Ultrafast Local Outlier Detection from a Data Stream with Stationary Region Skipping

Susik Yoon
KAIST
Daejeon, Korea
susikyoon@kaist.ac.kr

Jae-Gil Lee*
KAIST
Daejeon, Korea
jaegil@kaist.ac.kr

Byung Suk Lee
University of Vermont
Burlington, Vermont
bslee@uvm.edu

ABSTRACT

Real-time outlier detection from a data stream is an increasingly important problem, especially as sensor-generated data streams abound in many applications owing to the prevalence of IoT and emergence of digital twins. Several density-based approaches have been proposed to address this problem, but arguably none of them is fast enough to meet the performance demand of real applications. This paper is founded upon a novel observation that, in many regions of the data space, data distributions hardly change across window slides. We propose a new algorithm, abbr. **STARE**, which identifies local regions in which data distributions hardly change and then skips updating the densities in those regions—a notion called *stationary region skipping*. Two techniques, *data distribution approximation* and *cumulative net-change-based skip*, are employed to efficiently and effectively implement the notion. Extensive experiments using synthetic and real data streams as well as a case study show that STARE is several orders of magnitude faster than the existing algorithms while achieving comparable or higher accuracy.

CCS CONCEPTS

• **Computing methodologies → Anomaly detection**; • **Information systems → Data stream mining**.

KEYWORDS

Outlier detection; anomaly detection; data stream; local outlier; kernel density estimation

ACM Reference Format:
Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2020. Ultrafast Local Outlier Detection from a Data Stream with Stationary Region Skipping. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403171

1 INTRODUCTION

1.1 Background and Motivation

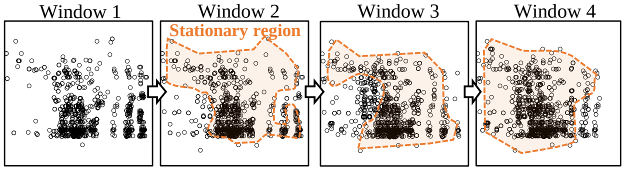


Figure 1: Example data distributions of a two-dimensional subspace of the HTTP data set [1], where a large proportion of the data space (inside the dashed line) has nearly the same data distribution each time a window slides.

Table 1: The ratio of the number of data points with (near-)stationary densities to the number of all data points, averaged across windows sliding over a data stream, for different data sets. (The data sets are described in Section 5.1.)

Data stream	A1	A2	HTTP	DLR	ECG	FDC	**Average**
100% stationary	0.58	0.65	0.88	0.72	0.62	0.61	**0.68**
≥ 99% stationary	0.65	0.76	0.99	0.91	0.97	0.96	**0.87**

Real-time detection of outliers from a data stream is an important problem rapidly gaining increasing attention, especially with the recent prevalence of IoT and emergence of digital twins [21]. Its solution is widely needed in many applications, such as intrusion detection from a network traffic stream [1], fall detection from a wearable sensor stream [12], and abnormal heartbeat detection from an ECG stream [11].

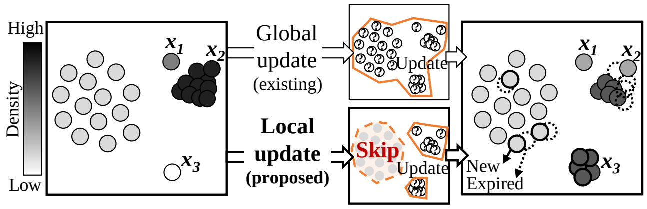
Since a data stream is inherently unbounded, it is a common practice in continuous outlier detection to use a *sliding window* to consider only the most recent data points [8, 25]. As a window slides, new data points are added to the window, and old data points expire from the window. Then, any data points significantly different from others in that window are labeled as outliers.

This work takes advantage of an important characteristic in real data streams that can potentially save much work for outlier detection: data points are skewed into a number of local regions in the data space and data distributions are nearly stationary (i.e., change insignificantly) in those regions for a certain duration of time. This observation is more pronounced in a *windowed* stream processing because a window typically slides at a fraction of the window size and, therefore, expired or new data points from a window slide have limited effect on the data distribution in the entire window (see Figure 1).

With data points typically skewed to local regions in the data space, outliers are likely to be identifiable only in the local region that they belong to, called *local outliers.* A *density-based* approach is able to find such local outliers effectively by labeling a data point as an outlier if it has a relatively lower density than its neighbors, where the density at a data point is determined by the data distribution in its local region [2, 9, 17].

*Jae-Gil Lee is the corresponding author.

KDD '20, August 23–27, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403171



(a) Previous window. (b) Density update. (c) Current window.

Figure 2: Two density-update approaches.

Thus, the densities at many data points tend to be *stationary* in a windowed stream processing, as confirmed in Table 1. In each sliding window, the densities are completely stationary for 68% of data points and nearly stationary (within 1% change across window slides) for 87% of data points, when averaged over all data sets. Despite this obvious opportunity to save work in density-based outlier detection, none of the existing algorithms recognizes the "stationarity" of the densities.

1.2 Main Ideas

Several density-based algorithms have been proposed to detect local outliers from a data stream [13–16], and they need to estimate the densities at data points. Oblivious to the *density stationarity*, the existing algorithms update the densities at *all* data points in a window repeatedly every time the window slides; this density estimation in the entire data space incurs quadratic time complexity [2, 6], which impairs timely outlier detection because of excessive latency. The key idea we employ to resolve this problem is to skip density updates in local regions in which densities at data points have hardly changed, namely *stationary regions*.

Example 1.1. (Local Update) In Figure 2, there are two outliers, x_1 and x_3, in the previous window (Figure 2a) and, after the window slides, in the current window (Figure 2c) x_2 becomes a new outlier, as it now has a lower density than its nearest neighbors, and x_3 becomes an inlier, as it now has a similar density to its nearest neighbors' densities. Between the previous and current windows, the densities at data points change only in local regions on the right. However, the existing algorithms globally update the densities at all data points (Figure 2b upper). These excessive updates can be avoided with local updates (Figure 2b lower), which allow for skipping the stationary regions on the left and estimate the densities only for the remaining local regions. □

Implementing the notion of *stationary region skipping*, however, poses significant challenges. First, tracking where and how the densities at data points change significantly should be done without actually calculating their densities in the data space, as it is an expensive operation. Second, skipping stationary regions should not damage the outlier detection accuracy as a result.

In this paper, we propose a novel algorithm, abbreviated as **STARE** (local outlier detection by STAtionary REgion skipping), that addresses the challenges. To the best of our knowledge, STARE is the first that fully exploits the density stationarity of a data stream towards achieving fast and accurate density updates. Specifically, STARE uses kernel density estimation (KDE) to compute densities at data points while employing the following two techniques for stationary region skipping.

Table 2: Notations used in this paper.

Notation	Description
$\mathcal{X}^d$	a set of d-dimensional data points
θ_R	the size (diagonal length) of a grid cell
θ_K	the threshold on the number of neighbors
$\mathcal{K}_h$	a kernel function with a bandwidth h
kc	a kernel center
$\mathcal{X}^d(kc)$	a set of data points represented by kc
$\mathbb{G}$	a weight-distribution grid of kernel centers
$\mathcal{D}$	a local density function
$\mathcal{S}$	a local outlier score function

- *Data distribution approximation*: KDE requires a set of kernel centers used to determine the local densities at their neighboring data points by a certain kernel function. By virtue of KDE, STARE can track only the change of the distribution of kernel centers, which is an indicator of the data density changes. Notably, kernel centers are derived from a set of fixed small regions partitioning the data space. Their fixed possible positions make it very efficient to maintain and update them.
- *Cumulative net-change-based skip*: STARE is built upon a systematic skipping framework based on a quantification of the changes of kernel center distribution across sliding windows. STARE updates the density only in the regions where the cumulative net-change of kernel center distribution becomes significant. The bounds on the density error resulting from skipping density updates is theoretically analyzed to provide both *exact* and *approximate* skipping strategies accordingly.

1.3 Summary

Main contributions in this paper are as follows.

- A new observation on the *density stationarity* in a data stream and a novel concept of *stationary region skipping* to exploit it for expediting local outlier detection.
- A novel algorithm **STARE** that fully implements the concept of stationary region skipping by way of data distribution approximation and cumulative net-change-based skip techniques.
- Comprehensive experiments using one synthetic data set and five real data sets; the results showed that STARE detects local outliers 11 times faster than the state-of-the-art algorithm on average while achieving comparable or higher accuracy and robust performance for varying parameter values.

In the rest of the paper, Section 2 provides background knowledge, Section 3 reviews related work, Section 4 proposes the STARE algorithm, Section 5 presents the experiments, and Section 6 concludes the paper. Table 2 summarizes the notations used in the paper.

2 REVIEW: KDE-BASED LOCAL OUTLIERS

Kernel density estimation (KDE) (see Definition 2.1) is nonparametric estimation of the probability density function of a random variable from a set of d-dimensional sample data points [20].

Definition 2.1. (Kernel Density Estimation) Given a set of kernel centers $\mathcal{KC} = \{kc_1, \ldots, kc_m\}$ and a kernel function $\mathcal{K}_h$, the *density* at a data point $x \in \mathcal{X}^d$ is estimated as $\frac{1}{m}\sum_{i=1}^{m}\mathcal{K}_h(dist(x, kc_i))$, where $dist(x, kc_i)$ is the Euclidean distance between x and kc_i. □

Table 3: Existing algorithms for local outlier detection from a data stream.

Name	Base model	Main technique employed	Update scope
iLOF [14]	LOF	Incremental update	*All* new points
MiLOF [16]	LOF	Clustering-based summary	*All* new points
DILOF [13]	LOF	Sampling-based summary	*All* new points
KELOS [15]	KDE	Clustering-based pruning	*All* regions

The kernel centers are usually sampled from $\mathcal{X}^d$, and the density contribution of each kernel center kc_i to a target data point x is estimated by a kernel function. A Gaussian function $\mathcal{K}_h(u) = (h\sqrt{2\pi})^{-1}e^{-u^2/2h^2}$ is widely used as such a kernel function, where u is $dist(x, kc_i)$ and h is the bandwidth which determines the smoothness of the estimated density function.

To implement the concept of local outliers, we adopt the notion of *"local density"* [10, 15, 17] (see Definition 2.2), whose estimation is varied from the KDE as follows: (1) only the θ_K nearest kernel centers are used (instead of all kernel centers) to estimate the local density at each data point [22]; (2) density contributions from kernel centers are adjusted by their weights [7]; (3) the density at a multidimensional data point is calculated as the product of densities estimated in individual dimensions [18].

Definition 2.2. (Local Density Estimation) Given a set of weighted kernel centers $\mathcal{KC} = \{\langle kc_1, w_1\rangle, \ldots, \langle kc_m, w_m\rangle\}$, the *local density* $\mathcal{D}(x)$ of a data point $x \in \mathcal{X}^d$ is estimated as

$$\mathcal{D}(x) = \sum_{i=1}^{\theta_K} \frac{w_i}{\sum_{j=1}^{\theta_K} w_j} \prod_{l=1}^{d} \mathcal{K}_{h^l}(dist(x^l, kc_i^l)), \quad (1)$$

where $kc_1, kc_2, \ldots, kc_{\theta_K}$ are the θ_K nearest kernel centers of x; and h^l, x^l, and kc^l are the bandwidth, the value of x, and the value of kc, respectively, in the l-th dimension ($1 \le l \le d$); here, the bandwidth h^l is set to the average of distances to the θ_K nearest kernel centers in the l-th dimension [17, 19]. □

Then, the *local outlier score* of a data point is calculated based on how much the local density at the data point is different from the local densities at its θ_K nearest kernel centers [15, 17] (see Definition 2.3).

Definition 2.3. (Local Outlier Score) The *local outlier score* of a data point x is calculated as $\mathcal{S}(x) = (\mu - \mathcal{D}(x))/\sigma$, where μ and σ are the mean and standard deviation of the local densities at the θ_K nearest kernel centers of x. □

A local outlier score ranges from $-\infty$ to $+\infty$, where a higher score means lower density relative to the θ_K nearest neighbors and so more likely to be an outlier.

3 RELATED WORK

The existing algorithms can be categorized into two kinds depending on the model used to estimate the local density at a data point: local outlier factor (LOF)-based and kernel density estimation (KDE)-based (see Table 3).

3.1 LOF-based Algorithms

The local outlier factor (LOF) [2] is a well-known model for measuring local density at a data point based on the average distance to its neighbors. Specifically, it introduces the notion of *reachability distance* from a data point x to another data point y as the larger between the distance between x and its k-th nearest neighbor and the distance between x and y, and then defines *local reachability density* at x as the inverse of the average reachability distance from x to its k nearest neighbors. Then, a data point with a lower local reachability density than its neighbors has a higher LOF score, and so more likely to be a local outlier. LOF's running time complexity is quadratic with the number of data points, which makes it challenging to be applied in a streaming environment.

There have been efforts made to address the efficiency concern of LOF [13, 14, 16], when using a landmark window where data points never expire. Instead of updating LOF scores of all data points as the landmark window expands, iLOF [14] incrementally updates the LOF scores of only those data points that are direct or indirect neighbors of new data points. While more efficient than LOF, iLOF still suffers from memory shortage as well as excessively long running time as the number of data points keeps increasing. MiLOF [16] and DILOF [13] address these problems by compressing old data points with clustering and sampling, respectively, to reduce the overhead of local density updates while closely approximating the data distribution. However, they only focus on efficiently computing the LOF scores of new data points rather than all data points in a window. Besides, they do not consider data expiration.

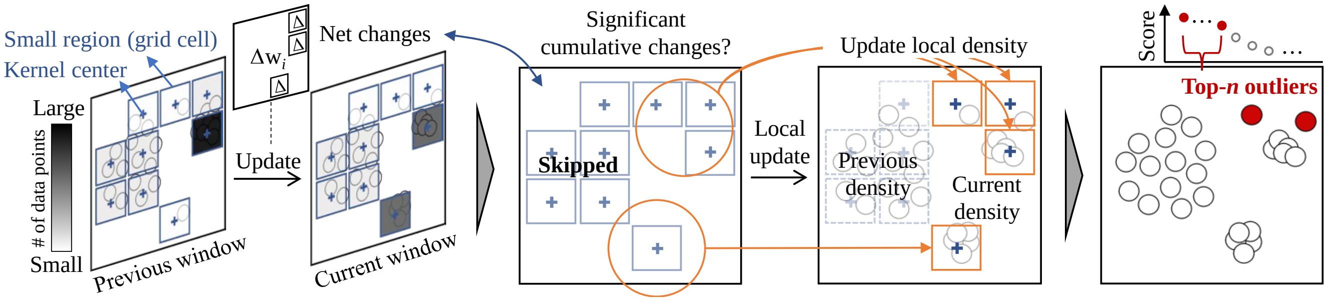
3.2 KDE-based Algorithms

The kernel density estimation (KDE) for local outlier detection was first proposed by Latecki et al. [10], and was improved further in the recent work [6, 15, 17]. KDEOS [17] was the first work detecting local outliers using KDE considering only k nearest neighbor kernel centers. However, it is not meant for a data stream environment, and, moreover, its quadratic running time complexity with the number of kernel centers can be problematic, as every data point is considered a kernel center.

Recently, KELOS [15] was proposed to detect local outliers from a data stream. Similar to the clustering-based outlier pruning in LOF-based approaches [9], KELOS clusters data points by micro-clustering and uses cluster centroids as kernel centers for KDE. By bounding the local densities and local outlier scores of data points in each cluster, KELOS first prunes some data points that are unlikely to be outliers at the cluster level and then inspects the remaining data points to find top-n local outliers. This clustering-based outlier pruning saves considerable computation cost. However, the update of clusters still requires quadratic computation time; besides, their bounds on local densities and local outlier scores always need to be recomputed in every sliding window even if there has been little or no change from the previous window.

4 THE ALGORITHM "STARE"

Problem setting: The generic problem underlying the work presented in this paper is *density-based top-n local outlier detection from a windowed data stream*, whereby n local outliers with the highest local outlier scores are detected from a window each time the window slides. In this paper, the size of a window and a slide is measured as the number of data points in them, as done in other related studies [15, 23]. An *expired slide* and a *new slide* refer to the data points in them.



(a) Data distribution update.

(b) Stationary region skip.

(c) Top-n outlier detection.

Figure 3: Overall procedure of STARE.

4.1 Overview

To support fast detection of local outliers, STARE approximates the data space into a set of fixed small regions. These regions are used to derive kernel centers as well as to approximate data points, as will be detailed in Section 4.2. STARE works in the following three phases in every sliding window, as outlined in Algorithm 1 and illustrated in Figure 3.

Phase 1. Data distribution update (Section 4.3): STARE keeps track of the change by counting the number of data points in each small region and using the count as the weight of the kernel center derived from the region. A small region is implemented as a grid cell partitioning the data space, and we call the resulting grid a *weight distribution grid*. The weights are updated efficiently by reflecting the *net change* of the count for each small region between the expired slide and the new slide to the weight distribution grid of the previous window (see Figure 3a).

Phase 2. Stationary region skip (Section 4.4): STARE examines the changes of weights in the weight distribution grid between consecutive sliding windows and identifies stationary regions, where the cumulative changes of the nearby kernel centers are not significant. Then, it skips updating the local densities at data points in those stationary regions (see Figure 3b) and instead reuses the local densities estimated in previous windows.

Phase 3. Top-*n* outlier detection (Section 4.5): STARE chooses the top-n local outliers based on the local outlier scores of data points (see Figure 3c), while efficiently pruning small regions and data points that have low scores.

4.2 Density Approximation

The existing algorithms [10, 15, 17] use KDE mainly to estimate the density at data points by using kernel centers, which are derived from data points and, therefore, vary across windows. STARE also uses KDE for the same purpose and, additionally, to detect changes of local densities between consecutive windows; it is facilitated by fixing kernel centers to the center of a small region (independently of the data points) and associating the number of data points in a small region as the data density at the kernel center of the region (see Definition 4.1).

Definition 4.1. (Kernel Center) Consider the data space of $\mathcal{X}^d$ partitioned into d-dimensional grid cells whose diagonal length is θ_R.[1] Then, a *kernel center* is defined as a pair $\langle kc, w\rangle$ for each non-empty grid cell such that kc is the center coordinate of the grid cell and w is the number of data points in the grid cell, used as the weight of the kernel center. □

Algorithm 1 Overall Procedure of STARE

```
Input: a data stream $DS$, the number $n$ of outliers to find
Output: a set $O$ of top-$n$ outliers for each window
for each window $\mathcal{W}$ sliding on $DS$ do
    $\mathcal{S}_{exp} \leftarrow$ the expired slide; $\mathcal{S}_{new} \leftarrow$ the new slide;
    $\mathbb{G}^{prev} \leftarrow$ the previous weight distribution grid;
    /* 1. Data Distribution Update */
    $(\mathbb{G}^{curr}, \Delta\mathbb{G}) \leftarrow$ UpdateDistribution($\mathbb{G}^{prev}, \mathcal{S}_{exp}, \mathcal{S}_{new}$);
    /* 2. Stationary Region skip */
    $\mathcal{W} \leftarrow$ UpdateChangedRegion($\mathcal{W}, \mathbb{G}^{curr}, \Delta\mathbb{G}$);
    /* 3. Top-$n$ Outliers Detection */
    $O \leftarrow$ DetectLocalOutliers($\mathcal{W}, n$);
    return $O$;
end for
```

A kernel center kc acts as the representative of a grid cell. Let $\mathcal{X}^d(kc)$ denote the set of data points in a d-dimensional grid cell represented by kc. Then, the set of kernel centers in the data space, $\mathcal{KC} = \{\langle kc_i, w_i\rangle \mid i = 1, \ldots, m\}$, is managed in a grid-based structure $\mathbb{G}$, called a *weight distribution grid*.

The local density at $x \in \mathcal{X}^d(kc)$ and the local outlier score of x are estimated by substituting the θ_K nearest kernel centers of x (in Definition 2.2 and Definition 2.3) with those of kc that contains x. Then, the local densities at all data points within the grid cell are bounded as stated in Theorem 4.2.

Theorem 4.2. (Bounds on Local Density) *Given a grid cell c and the kernel center kc representing c, the local density $\mathcal{D}(x)$ at every $x \in \mathcal{X}^d(kc)$ is bounded as $\mathcal{D}_{low}(c) \le \mathcal{D}(x) \le \mathcal{D}_{up}(c)$, where*

$$\mathcal{D}_{low}(c) = \sum_{i=1}^{\theta_K} \frac{w_i}{\sum_{j=1}^{\theta_K} w_j} \prod_{l=1}^{d} \mathcal{K}_{h^l}\Big(dist(kc^l, kc_i^l) + \frac{\theta_R}{2\sqrt{d}}\Big) \text{ and}$$
$$\mathcal{D}_{up}(c) = \sum_{i=1}^{\theta_K} \frac{w_i}{\sum_{j=1}^{\theta_K} w_j} \prod_{l=1}^{d} \mathcal{K}_{h^l}\Big(dist(kc^l, kc_i^l) - \frac{\theta_R}{2\sqrt{d}}\Big); \quad (2)$$

kc_i is the i-th of the θ_K nearest kernel centers of kc.

Proof. The distance between kc and $x \in \mathcal{X}^d(kc)$ on the l-th dimension is at most $\theta_R/2\sqrt{d}$. Consider another kernel center kc_o. By the triangle inequality, $dist(kc^l, kc_o^l) + \theta_R/2\sqrt{d} \ge dist(x^l, kc_o^l)$. Since the estimated local density is smaller when the distance to

[1] See Appendix C.1 for the guideline of determining θ_R.

Algorithm 2 UpdateDistribution

```
INPUT: $\mathbb{G}^{prev}$, $\mathcal{S}_{exp}$, $\mathcal{S}_{new}$
OUTPUT: $\mathbb{G}^{curr}$, $\Delta\mathbb{G}$ /* weight and net-weight distribution grids */
$\mathbb{G}_{exp} \leftarrow$ add $x \in \mathcal{S}_{exp}$ to the cell containing the coordinate of $x$;
$\mathbb{G}_{new} \leftarrow$ add $x \in \mathcal{S}_{new}$ to the cell containing the coordinate of $x$;
$\Delta\mathbb{G} \leftarrow \mathbb{G}_{new} - \mathbb{G}_{exp}$; /* net changes of $kc$ weights per cell */
$\mathbb{G}^{curr} \leftarrow \mathbb{G}^{prev} + \Delta\mathbb{G}$;
return $\mathbb{G}^{curr}$ and $\Delta\mathbb{G}$;
```

kc_o is longer, the longest distance to kc_o gives the lowest local density. Likewise, the shortest distance to kc_o, $dist(kc^l, kc_o^l) - \theta_R/2\sqrt{d}$, gives the highest local density. □

Based on these bounds on the local density within a grid cell, the bounds on the local outlier score is given by Corollary 4.3.

COROLLARY 4.3. (BOUNDS ON LOCAL OUTLIER SCORE) *Given a grid cell c and the kernel center kc representing c, the local outlier scores $\mathcal{S}(x)$ of every $x \in \mathcal{X}^d(kc)$ is bounded as*

$$\mathcal{S}_{low}(c) = \frac{\mu - \mathcal{D}_{up}(c)}{\sigma} \leq \mathcal{S}(x) \leq \mathcal{S}_{up}(c) = \frac{\mu - \mathcal{D}_{low}(c)}{\sigma}, \quad (3)$$

where $\mathcal{D}_{low}(c)$ and $\mathcal{D}_{up}(c)$ are the local density bounds in Theorem 4.2, and μ and σ are the average and standard deviation, respectively, of the local densities at the θ_K nearest kernel centers of kc.

PROOF. Straightforward from Definition 2.3. □

4.3 Phase 1: Data Distribution Update

Every time a window slides, old data points expire and new data points enter the window; thus, the data distribution of the window, managed in the weight distribution grid, should be updated accordingly. STARE efficiently performs this update by managing a net-weight distribution grid, denoted as $\Delta\mathbb{G}$, and calculating the net changes in each grid cell between the expired and new slides, as shown in Algorithm 2. Each data point in a slide is indexed into a grid cell in constant time, and the net changes of grid cells are calculated in linear time with the number of non-empty grid cells, as it is implemented by a simple matrix addition.

4.4 Phase 2: Stationary Region Skip

Algorithm 3 outlines the steps of this phase. The net-weight distribution grid $\Delta\mathbb{G}$ from Phase 1 contains the essential information regarding the data distribution changes. STARE can selectively update the local densities at data points in a window by using $\Delta\mathbb{G}$ only, as justified by Theorem 4.4.

THEOREM 4.4. ($\Delta\mathbb{G}$-BASED DENSITY UPDATE) *Given* non-zero *weight net-changes in $\Delta\mathbb{G}$, let $\mathcal{X}_{affected}$ be a set of data points "close" to kernel centers in $\Delta\mathbb{G}$, specifically, $\mathcal{X}_{affected} = \{x \mid \exists(\langle kc_i, \Delta w_i \rangle \in \Delta\mathbb{G}) \wedge (dist(x, kc_i) \leq dist(x, kc_{\theta_K}))\}$, where kc_{θ_K} is the θ_k-th nearest kernel center in the previous window. Then, all data points at which local densities have changed must be in $\mathcal{X}_{affected}$.*

PROOF. By Definition 2.2, if the θ_K nearest kernel centers of a data point x and their weights remain the same, then the local density at x does not change. Therefore, since $\mathcal{X}_{affected}$ contains all data points close to the θ_K nearest kernel centers whose weights have changed (i.e., non-zero net-changes), it contains all data points at which the local densities have changed. □

Algorithm 3 UpdateChangedRegion

```
INPUT: $\mathcal{W}$, $\mathbb{G}^{curr}$, $\Delta\mathbb{G}$, $\gamma$ /* error allowance threshold, 0 if not specified */
OUTPUT: $\mathcal{W}$ /* updated window */
/* Stationary region skip in Definition 4.6 */
for each $kc$ in $\mathbb{G}^{curr}$ do
    $E(kc) \leftarrow$ the cumulative error of $kc$ in Definition 4.5;
    if $E(kc) > \gamma$ then
        $(\mathcal{D}_{low}(c), \mathcal{D}_{up}(c)) \leftarrow$ update density bounds of a grid cell $c$
            represented by $kc$;
        $\mathcal{D}(x) \leftarrow$ update densities at all data points $x \in \mathcal{X}^d(kc)$;
    end if
end for
return $\mathcal{W}$;
```

Thus, based on Theorem 4.4, STARE updates local densities only at data points in $\mathcal{X}_{affected}$ to keep the densities up to date in the current window. Further, STARE relaxes the density update to skip it at data points until the resulting error accumulated over sliding windows reaches a certain threshold. This strategy allows STARE to exploit the *near-* as well as exact stationarity of densities.

Calculating the local density errors directly at data points is computationally expensive, and therefore STARE instead uses errors on the weights of kernel centers near the data points in the same grid cell. In the current implementation, it quantifies the change in the weight distribution of the kernel centers within the distance to the θ_K-th nearest kernel centers of a data point from the last update (see Definition 4.5).

Definition 4.5. (CUMULATIVE ERROR) Let $t(= 1, \ldots, t_c)$ be the index of a sliding window and $\Delta\mathbb{G}_t$ be the $\Delta\mathbb{G}$ between the windows $t-1$ and t. Then, the *cumulative error* $E(x; t_c, t_l)$ on the local density at a data point x in the current window t_c since the last density update in the window t_l is calculated as

$$E(x; t_c, t_l) = \sum_{t=t_l,\ldots,t_c} \frac{\sum_{\Delta w_j \in \Delta\mathcal{W}_t(x;t_l)} |\Delta w_j|}{\sum_{w_i \in \mathcal{KC}(x;t_l)} w_i}, \quad (4)$$

where $\mathcal{KC}(x; t_l)$ is the set of θ_K nearest kernel centers of x in the window t_l and $\Delta\mathcal{W}_t(x; t_l) = \{\Delta w_i \mid (\langle kc_i, \Delta w_i \rangle \in \Delta\mathbb{G}_t) \wedge (dist(x, kc_i) \leq dist(x, kc_{\theta_K}; t_l))\}$, where $kc_{\theta_K}; t_l$ is the θ_k-th nearest kernel center in the window t_l. $\Delta\mathcal{W}_t(x; t_l)$ contains weight net-changes of the kernel centers affecting the local density at x. □

Definition 4.6 gives a formal specification of *stationary region skipping*. The upper bound on the consequential local density error is given in Theorem 4.7.

Definition 4.6. (STATIONARY REGION SKIP) Given a set $\mathcal{X}^d$ of data points in the current window and an error allowance threshold γ, the update of local density at a data point $x \in \mathcal{X}^d(kc)$ is skipped if $E(kc) \leq \gamma$, where kc is a kernel center containing x. □

THEOREM 4.7. (UPPER BOUND ON DENSITY ERROR) *Applying stationary region skipping with an error allowance threshold γ to a data point $x \in \mathcal{X}^d$ gives the upper bound on the density estimation error,*

$$|\Delta\mathcal{D}(x)| = |\mathcal{D}_{curr}(x) - \mathcal{D}_{last}(x)| \leq |\frac{\gamma(\mathcal{K}_{\tilde{h}}(0)^d - \mathcal{D}_{last}(x))}{1 + \gamma}|, \quad (5)$$

where $\mathcal{D}_{last}(x)$ is the last updated local density, $\mathcal{K}_h$ is the corresponding kernel function, and $\tilde{h} = min(h^1, \ldots, h^d)$.

Algorithm 4 DetectLocalOutliers

```
INPUT: $\mathcal{W}$ /* current window */, $n$ /* number of outliers to find */
OUTPUT: a set $O$ of top-$n$ outliers
$\mathcal{S}_{low}(c), \mathcal{S}_{up}(c) \leftarrow$ update score bounds of all grid cells $c$ in $\mathcal{W}$;
$\mathcal{S}(x) \leftarrow$ update scores of all data points $x$ in $\mathcal{W}$;
/* CELL-LEVEL DETECTION */
Let $C$ be the set of grid cells in $\mathcal{W}$;
$C_{cand} \leftarrow$ find candidate grid cells from $C$ according to Definition 4.8;
/* POINT-LEVEL DETECTION */
Let $X_{cand}$ be the set of data points in the grid cells in $C_{cand}$;
$O \leftarrow$ pick top-$n$ local outliers from $X_{cand}$;
return $O$;
```

PROOF. $\Delta\mathcal{D}(x)$ is determined by the weight net-changes of nearest kernel centers of x and the density estimation by a kernel function. The former can be approximated by $E(x)$ in Eq. (4), and the latter can be bounded by using the monotonicity property of a kernel function. See Appendix B.1 for the detailed proof. □

γ is a significant parameter influencing the performance and accuracy of STARE. $\gamma = 0$ leads to skipping density updates only where there is absolutely no density change at all, thereby incurring no density error, and increasing γ leads to skipping updates in more regions at the expense of higher local density errors. The optimal γ value balancing the trade-off would be application-dependent. An empirical study using real data streams (see Section 5.3) showed that $\gamma = 0.1$ was optimal, minimizing the processing time while guaranteeing near the baseline accuracy.

4.5 Phase 3: Outlier Detection

As outlined in Algorithm 4, STARE takes advantage of the bounds on the local outlier scores of data points in a grid cell (see Corollary 4.3) to perform outlier detection at two levels: *cell-level* and *point-level.* That is, it first identifies at the candidate grid cells (see Definition 4.8) that are guaranteed to contain the top-n outliers and then retrieves the top-n outliers from these grid cells.

Definition 4.8. (CANDIDATE GRID CELL) Given the number n of outliers and a set C of grid cells, a grid cell c becomes a *candidate* if it is a member of C_{cand} ($\subseteq C$) determined by

$$\begin{aligned} \operatorname*{argmin}_{C_{cand} \subseteq C} \{ |C_{cand}| \text{ such that } |C_{cand}| \geq n \land \\ \min(\{\mathcal{S}_{low}(c) \mid c \in C_{cand}\}) > \max(\{\mathcal{S}_{up}(c) \mid c \in C - C_{cand}\}) \}, \end{aligned} \tag{6}$$

where $\mathcal{S}_{low}(c)$ and $\mathcal{S}_{up}(c)$ are defined in Corollary 4.3. □

Since, by definition, the minimum outlier score of data points in C_{cand} is greater than the maximum outlier score of data points in $C - C_{cand}$, all top-n outliers must be in C_{cand} as long as there are at least n data points in C_{cand}.

4.6 Time Complexity of STARE

Theorem 4.9 shows the worst-case time complexity of STARE.

THEOREM 4.9. *Given the number W of data points in a window, the number N_G of non-empty grid cells in a window, the ratio r of the number of changed grid cells over N_G in Phase 2, and the ratio p of the number of data points in candidate grid cells over W in the point-level detection step of Phase 3, the time complexity of STARE is $O(W + rN_G^2)$.*

Table 4: Time complexity analysis of the four steps: Phase (1) data distribution update, (2) stationary region skip, (3-1) cell (cluster)-level detection, and (3-2) point-level detection.

Phase	LOF	KELOS	**STARE**
1	-	$O(WN_M)$	$O(W)$
2	-	-	$O(W + rN_G^2)$
3-1	-	$O(N_M^2)$	$O(N_G)$
3-2	$O(W^2)$	$O(pWN_M)$	$O(pW)$
Total	$O(W^2)$	$O(WN_M + N_M^2)$	$\mathbf{O(W + rN_G^2)}$

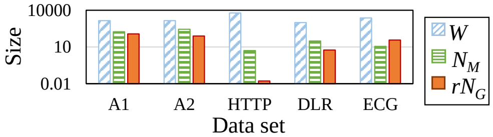


Figure 4: Comparison of the window size W, the number N_M of micro-clusters, and the number rN_G of changed grid cells in five real data sets.

PROOF. See Appendix B.2 for the proof. □

In Table 4, we compare the time complexity of STARE with those of LOF [14] and KELOS [15], which are considered the representatives of LOF-based algorithms and KDE-base algorithms, respectively. N_M is the number of micro-clusters in a window used by KELOS. STARE has the lowest complexity since $W > N_M > rN_G$ typically holds, as empirically shown in Figure 4.

5 EXPERIMENTS

We conducted thorough experiments to evaluate the performance of STARE. The results clearly demonstrate the superiority of STARE as follows.

- Orders-of-magnitude faster than the existing algorithms (Section 5.2 and Section 5.3).
- Robust to the variation of parameter values (Section 5.4 and Appendix C.2).
- Case-proved for a real industry application (Section 5.5).

5.1 Experiment Setup

Data sets: We used one synthetic data set (YahooA2 [24]) and four real data sets (YahooA1 [24], HTTP [1], DLR [5], and ECG [11]), all commonly cited in the literature. In addition, we used a real data set FDC (proprietary to an industrial sponsor) for a case study. YahooA1 and YahooA2 are provided by Yahoo! for time-series anomaly detection tasks; YahooA1 contains metrics for Yahoo! services with human-labeled outliers, and YahooA2 contains a synthetic data stream generated with varying trend, noise, and seasonality. HTTP contains network traffic including various network attacks (e.g., denial-of-service) labeled as outliers. DLR has been collected for an activity recognition system and contains measurements from the sensors attached to human subjects; it has 12 different activities (e.g., running, walking, falling, etc.) as labels, and "falling" is labeled as an outlier because fall detection is one of popular applications of outlier detection [12]. ECG contains features extracted from electrocardiogram signals, and the signals from abnormal heartbeats are labeled as outliers. FDC contains sensor readings collected from facilities in semiconductor factory facilities.

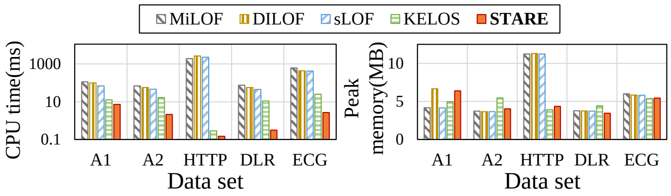


Figure 5: CPU time (ms) and peak memory (MB) results

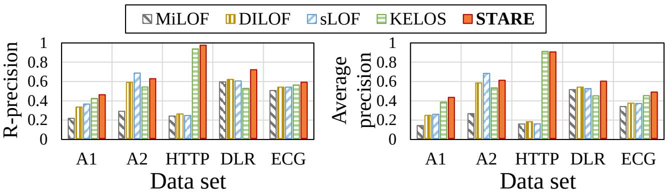


Figure 6: Accuracy results.

Table 5: Data sets and default parameter values.

Data set	Dim.	Size	Window size	Slide size	Outlier ratio
YahooA1 [24]	1	95K	1,415	71	1.7%
YahooA2 [24]	1	142K	1,421	71	0.3%
HTTP [1]	3	567K	6,000	300	0.3%
DLR [5]	9	23K	1,000	50	2.2%
ECG [11]	32	112K	2,237	117	16.3%
FDC	32	1.6K	534	24	0.2%

Parameters: The window size and the slide size specify the amount of data reflecting the most recent context of an application, so usually application-dependent. We set the window size to the default value suggested by the data providers for YahooA1/A2, HTTP, and ECG; to 10 seconds for DLR so that a window contains at least two activities. The slide size was set to 5% of the window size by default, following a relevant survey by Tran et al. [23]. The number of neighbors θ_K and the size of a grid cell θ_R (the size of a micro-cluster for KELOS) are *common* hyper-parameters used by existing local outlier detection algorithms, so they were tuned to achieve the peak accuracy in each experiment. Unless otherwise specified, the search space of θ_K was set to 10% of the number of data points in a window in accordance with the context of *local* outlier, and that of θ_R was set as suggested in Appendix C.1. The error allowance threshold γ for stationary region skipping, which is the *only one* additional parameter in STARE, was fixed to 0.1 as explained in Section 4.4 and Section 5.3.

Algorithms: For comparison with STARE, we chose five existing algorithms, LOF [2], iLOF [14], MiLOF [16], DILOF [13], and KELOS [15] (state-of-the-art). Because LOF was designed for static data, we ran it for every window to adapt to data streams and renamed it to sLOF. Since iLOF, MiLOF, and DILOF did not support data expiration, we also ran them for every window for fair comparison, the same as done by Qin et al. [15]. Because the five algorithms were originally implemented in different languages, we re-implemented them in JAVA from the source codes provided by the authors. STARE was also implemented in JAVA, and the source code is available at https://github.com/kaist-dmlab/STARE.

Computing platform: We conducted experiments on an Amazon AWS c5d.xlarge instance with four vCPUs (3GHz), 8GB of RAM, and 100GB of SSD. Amazon Linux AMI 2018.03 and JDK 1.8.0 are installed in the instance.

Performance metrics: The run time metrics are CPU time to retrieve outliers, averaged per window, and maximum memory consumed (or peak memory). The accuracy metrics are R-precision [4] and average precision [26], which are widely used for evaluating top-*n* outlier detection [3]. *R-precision* is the proportion of the retrieved true outliers out of all true outliers reported for all windows, and thus indicates how many true outliers are found. *Average precision* is the precision of the ranks of the retrieved true outliers *averaged* over all of them, and indicates how high the true outliers are ranked in the result.

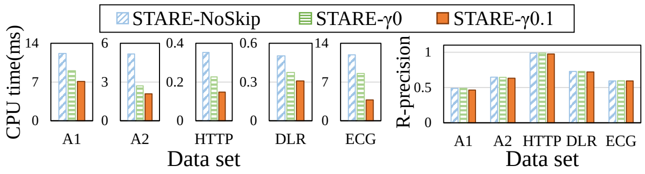


Figure 7: Performance comparison of STARE variants.

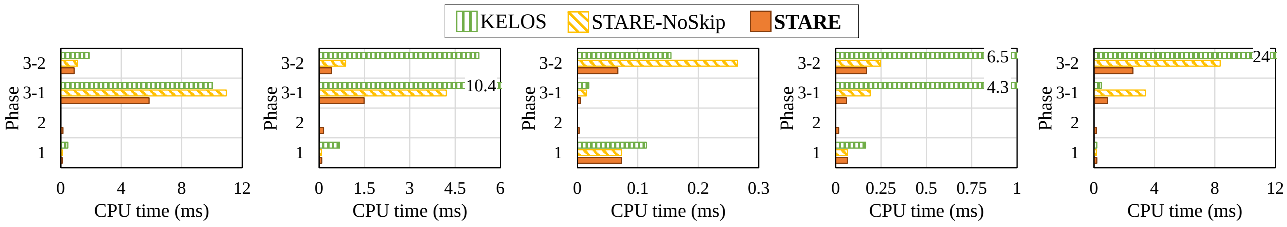
5.2 Overall Performance Comparison

We compared the performance of STARE with other algorithms for all data sets with the default parameter value setting shown in Table 5 and Appendix A.1. The CPU time and the peak memory of the five algorithms are shown in Figure 5, and the accuracy of their results is shown in Figure 6. The results of iLOF are not included due to unacceptable execution time. Note the logarithmic scale of the CPU time. It is evident that STARE was the fastest for all data sets, outperforming sLOF by 3,107 times and KELOS by 11 times when averaged over all data sets. Despite this notable speedup, STARE consumed comparable memory space while achieving the highest outlier detection accuracy. (YahooA2 may be a slight exception, but the accuracy of STARE was still comparable to the highest accuracy, achieved by sLOF.) This remarkable performance of STARE demonstrates the merit of stationary region skipping.

5.3 Analysis of STARE

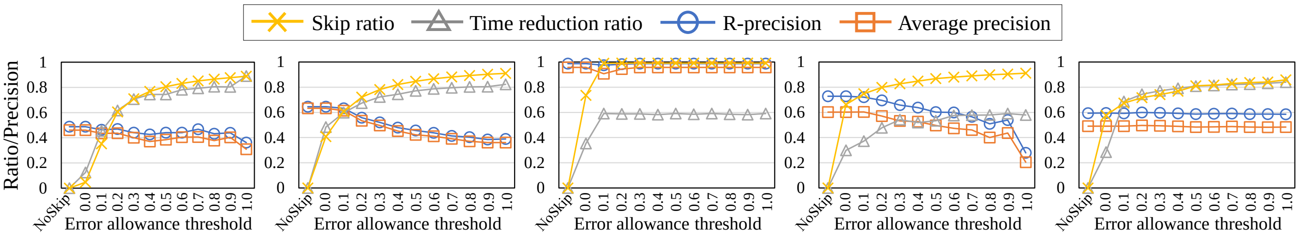
Skipping strategy: Figure 7 shows the performance of STARE for different degrees of region skipping: no skipping, $\gamma = 0$, and $\gamma = 0.1$. To save space, we show only R-precision for the accuracy because average prevision showed similar trends. As expected, STARE-γ0 achieved exactly the same accuracy as STARE-NoSkip while saving 25%–48% of the CPU time. Noticeably, STARE-γ0.1 sustained almost no accuracy loss compared with STARE-NoSkip while saving 39%–68% of the CPU time. This improvement is consistent with the stationary density ratio shown in Table 1, which strongly indicates that the stationary region skipping is the key factor.

Run time breakdown: To further analyze the improvement of STARE over the state-of-the-art algorithm KELOS, in Figure 8, we compared the CPU time of Figure 5 for the three phases common to both algorithms: data distribution update (Phase 1), cell/cluster-level detection for STARE/KELOS (Phase 3-1), and point-level detection (Phase 3-2). Note that region skipping is unique for STARE; the CPU time for the region skipping itself was very small



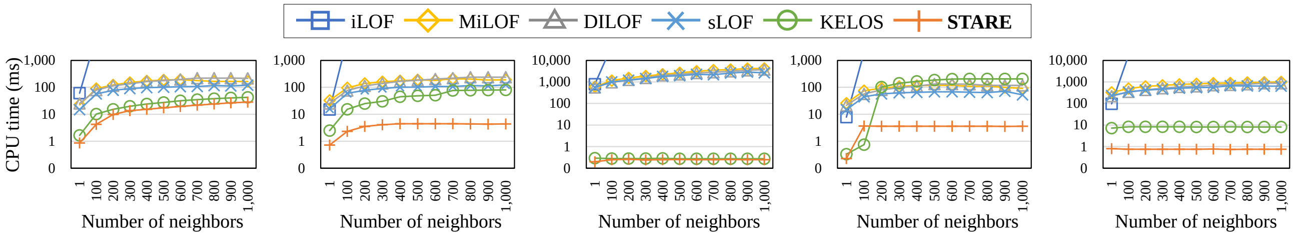
(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 8: Breakdown of CPU time into the four phases: (1) data distribution update, (2) stationary region skip, (3-1) cell-level detection (cluster-level detection for KELOS), and (3-2) point-level detection. Phase 2 is only applicable to STARE.



(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 9: Effects of varying the error allowance threshold γ.



(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 10: CPU time (ms) with varying number of neighbors θ_K.

and was shown for reference purposes. We also added STARE-NoSkip, to highlight the impact of region skipping in each step. The CPU time on the data distribution update was the smallest for all algorithms except HTTP. In the case of YahooA2, because the cell-level detection was the largest for all algorithms, most of data points were pruned, and the workload for the point-level detection was significantly reduced. On the contrary, the case of HTTP showed the opposite trend for the two-level detection. It is notable that STARE outperformed KELOS in most cases, showing that stationary region skipping was effective in reducing the CPU time on both cell-level detection and point-level detection. Interestingly, STARE-NoSkip also spent less time than KELOS in some data sets (e.g., YahooA2 and DLR), which suggested that the kernel center idea itself was beneficial for updating data distribution (Phase 1) and detecting outliers (Phase 3).

Error allowance threshold: To analyze the impact of stationary region skipping on the performance and the trade-off between efficiency and accuracy, we measured R-precision, average precision, skip ratio, and time reduction ratio while increasing the threshold γ from 0 to 1. (Skip ratio is the ratio of the number of skipped grid cells to that of unskipped grid cells; time reduction ratio is the ratio of the saved CPU time to the entire CPU time.) The other parameters were set to the default values. In Figure 9, both skip ratio and time reduction ratio increased as the threshold increased for all data sets. On the other hand, the accuracy decreased as more grid cells were skipped. As discussed in Section 4.4, the threshold 0.1 led to the accuracy close to the best (i.e., the baseline accuracy with no skipping) and the speedup by 1.6–3.2 times for all data sets. Interestingly, HTTP and ECG did not suffer from accuracy loss even with a high skip ratio. It happened because the density distributions of the two data sets barely changed over time, so the local outliers could be detected effectively even with the outdated data distribution of a window.

5.4 Effect of the Number of Neighbors

We analyzed the effect of the number of neighbors, a parameter common to all algorithms, by varying it from 1 to 1,000. As shown in Figure 10, the CPU time of all algorithms increased with θ_K, as finding more neighbors required more processing time. While the CPU time of the LOF-based algorithms (iLOF, MiLOF, DILOF, and sLOF) kept increasing, that of the KDE-based algorithms (KELOS and STARE) converged at some point since the neighbor search was conducted over kernel centers which were much fewer than data points. STARE outperformed all other algorithms in the entire range of θ_K (with minute exception for DLR), thereby demonstrating the robustness of STARE against θ_K.

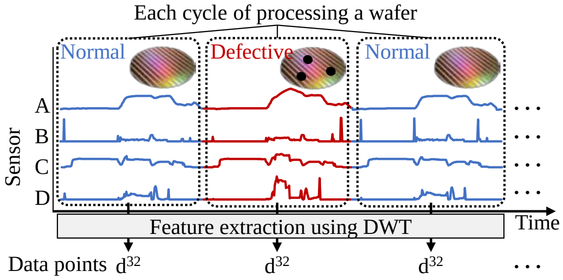


Figure 11: Feature extraction from a sensor data stream.

Table 6: Performance results in FDC.

	CPU time (ms)	R-precision	Average precision
sLOF	13.6	0.77	0.69
KELOS	1.21	0.81	0.81
STARE	**0.81**	**1.00**	**1.00**

5.5 Case Study

The case study is from the domain of smart factories, where multidimensional data streams are commonly generated by digital twins [21] with sensors attached to the facilities. Detecting anomalies from the data streams *with the minimum latency* (e.g., less than a millisecond) is crucial to guarantee the quality of products and to minimize the downtime of a production line.

The specific case study chosen is on the fault detection and classification (FDC) data set provided by Samsung Electronics Co., Ltd. to verify the effectiveness of STARE. The data set consists of four sensor-generated data streams collected during the semiconductor manufacturing process (see Figure 11). The manufacturing process cycle is repeated for each wafer, and the cycle that produces a defective wafer is annotated as an outlier. So, using discrete wavelet transform (DWT), we transformed each cycle containing four data stream segments to a 32-dimensional feature vector (consisting of eight DWT coefficients for each segment). The window size is 534 (the number of wafers processed in a day), and the slide size is 24 (the number of wafers in a lot).

Then, we evaluated the performances of sLOF, KELOS, and STARE for detecting outliers. Table 6 shows that STARE outperforms the other two algorithms in both speed and accuracy. Notably, STARE was able to detect *all* outliers only within a millisecond. This case study thus demonstrates the outstanding usability of STARE for a real manufacturing industry.

6 CONCLUSION

This paper proposed **STARE**, a very fast density-based algorithm for local outlier detection from a windowed data stream. It achieved several orders-of-magnitude speedup over the existing algorithms. This remarkable improvement was enabled by the novel concept, *stationary region skipping*, which exploits the near-stationarity of data densities in the windowed stream processing. We realized this concept using (i) data distribution approximation for reliably and efficiently estimating the change of data densities and (ii) cumulative net-change-based skip for accurately determining the regions to skip updating data densities. Through extensive experiments using real data streams, STARE was shown to outperform the state-of-the-art algorithm by 11 times on average while achieving comparable or higher accuracy. Its practical merit was also demonstrated in a case study with a semiconductor manufacturing industry.

ACKNOWLEDGMENTS

This work was partly supported by Samsung Electronics Co., Ltd. and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00862, DB4DL: High-Usability and Performance In-Memory Distributed DBMS for Deep Learning). We thank Samsung Electronics Co., Ltd. for the opportunity to verify the proposed method with their real data sets provided through the Strategic Collaboration Academic Program.

REFERENCES

[1] KDD Cup 99. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed: 2020-06-01.

[2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, 2000.

[3] 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. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

[4] N. Craswell. *R-Precision, Encyclopedia of Database Systems.* Springer US, Boston, MA, 2009.

[5] K. Frank, M. J. Vera Nadales, P. Robertson, and T. Pfeifer. Bayesian recognition of motion related activities with inertial sensors. In *Proc. UbiComp*, pages 445–446, 2010.

[6] E. Gan and P. Bailis. Scalable kernel density classification via threshold-based pruning. In *Proc. SIGMOD*, pages 945–959, 2017.

[7] F. J. G. Gisbert. Weighted samples, kernel density estimators and convergence. *Empirical Economics*, 28(2):335–351, 2003.

[8] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Trans. on Knowledge and Data Engineering*, 26(9):2250–2267, 2013.

[9] W. Jin, A. K. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proc. KDD*, pages 293–298, 2001.

[10] L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *Proc. MLDM*, pages 61–75, 2007.

[11] Y. Lin, B. S. Lee, and D. Lustgarten. Continuous detection of abnormal heartbeats from ECG using online outlier detection. In *Proc. SIMBig*, pages 349–366, 2018.

[12] M. Mubashir, L. Shao, and L. Seed. A survey on fall detection: Principles and approaches. *Neurocomputing*, 100:144–152, 2013.

[13] G. S. Na, D. Kim, and H. Yu. DILOF: Effective and memory efficient local outlier detection in data streams. In *Proc. KDD*, pages 1993–2002, 2018.

[14] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *Proc. CIDM*, pages 504–515, 2007.

[15] X. Qin, L. Cao, E. A. Rundensteiner, and S. Madden. Scalable kernel density estimation-based local outlier detection over large data streams. In *Proc. EDBT*, pages 421–432, 2019.

[16] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan, and X. Zhang. Fast memory efficient local outlier detection in data streams. *IEEE Trans. on Knowledge and Data Engineering*, 28(12):3246–3260, 2016.

[17] E. Schubert, A. Zimek, and H.-P. Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proc. SDM*, pages 542–550, 2014.

[18] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons, 2015.

[19] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690, 1991.

[20] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* 2018.

[21] F. Tao, H. Zhang, A. Liu, and A. Y. Nee. Digital twin in industry: State-of-the-art. *IEEE Trans. on Industrial Informatics*, 15(4):2405–2415, 2018.

[22] G. R. Terrell, D. W. Scott, et al. Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265, 1992.

[23] L. Tran, L. Fan, and C. Shahabi. Distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment*, 9(12):1089–1100, 2016.

[24] Yahoo! Webscope. ydata-labeled-time-series-anomalies-v1_0. https://webscope.sandbox.yahoo.com. Accessed: 2020-06-01.

[25] S. Yoon, J. G. Lee, and B. S. Lee. NETS: Extremely fast outlier detection from a data stream via set-based processing. *Proceedings of the VLDB Endowment*, 12(11):1303–1315, 2019.

[26] E. Zhang and Y. Zhang. *Average Precision, Encyclopedia of Database Systems.* Springer US, Boston, MA, 2009.

A DETAILED EXPERIMENT SETTING

A.1 Default Parameter Values

Table 7 summarizes the default values of the neighbor count threshold, θ_K, and the size of a grid cell (or the size of a micro-cluster for KELOS), θ_R, in the five algorithms used for the overall performance comparison in Section 5.2. As stated in Section 5.1, their values were tuned to achieve the peak accuracy while θ_K was varied up to 10% of the number of data points in a window and θ_R was set according to the guideline stated in Appendix C.1.

Table 7: Default parameter values of the five algorithms in each data set.

	MiLOF	DILOF	sLOF	KELOS		STARE	
	θ_K	θ_K	θ_K	θ_R	θ_K	θ_R	θ_K
YahooA1	140	140	140	30	140	60	140
YahooA2	50	50	50	27.5	100	65	50
HTTP	600	600	600	5	2	24	5
DLR	100	100	100	2.5	100	18.8	2
ECG	200	200	200	4.3	6	13.5	2

A.2 Data Set Preprocessing

The original YahooA1 data set is composed of 67 CSV files named as "real_1," "real_2," . . ., and "real_67," respectively. We concatenated them sequentially and used the concatenated file as a single data set. Similarly, the original YahooA2 data set is composed of 100 CSV files, and we concatenated them into a single data set. For the HTTP data set, we followed the same instruction given in Outlier Detection Data Set (ODDS)[2] to preprocess the original KDD Cup 99 data set. For the DLR data set, we specifically used the data set named "ARS_Christine_Test_JmpFall_Sensor_Right" as it contains the "falling" activity, which we labeled as an outlier. For the ECG data set, we used the original data set given by the source paper [11].

B PROOF OF THEOREMS

B.1 Proof of Theorem 4.7 (Density Error Bound)

Let t_l be the index of the window where the local density at a data point x was last updated and let t_c be the index of the current window. Additionally, let us denote the sets of θ_K nearest kernel centers of x in the window t_l and the window t_c as $\mathcal{KC}(x;t_l)$ and $\mathcal{KC}(x;t_c)$, respectively, and the net change between them as $\Delta\mathcal{KC}(x;t_l,t_c)$. Because $\Delta\mathcal{KC}(x;t_l,t_c)$ is decided by both the expired kernel centers and the new kernel centers, it reflects the exact net change of the weight distribution of the nearest kernel centers. Then, by Definition 2.2, the local density at x in the current window can be expressed as

$$\begin{aligned}\mathcal{D}_{curr}(x) &= \frac{\sum_{kc_i,w_i\in\mathcal{KC}(x;t_c)} w_i \prod_{l=1}^{d} \mathcal{K}_{h^l}(dist(x^l,kc_i^l))}{\sum_{w_i\in\mathcal{KC}(x;t_c)} w_i} \\ &= \frac{\mathcal{D}_{last}(x)\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i + \sum_{\Delta w_i\in\Delta\mathcal{KC}(x;t_c,t_l)} \Delta w_i} + \mathcal{D}_{new}(x),\end{aligned} \tag{7}$$

where $\mathcal{D}_{last}(x)$ and $\mathcal{D}_{new}(x)$ are the local densities contributed by the original kernel centers in $\mathcal{KC}(x;t_l)$ and the changed kernel centers in $\Delta\mathcal{KC}(x;t_c,t_l)$, respectively, such that

$$\mathcal{D}_{last}(x) = \frac{\sum_{kc_i,w_i\in\mathcal{KC}(x;t_l)} w_i \prod_{l=1}^{d} \mathcal{K}_{h^l}(dist(x^l,kc_i^l))}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i} \text{ and} \tag{8}$$

$$\mathcal{D}_{new}(x) = \frac{\sum_{\Delta w_i\in\Delta\mathcal{KC}(x;t_c,t_l)} \Delta w_i \prod_{l=1}^{d} \mathcal{K}_{h^l}(dist(x^l,kc_j^l))}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i + \sum_{\Delta w_i\in\Delta\mathcal{KC}(x;t_c,t_l)} \Delta w_i}. \tag{9}$$

Since the value of a kernel function $\mathcal{K}_h$ monotonically decreases from the center,

$$\mathcal{D}_{new}(x) \le \frac{\sum_{\Delta w_j\in\Delta\mathcal{KC}(x;t_c,t_l)} |\Delta w_j|\mathcal{K}_{\tilde{h}}(0)^d}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i + \sum_{\Delta w_j\in\Delta\mathcal{KC}(x;t_c,t_l)} \Delta w_j}. \tag{10}$$

Because of the near-stationarity of data distribution, the distance to the θ_K-th nearest kernel center of x does not change significantly. Thus, the ratio of the sum of weight net-changes is approximated by the cumulative error, $E(x;t_c,t_l)$, in Definition 4.5. Furthermore, the cumulative error must be less than or equal to γ by the definition of stationary region skipping in Definition 4.6, i.e.,

$$\begin{aligned}\frac{\sum_{\Delta w_j\in\Delta\mathcal{KC}(x;t_c,t_l)} \Delta w_j}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i} &\le \frac{\sum_{\Delta w_j\in\Delta\mathcal{KC}(x;t_c,t_l)} |\Delta w_j|}{\sum_{w_i\in\mathcal{KC}(x;t_l)} w_i} \\ &\le E(x;t_c,t_l) \\ &\le \gamma.\end{aligned} \tag{11}$$

By consolidating Eq. (7)–Eq. (11), we derive Eq. (5) with straightforward mathematics as follows:

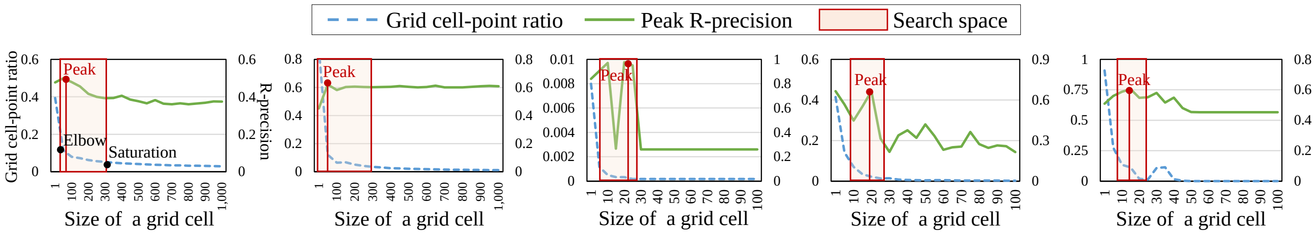
$$\begin{aligned}|\Delta\mathcal{D}(x)| &= |\mathcal{D}_{curr}(x) - \mathcal{D}_{last}(x)| \\ &\le |\frac{\mathcal{D}_{last}(x)}{1+\gamma} + \mathcal{D}_{new}(x) - \mathcal{D}_{last}(x)| \\ &\le |\frac{-\gamma\mathcal{D}_{last}(x)}{1+\gamma} + \frac{\gamma\mathcal{K}_{\tilde{h}}(0)^d}{1+\gamma}| \\ &= |\frac{\gamma(\mathcal{K}_{\tilde{h}}(0)^d - \mathcal{D}_{last}(x))}{1+\gamma}|.\end{aligned} \tag{12}$$

□

B.2 Proof of Theorem 4.9 (Time Complexity)

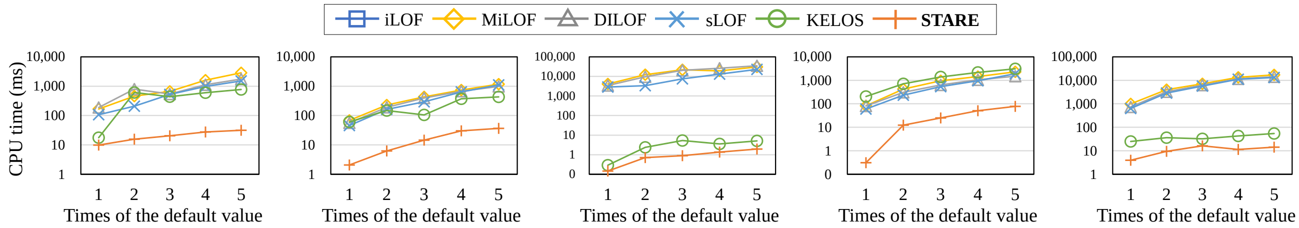
The time complexity of the first phase, data distribution update, is $O(W)$ because it takes constant time for a data point to find the grid cell where the data point falls in given that the size of a grid cell is fixed. Then, the time complexity of the second phase, stationary region skipping, is $O(N_G) + O(rN_G^2) + O(W)$, where $O(N_G)$ is for cumulative error computation because the non-zero weight net-changes in $\Delta\mathbb{G}$ are negligible, $O(rN_G^2)$ is for local density updates of grid cells, and $O(W)$ is for local density updates for data points. Finally, the time complexity of the third phase, outlier detection, is $O(N_G) + O(pW)$. Since the number n of outliers to find is very small, the time complexity of cell-level detection is $O(N_G)$, and that of point-level detection is $O(pW)$, as pW is the number of data points in the candidate grid cells. Overall, the dominant time complexity of STARE is $O(W + rN_G^2)$. □

[2] http://odds.cs.stonybrook.edu/http-kddcup99-dataset/



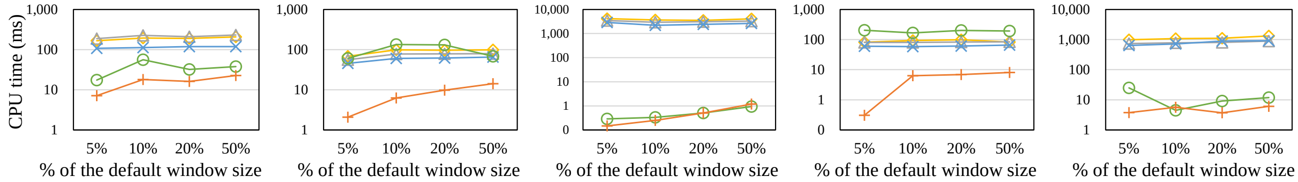
(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 12: Effects of varying the size θ_R of a grid cell.



(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 13: CPU time (ms) with varying the window size (times of the default value).



(a) YahooA1. **(b) YahooA2.** **(c) HTTP.** **(d) DLR.** **(e) ECG.**

Figure 14: CPU time (ms) with varying the slide size (percentage of the default window size).

C ADDITIONAL EXPERIMENTS

C.1 Guideline on the Size of a Grid Cell

The size of a grid cell, θ_R, is relevant to the degree of approximating data points. We set the ratio of the number of non-empty grid cells to the number of data points in a window as an indicator of the degree of approximation. The optimal θ_R should make the ratio fairly small to reduce the computation overhead but not too small, in order to preserve the outlier detection accuracy. To this end, we find the first elbow of the ratio curve by increasing θ_R from a sufficiently small value, as shown in Figure 12. The range between the first elbow and the saturation point is likely to include the best value of θ_R in terms of the peak R-precision, and thus the user can easily determine θ_R by grid search within this range.

C.2 Effect of Window Size and Slide Size

While the window size and the slide size are commonly provided and/or derived from the data set and application in hand, their effects on the processing time is worth examining. We compared the performance of STARE with other algorithms (except the iLOF with unacceptable execution time). The window size and the slide size were respectively set to the default value (see Table 5) while the other size was varied in each experiment. The algorithm-dependent parameters θ_K and θ_R were tuned from 1 to 1,000 and from 1 to 100, respectively, to achieve the peak R-precision.

Varying the window size (see Figure 13): The window size is shown as the number of times of the default window size. All algorithms spent more CPU time in detecting outliers with a larger window size in most cases. STARE was still the fastest among all algorithms in the entire rage of the window size, showing that STARE was effective regardless of the window size.

Varying the slide size (see Figure 14): The slide size is shown as the percentage of the default slide size. All algorithms except STARE and KELOS were hardly affected by the slide size, because they were not subject to the effects of the net changes in STARE or microclustering in KELOS. The trend of KELOS appeared to be inconsistent for different data sets, which we believe is attributed to the different effects of microclustering under different data distributions. In contrast, STARE spent increasingly more time as the slide size increased, because of the increase in net changes between window slides and the consequential decrease in region skipping. Yet, STARE showed the smallest CPU time in most of the range.