



# SDOoop: Capturing Periodical Patterns and Out-of-phase Anomalies in Streaming Data Analysis

Alexander Hartl   
Institute of Telecommunications  
TU Wien  
1040 Wien, Austria  
me@alexhartl.eu

Félix Iglesias Vázquez   
Institute of Telecommunications  
TU Wien  
1040 Wien, Austria  
felix.iglesias@tuwien.ac.at

Tanja Zseby   
Institute of Telecommunications  
TU Wien  
1040 Wien, Austria  
tanja.zseby@tuwien.ac.at

**Abstract**—Streaming data analysis is increasingly required in applications, e.g., IoT, cybersecurity, robotics, mechatronics or cyber-physical systems. Despite its relevance, it is still an emerging field with open challenges. SDO is a recent anomaly detection method designed to meet requirements of speed, interpretability and intuitive parameterization. In this work, we present SDOoop, which extends the capabilities of SDO’s streaming version to retain temporal information of data structures. SDOoop spots contextual anomalies undetectable by traditional algorithms, while enabling the inspection of data geometries, clusters and temporal patterns. We used SDOoop to model real network communications in critical infrastructures and extract patterns that disclose their dynamics. Moreover, we evaluated SDOoop with data from intrusion detection and natural science domains and obtained performances equivalent or superior to state-of-the-art approaches. Our results show the high potential of new model-based methods to analyze and explain streaming data. Since SDOoop operates with constant per-sample space and time complexity, it is ideal for big data, being able to instantly process large volumes of information. SDOoop conforms to next-generation machine learning, which, in addition to accuracy and speed, is expected to provide highly interpretable and informative models.

**Index Terms**—Contextual Anomalies, Streaming Data Analysis, Anomaly Detection, Communication Networks, Critical Infrastructures

## I. INTRODUCTION

In data stream processing, data points  $v_j \in \mathbb{R}^D$  consistently arrive at monotonically increasing times  $t_j \in \mathbb{R}$  for  $j = 1, 2, \dots$ . Due to this steady acquisition, analysis algorithms face the challenge of discovering knowledge in unbounded data that substantially accumulates in a short time. In such a context, real-life applications dismiss batch-mode operation while demanding fast online processing able to update models and parameters to *concept drift*. Here, “updating models and parameters” does not only mean adapting to new patterns and classes, but also forgetting those that have become obsolete.

In anomaly/outlier detection (OD), we commonly set a sliding window (or an observation horizon)  $w$  that establishes the memory length for which space geometries are remembered. Hence, the anomaly is defined: (a) either based on the neighborhood of a data point within  $w$  (e.g., Exact- and Approx-STORM [1]), or (b) by comparing to a model that evolves with  $w$  (e.g., Robust Random Cut Forests [2]). In both cases, note that the comparison reference is purely static (or

geometric) relative to the point of comparison at the instant of comparison. That is, the data within  $w$  (or the model used instead) is a snapshot. While this is not a problem for many types of anomalies, most traditional methods are blind to identify contextual anomalies. A contextual (aka. conditional or out-of-phase) anomaly “occurs if a point deviates in its local context” [3], i.e., if it happens outside its usual time. Consider a method whose  $w$  spans a one-week period. If a cluster occurs exclusively during weekends, but a data point of this cluster accidentally appears on a Wednesday, this method *will not* identify it as an anomaly, but as a normal inlier instead.

Here, we present SDOoop (SDO out-of-phase), an algorithm for OD in streaming data whose models store temporal information. While retaining constant per-sample space and time complexity and keeping intact the functionalities to detect other types of anomalies, SDOoop is also able to identify contextual anomalies and capture periodical patterns that explain the time behavior of the data bulk. SDOoop builds models by sampling a fixed number of data points at representative locations in feature space, called *observers*. To escape dependence of the data volume, it uses an exponentially weighted moving average (EWMA) to estimate model information from the arriving data mass. At the same time, observers hold temporal information as coefficients of Fourier transforms (FT). Thus, for a specific time of interest

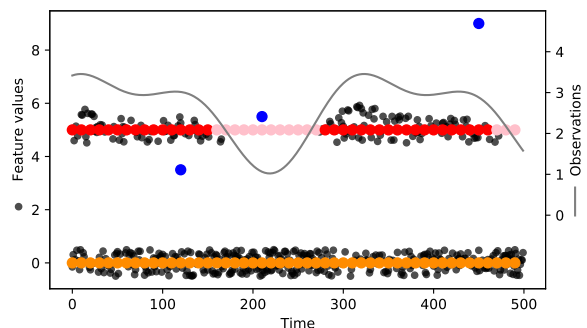


Fig. 1. Example of a data stream, a model with two observers (red and orange), and three types of anomalies (blue): local (left), contextual (middle), global (right).

$t$ , observers “twinkle” to show only the most representative model for time  $t$ . The simple example in Fig. 1 can give an intuition of the distinctive capabilities of SDOoop when compared with alternatives. In the figure, the internal model (incrementally updated) is formed by the red and orange points, which represent observers. Observers are placed in areas of considerable density to see the data mass around. Note that, while the orange observer stands for a cluster that occurs at a continuous pace, the red observer represents a cluster that exhibits a temporal behavior. Consequently, the red observer *twinkles* accordingly, the drawn gray curve showing the inverse FT of its captured FT coefficients. If a contextual anomaly happens, closest observers will not be awake, hence it will be detected as an anomaly by far observers.

Our work advances the research on observers-based unsupervised learning, which originated SDO [4], SDOstream [5] and SDOclust [6]. The remainder of this paper is structured as follows: In Section II, we introduce observers-based OD. Section III describes SDOoop and explains its parameters. SDOoop is evaluated in Section IV. In Section V we explore related research efforts and contrast them to the problem solved here. Finally, we summarize the main ideas and contributions in Section VI. To enable reproducibility, we make all our source code available in our repository <https://github.com/CN-TU/tpsdos-experiments>.

## II. OBSERVERS-BASED OUTLIER DETECTION

The Sparse Data Observers (SDO) method [4] for OD on static datasets is the foundation of our current proposal. In a nutshell, SDO works as follows: (a) Randomly sample points from the dataset, which will be called “observers”. (b) Each data point in the dataset is *only* observed by the  $x$  nearest observers, resulting in each observer performing a different number of observations. (c) Remove *idle* observers, i.e., with the smallest number of observations. Remaining observers are termed *active*. (d) For each data point, compute an outlier score as median distance to the  $x$  nearest *active* observers. Hence, observers capture the main shapes of the data in a low-density model and outlier scores are calculated as distances to points in this model. Removing idle observers minimizes the chances of outliers being part of the model. The observer-approach also holds for SDOstream [5], which adapts the algorithm for a streaming setting by continuously sampling new observers and using EWMA for computing observations.

In this paper, we predominantly adhere to the notation of [5]. Hence, we denote the observers set as  $\Omega$  and, accepting a slight abuse of notation, we denote by  $\omega \in \Omega$  both an abstract observer and its feature vector. Furthermore,  $P_\omega$  denotes  $\omega$ ’s observations, where  $P_\omega \in \mathbb{N}_0$  for SDO and  $P_\omega \in \mathbb{R}_0^+$  for SDOstream. Hence,  $P_\omega$  counts the number of data points for which  $\omega$  belongs to the  $x$  nearest observers with an algorithm parameter  $x \in \mathbb{N}$ . Observers with insufficient  $P_\omega$  are thus disregarded for outlier scoring. In contrast to SDO and SDOstream, SDOoop replaces the number of observations  $P_\omega$  with a temporal function, allowing active observers to become

TABLE I  
SYMBOLS AND NOTATION

Algorithm Parameters	$k \in \mathbb{N}$	Number of observers.
	$x \in \mathbb{N}$	Number of nearest observers.
	$T \in \mathbb{R}^+$	EWMA time constant.
	$T_0 \in \mathbb{R}^+$	FT base period.
	$N_{\text{bins}} \in \mathbb{N}$	Number of frequency bins.
	$q_{\text{id}} \in [0, 1]$	Observer idle-active fraction.
Algorithm State	$\Omega$	Observers set.
	$P_{\omega,n} \in \mathbb{C}$	Fourier coefficients for $\omega$ ’s observations.
	$H_\omega \in \mathbb{R}^+$	Reference for age-normalization of $P_{\omega,0}$ .
	$i_{\text{LAO}} \in \mathbb{N}$	Index of last added observer.
Further Notation	$\omega \in \Omega$	An observer.
	$d(\cdot, \cdot)$	A distance function.
	$\mathcal{N} \subset \Omega$	Set of nearest observers.
	$\mathcal{N}_a \subset \Omega$	Set of nearest active observers.
	$n \in [N_{\text{bins}}]$	The frequency index.
	$\in_R$	Uniformly random sampling from a given set.
	$\mathbb{R}, \mathbb{C}$	Sets of real and complex numbers, respectively.
	$\Re(\cdot), \Im(\cdot)$	Real and imaginary part, respectively.

temporarily idle (i.e., *asleep*) and reappear dynamically in accordance with the temporal pattern of the underlying clusters. Therefore, it is possible to construct an active observers set representative for the data stream at the current time and, hence, to detect data points that do not meet the established temporal pattern, i.e., contextual anomalies.

## III. SDOOOP

We describe the construction of our proposal. Main symbols and notation are shown in Table I. We denote by  $[N_{\text{bins}}]$  with  $N_{\text{bins}} \in \mathbb{N}$  the set  $\{0, \dots, N_{\text{bins}} - 1\}$  and by  $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}^+$  a distance function (e.g., Euclidean). Our method enables the model to absorb temporal patterns in processed data streams. To describe this, we consider data streams satisfying the following definition.

*Definition 1:* For a given data stream, let  $\gamma(\mathbf{v}, t) \in \mathbb{R}_0^+$  denote the expected rate of arriving data points at location  $\mathbf{v} \in \mathbb{R}^D$  and time  $t \in \mathbb{R}$ . Therefore,  $\gamma(\mathbf{v}, t)\Delta\mathbf{v}\Delta t$  stands for the expected number of data points seen in a volume  $\Delta\mathbf{v}$  and time interval  $\Delta t$ . We say that the stream exhibits  $T_0$ -periodic patterns with  $T_0 \in \mathbb{R}^+$  if  $\gamma(\mathbf{v}, t)$  is  $T_0$ -periodic, i.e.,  $\gamma(\mathbf{v}, t) = \gamma(\mathbf{v}, t + T_0)$  for all  $\mathbf{v} \in \mathbb{R}^D, t \in \mathbb{R}$ .

Definition 1 is based on the *expected* rate of arriving data points. This means that, to reason about periodic behaviors, the random stream is modeled as generated by an underlying deterministic process. In particular, a stationary stream exhibits  $T_0$ -periodic patterns for any  $T_0$ . Note that Definition 1 does not include concept drift, which is tackled by SDOoop with an exponential sliding window.

To capture temporal patterns, we allow the observers’ observations to be  $T_0$ -periodic. We represent and store the associated temporal functions in terms of their FT coefficients  $P_{\omega,n} \in \mathbb{C}$  with  $\omega \in \Omega, n \in [N_{\text{bins}}]$ . To extract observers relevant for the current point in time from the model, we first

define the  $q_{id}$ -percentile  $P_{thr} \in \mathbb{R}^+$  of the observers' average observations  $P_{\omega,0}$ , i.e.,

$$P_{thr} = \max \left\{ \rho \in \mathbb{R}^+ \mid \left| \{ \omega \in \Omega \mid P_{\omega,0} < \rho \} \right| \leq q_{id} |\Omega| \right\}. \quad (1)$$

Similar to previous work,  $P_{thr}$  allows us to require active observers to have a minimum number of observations in relation to the total time-averaged observation count. Hence, we construct a view yielding the currently *active* observers

$$\Omega_a = \left\{ \omega \in \Omega \mid \Re \left\{ \sum_{n \in [N_{bins}]} P_{\omega,n} \right\} \geq P_{thr} \right\} \quad (2)$$

in terms of a lower-bound for the inverse FT, where  $\Re \left\{ \sum_{n \in [N_{bins}]} P_{\omega,n} \right\}$  evaluates the temporal shape of the observers' observations at the current time.

To narrow the scope to the most relevant information, we form sets from both  $\Omega$  and  $\Omega_a$  that only contain the  $x$  nearest points. Hence, for a point  $\mathbf{v} \in \mathbb{R}^D$  we specify the set of nearest observers  $\mathcal{N}(\mathbf{v}) \subset \Omega$  with  $|\mathcal{N}| = \min(x, |\Omega|)$  and the set of nearest *active* observers  $\mathcal{N}_a(\mathbf{v}) \subset \Omega_a$  with  $|\mathcal{N}_a| = \min(x, |\Omega_a|)$ , i.e.

$$d(\tilde{\omega}, \mathbf{v}) \leq d(\omega, \mathbf{v}) \forall \tilde{\omega} \in \mathcal{N}(\mathbf{v}), \omega \in \Omega \setminus \mathcal{N}(\mathbf{v}) \quad \text{and} \quad (3)$$

$$d(\tilde{\omega}, \mathbf{v}) \leq d(\omega, \mathbf{v}) \forall \tilde{\omega} \in \mathcal{N}_a(\mathbf{v}), \omega \in \Omega_a \setminus \mathcal{N}_a(\mathbf{v}). \quad (4)$$

Algorithm 1 depicts the core process, discussed as follows.

---

**Algorithm 1** Processing a data point  $(\mathbf{v}_i, t_i)$ .

---

- 1: Find  $x$  nearest observer sets  $\mathcal{N}(\mathbf{v}_i)$  and  $\mathcal{N}_a(\mathbf{v}_i)$
  - 2: **report**  $\text{median}_{\omega \in \mathcal{N}_a} d(\omega, \mathbf{v}_i)$  as outlier score
  - 3: Set  $H_\omega \leftarrow H_\omega \exp(-\frac{t_i - t_{i-1}}{T}) + 1 \quad \forall \omega \in \Omega$
  - 4: Set  $P_{\omega,n} \leftarrow P_{\omega,n} \left[ \exp(-\frac{1}{T} + \frac{jn2\pi}{T_0}) \right]^{t_i - t_{i-1}} \quad \forall \omega \in \Omega, n \in [N_{bins}]$
  - 5: Set  $P_{\omega,n} \leftarrow P_{\omega,n} + 1 \quad \forall \omega \in \mathcal{N}, n \in [N_{bins}]$
  - 6: **if**  $|\Omega| = 0$  **or**  $r \leq \frac{k^2}{Tx} \frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}} \frac{t_i - t_{iLAO}}{i - i_{LAO}}$  **with**  $r \in_R [0, 1]$  **then**
  - 7:   Remove  $\arg \min_{\omega \in \Omega} \frac{P_{\omega,0}}{H_\omega}$  from  $\Omega$  **if**  $|\Omega| = k$
  - 8:   Add  $\mathbf{v}_i$  to  $\Omega$
  - 9:   Set  $i_{LAO} \leftarrow i, H_{\mathbf{v}_i} \leftarrow 1$  and  $P_{\mathbf{v}_i,n} \leftarrow 1 \quad \forall n \in [N_{bins}]$
  - 10: **end if**
- 

### A. Algorithm Construction

Algorithm 1 can be divided into three parts: establishing active observers  $\Omega_a$  (line 1), scoring outlierness (line 2), and updating the model (lines 3-10). The core concept, which allows to capture periodical patterns, is based on Lemma 1.

*Lemma 1:* For an observer  $\omega \in \Omega$ , let  $g(t) \in \mathbb{R}^+$  denote the expected rate of arriving data points, for which  $\omega$  is contained in  $\mathcal{N}$  at time  $t$ . If  $g(t)$  is a  $T_0$ -periodic function and  $T \gg T_0$ , observations  $P_{\omega,n}$  approximate a Fourier transform  $E\{P_{\omega,n}\} \approx \int_{-T_0}^0 g(\tau - t) \exp(-j2\pi n\tau/T_0) d\tau$  up to a constant factor.

We prove the lemma in Appendix A. Lemma 1 shows that temporal information about how frequently observers are used can be extracted from  $P_{\omega,n}$  in terms of an inverse FT. To obtain the current set of active observers  $\Omega_a$ , it suffices by

selecting observers that have been used most often in the past. Here, observer activity is mainly evaluated at time  $t - T_0$ , which is reasonable due to  $T_0$ -periodicity. However, due to inherent interpolation, also the very recent activity of observers is considered, which is particularly relevant for setting new observers. In Theorem 1, we show that our method applies this approach for constructing  $\Omega_a$ .

*Theorem 1:* At time  $t$ , for data streams with  $T_0$ -periodic patterns, the active observers set  $\Omega_a$ , as used by Algorithm 1, contains observers with highest  $g(t)$ .

*Proof 1:* Equation 2 constructs the set  $\Omega_a$  by selecting observers from  $\Omega$ , for which  $\Re\{\sum_{n \in [N_{bins}]} P_{\omega,n}\}$  is highest. If  $P_{\omega,n}$  yields the FT of  $g(t)$  according to Lemma 1, the theorem follows immediately, since  $\Re\{\sum_{n \in [N_{bins}]} P_{\omega,n}\}$  performs the inverse FT at time  $t = 0$  relative to the current time.

Theorem 1 allows us to use  $\Omega_a$  for assessing outlierness of arriving data points by leveraging nearest-observer distances. Hence, in line 1,  $\mathcal{N}$  and  $\mathcal{N}_a$  are constructed. Based on [7], we compute an outlier score with the median of distances to the  $x$  closest observers. The final part of Algorithm 1 handles model updating, which involves replacing the less ‘‘active’’ observer. Updating of  $P_{\omega,n}$  in line 4 follows an exponential shape set by time  $T$ . We show in Theorem 2 that replacing observers proceeds with the same pace, which is necessary as observers otherwise would not be able to build meaningful  $P_{\omega,n}$  values.

*Theorem 2:* For data streams exhibiting  $T_0$ -periodic patterns, Algorithm 1 on average samples  $k$  data points during a time period  $T$  as new observers.

We prove the lemma in Appendix B. Note that the factor  $\frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}/x}{\sum_{\omega \in \Omega} P_{\omega,0}/k}$  occurring in line 6 of Algorithm 1 might be omitted without invalidating Theorem 2. However, we include it to promote representativity of the observers set. Hence, underrepresented observers in a neighborhood cause the observation count in this neighborhood to increase, leading to a higher sampling probability, while overrepresented observers lead to a lower sampling probability. Moreover, this factor is stronger during the transient starting phase, ensuring that the model soon reaches its full size, but at the same time avoiding that it is fulfilled with the first points, which in many cases would build unrepresentative models.

During time  $T$  the model is replaced one time on average according to Theorem 2. Since we use a fixed-size model, an observer has to be removed when adding a new one, picking the removal candidate based on its  $P_{\omega,n}$ . To avoid new observers constantly replaced due to the stronger inertia of old observers, we use an age-normalized observation count  $\frac{P_{\omega,0}}{H_\omega}$  for selecting the observer to remove in line 7. By updating  $H_\omega$  as depicted in line 3,  $H_\omega$  denotes the maximum  $P_{\omega,0}$  that an observer  $\omega$  might have reached over time. Thus,  $\frac{P_{\omega,0}}{H_\omega} \in [0, 1]$ , where 1 is only scored if  $\omega$  has always been in  $\mathcal{N}$  since it was assimilated in the model.

### B. Interpreting the Learned Model

Direct analysis of patterns in streams from a manual perspective is inherently complicated. Experts quickly run into difficulties regarding how to observe the data, what reference

points to take, for how long, how much data to use, how to do this incrementally, etc. SDOoop solves all these issues in a natural and elegant way. At any point in time, the observers set  $\Omega$  allows prompt access to highly representative data points that additionally retain temporal information. The temporal shape  $g_\omega(t)$  of observed data points in a neighborhood of a given observer  $\omega$  can be efficiently recovered in terms of an inverse FT,

$$g_\omega(t) = \Re \left\{ \sum_{n \in [N_{\text{bins}}]} P_{\omega,n} \exp(jtn2\pi/T_0) \right\}. \quad (5)$$

SDOoop is designed to be embedded in the data processing pipeline of stand-alone systems. In other words, its primary purpose is feeding subsequent analysis phases, e.g., visualization or clustering techniques to extract further knowledge. Nevertheless,  $\Omega$  is commonly small enough for manual inspection, meaning that we can explore the set of observers along different time spans and study their periodicities, but also isolate a given instant in time to focus only on its stream characteristics. By analyzing observers in  $\Omega_a$ , the data analyst obtains an immediate depiction of the data model to easily interpret both the ground of the outlieriness scoring and how data are (or are expected to be) as a whole.

### C. Costs and Parameters

The computational cost lies primarily in the comparison of incoming data points with the learned model. Assuming that distance computation is  $\mathcal{O}(D)$  with the number of dimensions  $D$ , building  $\Omega_a$  (equation 2) implies time complexity  $\mathcal{O}(kN_{\text{bins}}) + \mathcal{O}(kD)$ . Holding the model in memory requires storing the observers  $\omega$  and storing their observations  $P_{\omega,n}$ , similarly resulting  $\mathcal{O}(kN_{\text{bins}}) + \mathcal{O}(kD)$  as space complexity. Therefore, per-point space and time complexity linearly depend on model size  $k$ , which is a pre-fixed parameter. This makes SDOoop suitable for big data with highly demanding processing.

Temporal behavior is captured by  $T$ ,  $T_0$  and the number of frequency bins  $N_{\text{bins}}$ .  $T$  is the time constant of the exponential windowing mechanism. It governs memory length, therefore equivalent to the window length of sliding window algorithms.  $T_0$  denotes the period of the FT base frequency. Periodicities

can be captured best if  $T_0$  is an integer multiple of expected periodicities. For instance, in many real-world applications, it might be reasonable a  $T_0$ -value of one week, so that weekly and diurnal patterns can be detected. Furthermore, to ensure that the EWMA approximates a Fourier integral,  $T_0$  should be reasonably smaller than  $T$ .  $N_{\text{bins}}$  determines the maximum frequency that can be captured by the model, hence also fixing the temporal resolution of the learned temporal shapes.

$T$ ,  $T_0$  and  $N_{\text{bins}}$  are intuitive parameters and can be easily adjusted based on domain knowledge.  $q_{\text{id}}$ ,  $k$  and  $x$  are discussed extensively in [4] and [5]. Here suffice it to mention that further experimentation confirms  $q_{\text{id}}$ ,  $k$  and  $x$  robustness, meaning that performances are stable for a wide range of values and that most applications work properly with default configurations. The setting of  $k$  depends on the expected variability and degree of representation, but several hundred observers is sufficient in most cases.  $x$  inherits from nearest-neighbor algorithms, with similar tuning strategies [8]. In our experiments, values between  $k \in [100, 1000]$ ,  $x \in [3, 9]$  and  $q_{\text{id}} \in [0.1, 0.3]$  have shown excellent results.

## IV. EXPERIMENTAL EVALUATION

In this section, we discuss the experimental evaluation of SDOoop. Based on a proof of concept, we first demonstrate its capability to detect contextual outliers. We then proceed by benchmarking OD performances on public datasets. We finally show the discovery and modeling of temporal patterns in real-life cases.

### A. Contextual Anomalies: Proof of Concept

For this proof of concept, we use MDCGen [7] to generate a synthetic data stream of five clusters that vanish and reappear at different times and periods. We add both spatial and contextual outliers into the stream. Fig. 2 shows an excerpt of the generated data stream with 0.5% of contextual outliers. Hence, while normal outliers are distributed across the entire feature space, outliers occurring out of phase fall in the same spatial location as clustered data, but their time of appearance does not meet the temporal shape of clustered data.

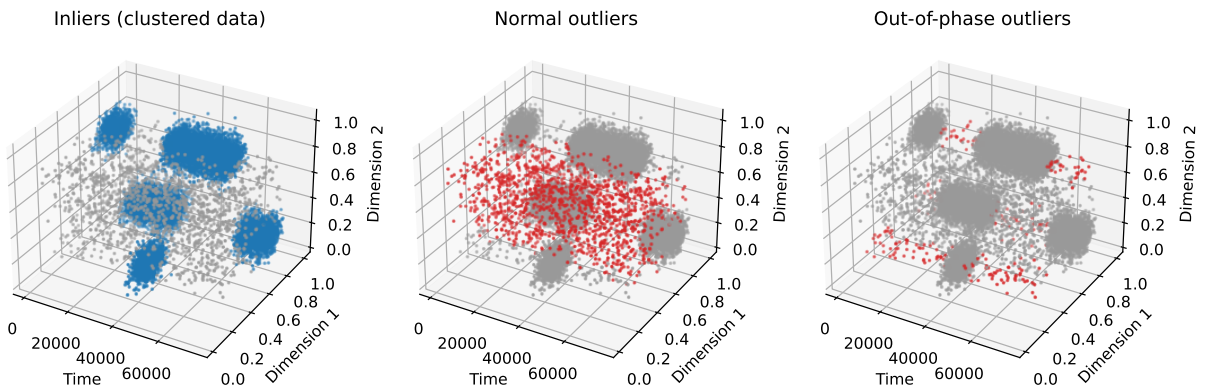


Fig. 2. Normal outliers and contextual outliers (aka out-of-phase outliers) in synthetic data for a fraction of contextual outliers of 0.5%.

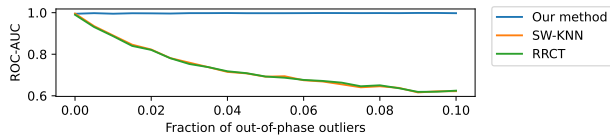


Fig. 3. OD performance vs contextual outlier rate in the proof of concept.

In Fig. 3, we plot the area under the ROC curve (AUC) for different ratios of contextual outliers to data points in one active cluster. We compare SDOoop with two consolidated OD methods for streaming data: SW- $k$ NN (the sliding-window implementation of  $k$ NN for OD [9]) and RRCT [2]. All algorithms have been properly tuned to capture at least one full period. The more outliers occur out of phase, the more the performance of traditional algorithms plummets, whereas our method retains the highest AUC at all times. This clearly indicates that SDOoop is the only algorithm capable of detecting contextual outliers.

### B. OD Performance with Evaluation Datasets

To compare our method with state-of-the-art stream OD algorithms, we selected popular OD datasets of sufficient length and with timestamped data points.

**Datasets and metrics.** The KDD Cup’99 dataset [10] aims at detecting network intrusions based on a number of network and host features and, similar to previous work [11], we considered User to Root (U2R) attacks as outliers over normal traffic, resulting in 976,414 data points with an outlier proportion of 0.4%. Additionally, we selected the recent SWAN-SF [12] dataset, which collects data about solar flares, and used preprocessing scripts provided by Ahmadzadeh and Aydin [13]. For SWAN-SF, we assigned a normal label to the majority class and an outlier label to the remaining classes, resulting in 331,185 data points with an outlier portion of 17.2%. In both experiments, we randomly sampled 50% of the data stream for randomized hyperparameter search and the other half for evaluation. For an overview of the ranges of hyperparameters, we refer to the code repository of this paper. Metrics for evaluation are Adjusted Average Precision (AAP), Adjusted Precision at  $n$  (AP@ $n$ ), and AUC [11].

**Algorithms and experimental setups.** We used the dSalmon framework [14], which provides efficient versions of several stream OD algorithms. Since ensembles commonly exhibit superior accuracy, we used an ensemble of nine for SDOoop, yet noticing almost no difference compared to a single SDOoop detector.

**OD performances.** Experiment results in Table II show how our method matches and even outperforms state-of-the-art algorithms for streaming OD. The strongest competitor is RS-Hash [15]. In the SWAN-SF case, SDOoop ranks among the best performers, while, in the KDD Cup’99 dataset, it clearly stands out, particularly in AAP and AP@ $n$ . The higher AP@ $n$  also indicates that our method finds several true outliers that pass unperceived for the competitors.

**Disclosing insights about the data.** Obtained results seem consistent with data contexts. Considering how outliers have

TABLE II  
PERFORMANCE COMPARISON WITH DIFFERENT OD ALGORITHMS

	SWAN-SF [12]			KDD Cup’99 [10]		
	AAP	AP@ $n$	AUC	AAP	AP@ $n$	AUC
SW- $k$ NN	0.69	<b>0.56</b>	<b>0.91</b>	0.07	0.15	0.72
SW-LOF	0.15	0.12	0.58	-0.00	-0.00	0.67
LODA [16]	0.72	0.54	<b>0.91</b>	0.10	0.13	0.92
RS-Hash [15]	<b>0.73</b>	0.55	<b>0.91</b>	0.13	0.15	0.95
RRCT [2]	0.23	0.19	0.69	0.07	0.05	0.85
SDOoop	<b>0.73</b>	0.55	<b>0.91</b>	<b>0.33</b>	<b>0.54</b>	<b>0.97</b>

been defined in the SWAN-SF dataset, we do not expect that outliers break possible temporal periodicities in samples from solar flares. On the other hand, the KDD Cup’99 dataset describes events in a computer network, which are expected to exhibit strong temporal patterns due to human activity. Patterns may be broken by attack traffic, leading to contextual outliers, behaviors that can be spotted by our method. Here, the superior detection of SDOoop not only indicates that the data show temporal patterns, but also that some U2R attacks are indeed contextual outliers.

### C. Temporal Patterns in Machine-to-Machine Communication

**Application context.** We study network traffic captured in a critical infrastructure, in particular, of an energy supply company that connects charging stations for electric vehicles. The network communication satisfies management, accounting and maintenance aspects<sup>1</sup>. Network communication for these purposes usually adopts the OCPP protocol. Due to the large portion of machine-to-machine communications, we expected to discover distinct periodic patterns.

**Preprocessing and parameters.** We preprocessed data with the feature vector described in [17], resulting in 13 million flows during a 1 month period. We parameterized the algorithm using  $T = 1$  week, 2000 frequency bins and  $T_0 = 2000$  minutes, obtaining a minimum period of 1 minute. We used 400 observers.

**Capturing periodical patterns/clusters.** Fig. 4 shows on the left side examples for the frequency spectrum (magnitude) learned by observers. Hence, different clusters show diverse temporal patterns. While observer 1 shows no or just weak periodicities, observer 2 shows a clear 5 minute periodicity and observers 3 and 4 show a 10 minute periodicity. From the learned FT, temporal shapes can be constructed in terms of an inverse FT as depicted in equation 5. Fig. 4 also shows the reconstructed temporal shape plotted over a 1 hour and 24 hour period. Hence, beneath the periodicities already found when inspecting the FT directly, the temporal shape for observers 3 and 4 additionally shows periodicities of a longer period of approximately 2.5 hours.

**Interpreting clusters in the application.** The manual examination of network flows represented by observers confirmed the soundness of discovered temporal patterns. For

<sup>1</sup>While we embrace reproducible research, issues related to confidentiality, security and privacy prevent us from making these data publicly available.



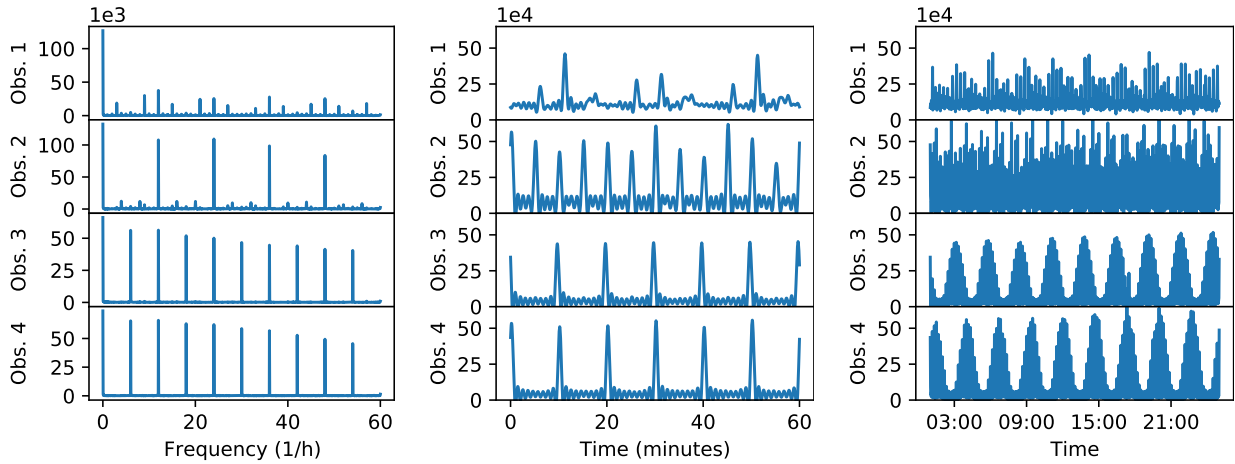


Fig. 4. Learned magnitude spectrum (left), one-hour temporal plots (middle) and 24-hour temporal plots (right) for four exemplary observers when processing network data captured in an e-charging infrastructure.

example, observer 3 corresponds to ICMP pings that happen regularly to ensure that network devices are alive. Observer 4 identifies DNS requests that charging stations perform to resolve the name of the OCPP server to its IP address and transmit meter readings. For observer 4, the periodicity emerges from DNS caching, so that every second request for transmitting meter readings can be performed without having to perform a DNS lookup.

Observer 1 corresponds to protocol heartbeat messages. The fact that it does not show a clear periodicity might be due to the requesting devices not being time-synchronized or by deviating device configurations. Alternatively, heartbeat messages might take place with a very high frequency, so that no periodicities can be observed at the analyzed time scale.

**Identifying outliers in the application.** Fig. 5 shows outlier scores of points in time order. The manual inspection of flows with highest outlieriness (in the center of Fig. 5) revealed that they are firmware update processes. Since updates took place only during two days in the monitored time span, then high outlier scores are consistent (yet they are not contextual anomalies).

**Learning stability.** Finally, we investigated whether our results meet the expected algorithm behavior with respect to the sampling of new observers. Fig. 6 shows how many data points have been sampled as new observers during the first two weeks. With  $T = 1$  week and  $k = 400$  observers,  $400/7 \approx 57$  observers should be sampled each day according to equation 2. This theoretical conjecture shows good agreement with the

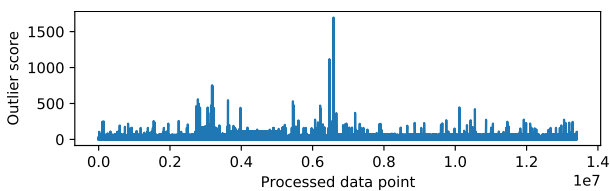


Fig. 5. Outlier scores of network data from an e-charging infrastructure.

empirical results. Fig. 6 also shows that the model is not instantly filled with observers in the first hours, but it is instead built up during the first days. Since data seen within the first couple of hours might not be representative for the remaining data, this transient behavior boosts the swift discovery of a representative model while achieving a fast model buildup with a high sampling rate at the beginning.

#### D. Discovery of Temporal Patterns: Darkspace Data

**Application context and parameterization.** We additionally tested our method on the publicly available CAIDA “Patch Tuesday” darkspace dataset [18]. During preprocessing, we aggregated features by source IP address using the AGM feature vector [19], specifically proposed for analyzing darkspace data. We applied our algorithm with  $T_0 = 1$  week and  $T = 10$  weeks and 100 observers and 100 frequency bins, resulting in a minimum period length of about 100 minutes.

**Capturing and identifying periodical patterns/clusters.** Fig. 7 shows the magnitude of the Fourier coefficients of the three strongest observers. Peaks in Fig. 7 occur at the 7th and 14th frequency bins, which are diurnal and semi-diurnal periodicities. This coincides with previous studies of the darkspace [19] that, among others, identified Conficker.C worm attacks or BitTorrent misconfigurations for diurnal patterns, and horizontal scan, vertical scan and probing activities on the UDP protocol for semi-diurnal patterns.

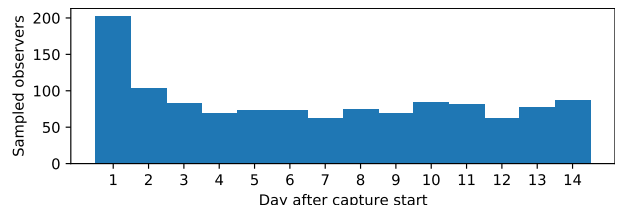


Fig. 6. Sampling of arriving data points as new observers when processing network data captured in an e-charging infrastructure.

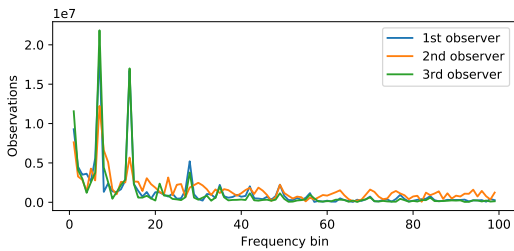


Fig. 7. FT of the top three observers after processing darkspace data.

## V. RELATED WORK

The problem addressed in this paper is covered in diverse fields, yet differing in some core aspects. In this section, we provide an overview of related concepts and precedents to place SDOoop in the landscape of existing work.

**Time Series Analysis and Contextual Anomalies.** A time series is a temporal sequence of observations of specific measurement variables. Time series have been studied in multiple domains, e.g., finance and econometrics [20] weather forecasting [21], electric load forecasting [22]. Traditionally, time series have been analyzed with mathematical tools [20], however, in multivariate time series complexity increases dramatically and experts usually resort to nonlinear machine learning, e.g., [21], [23] Multivariate time series and streaming data are frequently considered synonyms, showing small differences open to discussion [24].

Contextual anomalies have been tackled mainly in the time series analysis domain [25], but here experts also emphasize the low attention given to contextual anomalies in spite of its relevance for cybersecurity, healthcare sensory and fraud detection [26]. Latest research tends to expand the focus and, besides point anomalies, face *anomalous sequences* within the whole time series context, giving rise to fast model-based detectors [27]. However, when considered, contextual anomalies are confronted mainly from a univariate time series perspective. An exception is the recent work by Pasini et al. [28], which copes with low-dimensional multivariate time series and proposes a global *contextual variance* score by weighting *feature-wise contextual variances* based on Mahalanobis distances. Note that such approach assumes feature independence though.

**OD in Streaming Data** The trend in OD of recent years is to build models to process streaming data with constant memory complexity. In addition to distance-based methods like SDOstream [5], OD in streams is grounded on tree-based methods [2], half-space chains [29], histograms [16], randomized hashing [15], or simply based on nearest neighbors in a sliding window [1]. A thorough comparison of these methods can be consulted in [30].

When compared to SDOoop, beyond the core approach for calculating point outlieriness, the main difference is that earlier algorithms establish a temporally evolving model (or a set of reference data points) that is deemed stationary at time scales smaller than a pre-fixed time parameter. Hence, contextual

TABLE III  
CHARACTERISTICS OF OD ALGORITHMS FOR EVOLVING DATA STREAMS.

	SW- $k$ NN	SW-LOF	LODA [16]	xStream [32]	RS-Hash [15]	RRCT [2]	SDOstream [5]	SDOoop
Fixed time complex.	~	×	✓	✓	✓	~	✓	✓
Fixed space complex.	×	×	×	✓	✓	×	✓	✓
Interpretability	✓	~	×	×	×	×	✓	✓
Detect temp. patterns	×	×	×	×	×	×	×	✓
Detect context. anom.	×	×	×	×	×	×	×	✓

outliers, i.e. data points that occur at an atypical time (out-of-phase), are wrongly classified as normal inliers. Another important property of OD methods is interpretability of returned outlier scores. In fact, many modern techniques like forest-based methods require space transformations that inevitably sacrifice interpretability. In SDOoop, outlier scores can be directly interpreted as distance-to-observers, i.e., distance-to-normality. The model is small enough for manual inspection, allowing the analyst to draw conclusions about the data mass based on main model patterns. On the other hand, obtained models are also suitable for stand-alone systems or frameworks where knowledge must be integrated with decision-making modules or other types of knowledge.

Table III shows a summarized comparison of recent algorithms for evolving stream OD with regard to key properties. SW- $k$ NN and SW-LOF denote sliding-window implementations of the popular  $k$ NN [9] and LOF [31] algorithms.

**Periodic Pattern Mining.** The detection of periodicities in sequences has also been investigated in the context of periodic pattern mining [33]. Periodic pattern mining can be applied to spatiotemporal data [34] to detect periodicities in the movement of objects. In contrast, SDOoop is able to detect periodicities of arbitrary clusters even if the corresponding data points are mixed up with data points from other clusters with different temporal patterns or no patterns at all. To the best of our knowledge, this problem has not been explored before.

## VI. CONCLUSION

Big data frequently arrives in data streams and requires online processing and analysis. We proposed SDOoop, a method for knowledge discovery in data streams that is able to capture coexisting periodicities regardless of data geometries. Our method performs a single pass through the data and builds a fixed-size model consisting of representative point locations along with their temporal behavior in Fourier space. We showed equal or superior performances compared to state-of-the-art algorithms when testing OD in established evaluation datasets. Moreover, we showed that our method can be an important tool for understanding and visualizing the spatiotemporal behavior of steadily arriving real-world data, particularly in network security and critical infrastructures communications.

## REFERENCES

- [1] F. Angiulli and F. Fassetto, "Detecting distance-based outliers in streams of data," in *Proc. of the Sixteenth ACM Cong. on Information and Knowledge Management*. New York, NY, USA: Assoc. for Comp. Mach., 11 2007, pp. 811–820.
- [2] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Int. Conf. on Mach. Learn.* PMLR, 2016, pp. 2712–2721.
- [3] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.
- [4] F. Iglesias Vázquez, T. Zseby, and A. Zimek, "Outlier detection based on low density models," in *ICDMW*, 2018, pp. 970–979.
- [5] A. Hartl, F. Iglesias, and T. Zseby, "SDOstream: Low-density models for streaming outlier detection," in *ESANN 2020 proceedings*, 2020, pp. 661–666.
- [6] F. Iglesias, T. Zseby, A. Hartl, and A. Zimek, "Sdoclust: Clustering with sparse data observers," in *Similarity Search and Applications*, O. Pedreira and V. Estivill-Castro, Eds. Cham: Springer Nature Switzerland, 2023, pp. 185–199.
- [7] F. Iglesias, T. Zseby, D. Ferreira, and A. Zimek, "Mdcgen: Multidimensional dataset generator for clustering," *Journal of Classification*, vol. 36, no. 3, pp. 599–618, 2019.
- [8] P. Hall, B. U. Park, R. J. Samworth *et al.*, "Choice of neighbor order in nearest-neighbor classification," *The Annals of Statistics*, vol. 36, no. 5, pp. 2135–2152, 2008.
- [9] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, no. 2, p. 427–438, may 2000.
- [10] "Kdd cup 1999 data," <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>, accessed: 2021-03-04.
- [11] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *DAMI*, vol. 30, no. 4, pp. 891–927, 2016.
- [12] R. A. Angryk, P. C. Martens, B. Aydin, D. Kempton, S. S. Mahajan, S. Basodi, A. Ahmadzadeh, X. Cai, S. Filali Boubrahimi, S. M. Hamdi, M. A. Schuh, and M. K. Georgoulis, "Multivariate time series dataset for space weather data analytics," *Scientific Data*, vol. 7, no. 227, 2020.
- [13] A. Ahmadzadeh and B. Aydin, "Multivariate Timeseries Feature Extraction on SWAN Data Benchmark (SWAN\_Features)," 2020, GSU Data Mining Lab, Bitbucket repository. [Online]. Available: [https://bitbucket.org/gsdmlab/swan\\_features](https://bitbucket.org/gsdmlab/swan_features)
- [14] A. Hartl, F. Iglesias, and T. Zseby, "dSalmon: High-speed anomaly detection for evolving multivariate data streams," in *Performance Evaluation Methodologies and Tools (VALUETOOLS 2023)*. Springer, 2024, pp. 153–169, <https://github.com/CN-TU/dSalmon>.
- [15] S. Sathe and C. C. Aggarwal, "Subspace outlier detection in linear time with randomized hashing," in *2016 IEEE 16th Int. Conference on Data Mining (ICDM)*. New York, NY, USA: IEEE, 2016, pp. 459–468.
- [16] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.
- [17] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.
- [18] CAIDA, "The UCSD network telescope "patch tuesday" dataset," [http://www.caida.org/data/passive/telescope-patch-tuesday\\_dataset.xml](http://www.caida.org/data/passive/telescope-patch-tuesday_dataset.xml), acc.: 2021-03-09.
- [19] F. Iglesias and T. Zseby, "Pattern discovery in internet background radiation," *IEEE Trans. on Big Data*, vol. 5, no. 4, pp. 467–480, 2017.
- [20] J. D. Hamilton, *Time series analysis*. Princ., NJ, USA: Princeton Univ. press, 2020.
- [21] G. Mahalakshmi, S. Sridevi, and S. Rajaram, "A survey on forecasting of time series data," in *Int. Conf. on Comp. Tech. and Int. Data Eng.*, 2016, pp. 1–8.
- [22] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.
- [23] T. W. Liao, "Clustering of time series data—a survey," *Pat. Rec.*, vol. 38, no. 11, pp. 1857–1874, 2005.
- [24] J. Read, R. A. Rios, T. Nogueira, and R. F. de Mello, "Data streams are time series: Challenging assumptions," in *Intelligent Systems*, R. Cerri and R. C. Prati, Eds. Cham: Springer International Publishing, 2020, pp. 529–543.
- [25] K. Shaikat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed, "A review of time-series anomaly detection techniques: A step to future perspectives," in *Adv. in Inf. & Com.*, K. Arai, Ed. Springer, 2021, pp. 865–877.
- [26] K. Golmohammadi and O. R. Zaiane, "Time series contextual anomaly detection for detecting market manipulation in stock market," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–10.
- [27] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, "Sand: Streaming subsequence anomaly detection," *Proc. VLDB Endow.*, vol. 14, no. 10, p. 1717–1729, jun 2021.
- [28] K. Pasini, M. Khoudjia, A. Samé, M. Trépanier, and L. Oukhellou, "Contextual anomaly detection on time series: A case study of metro ridership analysis," *Neural Comput. Appl.*, vol. 34, no. 2, p. 1483–1507, jan 2022.
- [29] S. Tan, K. Ting, and F. T. Liu, "Fast anomaly detection for streaming data," in *22nd Int. Joint Conf. on Artificial Intelligence*, 2011, pp. 1511–1516.
- [30] F. Iglesias, A. Hartl, T. Zseby, and A. Zimek, "Anomaly detection in streaming data: A comparison and evaluation study," *ESWA*, vol. 233, p. 120994, 2023.
- [31] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, May 2000.
- [32] E. Manzoor, H. Lamba, and L. Akoglu, "xStream: Outlier detection in feature-evolving data streams," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Know. Disc. & Data Mining*. New York, NY, USA: Assoc. for Comp. Mach, 2018, p. 1963–1972.
- [33] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recog.*, vol. 1, no. 1, pp. 54–77, 2017.
- [34] D. Zhang, K. Lee, and I. Lee, "Periodic Pattern Mining for Spatio-Temporal Trajectories: A Survey," in *2015 10th Int. Conf. on Int. Systems and Knowledge Engin. (ISKE)*. New York, NY, USA: IEEE, Nov. 2015, pp. 306–313.

## APPENDIX A PROOF OF LEMMA 1

Let  $i_o \in \mathbb{N}$  denote the index of a data point, for which  $\omega$  is contained in  $\mathcal{N}$  and  $i_c$  is the index of the currently processed data point, i.e.,  $i_o < i_c$ . Then, the contribution of  $i_o$  to  $P_{\omega,n}$  according to line 4 of Algorithm 1 has been multiplied by  $\prod_{i=i_o+1}^{i_c} (\exp(-T^{-1} + jn2\pi/T_0))^{t_i - t_{i-1}} = (\exp(-T^{-1} + jn2\pi/T_0))^{t_{i_c} - t_{i_o}}$ . Summing over all points that have arrived in  $\omega$ 's neighborhood, we can write

$$E\{P_{\omega,n}\} = \int_{-\infty}^t g(\tau) \left( \exp(-T^{-1} + jn2\pi/T_0) \right)^{t-\tau} d\tau.$$

Splitting the integral into intervals of length  $T_0$ , we obtain

$$\begin{aligned} E\{P_{\omega,n}\} &= \sum_{l=0}^{\infty} \int_{t-T_0}^t g(\tau-lT_0) (\exp(-T^{-1} + jn2\pi/T_0))^{t-\tau-lT_0} d\tau \\ &= \left( \sum_{l=0}^{\infty} \exp(-T^{-1}lT_0) \right) \int_{t-T_0}^t g(\tau) (\exp(-T^{-1} + jn2\pi/T_0))^{t-\tau} d\tau \end{aligned}$$

due to  $T_0$ -periodicity of  $g(t)$  and  $\exp(jn2\pi) = 1$ . Abbreviating the constant factor and substituting  $\tau' = \tau - t$ , we obtain

$$\begin{aligned} E\{P_{\omega,n}\} &= c \int_{-T_0}^0 g(\tau' - t) (\exp(-T^{-1} + jn2\pi/T_0))^{-\tau'} d\tau' \\ &\stackrel{T \gg T_0}{\approx} c \int_{-T_0}^0 g(\tau' - t) \exp(-jn2\pi\tau'/T_0) d\tau'. \quad \blacksquare \end{aligned}$$



APPENDIX B  
PROOF OF THEOREM 2

Taking line 6 in Algorithm 1 as starting point, the probability of selecting a newly seen point as observer is  $\min\left(1, \frac{k^2}{Tx} \frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}} \frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}\right)$ . Since we target specifically data streams with high rates of arriving data points, we can safely assume this probability to be small. Hence,  $\Pr\left\{1 < \frac{k^2}{Tx} \frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}} \frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}\right\}$  is negligible and we can write for the average probability of sampling a new point as observer  $P_s \approx E\left\{\frac{k^2}{Tx} \frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}} \frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}\right\}$ . Under the same assumption, we observe that the term  $\frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}$  depends on the current time, but, since points belonging to different neighborhoods arrive in an interleaved manner, does not depend on a point's neighborhood. Since  $\frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}}$  does not depend on time, we can split the term to  $P_s \approx E\left\{\frac{k^2}{Tx} \frac{\sum_{\omega \in \mathcal{N}} P_{\omega,0}}{\sum_{\omega \in \Omega} P_{\omega,0}}\right\} E\left\{\frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}\right\}$  due to stochastic independence of both terms.  $\sum_{\omega \in \mathcal{N}} P_{\omega,0}/x$  expresses the average observation count in the current neighborhood. The algorithm implements several mechanisms to make the observer density agree with the time-averaged point density, rendering the time-averaged local average observation count  $E\left\{\sum_{\omega \in \mathcal{N}} P_{\omega,0}/x\right\}$  equal to the total average observation count of all observers  $\sum_{\omega \in \Omega} P_{\omega,0}/k$ , hence  $P_s \approx \frac{k}{T} E\left\{\frac{t_i - t_{i_{\text{LAO}}}}{i - i_{\text{LAO}}}\right\} = \frac{k}{T} \overline{\text{IAT}}$ , where the average inter-arrival time of two data points is termed  $\overline{\text{IAT}}$ . During a time period of  $T$ ,  $T/\overline{\text{IAT}}$  data points arrive, yielding an average number of sampled points of  $P_s T / \overline{\text{IAT}} = k$ . ■