Streaming Time Series Subsequence Anomaly Detection: A Glance and Focus Approach

Wenjing Wang
Huazhong University of Science and Technology
wenjingwang@hust.edu.cn

Ziyang Yue
Huazhong University of Science and Technology
ziyangyue@hust.edu.cn

Bolong Zheng*
Huazhong University of Science and Technology
bolongzheng@hust.edu.cn

ABSTRACT

Subsequence anomaly detection for time series is a crucial problem in various real-world applications. However, existing methods proposed so far design the anomaly score functions solely based on either local neighborhood or global patterns, leading to unsatisfactory detection accuracy. In addition, these methods either cannot adapt, or yield insufficient accuracy and efficiency in streaming scenario. Therefore, we propose Sirloin, an accurate and efficient streaming time series subsequence anomaly detection framework. First, Sirloin proposes a glance and focus anomaly score function that takes both global and local information into consideration, contributing to an accurate anomaly detection. Second, Sirloin dynamically maintains an inverted file index and product quantization codebooks to index and compress the subsequences, hence is able to cope with the time series evolution and to process streaming batches efficiently. In addition, a dual index optimization strategy is put forward that further improves the efficiency. An experimental study in 11 different datasets from 5 domains offers insight into the performance of Sirloin, showing that it improves throughput on average 4× and enhances accuracy 58.02% compared to the state-of-the-art streaming method.

PVLDB Reference Format:
Wenjing Wang, Ziyang Yue, and Bolong Zheng. Streaming Time Series Subsequence Anomaly Detection: A Glance and Focus Approach. PVLDB, 18(6): 1892 - 1904, 2025.
doi:10.14778/3725688.3725714

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Wangwenjing1996/Sirloin.

1 INTRODUCTION

Anomaly detection for time series is a fundamental function in various fields such as finance, healthcare, and industrial monitoring [9, 20]. Related applications include detecting abnormal transactions or market fluctuations, discovering deviations in patients' physiological parameters, and monitoring the equipment failures. Therefore, time series anomaly detection is crucial for maintaining system stability, enhancing safety, and optimizing operational efficiency, etc., hence has attracted widespread attention in both fields of industry and scientific research over decades [18, 25].

In time series, anomalies can manifest in various forms, such as spikes, drops, or deviations from expected behavior. In this study, we focus on the subsequence anomaly detection for time series, where the anomaly is not reflected by a single value, but by a continuous interval. Even in the case that each value in an interval falls within the normal range, the trend of all values may still exhibit anomaly [17]. For example, in industrial production, a series of consecutive normal points in system operations could indicate an impending equipment failure. Therefore, subsequence anomaly detection is a significant and challenging task.

To solve the subsequence anomaly detection problem, existing studies are proposed in two categories, namely neighborhood based methods [6, 7, 13, 15, 22, 30, 34] and pattern based methods [3–5, 14]. Neighborhood based methods measure the anomaly of a subsequence by its nearest neighbor(s). One of the most prevailing neighborhood based method is Matrix Profile [31]. It defines the anomaly as the subsequence that has the largest distance to its nearest neighbor in the time series, which is called discord. They performs well when the time series contains only one anomaly. However, these methods suffer from a so-called twin-freak problem, where repeated anomalies become the nearest neighbor of each other with a small distance, leading to their failure on these anomalies. Recently, pattern based methods attract widespread attention. They discover anomalies according to the global distribution. As a well-known method in pattern based methods, SAND [5] applies clustering to extract global patterns, and detects anomalies by comparing them with global patterns. Therefore, even if an anomaly appears multiple times, they can still detect it due to its sparse distribution. This is able to overcome the shortcoming of neighborhood based methods. However, we observe two problems occur when leveraging existing methods:

- **Insufficient accuracy.** Neighborhood based methods do not consider the global distribution of subsequences, while pattern based methods lack a focused view of the local neighborhood. Both approaches exhibit a form of tunnel vision, as they only rely on either the local or the global perspective, neglecting the complementary insights provided by the other. Although pattern based methods claim to solve the twin-freak problem, which is the shortcoming of neighborhood based methods, neither of them emerges as a front-runner. Therefore, designing an effective anomaly score function that combines both global and local information remains a significant challenge.

*Bolong Zheng is the corresponding author

Proceedings of the VLDB Endowment, Vol. 18, No. 6 ISSN 2150-8097.
doi:10.14778/3725688.3725714

- **Struggle to handle streaming data.** Most existing methods are designed for static scenarios, where the entire time series is available in advance. However, in real-world applications, time series data often arrives in a streaming fashion, requiring real-time anomaly detection based on historical data. The few methods developed for streaming scenarios typically struggle to deliver satisfactory performance.

To address these aforementioned challenges, we propose Sirloin, an accurate and efficient streaming subsequence anomaly detection framework. The novelty of Sirloin lies in three key aspects, namely a glance and focus anomaly score function, the Sirloin index, and the dual index optimization strategy. First, Sirloin employs a glance and focus anomaly score function that effectively integrates global and local information, avoiding the information loss that arises from existing approaches. This function allows Sirloin to accurately detect not only repeated anomalies but also subtle ones hidden within normal subsequences. Second, Sirloin constructs and dynamically updates its index to facilitate efficient approximate nearest neighbor search, capture global patterns accurately, and minimize memory usage. The update strategy provides flexibility in handling varying pattern numbers while maintaining quantization quality with high efficiency. Finally, the dual index optimization restructures the update and anomaly score computation stages for higher efficiency. By postponing part of the update until after anomaly score computation and enabling faster convergence of the best-so-far result during the search, this optimization ensures that pruning conditions are met more easily, further boosting performance.

The main contributions are summarized as follows:

- We carefully design a novel glance and focus anomaly score function that incorporates both global and local information, effectively addressing the shortcomings of existing methods.
- We develop the Sirloin index, integrating an inverted file index and product quantization for fast anomaly score computation. To handle the continuous evolution of time series, we propose an update strategy that dynamically maintains the Sirloin index in real-time. In addition, we introduce a dual index optimization strategy that further enhances efficiency.
- We conduct extensive experiments across 11 datasets from 5 different domains, thoroughly verifying the effectiveness and efficiency of Sirloin.

2 RELATED WORK

We proceed to review related studies of subsequence anomaly detection, which mainly fall into three categories: the neighborhood based methods, the pattern based methods, and the deep learning based methods.

Neighborhood based methods. Discord [7, 15, 30, 34] is one of the most prevailing anomaly definition for its concise representation, which is defined as the subsequence whose distance to its nearest neighbor is the largest in the time series. STAMP [30] proposes a technique, namely Matrix Profile, that effectively computes the Euclidean distances between all pairs of subsequences using Fast Fourier Transform (FFT), and finds the Discord based on the Matrix Profile. STAMPI is an online version of STAMP and is able to update the Matrix Profile incrementally in stream. DAMP [15] narrows the search space to only consider a part of subsequences that occur before the current subsequence. In this way, it improves the efficiency in a streaming fashion.

In addition to methods designed specifically for time series, outlier detection methods for multi-dimensional data are also proposed in this category. LOF [6] focuses on the outlier degree of a data point relative to the surrounding neighborhood. It defines outliers based on the ratio of its nearest neighbors' local densities over its own local density. ABOD [13] considers both angles and distances to measure the outlier, where an outlier can only form a small angle with any pair of its nearest neighbors.

However, these methods face a so-called twin-freak problem, where an anomaly appears multiple times [15]. In this case, multiple repeated anomalies become the nearest neighbors of each other, causing the neighborhood based methods fail.

Pattern based methods. Pattern based methods capture global patterns and discover anomalies based on the patterns. S2G [4] constructs a graph, where each vertex is a pattern formed by similar subsequences, and each edge connects two vertices with their temporal adjacency. NormA [3] and SAND [5] use clustering to extract normal patterns, and compute the anomaly score of each subsequence by the sum of its distances to all the cluster centroids. SAND further proposes a streaming subsequence anomaly detection framework to update the cluster-based normal patterns. Isolation Forest [14] employs random space partition trees to capture the patterns. Anomalies located in sparse regions tend to be isolated at shallow nodes.

Pattern based methods discover anomalies by considering the distribution of all subsequences. Therefore, even an anomaly may repeat multiple times, it is easy to distinguish it from normal patterns due to its sparse distribution, which helps to tackle the twin-freak problem. However, pattern extraction is often conducted in a coarse-grained manner such that subtle anomalies may mix with normal patterns and are difficult to discover.

Deep learning based methods. Recently, deep learning has been widely applied in time series anomaly detection, such as OmniAnomaly [24], USAD [2], Anomaly Transformer [29], TranAD [27], and SGAT-AE [33]. However, these methods require intensive training on massive labeled or anomaly free data. As deep learning based methods are orthogonal to our study, we omit the discussion due to the space limitations.

3 PRELIMINARIES

We proceed to introduce the preliminaries. Frequently used notation is summarized in Table 1.

3.1 Problem Setting

A time series refers to a sequence of data points collected in chronological order, and is typically used to describe the data changes over time. We formally define it as follows.

Definition 1 (Time Series). *A time series T is an ordered sequence consisting of n data points, represented as $T = [t_1, \ldots, t_n]$, where $t_i \in \mathbb{R}$, $i \in [1, n]$. The length of T is denoted as $|T| = n$.*

In many real world applications, the time series is collected in a streaming manner. The total length of the time series is unknown in advance and is potentially infinite. Therefore, the analysis is

Table 1: Summary of Notation

Notation	Definition
T	A time series
n	The length of T
l	Subsequence length
$\mathbb{T}$	The set of all subsequences in T of length l
B_i	The i-th batch in T
$\mathbb{B}_i$	The set of all subsequences in B_i
b	The length of batch
$\mathrm{AS}(\cdot)$	The anomaly score function

usually conducted on a newly arriving batch of data points, which is defined as follows:

Definition 2 (Batch). *A batch of data points in a time series T is a finite sequence, denoted as $B_i = [t_j, \dots, t_{j+b-1}]$, where b is the length of B_i. Note that, for two consecutive B_i and B_{i+1}, the last data point of B_i is immediately followed by the first data point of B_{i+1}.*

Subsequence anomaly detection is a critical task that aims to identify continuous anomalous data points in a time series rather than isolated anomalous points. This is because a single data point may fall within the normal range of the whole time series, but a sequence of similar points can form an anomalous pattern. Therefore, we focus on the subsequence anomaly detection, and we define the subsequence as follows:

Definition 3 (Subsequence). *A subsequence $T_{i,l}$ in a time series $T = [t_1, \dots, t_n]$ is defined as a sequence of l contiguous data points starting from t_i, denoted as $T_{i,l} = [t_i, \dots, t_{i+l-1}]$. We denote $T_{i,l}$ as T_i for simplicity when there is no ambiguity. For a batch $B = [t_i, \dots, t_{i+b-1}]$, we denote the set of all possible subsequences as $\mathbb{B} = \{T_j\}$, where $j \in [i, i+b-l]$.*

3.2 Problem Definition

We study the problem of detecting the η most abnormal subsequences in streaming time series, which is defined as follows:

Definition 4 (Streaming Subsequence Anomaly Detection (SSAD)). *Given a streaming time series T that data points arrive in batches, a subsequence length l, and a delicately designed subsequence anomaly score function $AS(\cdot)$, the streaming subsequence anomaly detection (SSAD) aims to dynamically maintain a set R of η most abnormal subsequences in T, i.e., for $\forall T_i \in R$ and $T_j \notin R$, $AS(T_i) \geq AS(T_j)$.*

For each arriving batch $\mathbb{B} = \{T_i\}$, we compute $AS(T_i)$ for each subsequence T_i, and update R if necessary.

Example 1. *Fig. 1(a) shows an example of the SSAD problem with $\eta = 1$. On the top is the time series, followed by anomaly scores computed by three different algorithms, namely SAND, Matrix Profile, and Kmeans, denoted as AS_{SAND}, AS_{MP}, and AS_{Kmeans}, respectively. The red cursors annotate positions of the highest anomaly scores, i.e., $AS_{SAND}(T[12354, 12434]) = 0.790$, $AS_{MP}(T[12545, 12625]) = 0.296$, and $AS_{Kmeans}(T[12362, 12442]) = 0.839$, as indicated by shadowed rectangles.*

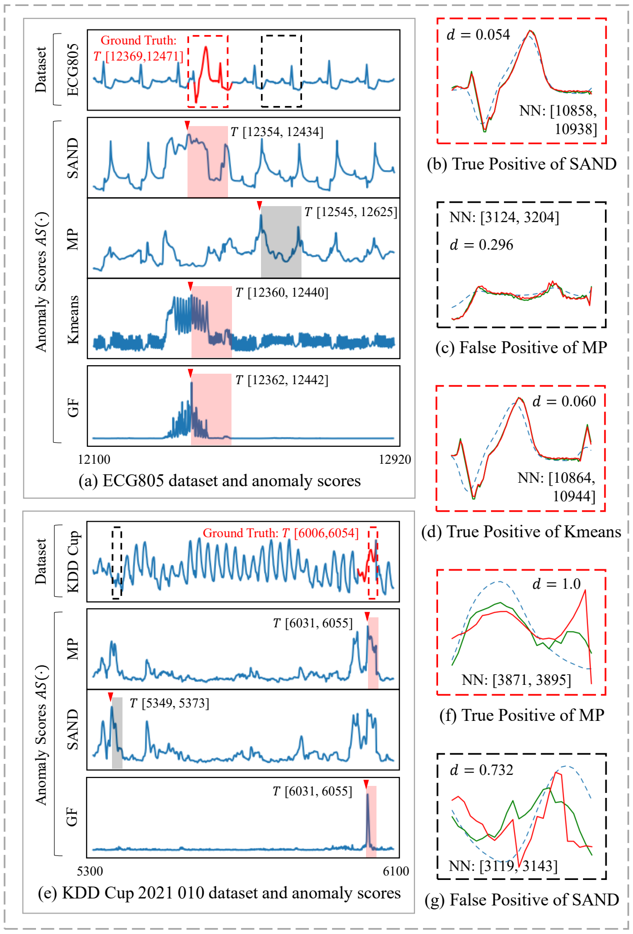


Figure 1: Pre-experiments in (a) ECG dataset and (e) KDD Cup 2021 dataset, including the time series and anomaly scores. The right plots the highlighted subsequence (red), its nearest neighbor (green), and its nearest centroid (blue).

As the subsequences reported by SAND and Kmeans have overlaps with the ground truth $T[12369, 12471]$, we consider they find a true positive and correctly discover the anomaly. In contrast, Matrix Profile reports a false negative that has no overlap with the ground truth, hence it fails to discover the anomaly.

4 A GLANCE AND FOCUS ANOMALY SCORE

We proceed to analyze the limitations of existing methods and propose a novel anomaly score function.

4.1 Limitations and Motivation

The SSAD problem usually involves two tasks, including designing an effective anomaly score function and finding anomalies efficiently. Existing anomaly score functions can be divided into two categories w.r.t. anomaly detection methods. Neighborhood based methods design the anomaly score functions by utilizing the local information, such as the Matrix Profile series methods, LOF, and ABOD. In contrast, pattern based methods employ the global information, including S2G, NormA, SAND, and IF. However, the performance ranking of methods varies considerably across different datasets, with no consistent front-runner. This phenomenon is also observed in the TSB-UAD benchmark [19, 26] and TSAD benchmark [32]. Next, we analyze the reason why some methods are only effective in specific datasets with pre-experiments in ECG [16] and KDD Cup 2021 [12] datasets.

Pre-experiments in ECG. ECG is an electrocardiogram dataset containing repeated anomalies generated by certain mechanisms, such as atrial fibrillation or premature beats. We conduct a pre-experiment on ECG805 dataset and show the results in Fig. 1(a). On the top is a subset of the dataset, followed by corresponding anomaly scores computed via two representative score functions of SAND and Matrix Profile, respectively. The red line is the annotation of the ground-truth anomaly, and the dashed boxes are the reported anomalies by different methods. We can see that SAND successfully distinguishes the anomaly (labeled in red), but Matrix Profile mistakes a normal subsequence (labeled in gray) for the anomaly. This is because Matrix Profile considers the local information, and defines an anomaly as a subsequence that deviates greatly from its neighborhood, hence suffers from the twin-freak phenomenon. For SAND, which designs the score function based on the data distribution, anomalies located in sparse areas can be distinguished easily, even when they repeat multiple times. Figs. 1(b) and (c) show the true positive $T[12354, 12434]$ and false negative $T[12545, 12625]$ along with their nearest neighbors (in green), respectively. Matrix Profile reports the false positive since its distance to the nearest neighbor $T[3124, 3204]$ is 0.296, and is the largest among all subsequences. The true positive and its nearest neighbor $T[10858, 10938]$ form a twin-freak with a distance of only 0.054, which is why Matrix Profile fails.

Pre-experiments in KDD Cup 2021. KDD Cup 2021 dataset consists of 250 univariate time series from various domains, and each containing only one anomaly. Fig. 1(e) shows pre-experiment results of the 10-th time series. Interestingly, the performance ranking of two anomaly score functions is reversed compared to the results observed in the ECG dataset. Matrix Profile successfully detects the anomaly, while SAND fails. This can be attributed to two main factors. First, the dataset lacks strong regularity and contains various types of normal patterns, making it challenging for clustering to accurately capture the distribution of all normal patterns. As a result, some normal subsequences also exhibit large distances to their nearest cluster centroids. Second, the anomaly in this dataset is rare and not sufficiently distinct from normal subsequences. Instead of forming an isolated cluster, the anomaly is scattered within a normal pattern, causing both normal and abnormal subsequences to have similar distances to the nearest centroids. In contrast, the anomaly exhibits a larger nearest neighbor distance compared to normal subsequences. The details of detected subsequences and their nearest neighbors are illustrated in Figs. 1(f) and (g).

Effects of abnormal patterns. In addition, anomalies can influence the patterns and form abnormal patterns. To explore the effects of abnormal patterns, we conduct another pre-experiment on ECG805 dataset. Specifically, we employ Kmeans to generate 32 clusters, and take the distance to the nearest centroid as the anomaly score function, denoted as $\mathsf{AS}_{\mathsf{Kmeans}}$. The anomaly score is also plotted in Fig. 1(a). Intuitively, if a pattern fits an anomaly behavior, anomalies belonging to this pattern should yield small anomaly scores and cannot be detected. However, we observe a counter-intuitive result.

Take Figs. 1(c) and (d) as examples. Fig. 1(c) shows a normal subsequence $T[12545, 12625]$, and Fig. 1(d) shows an abnormal subsequence $T[12360, 12440]$. We draw a blue dashed line to show their nearest centroids. Obviously, the centroid of the abnormal subsequence fits an abnormal pattern. However, this abnormal subsequence still has a larger distance to its nearest centroid.

As subsequences are obtained by sliding windows, two chronologically successive subsequences are highly-overlapping, with a small distance between them. Assume a normal and an abnormal patterns successively occur in time series, then a transition area exists between these two patterns in the subsequence space. When fitting the distribution of subsequences, the transition area can be assigned to either of them. In both cases, the centroid of the pattern being assigned the transition area needs to shift towards that transition area, which decreases the fitting accuracy of the corresponding pattern. Since the subsequences located in an abnormal pattern are far fewer than those in a normal pattern, clustering tends assign the transition area to the abnormal pattern, so that the total decrease of fitting accuracy is minimized. Therefore, we argue that there is no need to mitigate the influence of abnormal patterns.

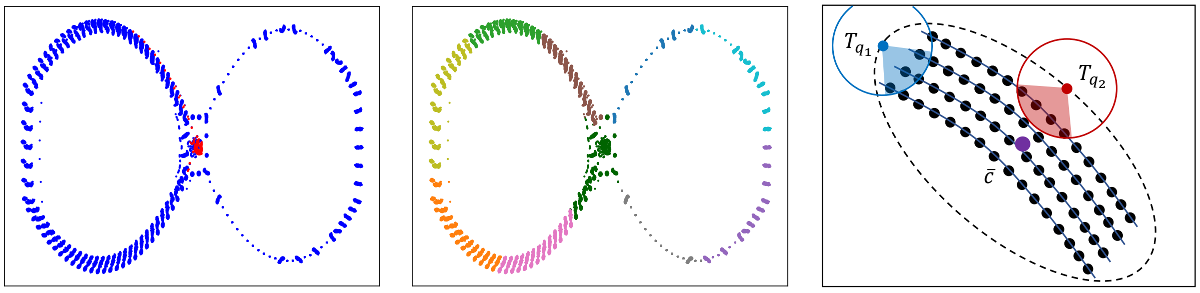
The pre-experiments indicate that existing anomaly score functions, whether based on global or local information, are only effective on datasets with specific characteristics. Relying solely on either global or local information is insufficient to handle different types of datasets. Next, we introduce a novel anomaly score function that considers both information.

4.2 Anomaly Score

Based on the discussion above, we reach the conclusion that focusing only on local information makes methods vulnerable to repeated anomalies, while relying solely on global information may miss subtle anomalies hidden among normal subsequences. Therefore, we propose a glance and focus novel anomaly score function, denoted as $\mathsf{AS}_{\mathsf{GF}}$, that considers both local and global information.

Considering local information. In Sec. 4.1, we demonstrate that global information alone is not enough to distinguish normal and abnormal subsequences. Therefore, we leverage the nearest neighbor distance as local information. Given a subsequence T_q, we denote its k nearest neighbors (kNN) as $\mathbb{N}_k(T_q) = \{T_{o_1}, \ldots, T_{o_k}\}$. The nearest neighbor distance $d(T_q, T_{o_i})$ implies to what extent a subsequence deviates from its neighborhood and serves as a **focused** compensation for the global information.

Considering global information. We choose Kmeans for clustering subsequences and extracting the global information, due to its intuitive interpretation and outstanding performance on subsequence anomaly detection [3, 5]. The accuracy evaluation demonstrates that using Kmeans is enough to achieve satisfactory performance, as indicated by Fig. 1(a). For a set of subsequences $\mathbb{T}$, we partition them into *nlist* disjoint clusters with *nlist* centroids, denoted as $\bar{c}_1, \ldots, \bar{c}_{nlist}$ and $C_1, \ldots, C_{nlist}$, respectively. Therefore, we have a **glance** at the subsequence distribution and each pattern is represented by a centroid. Each cluster contains similar subsequences that belong to a corresponding pattern. In the following,



(a) t-SNE visualization (b) t-SNE visualization with clustering (c) A toy example

Figure 2: (a) t-SNE visualization of a time series in 2D space, anomalies are highlighted in red; (b) Clustering of the time series, each color represents a cluster; (c) A toy example inspired by the t-SNE visualization.

we use pattern and centroid interchangeably. Given a subsequence T_q and its kNN set $\mathbb{N}_k(T_q)$, we first select clusters that cover T_q's neighborhood, i.e., $\{C_i\}$ that satisfy $C_i \cap \mathbb{N}_k(T_q) \neq \emptyset$. Then, the distances between T_q and the centroids $\bar{c}_i$ are employed to evaluate T_q's abnormality.

We differ from existing studies [3, 5] in two aspects. First, these studies compute distances to all patterns, whereas we only consider the patterns similar to T_q. Since both normal and abnormal subsequence can have one or more extremely dissimilar patterns, considering such patterns may cause deterioration in performance. Second, existing studies design mechanisms to mitigate the effect of abnormal patterns, such as assigning different weights to patterns. However, we observe in the pre-experiments (Sec. 4.1) that even a pattern fits an anomaly behavior, anomalies belong to this pattern still yield larger distance to the centroid. Therefore, We do not bother to introduce such mechanism that complicates the score function and may bring additional overhead.

Best of both worlds. After both global and local information are obtained, the remaining question is how to combine them. A straightforward way is using the product of these two distances. Given a subsequence T_q and its kNN set $\mathbb{N}_k(T_q)$, we define a naive anomaly score function as follows,

$$\mathrm{AS}_{\mathrm{naive}}(T_q) = \mathrm{E}\left[\bigcup_{T_{o_j}} \{d(T_q, \bar{c}_i) \cdot (T_q, T_{o_j})\}\right], \tag{1}$$

where $T_{o_j} \in \mathbb{N}_k(T_q)$, $\bar{c}_i$ is the centroid w.r.t. T_{o_j}, and $\mathrm{E}[\cdot]$ computes the mean value. Specifically, $\mathrm{AS}_{\mathrm{naive}}$ reports the average value of all products of the distances of T_q to its centroid and to its neighbors. In experiments, Table 5 shows that $\mathrm{AS}_{\mathrm{naive}}$ already outperforms the baseline methods.

However, we argue that this straightforward approach leads to information loss. By employing a more thoughtful and effective combination, the performance can be further improved. The intuition here is that simply measuring absolute deviations from patterns and neighborhood is insufficient to capture anomalies. The distribution of a neighborhood relative to the pattern also plays a crucial role. In subsequence data, clusters often extend along a specific direction, forming non-spherical shapes. Subsequences can be viewed as a trajectory in high-dimensional space, where adjacent subsequences are chronologically successive. Fig. 2(a) provides an example of subsequences in a time series, reduced to 2D space using t-SNE, where each point represents a subsequence and anomalies are highlighted in red. The subsequences clearly form a trajectory in the shape of an "∞". In Fig. 2(b), Kmeans is applied and different clusters are shown in distinct colors. Each cluster extends along the trajectory. When comparing Fig. 2(a) with Fig. 2(b), it becomes clear that some anomalies, such as those in the brown cluster, deviate orthogonally from the trajectory, residing on the cluster's border.

We plot this condition in a toy example (Fig. 2(c)). The dashed ellipse is a cluster with centroid $\bar{c}$, and T_{q_1}, T_{q_2} are two suspicious subsequences. Intuitively, T_{q_2} is more anomalous than T_{q_1}, because T_{q_1} aligns with the direction of the data distribution, whereas T_{q_2} deviates from it. However, compared with T_{q_2}, T_{q_1} has a similar nearest neighbor distance and a larger distance to the centroid, leading to a larger naive anomaly score (Eq. 1). The most essential difference between T_{q_1} and T_{q_2} is that the neighbors of T_{q_2} distribute more dispersedly in direction, thus forming a larger angle represented by the red sector in Fig. 2(c). In fact, some studies [13, 21] also find angle information useful in discovering data distribution. Since it is impossible to find such an angle that covers all neighbors in high dimensional space, we use the fluctuation of the angle formed by the subsequence, the centroid, and a nearest neighbor, to measure such dispersity. When neighbors distribute dispersedly, larger angles can be formed and contribute to the fluctuation. In addition, distance still matters and should be taken into consideration. Therefore, we define the anomaly score function as follows:

Definition 5 (A Glance and focus Anomaly Score). *Given a subsequence T_q, its kNN set $\mathbb{N}_k(T_q)$, the clusters $\{C_1, \ldots, C_{nlist}\}$, and the corresponding cluster centroids $\{\bar{c}_1, \ldots, \bar{c}_{nlist}\}$, the anomaly score function of T_q is defined as:*

$$AS_{GF}(T_q) = \mathrm{Var}\left[\bigcup_{T_{o_j}} \{\overrightarrow{T_q\bar{c}_i} \cdot \overrightarrow{T_qT_{o_j}}\}\right], \tag{2}$$

where $T_{o_j} \in \mathbb{N}_k(T_q)$, $\bar{c}_i$ is the centroid w.r.t. T_{o_j}, and $\mathrm{Var}[\cdot]$ *computes the variance.*

We choose the inner product as it satisfies all the necessary requirements for evaluating both distances and angles. Specifically, the inner product $\overrightarrow{T_q\bar{c}_i} \cdot \overrightarrow{T_qT_{o_j}} = ||\overrightarrow{T_q\bar{c}_i}|| \cdot ||\overrightarrow{T_qT_{o_j}}|| \cdot \cos\angle\bar{c}_iT_qT_{o_j}$, where the distance terms measure the deviation from the patterns

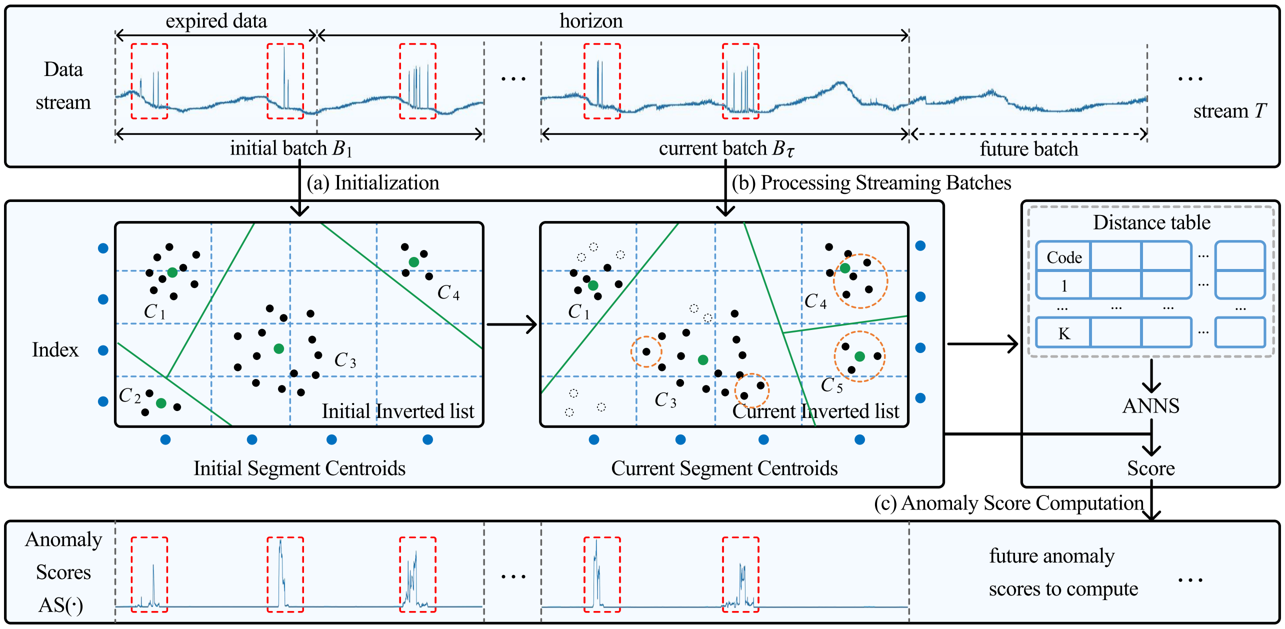


Figure 3: Framework Overview of Sirloin

and neighborhood, and the angle term measures the dispersity. In the case of an anomaly that deviates significantly or a normal subsequence within a cluster, distance terms dominate the variance and distinguish subsequences correctly. In Figs. 1(a) and (e), the glance and focus anomaly scores are illustrated at the bottom. Obviously, the proposed anomaly score function not only discovers both anomalies accurately, but also generates more distinguishable anomaly scores between the anomaly and normal subsequences.

5 SOLUTION

We proceed to introduce the proposed Sirloin to solve the SSAD problem, including initialization, anomaly score computation, and streaming batch processing. Fig. 3 shows the framework overview of Sirloin. First, we initialize Sirloin with the subsequences in the initial batch. Then, we introduce how to compute the glance and focus anomaly score for each subsequence with indexes of Sirloin. Third, for each arriving batch, the index updates to adapt to the evolution of the time series, with new subsequences inserted and expired subsequences deleted.

5.1 Initialization

As aforementioned, the glance and focus anomaly score function considers both global information and local information, derived from clustering and nearest neighbors, respectively. The inverted file product quantization (IVFPQ) [11]: (1) is one of the most prevailing approximate nearest neighbor search (ANNS) algorithms that efficiently finds the kNNs for vectors/data sequences; (2) partitions the data via clustering; (3) employs an effective data compression technique, hence is able to cope with gradually accumulated streaming batches with a low memory footprint. Therefore, we believe it is well-suited for addressing the SSAD problem. To be self-contained, we briefly introduce the initialization of inverted file (IVF) index and product quantization (PQ).

Initializing IVF index. When the initial batch $\mathbb{B}_1$ arrives, we employ Kmeans to partition the subsequences of $\mathbb{B}_1$ into *nlist* clusters, i.e., $\mathbb{C} = \{C_1, \ldots, C_{nlist}\}$, and build an inverted list for each cluster that stores the IDs of all subsequences assigned to that cluster, which significantly reduces the search space for ANNS while also capturing global patterns from the initial batch.

Initializing PQ. To reduce the memory footprint, PQ is employed to compress the subsequences. It splits each subsequence into smaller segments, and quantizes segments in groups. The quantization of a subsequence is then the concatenation of the quantization of each segment. Specifically, given a subsequence $T_o \in \mathbb{R}^l$, PQ evenly divides T_o into M segments, denoted as $T_o = [T_o^1, \ldots, T_o^M]$, where each segment $T_o^i \in \mathbb{R}^{l/M}, i \in [1, M]$. This divides the whole subsequence space into M segment spaces. In each segment space, Kmeans is applied to segments and generates K segment centroids, namely $W^i = \{w_1^i, \ldots, w_K^i\}$ in the i-th segment space. Each segment is quantized by the nearest segment centroid, denoted by an indicator $u^i(\cdot)$. For example, $u^i(T_o^i) = w_j^i$ indicates that T_o^i is quantized by the j-th segment centroid in the i-th segment space. This means T_o^i is approximated by w_j^i and encoded as a short integer j. In this way, PQ quantizes a subsequence T_o as $u(T_o) = [u^1(T_o^1), \ldots, u^M(T_o^M)]$ and obtains its PQ code consisting of M short integers. We denote $W = W^1 \times \cdots \times W^M$ as the codebook, where $\times$ means Cartesian product.

Example 2. *Fig. 3(a) shows an example of initialization in 2D space. The subsequences of length 2 are partitioned into 4 clusters, i.e.* C_1, C_2, C_3, C_4*, where green points represent the centorids. Each subsequence is divided into two segments of length 1. The blue points represent the segment centroids.*

It is worth noting that Sirloin does not employ an optimization technique of IVFPQ, called residuals [11]. This technique sacrifices efficiency for lower quantization loss and higher ANNS accuracy. However, on the one hand, Sirloin does not require such high quality results from ANNS. We compare Sirloin with and without residuals in static scenarios, and find that residuals improve the accuracy only marginally. The experimental results are shown in Table 4, where Sirloin$_{res}$ stands for Sirloin with residuals. On the other hand, this technique cannot be directly applied to Sirloin in streaming scenarios. Because it quantizes the differences between the subsequences and corresponding cluster centroids, instead of the original subsequences. However, contrast to the original subsequences that always remain unchanged, such differences continuously change due to the update of centroids. To overcome this issue, re-quantization is needed when the centroids update, which leads to a further decrease in efficiency.

5.2 Anomaly Score Computation

For a newly arriving batch $\mathbb{B}_\tau$, Sirloin computes the glance and focus anomaly scores for subsequences in a chronological order. Specifically, for each subsequence T_q, we first perform an ANNS to obtain both the global and local information, and we then compute the anomaly scores with almost no additional overhead.

Performing ANNS. Given T_q, Sirloin first computes its distance to the centroid of each inverted list, and selects *nprobe* inverted lists with the nearest *nprobe* centroids for further evaluation. Then, in each segment space (say the i-th), the distances between T_q^i and all corresponding segment centroids are computed and stored in a distance table. When evaluating a subsequence T_o, Sirloin approximates the Euclidean distance by the distance between T_q and the quantization of T_o, i.e.,

$$d(T_q, T_o) = d(T_q, u(T_o)) = \sum\nolimits_{i=1}^{M} d(T_q^i, u^i(T_o^i)), \tag{3}$$

where $d(T_q^i, u^i(T_o^i))$ is acquired by table look-up. The subseqences with top-k smallest distance are returned as the ANNS results.

Computing anomaly score. During the anomaly score computation, we also use the quantizations of the nearest neighbors instead of original subsequences, which are not stored. Therefore, the anomaly score function becomes:

$$\mathrm{AS}_{\mathrm{GF}}(T_q) = \mathrm{Var}\left[\bigcup\nolimits_{T_{o_j}} \{\overrightarrow{T_q\bar{c}_i} \cdot \overrightarrow{T_q u(T_{o_j})}\}\right], \tag{4}$$

where we do not distinguish approximate nearest neighbor set to avoid introducing additional notation. However, the straightforward computation of the inner product $\overrightarrow{T_q\bar{c}_i} \cdot \overrightarrow{T_q u(T_{o_j})}$ incurs computation cost that linearly increases with the subsequence length. To avoid this overhead, we transform the inner product as follows:

$$\overrightarrow{T_q\bar{c}_i} \cdot \overrightarrow{T_q u(T_{o_j})} = \frac{1}{2}\left(d^2(T_q, \bar{c}_i) + d^2(T_q, T_{o_j}) - d^2(\bar{c}_i, T_{o_j})\right), \tag{5}$$

where the first two terms are obtained during ANNS, and the third term can be precomputed and stored in the inverted list. In this way, the score computation requires only constant time and is independent from the subsequence length.

Although Sirloin aims to solve the SSAD problem, it is also able to solve subsequence anomaly detection in static scenario, with the whole time series as the initial batch.

5.3 Processing Streaming Batches

In the streaming scenario, the time series evolves over time. Therefore, the index structure of Sirloin should also evolve to accommodate the continuous evolution. However, it would lead to unbearable time overhead if reconstructing the IVF index and PQ codebooks from scratch every time a new batch arrives. To avoid this problem, we design a real-time update mechanism for Sirloin, including updating the IVF index and PQ codebooks.

Update of IVF index. For each following batch $\mathbb{B}_\tau$ ($\tau > 1$), the purpose of updating IVF index is to adjust the centroids of the inverted lists and reassign the historical subsequences along with new subsequences to the inverted lists, so that the IVF index can provide better global patterns for SSAD and more reasonable candidate set for ANNS.

Under a fixed *nlist*, an optimal way to update the IVF index is assigning new subsequences to the current centroids and running additional Kmeans iterations to optimize the clustering. However, optimizing the clustering by both historical and new subsequences incurs high computation overheads, let alone the need to decode the PQ codes back to subsequences, since Sirloin only stores the PQ codes rather than original subsequences. Another way is updating the centroids only once after assigning new subsequences. However, if new subsequences drift away from the historical ones in the subsequence space, they may be frequently assigned to several clusters and drag the corresponding centroids along the drift direction after centroid update, which forms a positive feedback. In the worst case, new subsequences in streaming batches may be totally assigned to a single cluster, making both functions of the IVF index lose efficacy, i.e. capturing patterns and restricting search space. We call this phenomenon as cluster explosion. In addition, *nlist* is a parameter that is impossible to properly set beforehand. The number of clusters should be more flexible to better adapt to the distribution of subsequences.

Therefore, we propose an online update strategy that (1) avoids processing historical subsequences, (2) is free of cluster explosion, (3) generates flexible number of clusters. Specifically, Sirloin only runs Kmeans on the arriving batch at first. Then, clusters of historical subsequences and new subsequences are emerged based on the similarity of clusters. At last, expired subsequences are deleted to avoid the effects of the out-of-date data.

When a batch $\mathbb{B}_\tau$ arrives, assume the current cluster set is $\mathbb{C}$. Note that $|\mathbb{C}| = nlist$ does not hold because the number of clusters can change during update. Sirloin applies Kmeans on $\mathbb{B}_\tau$ and achieves *nlist* clusters, denoted as $\mathbb{C}^\tau = \{C_1^\tau, \ldots, C_{nlist}^\tau\}$. Then, Sirloin does not merge all clusters into existing clusters, since we have demonstrated that this may lead to cluster explosion. Instead, for a cluster $C_i^\tau \in \mathbb{C}^\tau$, it is emerged into an existing cluster $C_j \in \mathbb{C}$, iff. C_i^τ is close enough to C_j. In other words, the distance between two centroids is small. Otherwise, it is treated as an independent cluster. However, the capabilities of existing clusters to absorb new clusters are different. For example, we tend to merge new clusters to dense clusters with more cautiousness compared to sparse clusters to avoid cluster explosion and obtain more balanced clustering. Therefore, for each cluster C_j, we employ a density parameter ρ_j to evaluate its density, which is initialized as the maximum distance between the subsequences located in C_j, i.e., $\rho_j = \max\left\{d(T_o, \bar{c}_j) | T_o \in C_j\right\}$.

The intuition is that a larger maximum distance indicates that the cluster is located in sparse area, only under which condition Kmeans assigns far subsequence to it to minimize the global loss. When the distance between two centroids is smaller than the density parameter, i.e. $d(\bar{c}_i^\tau, \bar{c}_j) < \rho_j$, C_i^τ is merged into C_j. After the merge, Sirloin updates the centroid by the weighted sum of corresponding values of the two clusters, where $\bar{c}'_j = \frac{|C_i^\tau|*\bar{c}_i^\tau+|C_j|*\bar{c}_j}{|C_i^\tau|+|C_j|}$ and $\rho'_j = \frac{|C_i^\tau|*\rho_i^\tau+|C_j|*\rho_j}{|C_i^\tau|+|C_j|}$. The unmerged clusters are inserted into $\mathbb{C}$ as independent clusters. Then new subsequences are appended to the corresponding inverted lists. Since only similar clusters are merged, the original centroid change slightly, so we do not need to reassign historical subsequences.

In addition, we only retain the most recent portion of subsequences and delete expired data, for three reasons. First, since the total length of time series is unknown and potentially infinite, storing all historical subsequences leads to an uncontrollable storage overhead. Second, the infinite growth of data volume in the index also causes continuous declines in the ANNS efficiency. Third, as the time series evolves over time, a normal pattern in history can become no more normal. Retaining all the historical subsequences in index does not increase but deteriorates the detection accuracy. Therefore, we only retain the subsequences from the latest window of length H. For the expired data, we remove it from the IVF index. When a cluster becomes empty, we delete it from $\mathbb{C}$.

Example 3. *Fig. 3(b) shows an example of the IVF index update. The subseuqences in the arriving batch form 4 clusters marked by the dashed orange circles. Three of them are merged into existing clusters C_3 and C_4, and one of them becomes an independent cluster. At last, the expired data represented by the dashed points are deleted, where C_2 becomes an empty cluster and is removed.*

Online PQ. Although PQ is effective in subsequence compression, it is a static method that cannot fit the time series evolution. In essence, the purpose of updating PQ is similar with updating IVF index. The difference lies in that, for PQ there is a fixed budget of segment centroids, i.e., the number of centroids in each segment space should remain the same during the update. Therefore, we employ online product quantization [28] to update the codebook for each batch $\mathbb{B}_\tau$. Online PQ quantizes new subsequences with the current codebook, and updates the segment centroids by the quantization error.

Given that time series drift occurs gradually, we assume that the subsequence distribution does not change dramatically in a short period. As a result, the current codebook W can be used to quantize the newly arriving subsequences. Namely, for a subsequence T_o, we obtain the quantization $u(o^i) = w_j^i$ in an arbitrary i-th segment space. Next is to update the codebook. If only considering one subsequence T_o, to minimize the quantization loss in the i-th segment space, w_j^i should be updated as $w_j^{i\prime} = w_j^i + (T_o^i - w_j^i)$. Take the historical data into consideration, the quantization loss is averaged by the number of segments quantized to w_j^i, i.e. $w_j^{i\prime} = w_j^i + (T_o^i - w_j^i)/n_j^i$.

However, updating the codebook in a subsequence-wise manner may cause frequent fluctuations in codebook and inconsistent quantization quality within a batch, which deteriorates the performance. Therefore, we update the codebook in a batch-wise manner that agrees with the streaming scenario. Specifically, we quantize all the subsequences in a batch and update the codebook by the total quantization loss, i.e.:

$$w_j^{i\prime} = w_j^i + \frac{1}{n_j^{i\prime}} \sum\nolimits_{u^i(T_o^i)=w_j^i} (T_o^i - w_j^i), \tag{6}$$

where $n_j^{i\prime}$ is the number of segments quantized to w_j^i including the current batch.

After the update is completed, we perform ANNS and compute anomaly scores in the same procedure as in static scenarios, which has been discussed in Sec. 5.2.

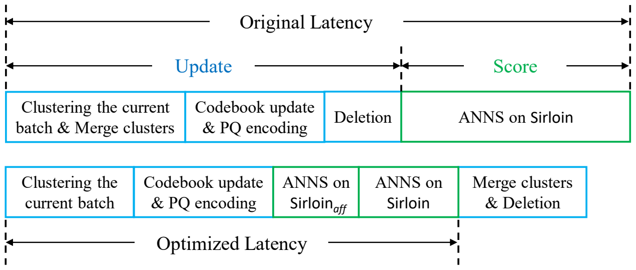

Original Latency
Update
Score
Clustering the current batch & Merge clusters
Codebook update & PQ encoding
Deletion
ANNS on Sirloin
Clustering the current batch
Codebook update & PQ encoding
ANNS on Sirloin$_{aff}$
ANNS on Sirloin
Merge clusters & Deletion
Optimized Latency

Figure 4: Dual Index Optimization

6 DUAL INDEX OPTIMIZATION

We introduce an optimization strategy that further reduces computational overhead by restructuring the update and scoring stages in a more efficient manner, referred to as Sirloin+. It accelerates the ANNS process through a dual index structure, utilizing pruning based on a distance lower bound provided by the triangle inequality. This significantly enhances the performance by reducing unnecessary distance computations.

Most of the query time of Sirloin is spent on the update stage. However, in the SSAD problem, the most urgent task when a new batch arrives is to score each subsequence with low latency, rather than focusing on maintaining a high-quality index. As for the update, it can be completed at any time in the interval between the arrivals of two successive batches without any negative effects. Therefore, an aggressive optimization is to switch the update and scoring stages, meaning subsequences could be scored based on the current index without updating it. However, this could result in a loss of critical information from the incoming batch. For instance, if a new normal pattern emerges in the incoming batch, subsequences from this pattern might be falsely recognized as anomalies due to their deviation from the original patterns. To avoid this issue, we still need to consider the global and local information in the current batch, which necessitates performing clustering and ANNS. However, operations like merging clusters and deleting expired data are not as time-sensitive and can be postponed. Therefore, we shift part of the update operations to occur after the scoring process.

This idea is illustrated in Fig. 4, where Sirloin+ leverages an affiliated Sirloin index to obtain the information in the current batch separately. Specifically, the affiliated Sirloin index, denoted as Sirloin$_{aff}$, is the same as Sirloin in essence, and indexes the subsequences from the incoming batch. Instead of initialization from scratch, Sirloin$_{aff}$ fully reuses the intermediate results in the update procedure to avoid incurring additional overheads. Recall that during update, we cluster the new batch, quantize the new subsequences, and update the codebook, which almost entirely covers the cost of initialization. Sirloin$_{aff}$ directly employs the clustering centroids, PQ codes, and codebook, hence is constructed with almost no additional overheads. For ANNS, the overhead is mainly from computing the distance table and performing a linear scan of subsequences in the probed clusters. Since Sirloin and Sirloin$_{aff}$ share the same codebook, the distance table only needs to be computed once and can be used by both indexes. For the linear scan, there is almost no difference between the number of candidate subsequences from two indexes and that from a single index without optimization. As a result, the dual index structure for ANNS also incurs no additional overheads during this process. In contrast, the postponed merge of clusters and deletion of expired data no more contribute to the detection latency.

Besides, in order to reduce distance computation overheads during ANNS, we propose a pruning strategy based on the dual index structure and a lower bound provided by the triangle inequality:

$$LB(T_o) = |d(T_o, \bar{c}_i) - d(T_q, \bar{c}_i)|, \tag{7}$$

where $d(T_o, \bar{c}_i)$ and $d(T_q, \bar{c}_i)$ have already been computed. Once $LB(T_o)$ exceeds the best-so-far nearest neighbor distance, T_o can be safely pruned.

However, in many cases pruning based on lower bound cannot improve but reduce efficiency. It is because the distance to the best-so-far k-th nearest neighbor converges too slowly to the final result, and always remains a relative large value. Hence the pruning condition is hard to meet and the increased overhead of lower bound computation goes beyond the reduced overhead of distance computation. On the contrary, the dual index structure makes the best-so-far distance converge faster. As shifts exist in time series, the nearest neighbors of a subsequence are temporally close to it with high probability. Therefore, we first perform ANNS on Sirloin$_{aff}$ index without pruning. In this way, a high quality best-so-far distance is achieved with a small search space, since Sirloin$_{aff}$ only indexes the incoming batch that makes up a small portion of all retaining subsequences. Next, ANNS continues on the main Sirloin index with pruning to obtain the final anomaly scores.

7 EXPERIMENTAL EVALUATION

7.1 Experimental Setup

We implement Sirloin in C++ and compile it using GCC 8.3.1. All experiments are run on a Linux machine with an Intel Xeon Gold 622R @ 2.90GHz processor and 92GB memory.

Datasets. We conduct the experiments in 11 datasets across 5 different domains. The details of datasets are shown in Table 2, where l_A represents the average length of anomalies, and η is the number of anomalies.

(1) *IOPS* [23] is non-periodic time series reflecting server performance metrics.
(2) *ECG* [16] is a periodic electrocardiogram dataset, containing atrial fibrillation and premature beats.
(3) *Dodgers* [10] is a loop sensor dataset from the Glendale of the 101 North freeway, where anomalies are unusual traffic after Dodgers games.
(4) *KPI-AIOPS* [8] comes from the 2018 International AIOps Challenge and contains KPI data from real scenarios in Internet companies.
(5) *SED* [1] is periodic simulated engine disk data that records the disk speed from NASA Rotary Dynamics Laboratory.

Table 2: Dataset Statistics

Datasets	n	l_A	l	η	Domain
IOPS_37aa	149K	16	16	63	Server Performance
IOPS_b37d	146K	14	16	59	Server Performance
IOPS_ee91	149K	12	16	59	Server Performance
ECG803	100K	75	80	62	Cardiology
ECG805	100K	75	80	133	Cardiology
ECG806	100K	75	128	27	Cardiology
ECG14046	100K	75	80	142	Cardiology
Dodgers	50K	42	64	133	Traffic
KPI-AIOPS_1	100K	84	64	88	KPI
KPI-AIOPS_2	100K	164	64	56	KPI
SED	100K	75	80	76	Electronic

Table 3: Comparison Methods

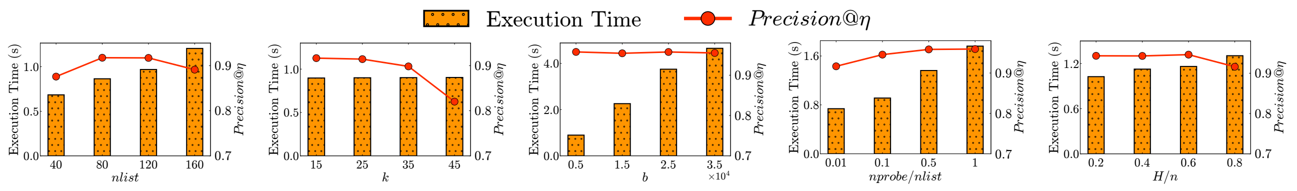
Methods	Categories	Scenarios
LOF	Neighborhood based	Static
ABOD	Neighborhood based	Static
Matrix Profile (MP)	Neighborhood based	Static
IF	Pattern based	Static
NormA	Pattern based	Static
S2G	Pattern based	Static
STAMPI	Neighborhood based	Streaming
DAMP	Neighborhood based	Streaming
SAND	Pattern based	Streaming

Baselines. We compare Sirloin with 6 static methods (the data input to methods is the entire time series) and 3 streaming methods (the data input to the methods is batches in time series). The details of comparison methods are shown in Table 3.

Metrics. We use *Precision@η* to evaluate the accuracy of all methods, which is the ratio of correctly detected anomalies among the top η subsequences with the highest anomaly scores. To ensure the result accuracy, we exclude surrounding subsequences when selecting the subsequences with the highest anomaly score to avoid the repeated selection of the same anomaly. In experiments, we set η to the number of true anomalies in time series. In addition, we measure the efficiency with the throughput per second and the execution time. The throughput is defined as the number of subsequences processed per second. The execution time is the time between the arrival of a batch and output of corresponding anomaly scores.

7.2 Parameter Settings

The subsequence length l is set identically for all methods, as specified in Table 2. We set $M = 8$ and $K = 256$ for all datasets, as recommended in PQ [11]. For other parameters, we tune them on



(a) Effects of *nlist* (b) Effects of *k* (c) Effects of batch size *b* (d) Effects of *nprobe* (e) Effects of *H*

Figure 5: Effects of Parameters

one dataset to obtain the optimal values, and then apply them to other datasets, which achieves satisfactory results. Thus, we consider these parameters to be relatively generalizable. To reproduce, users can adopt our parameter settings or tune them after collecting part of the data. We study the effects of these parameters when they vary around the optimal values in Fig. 5, including the number of clusters per batch *nlist*, the number of nearest neighbors k, the batch size b, the number of inverted lists to evaluate *nprobe*, and the length of the window retained H. For baselines, parameters are set to the default values as suggested in their studies.

Effects of varying *nlist*. As shown in Fig. 5(a), the execution time per batch of Sirloin increases steadily with enlarging *nlist*, because the time required to construct the index is proportional to *nlist*. In addition, the accuracy is affected by varying *nlist*. If *nlist* is too large or too small, the clustering may not align well with the data distribution. We set *nlist* to 100 for all datasets.

Effects of varying *k*. As shown in Fig. 5(b), the execution time remains stable as k increases, since most of the time is spent on ANNS, which is unaffected by k. However, the accuracy initially remains stable but gradually decreases as k becomes too large. This is because an excessive k reduces the emphasis on local information, which negatively impacts the accuracy. In our experiments, we set k to 25 for all datasets.

Effects of varying *b*. As shown in Fig. 5(c), the execution time per batch increases as batch size b grows, as more subsequences are processed in each batch. However, the accuracy remains stable with varying b, as changes in b have little effect on the neighbors of subsequences and clusters. We set b to 5000 for all datasets.

Effects of varying *nprobe*. As shown in Fig. 5(d), both the execution time per batch and accuracy increase as *nprobe* grows. Since a larger *nprobe* retrieves more candidate neighbors, which achieves more accurate ANNS results at the cost of more distance computations. Hence the accuracy is improved. In our experiments, the *nprobe* is 10 for all datasets.

Effects of varying *H*. As shown in Fig. 5(e), when H increases, the execution time also increases, while the accuracy initially remains stable but then begins to decline. A larger H makes Sirloin hold more subsequences, which reduces the ANNS efficiency. In addition, since normal patterns in history may become abnormal as the time series evolves, a larger H reduces the accuracy. In our experiments, we set H to 0.2 times of n for SED, and 0.5 times of n for other datasets.

7.3 Accuracy Evaluation

We evaluate the accuracy of proposed methods with *Precision@η* in different datasets under both the static and streaming scenarios.

Static scenarios. The left of Table 4 reports results of Sirloin and 6 competitive static methods in 11 datasets across 5 different domains under static scenarios. Overall, Sirloin performs better than other methods because it effectively combines both the local and global information, allowing it to adapt to a variety of datasets across different domains. Specifically, Sirloin achieves the best *Precision@η* in 9 out of 11 datasets. It detects only 1 fewer anomaly than NormA and S2G in ECG14046 and ECG806 datasets, while performing comparably or better than the comparison methods in the other datasets.

We observe that pattern based methods, such as IF, NormA, and SAND, struggle with non-periodic datasets (e.g., IOPS, Dodgers, and KPI AIOPS), because they compute anomaly scores by relying solely on the global information. These methods use clustering or tree structures to identify potential patterns, but they often fail to capture the intricacies of non-periodic data, resulting in large differences between normal subsequences and these patterns, making it difficult to distinguish normal and abnormal subsequences, which leads to poor anomaly detection results.

In contrast, neighborhood based methods, such as LOF and MP, generally perform worse than other methods in periodic datasets (ECG and SED), because they compute anomaly scores based solely on local information, focusing on the relationship between subsequences and their neighbors. In periodic datasets, anomalies often recur, and this local perspective can blur the distinctions between abnormal subsequences and their neighbors, which leads to the so-called twin-freak problem, and makes it hard to identify anomalies. Note that, MP performs poorly in IOPS datasets due to its use of z-normalized Euclidean distance for finding nearest neighbors. In non-periodic datasets, this normalization can make dissimilar subsequences appear similar, hindering MP's ability to detect anomalies. Additionally, while ABOD incorporates both distance and angle in its anomaly score design, its overall performance is still worse than Sirloin. This is because ABOD primarily focuses on the distance between subsequences and their neighbors, relying solely on local information, which limits its effectiveness in distinguishing anomalies.

Streaming scenarios. We simulate the streaming scenarios by dividing data points into batches. The right of Table 4 reports results of Sirloin and 3 competitive streaming methods in 11 datasets across 5 different domains under streaming scenarios.

Table 4: The Accuracy Evaluation of *Precision@η* under Static and Streaming Scenarios

	Static								Streaming			
Dataset	**LOF**	**ABOD**	**MP**	**IF**	**NormA**	**S2G**	**Sirloin$_{res}$**	**Sirloin**	**STAMPI**	**DAMP**	**SAND**	**Sirloin**
IOPS_37aa	0.7937	0.8730	0.0159	0.2810	0.0000	0.8730	**0.9206**	**0.9206**	0.0000	0.0159	0.0016	**0.9206**
IOPS_b37d	0.7321	0.7500	0.0000	0.2536	0.0000	0.8929	**0.9464**	**0.9464**	0.0000	0.0000	0.0000	**0.9643**
IOPS_ee91	0.7797	0.8136	0.0000	0.3898	0.0000	0.8814	**0.9661**	0.9492	0.0000	0.0000	0.0136	**0.9492**
ECG803	0.1452	0.7097	0.6129	1.0000	0.9887	1.0000	**1.0000**	**1.0000**	0.5806	0.2419	0.8968	**0.9838**
ECG805	0.5000	0.9103	0.0769	0.9872	0.9872	0.9872	**0.9872**	**0.9872**	0.0641	0.6410	0.5603	**0.9872**
ECG806	0.9200	0.8800	0.8800	0.5240	0.7000	**0.9600**	0.9200	0.9200	0.8800	0.2000	0.7200	**0.9200**
ECG14046	0.6475	0.8197	0.5820	0.9836	**1.0000**	0.9426	**1.0000**	0.9918	0.5574	0.7787	0.4027	**0.9836**
Dodgers	0.5263	0.6391	0.3910	0.6195	0.2902	0.7068	**0.8572**	0.8195	0.4135	0.3157	0.4586	**0.8271**
KPI-AIOPS_1	0.0114	0.8068	0.1250	0.4671	0.0068	0.9205	**1.0000**	**1.0000**	0.0113	0.1250	0.0534	**0.9886**
KPI-AIOPS_2	0.0000	0.0000	0.0000	0.3312	0.0000	0.8621	**0.9655**	**0.9655**	0.0000	0.0000	0.0000	**0.9655**
SED	0.9388	0.9592	0.9796	0.0408	0.9980	1.0000	**1.0000**	**1.0000**	1.0000	0.3061	1.0000	**1.0000**
Average	0.5450	0.7419	0.3276	0.5339	0.4519	0.9115	**0.9603**	0.9546	0.3188	0.2386	0.3734	**0.9536**

As shown in Table 4, Sirloin achieves comparable accuracy under streaming scenarios to that under static ones, and in some datasets, it even outperforms results of static scenarios. This highlights the effectiveness of online update strategy of Sirloin, which adapts to the continuously changing data distribution in streaming contexts and accurately captures global patterns, leading to more accurate detection results. Additionally, Sirloin has clear advantages over other methods. For example, STAMPI and DAMP evaluate the abnormality of subsequences using nearest neighbor distances, focusing solely on local information. This approach becomes ineffective when similar abnormal patterns emerge. Meanwhile, SAND only captures the global information, which hinders its ability to detect subtle anomalies and leads to poor performance in non-periodic datasets. In contrast, Sirloin effectively integrates both the local and global information, thereby avoiding the aforementioned shortcomings.

The experimental results reported in Table 4 fully demonstrate the superiority of Sirloin and also confirm the effectiveness of online update strategy of Sirloin.

7.4 Efficiency Evaluation

We study the efficiency of Sirloin and other competitive online methods by comparing throughput.

Fig. 6 illustrates the throughput across different datasets with varying batch sizes. Fig. 7 shows the average execution time per batch of different methods. Both STAMPI and DAMP use a similar approach, measuring the abnormality of subsequences using nearest neighbor distance, but DAMP computes approximate nearest neighbor distances, which leads to significantly higher efficiency compared to STAMPI. Therefore, we report only the results for DAMP and omit STAMPI to save space. We observe that Sirloin achieves much higher throughput and shorter execution time than both SAND and DAMP across different datasets. The improvements are two-fold. For SAND, Sirloin has much lower complexity on clustering. For DAMP, Sirloin has a smaller search space and computes distance faster.

Additionally, Sirloin+ outperforms Sirloin in both throughput and execution time due to its effective optimization strategy. This improvement meets the real-time requirements of anomaly detection, making Sirloin+ more suitable for real-world applications.

Table 5: The Accuracy of Different Anomaly Scores

Function	IOPS	ECG	Dodgers	KPI-AIOPS	SED
AS_{naive}	0.9286	0.8800	0.8271	0.9886	1.0000
AS_{GF}	**0.9464**	**0.9200**	**0.8346**	**0.9886**	**1.0000**

7.5 Improvements of Optimizations

Improvements of anomaly score function. As shown in Table 5, AS_{GF} outperforms AS_{naive}, highlighting the rationality and effectiveness of AS_{GF}. We combine the local and global information in a more sophisticated way than AS_{naive}, which considers the distribution of anomalies relative to its neighborhood and patterns. This enables AS_{GF} to detect subtle anomalies that are similar to normal subsequences. Fig. 8 shows additional subtle anomalies detected by AS_{GF}, which AS_{naive} misses. Even in datasets with complex patterns or subtle anomalies, AS_{GF} accurately identifies them. These experimental results highlight the superiority of AS_{GF}.

Improvements of dual index optimization. We evaluate the advantages of Sirloin+ over Sirloin and other baselines in terms of throughput and average execution time per batch across different datasets. As shown in Figs. 6 and 7, Sirloin+ outperforms all methods in throughput and execution time. Compared to Sirloin, the dual index structure pruning strategy in Sirloin+ effectively reduces distance computation in ANNS. Compared to SAND and DAMP, Sirloin+ has lower clustering time complexity and more efficient distance computations in ANNS.

7.6 Memory Evaluation

Fig. 9 shows the theoretical and actual peak memory usage for different methods. Although the actual memory usage is higher than the theoretical one due to practical factors such as caching and thread stacks, they show similar trends. Sirlon consumes the most memory, while the trends of STAMPI, DAMP and SAND vary across datasets. The theoretical memory usage is the sum of the

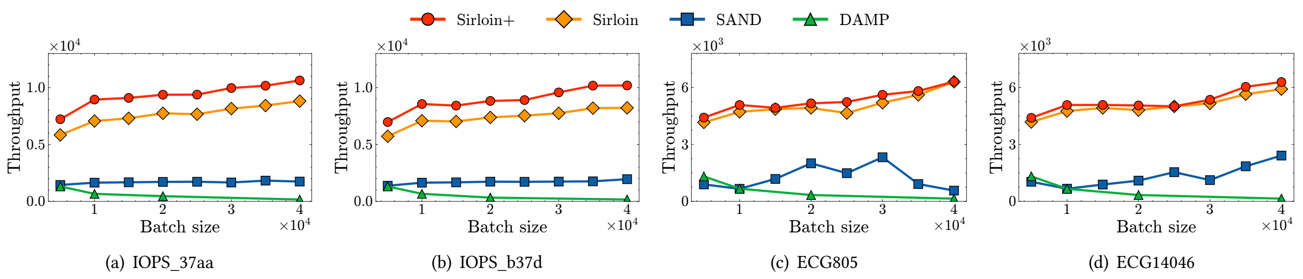


Figure 6: The Throughput in Different Datasets with Varying Batch Sizes

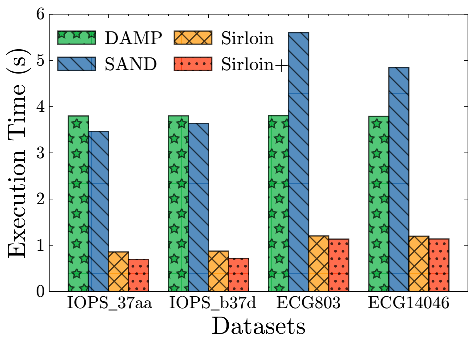


Figure 7: The Average Execution Time per Batch

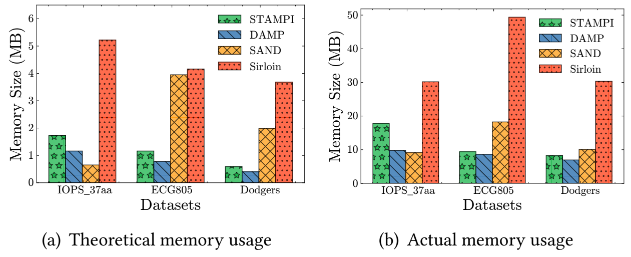


Figure 9: The Peak Memory Usage of Different Methods

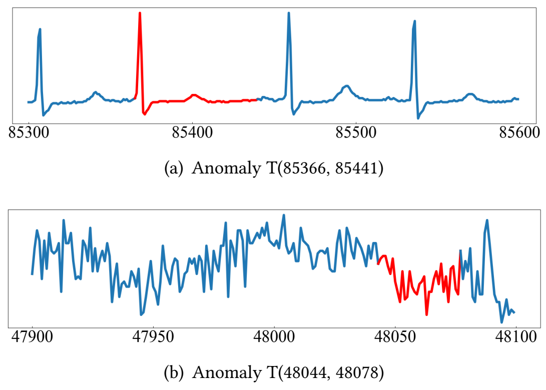


Figure 8: Anomalies in (a) ECG and (b) Dodgers

index size (for SAND and Sirloin) or raw data (for STAMPI and DAMP), and the space required by anomaly score computation. STAMPI and DAMP have no index structure, but need to store the whole time series. When computing the anomaly score, they store the distance of each subsequence to its top-1 nearest neighbor. Therefore, STAMPI and DAMP consume more memory on long time series such as IOPS. SAND avoids storing the raw data, but its index size scales quadratically with the subsequence length. Because it needs to store a $4l$ by $4l$ matrix for each pattern. During anomaly score computation, it stores the distances of each subsequence to all patterns. Therefore, it requires more memory on the datasets where the subsequence length is long, such as ECG. For Sirloin, the index size consists of the cluster centroids, the PQ codebook, and the PQ codes of the latest subsequences. For each subsequence, the anomaly score computation requires space for three kinds of distances in Eq. (5) for the top-k nearest neighbors. With our experiment settings, the anomaly score computation dominates the memory usage.

8 CONCLUSION

In this paper, we propose Sirloin for solving SSAD problem. First, Sirloin proposes a glance and focus anomaly score function that takes both global and local information into consideration, contributing to an accurate anomaly detection. Second, Sirloin dynamically maintains an IVF index and PQ codebooks to index and compress the subsequences, hence is able to cope with the time series evolution and to process streaming batches efficiently. In addition, a dual index optimization strategy is put forward that further improves the efficiency. Extensive experiments demonstrate that Sirloin is accurate and efficient compared to existing methods. Specifically, compared to the state-of-the-art method SAND in streaming scenarios, Sirloin achieves a 4× improvement in throughput on average and a 58.02% increase in anomaly detection accuracy, demonstrating its outstanding performance.

In future work, we plan to further investigate the storage and I/O efficiency of Sirloin, particularly its behavior on SSDs under high-throughput streaming scenarios.

ACKNOWLEDGMENTS

This work was supported in part by NSFC Grant No. 62372194.

REFERENCES

[1] Ali Abdul-Aziz, Mark R Woike, Nikunj C Oza, Bryan L Matthews, and John D lekki. 2012. Rotor health monitoring combining spin tests and data-driven anomaly detection methods. *Structural Health Monitoring.* 11, 1 (2012), 3–12.

[2] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *KDD.* 3395–3404.

[3] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and scalable subsequence anomaly detection in large data series. *VLDBJ.* 30, 6 (2021), 909–931.

[4] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. *PVLDB.* 13, 11 (2020), 1821–1834.

[5] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *PVLDB.* 14, 10 (2021), 1717–1729.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *SIGMOD Conference.* 93–104.

[7] Yingyi Bu, Oscar Tat-Wing Leung, Ada Wai-Chee Fu, Eamonn J. Keogh, Jian Pei, and Sam Meshkin. 2007. WAT: Finding Top-K Discords in Time Series Database. In *SDM.* 449–454.

[8] International AIOPS Challenges. 2018. https://competition.aiops-challenge.com/home/competition/1484452272200032281

[9] Medina Hadjem, Farid Naït-Abdesselam, and Ashfaq Khokhar. 2016. ST-segment and T-wave anomalies prediction in an ECG data using RUSBoost. In *Healthcom 2016.* 1–6.

[10] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. 2006. Adaptive event detection with time-varying poisson processes. In *KDD.* 207–216.

[11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[12] E. Keogh, T. Dutta Roy, U. Naik, and A. Agrawal. 2021. Multi-dataset Time-Series Anomaly Detection Competition. https://compete.hexagon-ml.com/practice/competition/39/

[13] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *KDD.* 444–452.

[14] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM.* 413–422.

[15] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A. Zuluaga, and Eamonn J. Keogh. 2022. Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams. In *KDD.* 1173–1182.

[16] George B Moody and Roger G Mark. 2001. The impact of the MIT-BIH arrhythmia database. *IEEE engineering in medicine and biology magazine.* 20, 3 (2001), 45–50.

[17] Themis Palpanas. 2024. Time Series Anomaly Detection. *SIGMOD Blog* (2024).

[18] Themis Palpanas and Volker Beckmann. 2019. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGMOD Rec.* 48, 3 (2019), 36–40.

[19] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. 2022. TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection. *PVLDB.* 15, 8 (2022), 1697–1711.

[20] Tuomas Pelkonen, Scott Franklin, Paul Cavallaro, Qi Huang, Justin Meza, Justin Teller, and Kaushik Veeraraghavan. 2015. Gorilla: A Fast, Scalable, In-Memory Time Series Database. *PVLDB.* 8, 12 (2015), 1816–1827.

[21] Dehua Peng, Zhipeng Gui, Dehe Wang, Yuncheng Ma, Zichen Huang, Yu Zhou, and Huayi Wu. 2022. Clustering by measuring local direction centrality for data with heterogeneous density and weak connectivity. *Nature communications.* 13, 1 (2022), 5455.

[22] Ninh Pham and Rasmus Pagh. 2012. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *KDD.* 877–885.

[23] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. *PVLDB.* 15, 9 (2022), 1779–1797.

[24] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *KDD.* 2828–2837.

[25] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. 2006. Online Outlier Detection in Sensor Data Using Non-Parametric Models. *PVLDB.* (2006), 187–198.

[26] Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos E. Trahanias, and Themis Palpanas. 2023. Choose Wisely: An Extensive Evaluation of Model Selection for Anomaly Detection in Time Series. *PVLDB.* 16, 11 (2023), 3418–3432.

[27] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *PVLDB.* 15, 6 (2022), 1201–1214.

[28] Donna Xu, Ivor W. Tsang, and Ying Zhang. 2018. Online Product Quantization. *IEEE Trans. Knowl. Data Eng.* 30, 11 (2018), 2185–2198.

[29] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *ICLR.*

[30] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *ICDM.* 1317–1322.

[31] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. 2018. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Min. Knowl. Discov.* 32, 1 (2018), 83–123.

[32] Aoqian Zhang, Shuqing Deng, Dongping Cui, Ye Yuan, and Guoren Wang. 2023. An Experimental Evaluation of Anomaly Detection in Time Series. *PVLDB.* 17, 3 (2023), 483–496.

[33] Bolong Zheng, Lingfeng Ming, Kai Zeng, Mengtao Zhou, Xinyong Zhang, Tao Ye, Bin Yang, Xiaofang Zhou, and Christian S. Jensen. 2024. Adversarial Graph Neural Network for Multivariate Time Series Anomaly Detection. *IEEE Trans. Knowl. Data Eng.* 36, 12 (2024), 7612–7626.

[34] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth J. Funning, Abdullah Mueen, Philip Brisk, and Eamonn J. Keogh. 2016. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In *ICDM.* 739–748.