$\mathcal{IDK}$-$\mathcal{S}$: Incremental Distributional Kernel for Streaming Anomaly Detection

Yang Xu[1,2], **Yixiao Ma**[1,2], **Kaifeng Zhang**[1,2], **Zuliang Yang**[1,2], **Kai Ming Ting**[1,2*]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2] School of Artificial Intelligence, Nanjing University, Nanjing, China
{xuyang, mayx, zhangkf2022, yangzl}@lamda.nju.edu.cn, tingkm@nju.edu.cn

Abstract

Anomaly detection on data streams presents significant challenges, requiring methods to maintain high detection accuracy among evolving distributions while ensuring real-time efficiency. Here we introduce $\mathcal{IDK}$-$\mathcal{S}$, a novel **I**ncremental **D**istributional **K**ernel for **S**treaming anomaly detection that effectively addresses these challenges by creating a new dynamic representation in the kernel mean embedding framework. The superiority of $\mathcal{IDK}$-$\mathcal{S}$ is attributed to two key innovations. First, it inherits the strengths of the Isolation Distributional Kernel, an offline detector that has demonstrated significant performance advantages over foundational methods like Isolation Forest and Local Outlier Factor due to the use of a data-dependent kernel. Second, it adopts a lightweight incremental update mechanism that significantly reduces computational overhead compared to the naive baseline strategy of performing a full model retraining. This is achieved without compromising detection accuracy, a claim supported by its statistical equivalence to the full retrained model. Our extensive experiments on thirteen benchmarks demonstrate that $\mathcal{IDK}$-$\mathcal{S}$ achieves superior detection accuracy while operating substantially faster, in many cases by an order of magnitude, than existing state-of-the-art methods.

Introduction

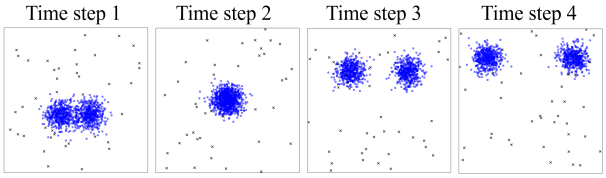
Anomaly detection, also known as outlier detection, is an essential task in various domains, identifying unusual patterns that do not conform to expected behavior (Chandola, Banerjee, and Kumar 2009). This technique is widely applied in many areas, such as financial fraud detection (Hilal, Gadsden, and Yawney 2022), network security (Ahmed, Mahmood, and Hu 2016), and industrial fault monitoring (Stojanovic et al. 2016), where anomalous behaviors can signify important and often costly events.

With the exponential growth of data generation, many modern applications in anomaly detection require processing data not as static datasets, but as continuous data streams. This is often referred to as online anomaly detection which imposes strict constraints: methods must process each data instance upon arrival with limited memory and must be highly adaptable to evolving data distributions, a phenomenon known as concept drift (Cao et al. 2025). In such dynamic environments, offline anomaly detection solutions are rendered ineffective, as their static models quickly become obsolete and fail to represent the current data normality. Although a straightforward approach is to continuously retrain the model as new data emerge, this strategy incurs substantial computational overhead and latency, making it impractical for high-throughput, real-time applications.

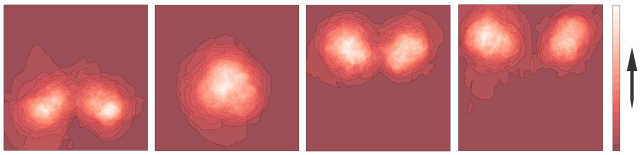
The prevailing strategy for developing streaming anomaly detectors is to adapt an established offline method, modifying its model to handle data incrementally and thus avoid the repeated high cost of reconstructing a full model (Lu, Wang, and Zhao 2023). Existing state-of-the-art methods are built upon a few foundational offline detectors, including Isolation Forest (iForest) (Liu, Ting, and Zhou 2008), Local Outlier Factor (LOF) (Breunig et al. 2000), and k-Nearest Neighbors (kNN) detector (Ramaswamy, Rastogi, and Shim 2000). Recent prominent examples of this trend include online Isolation Forest (oIFOR) (Leveni et al. 2024) and streaming Isolation Forest (SiForest) (Liu et al. 2025), both of which extend iForest to the streaming context. For example, oIFOR implements an efficient incremental update by modeling the data distribution with an ensemble of dynamic, multi-resolution histograms, which adapt their structure to learn and forget instances from a sliding buffer. This adaptation-based design is popular as it leverages the well-understood properties of established offline methods. However, an often overlooked consequence of this design is that the effectiveness of such a streaming algorithm depends on the capabilities of its base detector. This means that not only the strengths of its base detector are inherited, but so are its weaknesses. For example, streaming methods built upon iForest are likely to inherit its weaknesses with varied densities and local anomalies (Aryal et al. 2014; Bandaragoda et al. 2014), while LOF-based methods are typically sensitive to its k parameter setting in different data sizes (Ting et al. 2021a). These inherent weaknesses, inherited from their offline counterparts, create a performance ceiling that current streaming detectors struggle to break.

Isolation Distributional Kernel (IDK) (Ting et al. 2021b) is an anomaly detector based on kernel mean embedding (Muandet et al. 2017). IDK maps a given static dataset to a single point in a Hilbert space, which represents the data distribution of the dataset. To score any test instance, its similarity w.r.t. the data distribution of the entire dataset

*Corresponding author.



(a) Data stream over four time steps.



(b) The distribution of normal scores over four time steps.

Figure 1: An illustration of $\mathcal{IDK}$-$\mathcal{S}$'s adaptivity to concept drift. (a) A data stream where the normal distribution (blue clusters) shifts over four time steps. (b) The corresponding normal score distribution estimated by $\mathcal{IDK}$-$\mathcal{S}$. The heatmap (brighter areas indicate higher normal scores, i.e., higher normality) dynamically follows the evolving normality, demonstrating the model's ability to maintain accurate detection in a non-stationary online environment.

is computed as a dot product in Hilbert space. The instances having low similarity are identified as anomalies. The superiority of IDK over other detectors derives from its use of a data-dependent kernel (Ting et al. 2021b). Specifically, the data-dependent property of the kernel denotes that any two instances, with equal inter-instance distance, are more similar when they are located in a sparse data region than in a dense one. This characteristic is crucial for successful anomaly detection in datasets having varied densities and local anomalies, a known challenge for many traditional detectors (Boukerche, Zheng, and Alfandi 2020). Furthermore, its use of an exact, finite-dimensional feature map ensures highly efficient, linear-time computation. Empirical studies have shown that IDK exhibits significantly superior performance compared to traditional offline detectors such as iForest, LOF, and kNN-based methods (Ting et al. 2021a). In a recent survey, the retraining-based IDK model is found to have superior performance to state-of-the-art methods for streaming anomaly detection (Cao et al. 2025). However, this retraining-based model requires a high time cost, making it impractical for potentially infinite data streams.

The demonstrated superiority of the IDK as an offline detector motivates our work to adapt this model for streaming anomaly detection. Here we propose $\mathcal{IDK}$-$\mathcal{S}$, a novel **I**ncremental **D**istributional **K**ernel for **S**treaming anomaly detection. $\mathcal{IDK}$-$\mathcal{S}$ uses a simple yet effective incremental update mechanism that adapts incrementally to newly arriving data. The retraining-based IDK (Cao et al. 2025) must rebuild the entire set of data-dependent partitions in IDK that creates the data-dependent property. Instead, our method updates the data-dependent partitions by continuously replacing the obsolete partitions with new ones generated from new arriving data, ensuring that the model remains aligned with the latest data distribution. Figure 1 illustrates the adaptive capability of our method with a toy streaming data (the `TwoCluster` dataset used in the experiments section). Normal instances are depicted as dense blue clusters of instances, while anomalies are depicted as sparsely distributed instances. As the distribution of normal instances changes over time (i.e., concept drift), our proposed method effectively updates its estimated scores for the evolving data stream, as shown on the heatmaps.

In short, we highlight a critical yet often overlooked fact: *the performance ceiling of each existing streaming detector is fundamentally constrained by its offline counterpart.* To break through this ceiling, a more powerful base detector is essential. Motivated by this, our work makes the following contributions:

- Proposing $\mathcal{IDK}$-$\mathcal{S}$ that adapts the superior IDK to evolving data streams via a new incremental partition-update mechanism; thus addressing the performance limitations of methods based on weaker offline detectors.
- Demonstrating that $\mathcal{IDK}$-$\mathcal{S}$ reduces the theoretical time complexity from $\mathcal{O}(\omega\psi t)$ of the retraining-based IDK model to $\mathcal{O}(l\psi t)$, where ω and l are the data window size and the update step size respectively, and $l \ll \omega$.
- Showing the superiority of $\mathcal{IDK}$-$\mathcal{S}$ for both high detection accuracy and efficiency compared to state-of-the-art methods through experiments on thirteen benchmarks, including stationary and non-stationary distributions.

Preliminaries

Problem Formulation

We consider the streaming anomaly detection task in a potentially infinite multivariate data stream. A data stream $\mathcal{S}$ can be defined as a sequence of instances $x_T \in \mathbb{R}^d$:

$$\mathcal{S} = \{x_1, x_2, \ldots, x_T, \ldots\}, \tag{1}$$

where $T > 1$ and x_T is an instance generated from an underlying data distribution $\mathcal{P}_T$ at time instant T, i.e., $x_T \sim \mathcal{P}_T$. In a streaming context with concept drift, this distribution can evolve over time, i.e., $\mathcal{P}_T$ may differ from $\mathcal{P}_{T+1}$. In line with the existing online learning setting, we assume that the stream is processed in sliding windows. At any time step i, we have access to a finite data window $\mathbf{X}_i \subset \mathcal{S}$ of size ω, which contains the most recent instances from the stream. An online detector must score each instance in $\mathbf{X}_i$, and its model is updated with $\mathbf{X}_i$.

Let $\mathcal{P}_N$ denote the distribution of normal instances and $\mathcal{P}_A$ denote the distribution of anomalies. Given the data stream $\mathcal{S}$ and any instance $x \in \mathcal{S}$, the goal is to learn a model $\mathcal{F}_i(x)$ to assign an anomaly score (or normal score) to x, which identifies whether $x \sim \mathcal{P}_N$ or $x \sim \mathcal{P}_A$. Following established studies on anomaly detection (Liu, Ting, and Zhou 2008; Leveni et al. 2024), we assume that anomalies are characterized by two properties: "few" and "different". For any instance $x \in \mathcal{S}$,

(i) The "few" property implies that the probability that x is generated by the anomaly distribution $\mathcal{P}_A$ is much lower than that by the normal distribution $\mathcal{P}_N$, i.e., for $x \in \mathcal{S}$,

$$P(x \sim \mathcal{P}_A) \ll P(x \sim \mathcal{P}_N). \tag{2}$$

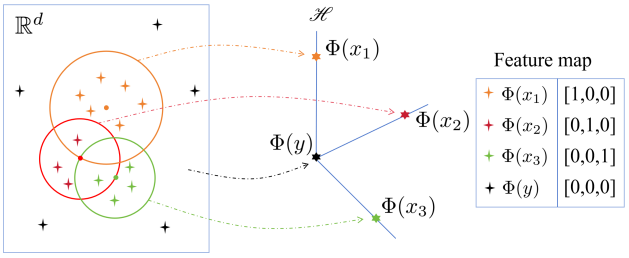


Figure 2: An illustration of the feature map Φ of IK in a Hilbert space $\mathscr{H}$ with one partitioning ($t = 1$) of three hyperspheres ($\psi = 3$), each centred at a blue dot that is randomly selected from the given dataset D. When a point $x \in D$ falls into an overlapping region, it is regarded to be in the hypersphere whose centre is closer to x.

(ii) The "different" property suggests that an anomalous instance x_a will exhibit low similarity to the population of normal instances. More formally, its expected similarity to the normal data distribution $\mathcal{P}_N$ is significantly lower than that of a normal instance x_n, i.e.,

$$E_{y\sim\mathcal{P}_N}[\kappa(x_a, y)] \ll E_{y\sim\mathcal{P}_N}[\kappa(x_n, y)], \tag{3}$$

where κ is a similarity function.

The challenge in streaming anomaly detection is to accurately and efficiently identify anomalies from unknown and evolving distributions under memory constraint, favoring incremental model updates over computationally expensive retraining.

Background: Isolation Distributional Kernel

IDK (Ting et al. 2021a,b) is a powerful measure to quantify the similarity between two distributions, based on a specific data-dependent point kernel called Isolation Kernel (IK) (Ting, Zhu, and Zhou 2018). It has been shown to improve the performance of various downstream tasks, such as density-based clustering (Qin et al. 2019), t-SNE visualization (Zhu and Ting 2021), and high-dimensional nearest neighbor search (Ting et al. 2024; Xu and Ting 2025), by mapping original data to a finite-dimensional feature map.

IK defines similarity based on random data-driven partitions, which can be implemented with hyperspheres. This mechanism is illustrated in Figure 2. It creates a random partitioning of the data space by first selecting a small subset of ψ points from a static dataset D. Then, for each selected point, a hypersphere is centered on it with a radius determined by its distance to its nearest neighbor within that subset. These ψ hyperspheres, along with the remaining unbounded space, form a complete partition of $\mathbb{R}^d$. IK uses an ensemble approach where this partitioning process is repeated t times. This procedure yields its exact, finite-dimensional feature map, $\Phi(x) \in \{0,1\}^{\psi\times t}$, where each value indicates if x falls into a specific hypersphere.

Then, based on the principle of KME (Muandet et al. 2017), IDK allows the similarity between two distributions, $\mathcal{P}_X$ and $\mathcal{P}_Y$, to be computed efficiently as the inner product of their mean kernel vectors (i.e., feature mean maps). The IDK is thus defined as:

$$\mathcal{K}_{\text{IDK}}(\mathcal{P}_X, \mathcal{P}_Y|D) = \frac{1}{t}\left\langle \widehat{\Phi}(\mathcal{P}_X), \widehat{\Phi}(\mathcal{P}_Y)\right\rangle, \tag{4}$$

where the kernel mean map $\widehat{\Phi}(\mathcal{P}_X) = \frac{1}{|X|}\sum_{x\in X}\Phi(x)$ is the centroid of the feature vectors of all points sampled from the distribution $\mathcal{P}_X$.

In offline anomaly detection, IDK computes a single mean feature vector, $\widehat{\Phi}(\mathcal{P}_D)$, which represents the centroid of distribution of the static dataset D. The normality of any given instance x is then measured by its similarity to the centroid of the overall data distribution. A high degree of similarity, computed via the inner product score$(x) = \frac{1}{t}\langle\Phi(x), \widehat{\Phi}(\mathcal{P}_D)\rangle$, results in a high normal score. Conversely, a low score indicates a poor fit with the overall data distribution, thus identifying a likely anomaly. The static nature of this design, however, presents a core challenge for dynamic data streams, which we address next.

Methodology

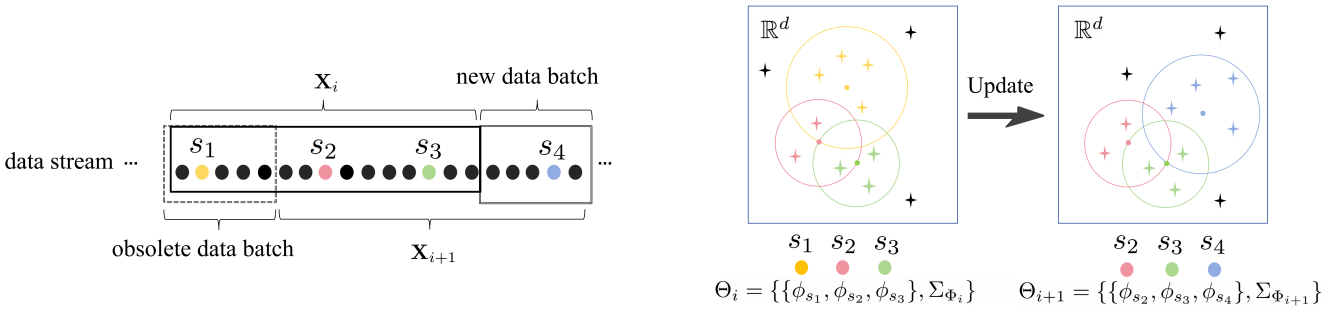
Insight: Incremental Distributional Kernel

The standard IDK is an offline method designed for static datasets. Its core data-dependent partitions are generated once from a fixed reference dataset and remain unchanged thereafter. When applied directly to a data stream where the underlying distribution evolves (i.e., concept drift), a model built on historical data quickly becomes obsolete. Consequently, the naive strategy for the standard IDK is to periodically discard the entire model and retrain it from scratch on a new window of recent data (Cao et al. 2025). Although straightforward, this retraining-based model is computationally expensive and impractical for real-time applications.

Our proposed incremental strategy is motivated by a key insight into IDK's structure: *the model can be treated as an ensemble of weak detectors, where each hypersphere partition acts as one such detector, which contributes one value to the entire feature map.* A value of 1 in this feature map signifies that an instance is considered "locally normal" by that specific partition, while a 0 means that it falls outside. The final anomaly score is not determined by any single detector, but by the aggregation of the entire feature map via KME. The power of this perspective is that since each weak detector is represented by a hypersphere, generated from a sampled point, the overall model can be updated incrementally by only replacing the detectors associated with obsolete sampled points. Specifically, as the data window slides, we identify the small subset of weak detectors whose generating samples came from the obsolete data batch. We then replace obsolete detectors with new ones generated from the new data batch. As the number of replaced detectors is small, this update strategy is far more efficient than retraining the entire IDK model. We term this incremental update mechanism the Incremental Distributional Kernel, which forms the core of our $\mathcal{IDK}$-$\mathcal{S}$ framework, as detailed below.

Proposed Framework: $\mathcal{IDK}$-$\mathcal{S}$

Given a data stream $\mathcal{S} = \{x_1, \ldots, x_T, \ldots\} \subset \mathbb{R}^d$, the proposed $\mathcal{IDK}$-$\mathcal{S}$ framework operates in two main phases: (1)



(a) New data batch arrives.

(b) $\mathcal{IDK}$-$\mathcal{S}$ model updates.

Figure 3: An illustration of the incremental update mechanism of $\mathcal{IDK}$-$\mathcal{S}$. As the data slides from $\mathbf{X}_i$ to $\mathbf{X}_{i+1}$, the model updates by discarding the partition tied to an obsolete sample (s_1, yellow) and generating a new partition from an incoming sample (s_4, blue). This process selectively modifies the set of hypersphere partitions, updating the model from Θ_i to Θ_{i+1}.

an one-off initialization phase to initialize the IDK model in the first window of the data stream, and (2) a continuous incremental update phase that adapts the model to the evolving data stream while scoring every new instance.

Initialization phase. Let $\mathbf{X}_0 = \{x_1, \ldots, x_\omega\} \subset \mathcal{S}$ denote the first data window in data stream $\mathcal{S}$, where ω is the fixed window size. $\mathcal{IDK}$-$\mathcal{S}$ first initializes the model on $\mathbf{X}_0$. As $\mathbf{X}_0$ can be treated as a given static data, this initialization phase is consistent with the IDK model (Ting et al. 2021b).

First, $\mathcal{IDK}$-$\mathcal{S}$ randomly selects ψ ($2 \leq \psi < \omega$) samples from $\mathbf{X}_0$ to get a subset $\mathcal{D} = \{s_1, \ldots, s_\psi\} \subset \mathbf{X}_0$. For any sample $s \in \mathcal{D}$, its hypersphere partition B_s centred at s with radius τ_s is defined to be $B_s = \{x \in \mathbb{R}^d : \{\|x - s\| < \tau_s\}$, where $\tau_s = \min_{y \in \mathcal{D} \setminus \{s\}} \|y - s\|$. Each hypersphere B_s is a weak anomaly detector, defined by a mapping function $\phi_s : \mathbb{R}^d \to \{0, 1\}$ to output a score (1 for normal, 0 for anomalous) for any instance $x \in \mathbb{R}^d$. As in IDK, $\phi_s(x) = 1$ if and only if $x \in B_s$ and x is closest to s compared to other samples in $\mathcal{D}$, i.e.,

$$\phi_s(x) = \mathbb{1}(x \in B_s, \|x - s\| = \min_{y \in \mathcal{D}} \|x - y\|), \quad (5)$$

where $\mathbb{1}(\cdot)$ is the indicator function.

Then repeat the above process t times, $\mathcal{IDK}$-$\mathcal{S}$ gets a set of t subsets $\{\mathcal{D}_i\}_{i=1}^t$ and a set of ψt detectors $\Phi_0 = \{\phi_{s_j}\}_{j=1}^{\psi t}$. For any $x \in \mathbb{R}^d$, its feature map is defined as

$$\Phi_0(x) = [\phi_{s_1}(x), \ldots, \phi_{s_{\psi t}}(x)], \quad (6)$$

where $\Phi_0(x) \in \{0, 1\}^{\psi \times t}$.

The normal score, as computed in IDK, is defined as

$$\text{score}(x) = \frac{1}{t} \langle \Phi_0(x), \widehat{\Phi}(\mathcal{P}_{\mathbf{X}_0}) \rangle, \quad (7)$$

where $\widehat{\Phi}(\mathcal{P}_{\mathbf{X}_0}) = \frac{1}{\omega} \sum_{y \in \mathbf{X}_0} \Phi(y)$ is the feature mean map that is used to represent the overall data distribution of $\mathbf{X}_0$ in a Hilbert space. To avoid a costly summation over the entire window each time, $\mathcal{IDK}$-$\mathcal{S}$ maintains a running sum of all feature vectors $\sum_{y \in \mathbf{X}_0} \Phi(y)$, abbreviated as Σ_{Φ_0}.

Let the model constructed by $\mathcal{IDK}$-$\mathcal{S}$ on $\mathbf{X}_0$ be denoted as $\Theta_0 = \{\Phi_0, \Sigma_{\Phi_0}\}$. For any data instance, its feature map is computed based on Φ_0 using Eq.(5) and Eq.(6). Its normal score is then computed through the dot product with Σ_{Φ_0} using Eq.(7). The core of $\mathcal{IDK}$-$\mathcal{S}$ is to continuously update Φ_0 and Σ_{Φ_0} through a sample replacement strategy to cope with data stream. It is important to note that, similar to other window-based streaming anomaly detectors, the effectiveness of the initial model Θ_0 is contingent on the representativeness of the first data window $\mathbf{X}_0$ (Leveni et al. 2024; Liu et al. 2025). This "cold start" consideration is a common aspect of online learning systems, where the initial state is learned from a limited subset of the entire data stream.

Incremental Update Phase. The incremental update phase is triggered when the data window slides from $\mathbf{X}_i$ to $\mathbf{X}_{i+1}$, by discarding the obsolete data batch and incorporating the new data batch. To adapt to this change without the high cost of full retraining, $\mathcal{IDK}$-$\mathcal{S}$ performs a targeted, three-step update (see Figure 3 for an illustrative example).

1. Updating samples and partitions: The core of the update refreshes the model's hypersphere partitions. First, $\mathcal{IDK}$-$\mathcal{S}$ identifies the set of sample points that were generated from the now-obsolete batch. These samples and their corresponding hypersphere partitions are removed from Φ_i. Second, an equal number of new sample points are randomly selected from the new data batch. New partitions are generated based on these new samples, resulting in an updated set of model partitions Φ_{i+1}.

2. Updating the feature map: With the new set of partitions, the model updates the feature map $\Phi_{i+1}(x)$ for every instance $x \in \mathbf{X}_{i+1}$. This update is highly efficient. For instances in the new data batch, $\mathcal{IDK}$-$\mathcal{S}$ computes their full feature maps against all partitions. For instances that remain in the window, $\mathcal{IDK}$-$\mathcal{S}$ only recompute the map entries corresponding to the updated partitions.

3. Computing the mean map and scoring: Finally, $\mathcal{IDK}$-$\mathcal{S}$ updates the sum of feature vectors from Σ_{Φ_i} to $\Sigma_{\Phi_{i+1}}$. This is achieved by subtracting the feature vectors from obsolete instances, adding the feature vectors of new instances, and incorporating the feature vector changes for persistent instances. The updated model is therefore $\Theta_{i+1} = \{\Phi_{i+1}, \Sigma_{\Phi_{i+1}}\}$. For scoring the new instances, their normal scores are computed using the updated feature mean map,

which is calculated on-the-fly as $\widehat{\Phi}(\mathcal{P}_{\mathbf{X}_{i+1}}) = \frac{1}{\omega}\Sigma_{\Phi_{i+1}}$.

By continually replacing the detectors associated with obsolete samples with new detectors generated from incoming data, $\mathcal{IDK}$-$\mathcal{S}$ effectively adapts to the current data distribution. The following proposition demonstrates that our proposed incremental strategy preserves a key statistical property of the retraining-based model.

PROPOSITION 1 (Sampling Distribution Equivalence)**.** *Given a data window $\mathbf{X}_i$ of size ω at any time step $i \in \mathbb{N}$. Let $\mathbf{S}_i \subset \mathbf{X}_i$ be the set of ψ unique sample points used to generate the model partitions Φ_i. The probability of obtaining a specific set $\mathbf{S}_i$ using the full retraining-based IDK is identical to the probability of obtaining it using $\mathcal{IDK}$-$\mathcal{S}$.*

$$P_{IDK}(\mathbf{S}_i \mid \mathbf{X}_i) = P_{\mathcal{IDK}\text{-}\mathcal{S}}(\mathbf{S}_i \mid \mathbf{X}_i) = \frac{1}{\binom{\omega}{\psi}}, \qquad (8)$$

where $\binom{\omega}{\psi}$ is the binomial coefficient, representing the total number of ways to choose ψ samples from the ω instances.

Proposition 1 demonstrates that at any time step, the probability distribution of the sample sets under both strategies is identical. This is because the incremental update mechanism in $\mathcal{IDK}$-$\mathcal{S}$ employs a random sampling process consistent with IDK, ensuring that the total probability of any final sample set appearing remains constant and equal to the uniform distribution probability of the retraining strategy. This theoretical guarantee is critical, as it distinguishes $\mathcal{IDK}$-$\mathcal{S}$ from many heuristic incremental methods that might trade model fidelity for speed (Leveni et al. 2024; Na, Kim, and Yu 2018), proving that our efficiency gains are achieved without compromising the statistical effectiveness of the model. The proof of Proposition 1 is provided in Appendix.

Complexity Analysis

The computational complexity is a crucial consideration in the streaming context, where data streams must be processed at high speed under low memory constraints. To quantify the efficiency of our proposed $\mathcal{IDK}$-$\mathcal{S}$, we analyze its time and space complexities in comparison to the retraining-based IDK strategy. The complexities are summarized in Table 1, with parameters defined as: window size w, update step size l, number of partitions t, hyperspheres per partition ψ, and dimensionality of the data instance d.

Time Complexity. The primary advantage of $\mathcal{IDK}$-$\mathcal{S}$ lies in its temporal efficiency. The time complexity of $\mathcal{IDK}$-$\mathcal{S}$ for an update step is $\mathcal{O}(l\psi t)$. This is derived from a detailed breakdown of the operations in its update phase.

First, compute the partitions corresponding to the updated samples. Let m denote the number of sample replacements in one update step. On average, $m = \frac{l}{\omega}\psi t$ samples are replaced. Computing the radii for the corresponding partitions incurs a cost of $\mathcal{O}(m \cdot \psi)$, i.e., $\mathcal{O}(\frac{l}{\omega}\psi^2 t)$. Second, update the feature map. For instances that remain in the window, their feature maps are updated according to the new partitions. The complexity of this operation is $\mathcal{O}(m \cdot (\omega - l))$, which simplifies to approximately $\mathcal{O}(l\psi t)$ (since $l \ll w$, the term $(\omega - l)/\omega$ is close to 1). For instances in the new data batch, computing their full feature maps against all partitions requires a time complexity of $\mathcal{O}(l\psi t)$. Third, score the new instance. Calculating the dot product of each new instance and the feature mean map requires time complexity $\mathcal{O}(l\psi t)$.

Therefore, the overall time complexity of $\mathcal{IDK}$-$\mathcal{S}$ is $\mathcal{O}(\frac{l}{\omega}\psi^2 t + l\psi t)$. As the sampling size is smaller than the window size, i.e., $\psi < \omega$, $\mathcal{IDK}$-$\mathcal{S}$ has an overall time complexity of $\mathcal{O}(l\psi t)$, as the second term dominates the expression. In contrast, the retraining-based IDK model needs to compute the feature maps of all ω instances via all ψt detectors, resulting in a time complexity of $\mathcal{O}(\omega\psi t)$. Given that $l \ll w$, the incremental strategy of $\mathcal{IDK}$-$\mathcal{S}$ is more efficient and suitable for real-time applications.

	Time	Space
$\mathcal{IDK}$-$\mathcal{S}$	$\mathcal{O}(l\psi t)$	$\mathcal{O}(\omega d + \psi t d + \omega\psi t)$
Retraining-based IDK	$\mathcal{O}(\omega\psi t)$	$\mathcal{O}(\omega d + \psi t d + \omega\psi t)$

Table 1: Complexity comparison of $\mathcal{IDK}$-$\mathcal{S}$ vs. IDK.

Space Complexity. In terms of memory, both the incremental $\mathcal{IDK}$-$\mathcal{S}$ and the retraining-based IDK have the same space complexity of $\mathcal{O}(\omega d + \psi t d + \omega\psi t)$. This is because both of them need to store the data in the current window $\mathcal{O}(\omega d)$, all samples $\mathcal{O}(\psi t d)$, and the feature maps of data instances in current window $\mathcal{O}(\omega\psi t)$. Crucially, for $\mathcal{IDK}$-$\mathcal{S}$, this memory requirement is dependent on the window size ω, not the total length of the data stream, making it viable for processing potentially infinite data streams.

Experiments

In this section, we compare $\mathcal{IDK}$-$\mathcal{S}$ with state-of-the-art methods on large-scale anomaly detection benchmarks, including both stationary and non-stationary distributions.

Datasets

Following oIFOR (Leveni et al. 2024), we use eleven large anomaly detection benchmarks, including the cardinality n ranging from thousands to hundreds of thousands, dimensionality d ranging from 3-48, and anomaly proportions ranging from 0.03% to 32%, to comprehensively test the performance of different methods on stationary data streams. Each stationary dataset is first randomly shuffled and then used for algorithm execution. We also use the `INSECTS` dataset (Frittoli, Carrera, and Boracchi 2023; Stucchi et al. 2023), which contains five real changes caused by temperature modifications that affect the insects' flying behavior, to test the performance on non-stationary data streams (Leveni et al. 2024). Moreover, we use the `TwoCluster` dataset shown in Figure 1 to test the performance on non-stationary data streams. Different from the sudden concept drift of INSECTS, the concept drift of `TwoCluster` occurs in a more gradual form. The properties of datasets used in our experiments are outlined in Table 2.

Competing Methods and Setups

We compare the performance of $\mathcal{IDK}$-$\mathcal{S}$ against five baselines, which represent various research lines to extend foun-

	Dataset	n	d	% of anomalies
Stationary	Donors	619,326	10	5.90
	Http	567,497	3	0.40
	ForestCover	286,048	10	0.90
	Fraud	284,807	29	0.17
	Mulcross	262,144	4	10.00
	Smtp	95,156	3	0.03
	Shuttle	49,097	9	7.00
	Mammography	11,183	6	2.00
	NYC_taxi_single	10,273	48	5.20
	Annthyroid	6,832	6	7.00
	Satellite	6,435	36	31.64
Non	INSECTS	212,514	33	5.50
	TwoCluster	1,000,000	2	5.00

Table 2: Datasets properties.

dational offline detectors to the streaming data paradigm. These competitors include two adaptations of the iForest detector, SiForest (Liu et al. 2025) and oIFOR (Leveni et al. 2024); DILOF (Na, Kim, and Yu 2018), which extends the density-based LOF detector; CPOD (Tran, Mun, and Shahabi 2020), which extends the distance-based detector; and Memstream (Bhatia et al. 2022), which combines a deep autoencoder with kNN for anomaly detection. It is worth noting that only Memstream runs on GPU in our experiment.

For all competing methods, we carefully tuned their parameters according to the suggestions in their papers. Following oIFOR (Leveni et al. 2024), we set the window size $\omega = 2048$ and the update step size $l = 100$ for all methods. Notably, for a fair comparison, the number of ensembles was set to the default $t = 100$ for the ensemble methods, including $\mathcal{IDK}$-$\mathcal{S}$, SiForest and oIFOR. For $\mathcal{IDK}$-$\mathcal{S}$, the number of samples ψ was searched in $[2^1, 2^2, \ldots, 2^6]$. Each method was executed 20 times on all datasets. We adopted the widely used ROC AUC score and runtime (in seconds) to evaluate the effectiveness and efficiency of the methods (Huang and Ling 2005). All experiments are conducted on the same Linux CPU machine: AMD 128-core CPU with each core running at 2 GHz and 1T GB RAM.

Results

Detection Effectiveness. In Table 3, we show the average AUC scores for each algorithm on all datasets. The results show that $\mathcal{IDK}$-$\mathcal{S}$ achieves the best AUC score in the most datasets, with the only exception of the Shuttle dataset. Interestingly, this result on Shuttle is consistent with the observation from existing research of the offline detectors, where the IDK (0.98) was also slightly outperformed by iForest (0.99) on this specific dataset, while IDK was superior on other datasets, including Smtp and ForestCover (Ting et al. 2021b). This observation is a reflection of our core argument that the performance of a streaming detector is fundamentally governed by the capabilities of its underlying offline base. The superiority of IDK is due to its use of a data-dependent kernel, which has been demonstrated to be more effective than the mechanisms in traditional detectors like iForest and LOF in most offline cases. This inherent advantage carries over directly to the streaming context. The proposed $\mathcal{IDK}$-$\mathcal{S}$ achieves the best mean AUC of 0.876, while its closest competitor, SiForest, obtains a mean of 0.853, showing a clear performance gap.

Adaptability to Concept Drift. Figure 4 investigates the adaptability of the methods on two non-stationary datasets. To assess instantaneous performance, we computed the AUC score at each time instant T using normal scores within a sliding evaluation interval $\{x_{T-\omega}, \ldots, x_{T+\omega}\}$ of size 2ω, centred at x_T. The length of this interval was chosen to guarantee that the resulting curves exhibit a satisfactory degree of smoothness. Overall, $\mathcal{IDK}$-$\mathcal{S}$ achieves the best performance compared to existing methods. On the INSECTS dataset, which features abrupt concept drifts (Figure 4a), all methods show temporary but significant drops in AUC scores after each drift, underscoring a common challenge for current streaming detectors. In contrast, on the TwoCluster dataset with gradual drifts (Figure 4b), $\mathcal{IDK}$-$\mathcal{S}$ and most other methods maintain stable performance. The notable exception is SiForest, whose accuracy degrades significantly as the distribution evolves. In our experiments, we found that SiForest sometimes retains obsolete data for too long due to its reservoir sampling update mechanism, preventing the model from adapting to the current distribution over time.

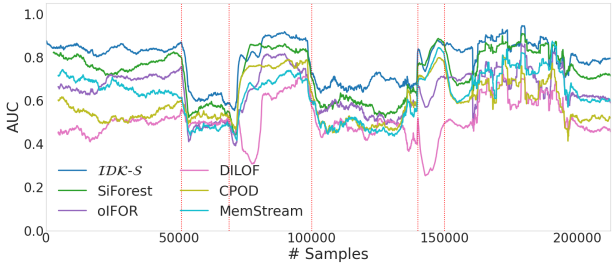
Computational Efficiency. Beyond superior detection performance, $\mathcal{IDK}$-$\mathcal{S}$ shows exceptional time efficiency, a critical requirement for real-world streaming applications. As detailed in Table 3, $\mathcal{IDK}$-$\mathcal{S}$ consistently records the lowest runtimes across all competing methods. This efficiency is also clearly evident even when compared to existing fastest baseline, oIFOR. On the six largest datasets, $\mathcal{IDK}$-$\mathcal{S}$ shows a speed-up of at least 5x. The runtime gap widens further on several of the smaller datasets (e.g., Shuttle, NYC_taxi_single, and Satellite), where the runtime of $\mathcal{IDK}$-$\mathcal{S}$ is reduced by a factor approaching or exceeding an order of magnitude. This superior time efficiency of $\mathcal{IDK}$-$\mathcal{S}$ is due to three key factors. First, its base detector IDK has linear-time complexity, making it inherently faster than methods based on LOF and kNN which often have quadratic complexity (Ting et al. 2021a). Second, our incremental update strategy avoids costly full retraining, which decouples the update complexity from the overall window size and makes it dependent only on the much smaller update step size. Third, compared to the iForest-based methods including SiForest and oIFOR with the same number of ensembles (t), $\mathcal{IDK}$-$\mathcal{S}$ achieves its optimal performance with a significantly smaller number of samples (ψ), which we discuss in detail in the following parameter analysis.

Parameter Analysis

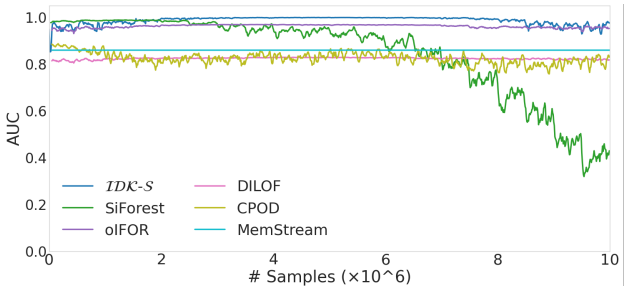
To further understand the efficiency of $\mathcal{IDK}$-$\mathcal{S}$, we analyze its sensitivity to the number of samples, ψ, a key parameter that influences both performance and computational cost. We compare $\mathcal{IDK}$-$\mathcal{S}$ with the iForest-based ensemble methods, SiForest and oIFOR, which have analogous parameters (*reservoir_samples* and *max_leaf_samples*, respectively). For comparison purposes, we set the number of ensemble estimators to $t = 100$. Figure 5 shows that the optimal sample size ψ of $\mathcal{IDK}$-$\mathcal{S}$ is concentrated in the range of $[2, 8]$, while

	Dataset	AUC (↑)						TIME (s) (↓)					
		$\mathcal{IDK}$-$\mathcal{S}$	SiForest	oIFOR	DILOF	CPOD	Memstream	$\mathcal{IDK}$-$\mathcal{S}$	SiForest	oIFOR	DILOF	CPOD	Memstream
Stationary	Donors	**0.802**	0.780	0.796	0.673	0.603	0.761	**57**	694	301	1492	4227	604
	Http	**0.999**	**0.999**	0.996	0.732	0.698	0.998	**29**	426	183	633	1492	439
	ForestCover	**0.926**	0.907	0.891	0.703	0.729	0.885	**19**	315	115	578	1619	294
	Fraud	**0.962**	0.941	0.931	0.897	0.874	0.936	**21**	378	177	591	1584	345
	Mulcross	**0.998**	0.949	0.994	0.800	0.810	0.981	**17**	289	87	443	1136	251
	Smtp	**0.911**	0.894	0.863	0.854	0.851	0.904	**6**	203	31	196	559	179
	Shuttle	0.976	**0.996**	0.990	0.912	0.902	0.973	**2**	169	19	128	413	155
	Mammography	**0.866**	0.840	0.859	0.829	0.803	0.842	**0.7**	51	4	57	128	43
	NYC_taxi_single	**0.732**	0.719	0.565	0.507	0.491	0.702	**1**	102	11	98	266	79
	Annthyroid	**0.752**	0.713	0.678	0.619	0.608	0.709	**0.7**	27	3	39	85	3
	Satellite	**0.726**	0.699	0.667	0.477	0.636	0.678	**0.5**	42	5	36	24	4
Non	INSECTS	**0.783**	0.752	0.718	0.596	0.614	0.665	**18**	713	368	1447	4531	734
	TwoCluster	**0.958**	0.896	0.940	0.885	0.829	0.904	**124**	2194	1528	4419	9473	2366
	Mean	**0.876**	0.853	0.838	0.730	0.727	0.841	**27**	432	218	781	1916	423
	Improvement (%) / Speedup (x)	-	2.2%	4.5%	20.0%	20.4%	4.2%	-	16x	8x	29x	71x	16x

Table 3: The average AUC scores and runtimes. The best row-wise results are highlighted in bold.



(a) INSECTS dataset (abrupt change).



(b) TwoCluster dataset (gradual change).

Figure 4: The performance of methods on non-stationary datasets where $\mathcal{P}_N$ and $\mathcal{P}_A$ change over time.

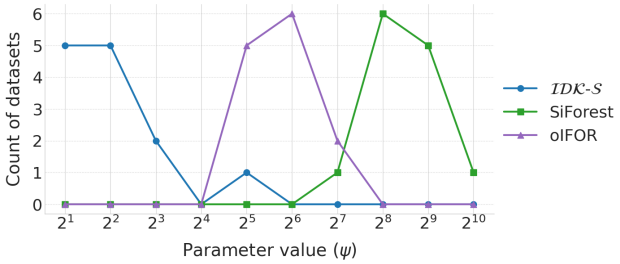


Figure 5: Distribution of the optimal sample size ψ.

Type	Method	Average AUC	Average time (s)
Online	$\mathcal{IDK}$-$\mathcal{S}$	0.877	14
	Retraining-based IDK	0.879	352
Offline	IDK	0.898	8

Table 4: Performance comparison of IDK variants.

that of oIFOR is $[32, 128]$ and that of SiForest is $[256, 512]$. This finding highlights a fundamental advantage of our approach. The ability of $\mathcal{IDK}$-$\mathcal{S}$ to operate effectively with a much smaller ψ is a key contributor to its superior time efficiency, as the complexity of both generating partitions and scoring instances is directly dependent on this parameter. It suggests that the data-dependent kernel of IDK is more sample-efficient in capturing the data distribution compared to the random space-partitioning mechanism of iForest.

Comparison with IDK Variants

We compare $\mathcal{IDK}$-$\mathcal{S}$ with its two variants. Table 4 summarizes their average performance on stationary datasets. The offline IDK, which processes each dataset as a single static batch, serves as a non-streaming performance reference. The results show that $\mathcal{IDK}$-$\mathcal{S}$ achieves results comparable to retraining-based IDK while being more than 20x faster, demonstrating the effectiveness of our approach.

Conclusion

We introduce $\mathcal{IDK}$-$\mathcal{S}$, a novel streaming anomaly detector that successfully adapts the powerful Isolation Distributional Kernel (IDK) to address the limitations of existing methods which are based on weaker offline detectors. Our lightweight incremental update strategy is not an approximation; it maintains statistical equivalence to full retraining while significantly reducing the computational complexity of IDK. Extensive experiments demonstrated that $\mathcal{IDK}$-$\mathcal{S}$ achieves both state-of-the-art accuracy and a speed-up of up to an order of magnitude over existing methods. This dual advantage is driven by the superior efficiency of its underlying data-dependent kernel. As a fast and accurate approach, $\mathcal{IDK}$-$\mathcal{S}$ is well-suited for real-world anomaly detection in evolving high-volume data streams. A future work would explore the challenge of improving robustness to abrupt concept drifts, a common weakness of current methods.

Acknowledgements

This work is supported by National Natural Science Foundation of China (Grant No. W2531050 and 92470116).

References

Ahmed, M.; Mahmood, A. N.; and Hu, J. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60: 19–31.

Aryal, S.; Ting, K. M.; Wells, J. R.; and Washio, T. 2014. Improving iforest with relative mass. In *Advances in Knowledge Discovery and Data Mining, PAKDD*, 510–521. Springer.

Bandaragoda, T. R.; Ting, K. M.; Albrecht, D.; Liu, F. T.; and Wells, J. R. 2014. Efficient anomaly detection by isolation using nearest neighbour ensemble. In *IEEE International Conference on Data Mining Workshop*, 698–705. IEEE.

Bhatia, S.; Jain, A.; Srivastava, S.; Kawaguchi, K.; and Hooi, B. 2022. Memstream: Memory-based streaming anomaly detection. In *Proceedings of the ACM Web Conference*, 610–621.

Boukerche, A.; Zheng, L.; and Alfandi, O. 2020. Outlier detection: Methods, models, and classification. *ACM Computing Surveys*, 53(3): 1–37.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 93–104.

Cao, Y.; Ma, Y.; Zhu, Y.; and Ting, K. M. 2025. Revisiting streaming anomaly detection: benchmark and evaluation. *Artificial Intelligence Review*, 58(1): 1–24.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys*, 41(3): 1–58.

Frittoli, L.; Carrera, D.; and Boracchi, G. 2023. Nonparametric and Online Change Detection in Multivariate Datastreams Using QuantTree. *IEEE Transactions on Knowledge and Data Engineering*, 35(08): 8328–8342.

Hilal, W.; Gadsden, S. A.; and Yawney, J. 2022. Financial fraud: a review of anomaly detection techniques and recent advances. *Expert Systems With Applications*, 193: 116429.

Huang, J.; and Ling, C. X. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3): 299–310.

Leveni, F.; Cassales, G. W.; Pfahringer, B.; Bifet, A.; and Boracchi, G. 2024. Online Isolation Forest. In *International Conference on Machine Learning*, 27288–27298. PMLR.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 eighth IEEE International Conference on Data Mining*, 413–422. IEEE.

Liu, J. J.; Cassales, G. W.; Liu, F. T.; Pfahringer, B.; and Bifet, A. 2025. Streaming Isolation Forest. In Wu, X.; et al., eds., *Advances in Knowledge Discovery and Data Mining, PAKDD*, volume 15870. Springer.

Lu, T.; Wang, L.; and Zhao, X. 2023. Review of anomaly detection algorithms for data streams. *Applied Sciences*, 13(10): 6353.

Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; Schölkopf, B.; et al. 2017. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2): 1–141.

Na, G. S.; Kim, D.; and Yu, H. 2018. Dilof: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1993–2002.

Qin, X.; Ting, K. M.; Zhu, Y.; and Lee, V. C. 2019. Nearest-neighbour-induced isolation similarity and its impact on density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4755–4762.

Ramaswamy, S.; Rastogi, R.; and Shim, K. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 427–438.

Stojanovic, L.; Dinic, M.; Stojanovic, N.; and Stojadinovic, A. 2016. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *2016 IEEE International Conference on Big Data*, 1647–1652. IEEE.

Stucchi, D.; Rizzo, P.; Folloni, N.; and Boracchi, G. 2023. Kernel quanttree. In *International Conference on Machine Learning*, 32677–32697. PMLR.

Ting, K. M.; Washio, T.; Wells, J. R.; and Zhang, H. 2021a. Isolation kernel density estimation. In *2021 IEEE International Conference on Data Mining*, 619–628. IEEE.

Ting, K. M.; Washio, T.; Zhu, Y.; Xu, Y.; and Zhang, K. 2024. Is it possible to find the single nearest neighbor of a query in high dimensions? *Artificial Intelligence*, 336: 104206.

Ting, K. M.; Xu, B.-C.; Washio, T.; and Zhou, Z.-H. 2021b. Isolation distributional kernel: A new tool for point and group anomaly detections. *IEEE Transactions on Knowledge and Data Engineering*, 35(3): 2697–2710.

Ting, K. M.; Zhu, Y.; and Zhou, Z.-H. 2018. Isolation kernel and its effect on SVM. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2329–2337.

Tran, L.; Mun, M. Y.; and Shahabi, C. 2020. Real-time distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment*, 14(2): 141–153.

Xu, Y.; and Ting, K. M. 2025. Voronoi Diagram Encoded Hashing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 87–103. Springer.

Zhu, Y.; and Ting, K. M. 2021. Improving the effectiveness and efficiency of stochastic neighbour embedding with isolation kernel. *Journal of Artificial Intelligence Research*, 71: 667–695.