deep representation learning for trajectory similarity computation, in: 2018 IEEE 34th International Conference on Data Engineering, ICDE, IEEE, 2018, pp. 617–628.
[36] C. Li, G. Feng, Y. Li, R. Liu, Q. Miao, L. Chang, DiffTAD: Denoising diffusion probabilistic models for vehicle trajectory anomaly detection, Knowl.-Based Syst. 286 (2024) 111387.
[37] T. Huang, P. Chen, R. Li, A semi-supervised VAE based active anomaly detection framework in multivariate time series for online systems, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1797–1806.
[38] D.P. Kingma, M. Welling, Auto-encoding variational bayes, 2013, arXiv:1312.6114.
[39] Y. Wang, A.C. Miller, D.M. Blei, Comment: Variational autoencoders as empirical Bayes, Statist. Sci. 34 (2) (2019) 229–233.
[40] A. Gu, T. Dao, S. Ermon, A. Rudra, C. Ré, Hippo: Recurrent memory with optimal polynomial projections, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1474–1487.
[41] A. Tustin, A method of analysing the behaviour of linear systems in terms of time series, J. Inst. Electr. Eng. -Part IIA: Autom. Regul. Servo Mech. 94 (1) (1947) 130–142.
[42] L. Zhou, M. Poli, W. Xu, S. Massaroli, S. Ermon, Deep latent state space models for time-series generation, in: International Conference on Machine Learning, PMLR, 2023, pp. 42625–42643.
[43] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv:1807.03748.
[44] D.W. Romero, A. Kuzina, E.J. Bekkers, J.M. Tomczak, M. Hoogendoorn, Ckconv: Continuous kernel convolution for sequential data, 2021, arXiv preprint arXiv:2102.02611.
[45] A. Gu, K. Goel, C. Ré, Efficiently modeling long sequences with structured state spaces, 2021, arXiv preprint arXiv:2111.00396.
[46] M.N. Rizve, K. Duarte, Y.S. Rawat, M. Shah, In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning, 2021, arXiv preprint arXiv:2101.06329.
[47] M.N. Rizve, N. Kardan, S. Khan, F. Shahbaz Khan, M. Shah, Openldn: Learning to discover novel classes for open-world semi-supervised learning, in: European Conference on Computer Vision, Springer, 2022, pp. 382–401.
[48] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, A. Joulin, Unsupervised learning of visual features by contrasting cluster assignments, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 9912–9924.
[49] E. Fini, E. Sangineto, S. Lathuiliere, Z. Zhong, M. Nabi, E. Ricci, A unified objective for novel class discovery, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9284–9292.
[50] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, G. Peng, Conditional Gaussian distribution learning for open set recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13480–13489.
[51] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, LSTM-based encoder-decoder for multi-sensor anomaly detection, 2016, arXiv preprint arXiv:1607.00148.
[52] T. Kieu, B. Yang, C. Guo, R.-G. Cirstea, Y. Zhao, Y. Song, C.S. Jensen, Anomaly detection in time series with robust variational quasi-recurrent autoencoders, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 1342–1354.
[53] Y. Sun, Z. Shi, Y. Li, A graph-theoretic framework for understanding open-world semi-supervised learning, Adv. Neural Inf. Process. Syst. 36 (2024).
[54] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, 1967, pp. 281–297.
[55] K. Cao, M. Brbic, J. Leskovec, Open-world semi-supervised learning, in: International Conference on Learning Representations, 2022, pp. 1–19.
[56] M.N. Rizve, N. Kardan, M. Shah, Towards realistic semi-supervised learning, in: European Conference on Computer Vision, Springer, 2022, pp. 437–455.
[57] D. Cai, X. He, J. Han, Locally consistent concept factorization for document clustering, IEEE Trans. Knowl. Data Eng. 23 (6) (2010) 902–913.
[58] K.Y. Yeung, W.L. Ruzzo, Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data, Bioinformatics 17 (9) (2001) 763–774.