Adaptive Temporal Windowing for Streaming Outlier Detection Under Dynamic Arrival Rates

Hend Maher[1], Mohamed Khafagy[2], Heba Nagaty[3]
Faculty of Computers and Artificial Intelligence, Department of Information Systems, Fayoum University, Egypt[1,2,3]
Faculty of Computer Science and Information Systems, Department of Information Systems,
October 6 University, Egypt[1]

***Abstract*—Streaming outlier detection requires adaptive mechanisms capable of handling continuously evolving data streams under dynamic arrival rates. Existing count-based approaches fail under bursty and irregular stream arrival patterns commonly observed in real-world systems, since they trigger model updates after a fixed number of instances without considering temporal dynamics. In this study, we propose an adaptive time-driven windowing framework for streaming outlier detection that decouples model updates from instance count and instead leverages elapsed time as the primary control mechanism. The proposed approach is based on the Density Incremental Local Outlier Factor (DILOF) and introduces a time-aware update strategy aligned with real-world streaming behavior. Extensive experiments on benchmark datasets demonstrate that the proposed method achieves robust and stable detection performance, with AUC values ranging up to 0.96. The results further show that time-based windowing provides a consistent trade-off between detection accuracy and computational efficiency, while offering a temporally grounded update mechanism for streams with variable arrival behavior. In addition, we analyze hybrid count-time strategies and demonstrate their limitations due to dominance effects. Repeated runs further indicate the robustness and consistency of the proposed framework. The findings highlight that temporal awareness is a critical factor in stream outlier detection and should be explicitly incorporated into windowing mechanisms, particularly in resource-constrained environments such as fog and edge computing.**

Keywords*—Streaming outlier detection; time-based windowing; Density Incremental Local Outlier Factor; data streams; anomaly detection; fog computing

I. Introduction

Outlier detection in data streams has become increasingly important in many real-world applications, such as network monitoring, financial fraud analysis, industrial process control, and sensor-based systems. In these scenarios, data is generated continuously and often at high speed, making it impractical to store and reprocess the entire stream [1], [2]. As a result, outlier detection methods that stream must operate incrementally while maintaining both efficiency and detection accuracy.

As illustrated in Fig. 1, outliers are typically characterized by their presence in low-density regions compared to normal data points, which tend to form dense clusters [3]. This observation forms the basis of density-based methods such as LOF, which evaluate the relative density of each data point with respect to its neighbors [4], [5]. Furthermore, the effectiveness of density-based outlier detection is closely related to how the local neighborhood is constructed and updated over time [8], [9]. In streaming environments, where data

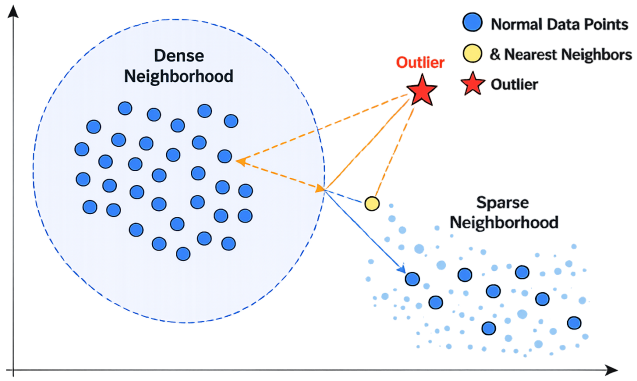

Fig. 1. Conceptual illustration of density-based outlier detection, where normal data points form dense regions while outliers reside in sparse regions with significantly lower local density.

arrives continuously, maintaining an accurate representation of these density regions depends on the chosen windowing mechanism [6]. This motivates the investigation of time-based windowing as a more adaptive approach for capturing the evolving structure of the data stream [10].

A central design issue in outlier detection of the stream is how to manage the active data window used for model updating. Most existing methods rely on count-based windowing, where the system updates its internal representation after processing a fixed number of instances [7], [12]. Although this strategy is widely used, it assumes that the number of arriving instances alone is sufficient to regulate the update process. However, in practice, data streams may exhibit irregular arrival rates, bursts of activity, or periods of relative sparsity. Under such conditions, a purely count-based mechanism may not reflect the actual temporal behavior of the stream.

These observations motivate the study of time-based windowing. Instead of triggering summarization after a fixed number of instances, time-based windowing uses elapsed time as the controlling factor. This allows the update process to become more sensitive to the temporal dynamics of the stream and may improve the trade-off between model freshness and computational cost.

In this study, we investigate the role of time-based windowing in streaming outlier detection using Density Incremental Local Outlier Factor (DILOF) as a baseline framework. We

implement a time-driven variant and evaluate its behavior on benchmark datasets. Our experiments show that the time-based strategy achieves strong and stable performance, while the optimal time window varies across datasets, suggesting that time-window selection should be data-dependent rather than fixed universally.

In addition to the time-based variant, we explore several hybrid strategies that combine count-based and time-based triggers. These include rule-based, score-based, and adaptive formulations. However, our empirical results indicate that simple hybridization mechanisms often become dominated by one trigger, preventing a genuinely balanced behavior. This finding highlights that combining count and time in streaming settings requires more principled formulations than straightforward trigger fusion.

The problem is also relevant in fog computing and edge environments, where data must often be processed locally with limited resources and low latency requirements [14], [15]. In such settings, an effective time-aware summarization policy can be valuable for maintaining timely anomaly detection without excessive computational overhead [17].

The main contributions of this study are summarized as follows:

- We introduce an adaptive time-driven windowing framework that redefines update triggering in streaming outlier detection by explicitly incorporating temporal dynamics.
- We develop a time-based variant of the Density Incremental Local Outlier Factor (DILOF) and evaluate its effectiveness on benchmark datasets with varying arrival behavior.
- We demonstrate that time-based windowing achieves robust and stable detection performance while providing a better balance between responsiveness and computational efficiency.
- We provide both theoretical and empirical analysis of time-based windowing and explain its advantages over conventional count-based mechanisms under dynamic arrival rates.
- We analyze hybrid count-time strategies and reveal their practical limitations due to trigger dominance effects.
- We discuss the applicability of the proposed framework in real-world streaming environments, including IoT, industrial monitoring, fog computing, and edge systems.

Despite extensive research in outlier detection, most existing methods still rely on instance-driven update mechanisms that fail to capture temporal dynamics in real-world data streams. This limitation leads to suboptimal update behavior under irregular arrival patterns, including bursty and sparse regimes. Unlike conventional approaches that merely adjust fixed count thresholds, the proposed framework redefines update triggering through temporal stream behavior, enabling adaptive model refresh under fluctuating data rates. By decoupling summarization from instance count, the proposed time-driven strategy provides a more realistic and practically meaningful mechanism for maintaining detection quality in streaming environments.

II. Related Work

Streaming outlier detection has been extensively studied in the context of real-time data processing, where models must operate incrementally under limited memory and computational constraints [16]. Early approaches focused on distance-based methods [8], [9], [11], which identify anomalies based on their distance from neighboring instances. However, these methods often suffer from scalability limitations in high-volume data streams.

Density-based approaches, particularly the Local Outlier Factor (LOF) [18], have gained significant attention due to their ability to capture local data characteristics. Several extensions have been proposed to adapt LOF to streaming environments, including incremental and memory-efficient variants, such as DILOF [13] and EILOF [23]. These methods improve efficiency by updating local density estimates incrementally while maintaining a sliding window of recent instances.

A key design component in streaming algorithms is the windowing mechanism used to control model updates. Most existing approaches rely on count-based windowing, where summarization is triggered after processing a fixed number of instances. While effective in maintaining bounded memory, this strategy assumes a uniform data arrival rate, which is often unrealistic in real-world scenarios characterized by bursty or irregular data flows.

Recent studies have explored adaptive and hybrid windowing strategies to address this limitation. However, many of these approaches still rely on instance-based triggers or simple combinations of count and time, which may not fully capture the temporal dynamics of streaming data [18],[20], [13].

In addition to LOF-based streaming methods, ensemble-based anomaly detection methods such as Isolation Forest have been widely adopted due to their efficiency and scalability. However, such methods are not specifically designed to incorporate temporal windowing mechanisms in streaming environments, and therefore do not explicitly address the challenges associated with dynamic data arrival patterns.

Despite these advancements, most existing approaches continue to rely on instance-driven update mechanisms, limiting their ability to adapt to temporal variations in real-world data streams. This highlights a critical gap in current research.

Table I summarizes the positioning of the proposed TD-ILOF framework relative to representative outlier detection methods of streaming. Existing approaches have made important progress in improving memory efficiency, incremental processing, and adaptability to evolving streams. However, their update mechanisms are generally driven by instance arrivals, adaptive context rules, or model-specific online updates, rather than by elapsed time as an explicit control variable for summarization. In contrast, TD-ILOF focuses on the role of time-driven windowing in density-based streaming outlier detection and systematically examines its behavior against conventional count-based and hybrid alternatives.

TABLE I. COMPARATIVE POSITIONING OF REPRESENTATIVE STREAMING OUTLIER DETECTION METHODS

Method	Year	Main Idea	Update / Windowing Mechanism	Temporal Adaptation	Key Limitation
DILOF [13]	2018	Density-based local outlier detection with memory-efficient summarization	Incremental LOF updates with instance-driven summarization	Does not explicitly use elapsed time as the primary update trigger	Assumes updates are governed mainly by processed instance counts
ASOD [6]	2024	Adaptive online stream outlier detection using context-aware modeling	Online incremental updates with adaptive context control	Addresses stream evolution and contextual adaptation	Does not focus on time-driven windowing for density-based summarization
EILOF [23]	2025	Efficient incremental LOF approximation for data streams	Updates LOF-related quantities primarily for newly arriving points	Improves online processing efficiency	Does not study time-based summarization as a control mechanism
EiForestASD [24]	2025	Tree-based streaming anomaly detection derived from Isolation Forest	Stream-oriented ensemble update strategy	Designed for efficient online anomaly detection	Not density-based and does not analyze temporal windowing in LOF-style summarization
TD-ILOF (Ours)	**2026**	**Density-based streaming outlier detection with time-driven summarization**	**Time-based, count-based, and hybrid update strategies**	**Explicitly models elapsed time as a primary update trigger**	**Targets the gap of temporal update control in density-based stream detection**

To address this limitation, this work focuses on time-based windowing as a primary mechanism for controlling model updates. By decoupling the update process from instance count, the proposed approach adapts naturally to variations in data arrival rates. To the best of our knowledge, a systematic empirical and theoretical evaluation of time-driven windowing in the context of density-based streaming outlier detection remains limited.

Unlike event-time processing frameworks, which organize stream computation according to timestamps attached to arriving records, the present work focuses on controlling the summarization schedule of a density-based outlier detector through elapsed processing time. Similarly, existing adaptive windowing studies usually aim to revise window boundaries or update policies at the stream-system level, whereas TD-DILOF targets the internal refresh mechanism of DILOF itself. Therefore, the contribution of this work is not a generic time-windowing replacement, but a detector-level temporal control strategy for density-preserving streaming summarization.

This study aims to bridge this gap by analyzing the effectiveness of time-based windowing and comparing it with conventional count-based and hybrid strategies.

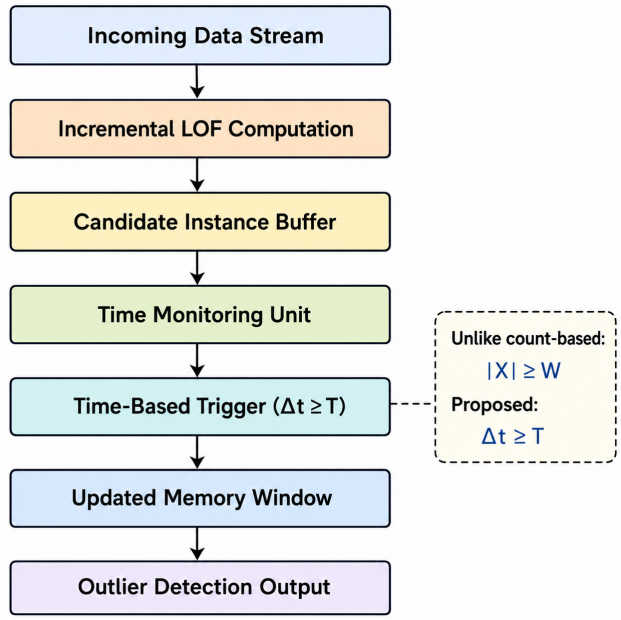

Fig. 2. Framework of the proposed time-based streaming outlier detection approach. Incoming data is processed incrementally and summarized based on elapsed time.

III. METHODOLOGY

In this section, we describe the baseline streaming outlier detection method and the proposed time-based windowing strategy.

A. Background on LOF-Based Methods

Streaming outlier detection aims to assign an anomaly score to each incoming data point in a continuous data stream $P = \{p_1, p_2, \dots\}$ [19], [22]. This process must operate under strict constraints, including limited memory capacity, real-time processing requirements, and evolving data distributions [26], [25].

A widely adopted technique for outlier detection is the Local Outlier Factor (LOF) algorithm, which evaluates the degree of abnormality of a data point based on its local density relative to its neighbors.

Let k denote the number of nearest neighbors, $dist(x, y)$ the Euclidean distance between two points, and $N_k(x)$ the set of k nearest neighbors of x. The computation of LOF is defined as follows.

Definition 1. The reachability distance is given by:

$$reach\_dist_k(x, y) = \max\{dist(x, y), dist_k(y)\} \quad (1)$$

Definition 2. The local reachability density is:

$$lrd_k(x) = \left(\frac{1}{k} \sum_{y \in N_k(x)} reach\_dist_k(x, y) \right)^{-1} \quad (2)$$

Definition 3. The LOF score is:

$$LOF_k(x) = \frac{1}{k} \sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)} \quad (3)$$

A data point is considered an outlier if its LOF score exceeds a predefined threshold.

Several extensions have been proposed to adapt LOF to streaming environments. The incremental LOF (iLOF) processes incoming data sequentially but requires storing all instances, which limits scalability. MiLOF addresses this limitation through clustering-based summarization but may lose important density information. More recently, DILOF improves upon these approaches by preserving density structures during summarization, making it a suitable baseline for streaming outlier detection. As shown in Fig. 2, the proposed approach replaces the traditional count-based trigger with a time-based mechanism.

B. Baseline: Density Incremental Local Outlier Factor

Density Incremental Local Outlier Factor (DILOF) is an incremental extension of the Local Outlier Factor (LOF) algorithm designed for streaming data. Instead of processing the entire dataset at once, DILOF updates the outlier scores incrementally as new instances arrive.

The method maintains a sliding window of recent data points and computes the local density of each instance based on its k-nearest neighbors. The LOF score is then used to determine whether a point is an outlier relative to its local neighborhood.

In the standard configuration, DILOF relies on a count-based windowing mechanism. A fixed window size W is used, and once the number of instances in the window reaches W, a summarization process is triggered. During summarization, a subset of representative instances is retained while less informative instances are removed to control memory usage and computational cost.

C. Count-Based Windowing

In the count-based approach, summarization is triggered when the number of processed instances reaches a predefined threshold W. Formally, let N denote the number of instances currently stored in the window. The summarization condition is defined as:

$$N \geq W \quad (4)$$

This mechanism ensures a bounded memory footprint but does not consider the temporal properties of the data stream. As a result, the update frequency depends solely on the number of arriving instances, which may lead to suboptimal behavior when the data arrival rate varies over time.

D. Proposed Time-Driven Density Incremental Local Outlier Factor

To address the limitations of count-based windowing, we propose a time-based mechanism in which summarization is triggered according to elapsed time rather than instance count.

Let T denote the predefined time window in milliseconds, and let Δt represent the elapsed time since the last summarization. The summarization condition is defined as:

$$\Delta t \geq T \quad (5)$$

In this approach, each incoming instance is processed incrementally, and a timer is used to track the elapsed time. Once the time threshold is reached, the summarization procedure is invoked, and the timer is reset. The distinction is therefore deeper than replacing the condition $N \geq W$ with $\Delta t \geq T$. In the proposed framework, the temporal trigger changes the rhythm of memory renewal and directly affects how many instances participate in each summarization cycle. This alters the interaction between stream arrival behavior, density preservation, and computational cost, which is not captured by a simple trigger substitution view. This mechanism allows the update frequency to adapt naturally to the temporal dynamics of the stream. In high-rate periods, more instances are processed within the same time interval, while in low-rate periods, fewer updates occur. This behavior provides a better balance between responsiveness and computational efficiency.

E. Hybrid Windowing Strategies

In addition to the time-based approach, we explored hybrid strategies that combine count-based and time-based triggers. These strategies include rule-based (logical OR), score-based combinations, and adaptive selection mechanisms.

However, our empirical evaluation shows that these naive hybrid formulations tend to be dominated by a single trigger, either count or time, depending on the configuration. As a result, they fail to provide a truly balanced integration of both mechanisms. This observation suggests that more principled approaches are required for effective hybrid windowing in streaming settings.

F. Theoretical Analysis of Time-Based Windowing

While time-based windowing provides an intuitive alternative to count-based mechanisms, it is important to analyze its behavior from a theoretical perspective in order to understand its impact on streaming outlier detection.

In count-based windowing, the update frequency is determined by the number of incoming instances. Assuming a fixed window size W, the expected update interval depends directly on the data arrival rate $r(t)$. When the arrival rate varies over time, the effective update interval becomes unstable, leading to inconsistent model refresh behavior.

In contrast, time-based windowing decouples the update mechanism from the instance count and instead relies on a fixed temporal interval T. Let N_T denote the number of instances observed within a time window T. Then:

$$N_T = \int_{t_0}^{t_0+T} r(t)\, dt \tag{6}$$

This formulation shows that the number of processed instances becomes a function of the arrival rate rather than a fixed constraint. As a result, the model adapts naturally to the dynamics of the data stream.

From a density estimation perspective, this behavior has important implications. During high-rate periods, the model receives more data within the same time window, leading to more accurate local density estimation. Conversely, during low-rate periods, fewer updates occur, reducing unnecessary computation while preserving temporal consistency.

G. Computational Complexity Analysis

The computational complexity of the proposed approach is influenced by both the LOF computation and the summarization process.

For each incoming instance, the LOF computation requires identifying the k-nearest neighbors, which typically has a complexity of $O(k \log n)$ depending on the data structure used.

In the count-based approach, summarization is triggered every W instances, resulting in a total of $O(N/W)$ summarization operations for a stream of size N.

In the time-based approach, the number of summarization operations depends on the total elapsed time rather than the number of instances. Let T_{total} denote the total processing time. Then the number of summarizations is approximately:

$$\frac{T_{total}}{T} \tag{7}$$

This leads to a more predictable and stable update frequency, particularly in environments with fluctuating data rates.

Moreover, as observed in the experimental results, increasing the time window reduces the number of summarization operations, which directly impacts execution time and computational cost.

H. Design Rationale and Trade-offs

The choice of time-based windowing introduces a fundamental trade-off between detection accuracy, responsiveness, and computational efficiency.

Smaller time windows result in more frequent updates, improving model freshness but increasing computational overhead. Larger time windows reduce the update frequency, improving efficiency but potentially delaying the detection of new anomalies.

This trade-off can be interpreted as a balance between temporal resolution and computational cost. Unlike count-based methods, which assume uniform data arrival, time-based windowing provides a mechanism that aligns more closely with real-world streaming conditions.

These characteristics make time-based windowing particularly suitable for deployment in resource-constrained environments such as fog and edge computing, where both latency and computational efficiency are critical.

1) Adaptive time window selection strategy: The experimental results indicate that the effectiveness of time-based windowing is influenced by the choice of the time window T, which appears to be dataset-dependent. While fixed time-window configurations provide stable performance, selecting an appropriate value of T remains a practical challenge, particularly in streaming environments with varying data arrival patterns.

To address this limitation, we outline a potential adaptive time window selection strategy that dynamically adjusts the window size based on the observed arrival rate of the data stream.

a) Motivation: In real-world streaming scenarios, the data arrival rate may fluctuate over time due to bursts, delays, or irregular generation processes. Under such conditions, a fixed time window may lead to inconsistent behavior. During high-rate periods, a fixed T may accumulate a large number of instances before summarization, increasing computational overhead. Conversely, during low-rate periods, the same window may contain too few instances, reducing the reliability of density estimation.

These observations suggest that adjusting the time window according to the temporal characteristics of the stream may improve both efficiency and stability.

b) Adaptive formulation: Let $r(t)$ denote the estimated data arrival rate at time t, defined as the number of instances observed per unit time. The adaptive time window $T(t)$ can be defined as:

$$T(t) = \alpha \cdot \frac{1}{r(t)} \tag{8}$$

where, α is a scaling factor controlling the balance between update frequency and computational cost.

Under this formulation, higher arrival rates lead to smaller time windows, resulting in more frequent updates. In contrast, lower arrival rates produce larger windows, reducing unnecessary computations. This behavior helps maintain a more consistent effective window content over time.

c) Integration with the framework: The adaptive mechanism can be incorporated into the proposed time-based framework by updating the time window prior to each summarization step:

```
if adaptive_mode:
    estimate r(t) over recent interval
    update T(t) = alpha / r(t)
```

```
if Delta_t >= T(t):
    perform summarization
    reset timer
```

In practice, the arrival rate $r(t)$ can be estimated using a sliding time interval or an exponential moving average to reduce sensitivity to short-term fluctuations.

d) Discussion: Although this adaptive strategy is not experimentally evaluated in the current study, it represents a natural extension of the proposed time-based windowing approach. It provides a principled direction for reducing sensitivity to dataset-specific configurations and may improve robustness in real-world streaming environments. Future work will focus on evaluating this strategy under real-time conditions and exploring more advanced adaptive formulations. Algorithm 1 presents the Time-Based Density Incremental Local Outlier Factor.

Algorithm 1 Time-Based Density Incremental Local Outlier Factor

Require: Infinite data stream $P = \{p_1, p_2, \dots, p_t, \dots\}$, time window T, LOF threshold θ, step size η, regularization constant λ, maximum number of iterations I
Ensure: Set of detected outliers O

$X \leftarrow \emptyset$ ▷ Data points in memory
$O \leftarrow \emptyset$ ▷ Detected outliers
startTime $\leftarrow$ *currentTime*()
for each $p_t \in P$ **do**
  $LOF_k(p_t) \leftarrow LOD(p_t, O, \theta)$
  if $LOF_k(p_t) > 0$ **then**
    $X \leftarrow X \cup \{p_t\}$
    $\Delta t \leftarrow$ *currentTime*() $-$ *startTime*
    if $\Delta t \geq T$ **then**
      $Z \leftarrow NDS(X, \eta, \lambda, I)$
      Remove the oldest $|X|/2$ data points from X
      $X \leftarrow X \cup Z$
      startTime $\leftarrow$ *currentTime*()
    end if
  end if
end for

IV. Experimental Setup

In this section, we describe the experimental configuration used to evaluate the proposed time-based windowing strategy.

A. Datasets

To assess the effectiveness of the proposed approach, we conducted experiments on two benchmark datasets commonly used in outlier detection research. These datasets differ in size, dimensionality, and anomaly construction, providing a suitable basis for evaluating the proposed method under different structural and computational conditions.

All datasets were processed in a simulated streaming environment, where instances were fed sequentially to emulate data arrival over time. No preprocessing was applied other than feature normalization, ensuring a fair and consistent evaluation across all methods.

TABLE II. Characteristics of the Benchmark Datasets used in the Experiments.

Characteristic	UCI Vowel	UCI Pendigit
Instances	1,456	3,498
Dimensions	12	16
Classes	11	10

Table II summarizes the main characteristics of the datasets used in the experiments.

The UCI Vowel and UCI Pendigit datasets were selected because they are commonly used benchmarks for evaluating density-based outlier detection methods. They also differ in size and dimensionality, allowing the proposed time-driven update policy to be examined under different data characteristics. In the Vowel dataset, one class was treated as the outlier class. In the Pendigit dataset, uniformly distributed noise was injected to simulate anomalous observations.

B. Evaluation Metric

The performance of the proposed method is evaluated using the Area Under the Receiver Operating Characteristic Curve (AUC). AUC is a widely used metric for outlier detection as it captures the trade-off between true positive rate (TPR) and false positive rate (FPR) across different threshold settings. Higher AUC values indicate better discrimination between normal instances and outliers.

C. Parameter Settings

For the time-based windowing strategy, different time windows ranging from 100 ms to 200 ms were evaluated. This range was selected to capture both frequent-update and moderate-update regimes in the simulated streaming environment, enabling analysis of the trade-off between responsiveness and computational overhead. The selected values provide a practical sensitivity analysis rather than assuming a universally optimal temporal configuration.

D. Implementation Details

All experiments were performed using an incremental streaming setup, where instances are processed sequentially in the order of arrival. The summarization process is triggered based on the selected windowing mechanism, either count-based or time-based.

For the time-based approach, a timer is used to measure the elapsed time between successive summarization operations. Once the predefined time threshold is reached, summarization is executed and the timer is reset. No artificial delays were introduced during the experiments, ensuring that the results reflect the natural processing time of the system. All experiments were conducted on a standard computing environment. Accordingly, the reported timing behavior should be interpreted as processing-time-triggered summarization within a controlled experimental environment, rather than as a full event-time implementation for distributed stream-processing systems.

E. Comparison Strategy

To evaluate the effectiveness of the proposed approach, we compare the time-driven windowing strategy with the traditional count-based mechanism and representative LOF-based streaming baselines. This comparison highlights the impact of incorporating temporal information into the summarization process. Since the proposed method is an extension of the DILOF framework, the evaluation focuses primarily on methodologically consistent baselines to ensure fair comparison. In addition, hybrid strategies combining count and time triggers are discussed qualitatively due to their unstable behavior observed during the experiments.

V. Results and Discussion

This section presents a comprehensive evaluation of the proposed time-based windowing strategy and analyzes its impact on streaming outlier detection performance under different configurations. To assess the robustness of the proposed framework, the experiments were repeated multiple times under the same parameter settings. The observed AUC values showed low variation across runs, indicating that the proposed method provides stable behavior rather than isolated performance gains. This consistency is particularly important in streaming settings, where irregular arrival patterns may otherwise introduce substantial variability in update behavior and anomaly scores.

A. Performance on Benchmark Datasets

Table III summarizes the AUC values obtained using different time-window configurations across the evaluated datasets. The results demonstrate that the proposed time-based strategy achieves consistently strong and stable performance.

For the Vowel dataset, the AUC values remain high across all configurations, ranging from 0.9428 to 0.9482. The best performance is achieved at a time window of 160 ms, indicating that moderate update intervals provide an effective balance between model responsiveness and stability.

In contrast, the Pendigit dataset exhibits a gradual improvement in performance as the time window increases. The AUC rises from 0.9384 at 100 ms to 0.9637 at 200 ms, suggesting that this dataset benefits from less frequent summarization and longer temporal aggregation.

These findings indicate that while time-based windowing is effective across different datasets, the optimal time window is data-dependent and influenced by the characteristics of the underlying data stream.

B. Comparison with Baseline Methods

To further assess the effectiveness of the proposed approach, we compare it with several baseline methods, including iLOF, MILOF, and the original count-based DILOF.

Fig. 3 and Fig. 4 present the performance comparison across different window configurations for the Vowel and Pendigit datasets, respectively.

On the Vowel dataset, the proposed TD-DILOF consistently achieves the highest AUC values across all window sizes. The iLOF baseline remains nearly constant around 0.90, indicating limited adaptability to changing window configurations. While MILOF [21] and count-based DILOF exhibit gradual improvements as the window size increases, their performance remains noticeably below that of the proposed approach.

Similarly, on the Pendigit dataset, the TD-DILOF demonstrates strong and stable performance across all configurations. Although count-based DILOF achieves competitive results at larger window sizes, it remains slightly inferior to the time-based variant, which consistently maintains higher AUC values.

Notably, the performance advantage of the proposed method is more pronounced at smaller window sizes. This suggests that time-based windowing is particularly effective in scenarios requiring frequent updates, where count-based approaches may suffer from insufficient or irregular sampling.

Overall, these results confirm that incorporating temporal dynamics into the windowing mechanism significantly enhances the robustness and effectiveness of streaming outlier detection compared to traditional count-based methods.

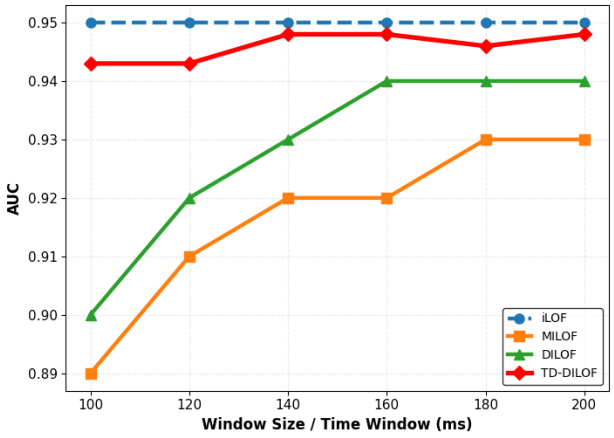

Fig. 3. AUC comparison of the baseline methods and the proposed TD-DILOF on the Vowel dataset.

C. Effect of Time Window on Summarization Behavior

In addition to detection accuracy, we analyze the impact of the time window on summarization behavior and computational cost.

The results show that increasing the time window reduces the number of summarization operations. This behavior is expected, as larger time windows delay the triggering of the summarization process, thereby reducing update frequency.

This behavior highlights a key advantage of time-based windowing, where update frequency is governed by temporal dynamics rather than instance count, enabling more efficient processing in streaming environments.

D. Discussion of Hybrid Strategies

We also investigated hybrid windowing strategies that combine count-based and time-based triggers, including rule-based, score-based, and adaptive formulations.

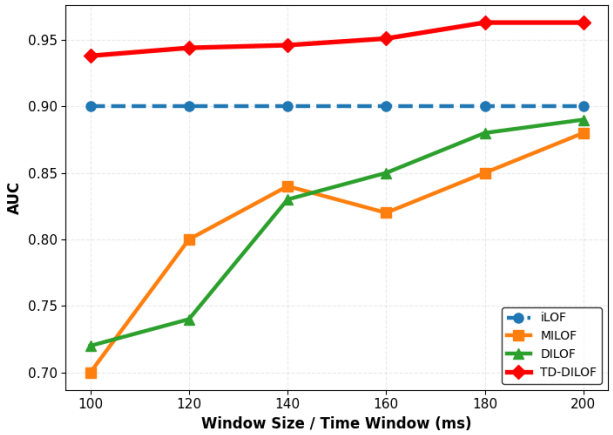

Fig. 4. AUC comparison of the baseline methods and the proposed TD-DILOF on the Pendigit dataset.

However, the experimental results indicate that these hybrid approaches fail to achieve a balanced integration of both mechanisms. In most cases, the hybrid strategy becomes dominated by either the count-based or the time-based trigger, depending on the configuration.

This behavior suggests that simple fusion of count and wall-clock time is insufficient for robust hybridization. More advanced designs, such as event-time-based windowing or adaptive mechanisms that dynamically adjust to stream characteristics, may be required to fully exploit the benefits of hybrid approaches.

Overall, the proposed time-based windowing strategy emerges as the most effective and stable configuration among all evaluated methods, providing a strong balance between accuracy, robustness, and computational efficiency.

TABLE III. AUC Performance Under Different Time-Window Configurations.

Time Window (ms)	Vowel Dataset	Pendigit Dataset
100	0.9429	0.9384
120	0.9428	0.9444
140	0.9481	0.9463
160	**0.9482**	0.9510
180	0.9464	0.9630
200	0.9480	**0.9637**

E. Practical Applications

The proposed framework is particularly suitable for real-world streaming scenarios in which data arrival rates are naturally irregular and bursty. Examples include IoT sensor monitoring, industrial fault detection, network intrusion detection, financial transaction surveillance, and edge-based anomaly detection. In such environments, time-driven summarization provides a more realistic control mechanism than fixed count-based updates, allowing the model to maintain temporal consistency while avoiding unnecessary computational overhead.

VI. Limitations

Despite the promising results, this study has several limitations. First, the experiments were conducted in a simulated streaming environment that approximates real-time data processing conditions. Although this setup enables controlled evaluation of the proposed framework, further validation in fully real-time deployment scenarios would provide additional practical insights.

Second, the time measurement used in this study is based on system processing time rather than event time. Consequently, the behavior of the proposed method may vary across hardware settings and deployment environments.

Third, the hybrid windowing strategies explored in this work rely on relatively simple formulations. These approaches were not sufficient to achieve a balanced integration between count-based and time-based triggers, indicating the need for more advanced designs.

These limitations open promising directions for future research, particularly for real-time deployment, event-time modeling, and evaluation on larger and more diverse streaming datasets.

VII. Conclusion

This study demonstrates that temporal awareness is a critical yet underexplored factor in streaming outlier detection. Using Density Incremental Local Outlier Factor (DILOF) as a baseline framework, we investigated a time-driven windowing strategy in which model summarization is triggered by elapsed time rather than by a fixed number of arriving instances.

The experimental results show that the proposed TD-DILOF framework consistently achieves strong and stable detection performance across different benchmark datasets and time-window configurations. The method produces high AUC values while also revealing that the optimal time window depends on the characteristics of the underlying data stream. This finding emphasizes that update triggering in streaming systems should account for temporal behavior rather than relying solely on instance count.

From a computational perspective, the proposed framework provides an improved balance between detection accuracy and efficiency by reducing unnecessary summarization operations under varying arrival rates. In addition, the analysis of hybrid strategies shows that simple combinations of count-based and time-based triggers are often dominated by one mechanism, limiting their practical effectiveness.

Overall, the findings suggest that time-driven windowing is not only beneficial but also practically meaningful for robust streaming outlier detection, particularly in real-world environments such as IoT monitoring, industrial systems, network security, and edge computing, where data arrival behavior is inherently dynamic and temporally irregular.

VIII. Future Work

Several directions can extend the present work. First, adaptive time window selection strategies should be investigated to dynamically adjust the window size according to stream

characteristics such as arrival rate, density variation, and concept drift. Such strategies may reduce sensitivity to dataset-specific configurations and improve robustness in practical deployments.

Second, more principled hybrid windowing mechanisms can be developed to integrate count-based and time-based information without suffering from trigger dominance. This may require dynamic control rules, event-time formulations, or learning-based update policies.

Third, replacing system-time processing with event-time-based windowing represents a promising direction, especially in distributed and real-time environments where temporal consistency is essential.

Finally, future studies should validate the proposed framework on a wider range of benchmark and real-world datasets, and compare it with additional state-of-the-art streaming anomaly detection methods to further assess scalability, generalizability, and deployment readiness.

REFERENCES

[1] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos, "A survey on the evolution of stream processing systems," *The VLDB Journal*, 2024.

[2] S. Anjum, S. K. Yadav, and S. Yadav, "Stream data model and architecture," in *Machine Learning: Navigating the Big Data Era*, Springer, 2024.

[3] R. Hu and Y. Wang, "An efficient outlier detection algorithm for data streaming," *arXiv preprint arXiv:2501.01061*, Jan. 2025.

[4] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, "A review of local outlier factor algorithms for outlier detection in big data streams," *Big Data and Cognitive Computing*, vol. 4, no. 3, p. 20, 2020.

[5] M. Sabha and B. Tugrul, "Stream data analysis and processing frameworks for detecting outliers in human activities: A review," *WSEAS Transactions on Information Science and Applications*, 2025.

[6] Z. Hu *et al.*, "ASOD: An adaptive stream outlier detection method using online strategy," *Journal of Cloud Computing*, 2024.

[7] I. Altaf and M. A. Chachoo, "Advances in density-based outlier detection algorithms: Exploration of LOF with experimental analysis," *Applied Soft Computing*, May 2025.

[8] L. Tran, L. Fan, and C. Shahab, "Distance-based outlier detection in data streams," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, 2016.

[9] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proc. ACM CIKM*, 2007, pp. 811–820.

[10] L. Tran, M. Y. Mun, and C. Shahabi, "Real-time distance-based outlier detection in data streams," 2021.

[11] L. Cao *et al.*, "Scalable distance-based outlier detection over high-volume data streams," in *Proc. IEEE ICDE*, 2014, pp. 76–87.

[12] A. Gounaris and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in *Proc. IEEE ICDE*, 2011, pp. 135–146.

[13] G. S. Na, D. Kim, and H. Yu, "DILOF: Effective and memory efficient local outlier detection in data streams," in *Proc. ACM SIGKDD*, 2018.

[14] F. Bonomi *et al.*, "Fog computing and its role in the IoT," in *Proc. MCC Workshop*, 2012.

[15] A. Ahmed and S. H. Mohammad, "A survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[16] S. Yi *et al.*, "Fog computing: Platform and applications," in *Proc. IEEE HotWeb*, 2015, pp. 73–78.

[17] S. Khan *et al.*, "A comprehensive survey on multifaceted fog computing resource management techniques, trends, applications and future directions," 2025.

[18] M. M. Breunig *et al.*, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD*, 2000.

[19] M. Hassaan *et al.*, "A fast and efficient algorithm for outlier detection over data streams," *International Journal of Advanced Computer Science and Applications*, 2021.

[20] D. Pokrajac *et al.*, "Incremental local outlier detection for data streams," in *IEEE CIDM*, 2007, pp. 504–515.

[21] M. Salehi *et al.*, "Fast memory efficient local outlier detection in data streams," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[22] A. Nilofer and S. Sasikala, "Enhanced hybrid filter based feature selection models for high dimensional streaming data," *IAENG International Journal of Computer Science*, vol. 52, no. 12, pp. 4678–4691, 2025.

[23] R. Hu *et al.*, "EILOF: An efficient incremental local outlier factor algorithm for data streaming," *arXiv preprint arXiv:2501.01061*, 2025.

[24] K. Moeenfar *et al.*, "EiForestASD: A fast and accurate tree-based approach for anomaly detection in streaming data," *International Journal of Data Science and Analytics*, 2025.

[25] A Duraj *et al.*, " Detection of anomalies in data streams using the LSTM-CNN model" *Sensors, mdpi*, 2025.

[26] QT Tran *et al.*, "Concept drift detection in image data stream: a survey on current literature, limitations and future directions" *Artificial Intelligence Review, Springer*, 2026.