Adaptive-GraphSketch: Real-Time Edge Anomaly Detection via Multi-Layer Tensor Sketching and Temporal Decay

Ocheme Anthony Ekle, *Student Member, IEEE*, and William Eberle, *Member, IEEE*
Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA
Email: {oaekle42, weberle}@tntech.edu

***Abstract*—Anomaly detection in dynamic graphs is essential for identifying malicious activities, fraud, and unexpected behaviors in real-world systems such as cybersecurity and power grids. However, existing approaches struggle with scalability, probabilistic interpretability, and adaptability to evolving traffic patterns. In this paper, we propose ADAPTIVE-GRAPHSKETCH, a lightweight and scalable framework for real-time anomaly detection in streaming edge data. Our method integrates temporal multi-tensor sketching with Count-Min Sketch using Conservative Update (CMS-CU) to compactly track edge frequency patterns with bounded memory, while mitigating hash collision issues. We incorporate Bayesian inference for probabilistic anomaly scoring and apply Exponentially Weighted Moving Average (EWMA) for adaptive thresholding tuned to burst intensity. Extensive experiments on four real-world intrusion detection datasets demonstrate that ADAPTIVE-GRAPHSKETCH outperforms state-of-the-art baselines such as ANOEDGE-G/L, MIDAS-R, and F-FADE, achieving up to 6.5% AUC gain on CIC-IDS2018 and up to 15.6% on CIC-DDoS2019, while processing 20 million edges in under 3.4 seconds using only 10 hash functions. Our results show that ADAPTIVE-GRAPHSKETCH is practical and effective for fast, accurate anomaly detection in large-scale streaming graphs.**

***Index Terms*—Anomaly Detection, Streaming, Real-time, Dynamic Graphs, Edge Streams, Tensor Sketching**

I. INTRODUCTION

Dynamic graph data is increasingly prevalent in real-time systems such as cybersecurity, social media, power grids, and fraud detection [1]–[3]. These systems generate massive, high-velocity streams of edges, representing relationships between nodes, and often exhibit evolving topologies and complex structural patterns. Detecting anomalies in such settings is critical for identifying unusual patterns.

However, many graph anomaly detection techniques, such as personalized random walks [4], matrix factorization [5], and snapshot aggregation [2], [6], struggle in streaming settings. They often require storing the full adjacency matrix or computing costly subgraph statistics, incurring high memory and latency, and lack adaptability to rapid network changes and interpretable, probabilistic outputs [3].

In this paper, we propose ADAPTIVE-GRAPHSKETCH, a lightweight, streaming anomaly detection framework that runs in real time without storing the full graph. Our method integrates temporal multi-tensor sketching [7] with Count–Min Sketch with Conservative Updates (CMS-CU) [8] to compactly track edge frequencies in bounded memory, while mitigating collisions. To enhance the detection of fast-changing anomalies, we incorporate Bayesian posterior scoring for uncertainty-aware and Exponentially Weighted Moving Average (EWMA) smoothing [9] for dynamic thresholding. The EWMA parameters are adaptively tuned based on burst intensity (i.e, edge spikes per time windows) to handle concept drift and volatility in graph streams.

Unlike sketch-based baselines such as MIDAS-R [1] and F-FADE [6], which lack probabilistic reasoning and struggle under rapidly evolving networks, ADAPTIVE-GRAPHSKETCH provides a unified real-time pipeline with Bayesian scoring and EWMA-based dynamic thresholding. To the best of our knowledge, it is the first edge-level streaming detection framework to combine these components into a probabilistic model with mathematically justified threshold adaptation.

The key contributions of this work are:

- Real-time framework using temporal multi-tensor sketching to compactly track edge frequencies with bounded memory.
- Collision-mitigated frequency estimation using Count–Min Sketch with Conservative Updates.
- Uncertainty-aware detection via Bayesian posterior scoring coupled with EWMA adaptive thresholding.
- Extensive experiments on four large datasets. ADAPTIVE-GRAPHSKETCH is competitive on DARPA and ISCX-IDS2012, and achieves up to +6.5% AUC (CIC-IDS2018) and +15.6% (CIC-DDoS2019), while processing 20 million edges in under 3.4s with 10 hash functions, demonstrating strong detection accuracy and runtime efficiency.

Reproducibility: Our code and datasets are available at `https://github.com/EkleTony/Graph-Sketch`.

II. RELATED WORK

In this section, we review existing methods for detecting anomalies in dynamic graphs. We group prior work into three main categories based on their algorithmic strategies. For broader coverage, we refer readers to [3], [13].

Edge Stream Methods: SEDANSPOT [10] detects sparse anomalies based on occurrence patterns; PENMINER [11] captures persistence in edge updates; F-FADE [6] uses

TABLE I

OUR METHOD VS. BASELINES: COMPARISON OF GRAPHSKETCH WITH PRIOR DYNAMIC GRAPH ANOMALY DETECTION METHODS.

Property \ Method	SedanSpot [10]	DenseStream [2]	PENminer [11]	F-FADE [6]	MIDAS-R [1]	AnoEdge [12]	OUR METHOD
Real-time detection*					✓	✓	✓
Edge anomalies	✓	✓	✓	✓	✓	✓	✓
Temporal Tensor Sketching							✓
Adaptive Bayesian Scoring							✓
Uncertainty modeling							✓
Memory-efficient					✓	✓	✓
Sudden edge changes	✓		✓	✓	✓	✓	✓

*Real-time = processing 20M edges within 10 seconds.

likelihoods estimation for frequency patterns; MIDAS-R [1] combines CMS with chi-squared for scoring. These methods lack probabilistic reasoning, struggle with bursts, and are computationally expensive. We address these via multi-tensor sketching, Bayesian inference, and adaptive thresholding.

***Probabilistic Sketch Methods*:** CMS [8] is widely used for scalable frequency estimation. RHSS [14] applies it to edge attributes; ANOEDGE [12] uses higher-order sketches; DECAYRANK [15] adds temporal decay to PageRank; ADAPTIVE-DECAYRANK [4] adds Bayesian updates and dynamic thresholds. Despite good memory use, these methods [1], [12], [14], [15] rely on fixed thresholds and lack uncertainty modeling, limiting performance on high-frequency edge streams. In contrast, ADAPTIVE-GRAPHSKETCH unifies 3D tensor sketching with CMS-CU, Bayesian scoring, and dynamic thresholding for real-time adaptation.

***Matrix Factorization Methods*:** DENSESTREAM [2] incrementally detect dense subtensors; EDGEMONITOR [16] model edge transitions with first-order Markov chains; MULTILAD [5] uses spectral decomposition for subgraph anomalies. While effective offline, these methods are costly and unsuitable for real-time settings. In contrast, our method avoid global recomputation and use bounded-memory summaries for efficient real-time edge anomaly detection.

As summarized in Table I, our method is the first to unify temporal sketching, conservative updates, Bayesian scoring, and adaptive thresholding in a single real-time pipeline.

III. PRELIMINARIES AND PROBLEM DEFINITION

Let $G = (V, E)$ be a dynamic graph with edge stream $E = \{e_i\}$, where $e_i = (u_i, v_i, t_i)$ denotes an interaction from u_i to v_i at time t_i. Table II summarizes the notation used throughout.

DEFINITION 1. (EDGE-LEVEL ANOMALY): An edge anomaly occurs at time t when the observed frequency of an edge (u, v) significantly deviates from its historical average. Given time window $t \geq 1$ and threshold $\alpha > 0$, the score is:

TABLE II

SUMMARY OF NOTATIONS USED IN ADAPTIVE-GRAPHSKETCH

Symbol	**Definition**
$e = (u, v, t)$	Edge from node u to v arriving at time t.
a, s	Observed and Cummulative frequency of edges.
$\hat{f}(x)$	Estimated frequency of item x via CMS-CU.
$h_i(x)$, v_i	Hash function of x and Counter value in sketch row i.
$C_{i,j}^{(t)}$	Value in sketch at row i, column j, time t.
$\boldsymbol{\mathcal{S}}_u$, $\boldsymbol{\mathcal{S}}_v$	Hash-based sketches of nodes u and v.
μ, σ^2	Expected frequency & Variance under normal behavior.
$\mu_A = \mu + \delta$	Mean under anomaly assumption.
$\sigma_A^2 = 4\sigma^2$	Increased variance under anomaly.
λ	Weight for smoothing past and current stats.
$\tilde{s}_t$	Smoothed cumulative frequency at time t.
γ	Decay factor ($0 < \gamma < 1$).
X_t, τ_t	Anomaly score and Dynamic threshold at time t.
α	EWMA smoothing constant ($0 < \alpha < 1$).
μ_t, σ_t	Mean and Std. deviation of X_t scores up to t.

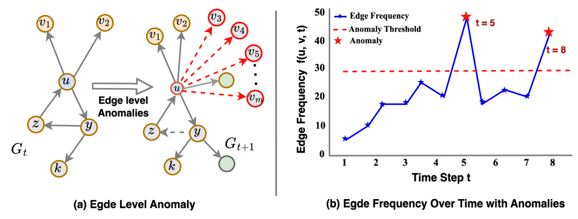


Fig. 1. **Edge anomalies over two steps** (t, t+1): **(a)** sudden bursts in edge activity and emerging microclusters as $G_t \rightarrow G_{t+1}$; **(b)** edge frequencies deviating from historical trends, with spikes at t=5, 8 suggest cyberattacks.

$$\text{Anomaly Score}(e, t) = \frac{(a - \frac{s}{t})^2 \cdot t}{s \cdot (t - 1)} \geq \alpha, \tag{1}$$

where a is the observed freq. at time t and s its history. In Fig. 1, edge anomalies appear as sudden bursts (e.g., $u \rightarrow \{v_1, \ldots, v_m\}$), while spikes at t=5, 8 indicate unusual surges.

A. Count-Min Sketch (CMS)

CMS [14] is a probabilistic structure for estimation with a 2D counter array and d hash functions. For item x,

$$\hat{f}(x) = \min_{i=1}^{d} \text{CM}_{i,h_i(x)}. \tag{2}$$

CMS achieves sublinear memory usage but suffers from overestimation due to hash collisions [17].

B. CMS with Conservative Update (CMS-CU)

CMS-CU [8] improves upon CMS [14] by reducing collision errors and updating only the counters at the current minimum value. For edge (u, v), we maintain two sketches: CMS-CU$_{\text{cur}}$ for the current window and CMS-CU$_{\text{total}}$ for the cumulative. For x, let v_i be the counter at $h_i(x)$, updated as $v_i \leftarrow v_i + 1$ if $v_i = \min_{j=1}^{d} v_j$. The frequency estimate is:

$$\hat{f}(x) = \min_{i=1}^{d} \text{CMS-CU}_{i,h_i(x)}. \tag{3}$$

C. Tensor Sketching

Tensor Sketching [7] generalizes pairwise edge tracking to higher-order interactions. For edge (u, v, t),

$$\hat{f}_{uv}^{(k)}(t) = \text{FFT}^{-1}(\text{FFT}(\boldsymbol{\mathcal{S}}_u) \circ \text{FFT}(\boldsymbol{\mathcal{S}}_v)), \tag{4}$$

with $\boldsymbol{\mathcal{S}}_u, \boldsymbol{\mathcal{S}}_v$ the sketches of u, v and $\circ$ element-wise multiplication. We avoid FFT (Fast Fourier Transform) by updating a 3D CMS for real-time, and efficient edge tracking.

D. Bayesian Anomaly Scoring

Bayesian inference [18] combines prior beliefs with data to estimate event likelihoods. We apply it to compute edge interactions and uncertainty-aware scores. The posterior is

$$P(\text{Anomaly} \mid a, \mathcal{H}) = \frac{P(a \mid \text{Anomaly}) \cdot P(\text{Anomaly})}{P(a)}, \tag{5}$$

where a is the observed frequency and $\mathcal{H}$ the historical data.

E. Problem Statement

Given edge stream $E = \{e_i = (u_i, v_i, t_i)\}$, the task is to assign each edge an anomaly score from historical deviations. An edge is *anomalous* if its score exceeds a dynamic threshold α adapting to balance sensitivity and false positives (Figure 1).

Problem 1. *Given a streaming graph E, compute an anomaly score for each edge $e = (u, v, t)$ by comparing its frequency with historical behavior. Higher scores indicate anomalous edges, such as rare spikes or bursts of interactions.*

IV. Method

In this section, we present our Adaptive-GraphSketch approach, illustrated in Figure 3. The key innovations are:

- **Multi-layer tensor sketch:** A compact 3D sketch $\mathcal{S} \in \mathbb{R}^{d \times w \times W}$ encodes edge frequencies via hash layers.
- **Frequency Estimation:** 3D CMSCU estimates counts over (d, w, t_w) while reducing overestimation.
- **Temporal Decay and Pruning:** Applies exponential decay (γ) and sliding window to focus on recent activity.
- **Bayesian Scoring & Thresholding:** Computes the posterior score with an EWMA threshold $\tau_t = \mu_t + k\sigma_t$.

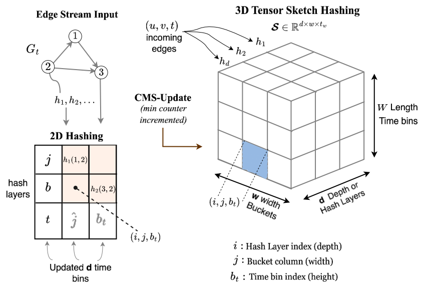


Fig. 2. **Multi-Layer Tensor Sketch Overview**. An edge (u, v, t) is hashed by $h_1, h_2, \ldots, h_d$, each h_i mapping it to a bucket j in the i-th sketch layer. Time is discretized into bins via $b_t = \lfloor t/\Delta \rfloor$. The 3D tensor $\mathcal{S}[i][j][b_t]$ is updated using CMS-CU, where only the min counter across layers is incremented.

A. Multi-Layer Tensor Sketching

To approximate evolving edge behavior, we use a **3D** tensor sketch $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times w \times W}$ that encodes frequency dynamics via *hash projection and time-aware indexing*. Building on prior tensor sketching for high-dimensional approximation with randomized hashing [7], we introduce a third axis for time indexing t_w. The sketch captures dynamics across d hash rows, w buckets, and W time bins with bounded memory. Figure 2 illustrates edge $(u, v, t) \in \mathcal{E}$ projected into the 3D tensor.

We define a dynamic graph as a sequence of timestamped edges $(u, v, t) \in \mathcal{E}$. The evolution is modeled by dividing the timeline into intervals Δ, assigning each edge to a time bin

$$b_t = \left\lfloor \frac{t}{\Delta} \right\rfloor, \quad b_t \in \{1, \ldots, W\}, \tag{6}$$

where t is the edge timestamp, Δ the bin width, and W the number of active bins in the tensor sketch $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times w \times W}$.

We use a family of d pairwise-independent hash functions $\{h_1, h_2, \ldots, h_d\}$, where each $h_i : \mathbb{N} \times \mathbb{N} \to \{1, 2, \ldots, w\}$ maps an edge (u, v) to a *bucket* in the i-th row. The sketch cell $\boldsymbol{\mathcal{S}}[i, h_i(u, v), b_t]$ stores the frequency of (u, v) in bin b_t.

Each edge arrival (u, v, t), triggers a *hash-based projection* into all d rows of the tensor sketch by updating:

$$\boldsymbol{\mathcal{S}}[i, h_i(u, v), b_t] \leftarrow \boldsymbol{\mathcal{S}}[i, h_i(u, v), b_t] + 1, \tag{7}$$

for $i = 1, \ldots, d$.

We adopt the CMS-CU [8], where only sketch counters with minimum value $\{\boldsymbol{\mathcal{S}}[j][h_j(u, v)][b_t]\}_{j=1}^{d}$ are incremented.

$$\text{If } \boldsymbol{\mathcal{S}}[i][h_i(u, v)][b_t] = \min_j \boldsymbol{\mathcal{S}}[j][h_j(u, v)][b_t], \tag{8}$$

where $h_i(u, v)$ is the i-th hash mapping edge (u, v) to a sketch column, and $b_t = \left\lfloor \frac{t}{\Delta} \right\rfloor$ the current time bin for timestamp t. The estimated frequency of edge (u, v) in the current bin is:

$$\hat{a}_{uv}(t) = \min_{i=1}^{d} \boldsymbol{\mathcal{S}}[i][h_i(u, v)][b_t], \tag{9}$$

where $\hat{a}_{uv}(t)$ is the estimated edge count in time bin b_t.

To maintain long-term context, we define the cumulative frequency $\hat{s}_{uv}(t)$ of edge (u, v) up to b_t as:

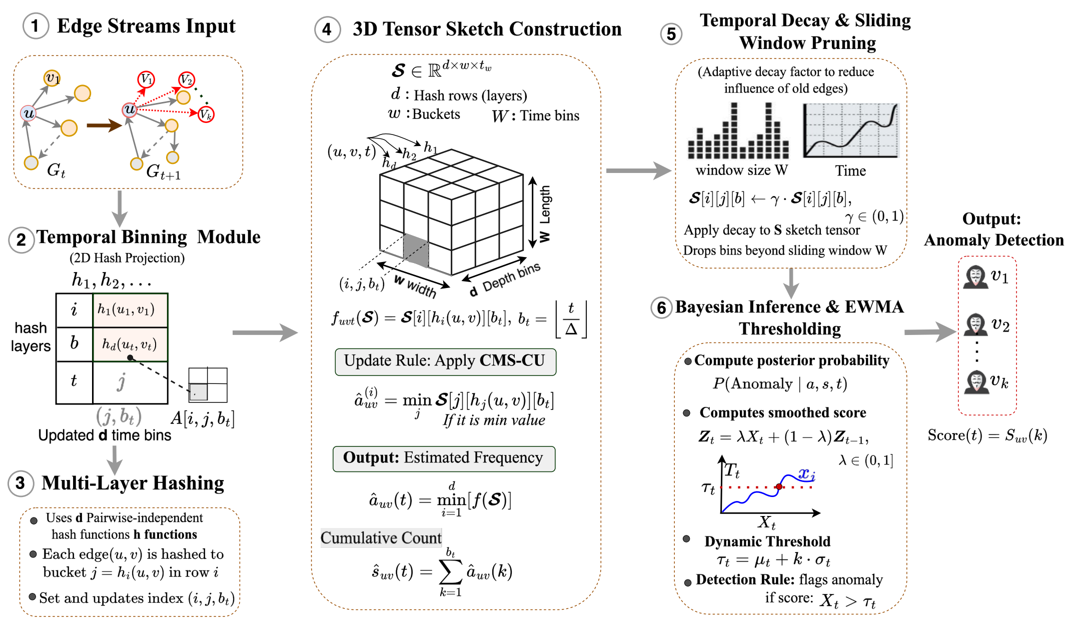
$$\hat{s}_{uv}(t) = \sum_{k=1}^{b_t} \hat{a}_{uv}(k). \tag{10}$$

Unlike traditional **2D** adjacency matrices $\boldsymbol{X} \in \mathbb{R}^{d \times w}$, which lack temporal granularity and grow with graph size, our **3D** sketch $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times w \times W}$ maintains fixed memory $\mathcal{O}(dwW)$ and supports real-time edge tracking in *sublinear space*. Edge updates run in constant time, $\mathcal{O}(1)$.

Algorithm 1 summarizes the edge processing pipeline, tensor updates, decay, sketching, and frequency estimation.

B. Frequency Estimation with Conservative Updates

We employ CMS-CU [8] as a lightweight data structure for tracking edge frequencies. For a given edge $e = (u, v, t)$, the *estimated frequency* of edge (u, v) at time t is denoted by $\hat{a}_{uv}(t)$, while the *cumulative historical freq.* up to t is $\hat{s}_{uv}(t)$.



1 Edge Streams Input
G_t
G_{t+1}
2 Temporal Binning Module
(2D Hash Projection)
$h_1, h_2, \dots$
hash layers
i $h_1(u_t, v_t)$
b $h_d(u_t, v_t)$
t j
(j, b_t)
$A[i, j, b_t]$
Updated d time bins
3 Multi-Layer Hashing
Uses d Pairwise-independent hash functions h functions
Each edge(u, v) is hashed to bucket $j = h_i(u, v)$ in row i
Set and updates index (i, j, b_t)
4 3D Tensor Sketch Construction
$\mathcal{S} \in \mathbb{R}^{d \times w \times t_w}$
d : Hash rows (layers)
w : Buckets
W : Time bins
(u, v, t)
W Length
w Width
d Depth bins
(i, j, b_t)
$f_{uvt}(\mathcal{S}) = \mathcal{S}[i][h_i(u, v)][b_t],\ b_t = \left\lfloor \frac{t}{\Delta} \right\rfloor$
Update Rule: Apply CMS-CU
$\hat{a}^{(i)}_{uv} = \min_j \mathcal{S}[j][h_j(u, v)][b_t]$ If it is min value
Output: Estimated Frequency
$\hat{a}_{uv}(t) = \min_{i=1}^{d}[f(\mathcal{S})]$
Cumulative Count
$\hat{s}_{uv}(t) = \sum_{k=1}^{b_t} \hat{a}_{uv}(k)$
5 Temporal Decay & Sliding Window Pruning
(Adaptive decay factor to reduce influence of old edges)
window size W
Time
$\mathcal{S}[i][j][b] \leftarrow \gamma \cdot \mathcal{S}[i][j][b]$, $\gamma \in (0, 1)$
Apply decay to S sketch tensor
Drops bins beyond sliding window W
6 Bayesian Inference & EWMA Thresholding
Compute posterior probability
$P(\text{Anomaly} \mid a, s, t)$
Computes smoothed score
$Z_t = \lambda X_t + (1 - \lambda) Z_{t-1}$, $\lambda \in (0, 1]$
τ_t T_t x_j X_t
Dynamic Threshold
$\tau_t = \mu_t + k \cdot \sigma_t$
Detection Rule: flags anomaly if score: $X_t > \tau_t$
Output: Anomaly Detection
v_1
v_2
v_k
$\text{Score}(t) = S_{uv}(k)$

Fig. 3. **Framework of ADAPTIVE-GRAPHSKETCH.** The model consist: (1) **Edge Inputs:** (u, v, t) form a dynamic stream. (2) **Temporal Binning:** Edges grouped into bins $b_t = \lfloor t/\Delta \rfloor$ via hash projection. (3) **Multi-Layer Hashing:** map edges to multiple hash layers $j = h_i(u, v)$, yielding (i, j, b_t) indices. (4) **3D Tensor Sketch:** Tensor $\mathcal{S} \in \mathbb{R}^{d \times w \times W}$ updated with CMS-CU for compact frequency tracking. (5) **Decay and Pruning:** Applies decay γ and retains W recent bins. (6) **Bayesian Inference:** with EWMA threshold τ_t flags anomalies where $\text{Score}_{uvt} > \tau_t$. Output is a ranked anomaly list of edges.

Algorithm 1 AGRAPHSKETCH: Multi-Layer Tensor Sketching

```
Input: edges $(u, v, t)$, decay $\gamma$, bin size $\Delta$, window $W$
Output: real-time anomaly score per edge $(u, v, t)$
1:  Initialize sketch tensor $\mathcal{S} \in \mathbb{R}^{d \times w \times W}$
2:  for each edge $(u, v, t)$, $b_t \leftarrow \left\lfloor \frac{t}{\Delta} \right\rfloor$        ▷ bin index
3:  for $i = 1$ to $d$ do                                ▷ CMS-CU update
4:      $j \leftarrow h_i(u, v)$
5:      if $\mathcal{S}[i][j][b_t] = \min_j \mathcal{S}[j][h_j(u, v)][b_t]$ then
6:          $\mathcal{S}[i][j][b_t] \leftarrow \mathcal{S}[i][j][b_t] + 1$
7:      end if
8:  end for
9:  for $b = 1$ to $W$ do                                ▷ decay + pruning
10:     $\mathcal{S}[i][j][b] \leftarrow \gamma \cdot \mathcal{S}[i][j][b]$
11:     if $b_t - b > W$ then $\mathcal{S}[i][j][b] \leftarrow 0$
12:     end if
13: end for
14: $\hat{a}_{uv}(t) \leftarrow \min_{i=1}^{d} \mathcal{S}[i][h_i(u, v)][b_t]$          ▷ freq. estimation
15: $\hat{s}_{uv}(t) \leftarrow \sum_{k=1}^{b_t} \hat{a}_{uv}(k)$
16: return BayesianScore$(\hat{a}_{uv}(t), \hat{s}_{uv}(t), t)$
```

To reduce bias from hash collisions, CMS-CU selectively increments only the counters with the current minimum values:

$$v_i \leftarrow v_i + 1 \quad \text{if} \quad v_i = \min_{j=1}^{d} v_j, \tag{11}$$

The estimated frequency of item x is then:

$$\hat{f}(x) = \min_{i=1}^{d} \text{CMS-CU}_{i, h_i(x)}. \tag{12}$$

C. Raw Anomaly Score Based on Frequency Deviation

Once the current frequency $\hat{a}_{uv}(t)$ and cumulative count $\hat{s}_{uv}(t)$ are estimated, we compute a *Raw Anomaly Score* to quantify how much recent edge activity deviates from expected behavior.

Let $a = \hat{a}_{uv}(t)$ and $s = \hat{s}_{uv}(t)$. The raw anomaly score is:

$$\text{RawScore}(u, v, t) = \frac{(a - \frac{s}{t})^2 \cdot t}{s \cdot (t - 1)}. \tag{13}$$

This formulation captures the squared deviation of current activity from historical average, normalized by variance, and highlights bursts or drops in edge behavior.

D. Adaptive Probabilistic Scoring via Bayesian Inference

Building on Equation 13, we model the current count a against historical trends using a Bayesian test. We assume mean $\mu = s/t$ and variance $\sigma^2 = s/t^2$ under normal behavior.

Anomalous behavior is captured by shifting the mean and inflating the variance, i.e., $\mu_A = \mu + \delta, \quad \sigma_A^2 = 4\sigma^2$.

$$P(a \mid \text{Anomaly}) = \frac{1}{\sqrt{2\pi\sigma_A^2}} \exp\left(-\frac{(a - \mu_A)^2}{2\sigma_A^2} \right). \tag{14}$$

Assuming a fixed prior $P(\text{Anomaly}) = p_0$, the posterior probability is given by Bayes' rule:

$$P(\text{Anomaly} \mid a) = \frac{P(a \mid \text{Anomaly}) \cdot p_0}{P(a)}, \tag{15}$$

where $P(a) = P(a \mid \text{Anomaly}) \cdot p_0 + P(a \mid \text{Normal}) \cdot (1 - p_0)$, and the posterior yields an adaptive anomaly score for (u, v).

E. Streaming Temporal Decay and Pruning

We apply two temporal mechanisms to emphasize recency and bound memory: (i) decay and (ii) sliding-window pruning.

a) Exponential Temporal Decay.: Before incoming edges at time t, the sketch tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d\times w\times W}$ is decayed along the temporal axis to reduce the impact of older interactions.

$$\boldsymbol{\mathcal{S}}[i][j][b] \leftarrow \gamma \cdot \boldsymbol{\mathcal{S}}[i][j][b], \quad \gamma \in (0,1), \qquad (16)$$
$$\forall\ i \in [1,d],\ j \in [1,w],\ b \in [1,W],$$

where b indexes the time bin and γ sets the decay rate. Smaller γ yield faster forgetting, giving more weight to recent edges.

b) Sliding Window Pruning.: To bound memory, we use a fixed-size sliding window that keeps only the most recent W bins (i.e., snapshots). Let b_t be the current bin at time t, and b an existing bin. Any bin outside the window is cleared:

$$\text{If } b_t - b > W \Rightarrow \boldsymbol{\mathcal{S}}[i][j][b] \leftarrow 0 \qquad (17)$$

Pruning operation removes stale slices, frees memory, and prevents scores from being biased by long-term drift or noise.

Algorithm 2 outlines how the posterior anomaly score is computed for each edge based on sketch-derived statistics.

Algorithm 2 Bayesian Posterior Anomaly Scoring

```
Input: Estimated frequency $\boldsymbol{a}$, cumulative $\boldsymbol{s}$, time bin $\boldsymbol{t}$
Output: Posterior anomaly score $P(\text{Anomaly} \mid a)$
1: Compute historical mean: $\mu \leftarrow \frac{s}{t}$, variance: $\sigma^2 \leftarrow \frac{s}{t^2}$
2: $P_{\text{normal}} \leftarrow \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)$    ▷ normal likelihood
3: $\mu_A \leftarrow \mu + \delta$
4: $\sigma_A^2 \leftarrow 4\sigma^2$
5: $P_{\text{anomaly}} \leftarrow \frac{1}{\sqrt{2\pi\sigma_A^2}} \exp\left(-\frac{(a-\mu_A)^2}{2\sigma_A^2}\right)$
6: $p_0 \leftarrow 0.05$    ▷ prior/posterior computation
7: $P(a) \leftarrow p_0 \cdot P_{\text{anomaly}} + (1-p_0) \cdot P_{\text{normal}}$
8: return $P(\text{Anomaly} \mid a) \leftarrow \frac{p_0 \cdot P_{\text{anomaly}}}{P(a)}$
```

F. Dynamic Thresholding with EWMA and FPR Control

Let $X_t = \text{Score}(u,v,t)$ be the anomaly score for edge (u,v) at time t. To smooth noise and control false positives, we apply thresholding EWMA [9]. The smoothed anomaly signal $\boldsymbol{Z}_t$ is:

$$\boldsymbol{Z}_t = \lambda X_t + (1-\lambda)\boldsymbol{Z}_{t-1}, \quad \lambda \in (0,1], \qquad (18)$$

where λ controls memory decay; larger values emphasize recent anomalies. The initial value is $\boldsymbol{Z}_0 = X_1$. The empirical mean μ_t and variance σ_t^2 of past scores up to time t are:

$$\mu_t = \frac{1}{t}\sum_{i=1}^{t} X_i, \qquad \sigma_t^2 = \frac{1}{t}\sum_{i=1}^{t}(X_i - \mu_t)^2 \qquad (19)$$

The dynamic threshold $\boldsymbol{\tau_t}$, with k sensitivity multiplier is:

$$\tau_t = \mu_t + k \cdot \sigma_t, \qquad (20)$$

a) Detection Rule.: An edge is anomalous if

$$X_t > \tau_t. \qquad (21)$$

b) False Positive Guarantee.: Using Chebyshev's inequality [19], the probability of false alarm is bounded as:

$$\text{FPR}_t \leq \frac{1}{k^2}. \qquad (22)$$

This guarantee in holds even under non-Gaussian noise.

Algorithm 3 Dynamic Thresholding and Anomaly Detection

```
Input: Score stream $\{X_t\}$, smoothing factor $\lambda$, sensitivity $k$
Output: Anomaly flags per timestamp $t$
1: Initialize $Z_0 \leftarrow X_1$, $\mu_0 \leftarrow X_1$, $\sigma_0 \leftarrow 0$
2: for $t = 2$ to $T$ do
3:     $Z_t \leftarrow \lambda X_t + (1-\lambda)Z_{t-1}$    ▷ EWMA update
4:     $\mu_t, \sigma_t \leftarrow$ empirical mean/variance of $\{X_i\}_{i=1}^t$
5:     $\tau_t \leftarrow \mu_t + k\sigma_t$
6:     Flag $(u,v,t)$ anomalous if $Z_t > \tau_t$, else normal
7: end for
8: return anomaly flags
```

Algorithm 3 implements the full EWMA-based adaptive thresholding procedure and variance for decision-making.

Lemma 1 (False Positive Control via Adaptive Thresholding)**.** *Let $\{X_t\}_{t=1}^T$ be anomaly scores with finite variance, and let μ_t, σ_t be the empirical mean and standard deviation computed up to time t (e.g., over a rolling window or EWMA-smoothed history). Using the adaptive threshold*

$$\tau_t = \mu_t + k \cdot \sigma_t \quad (k > 0),$$

the false positive rate (FPR) at time t satisfies

$$\text{FPR}_t = \Pr\big(X_t > \tau_t\big) \ \leq\ \frac{1}{k^2}. \qquad (23)$$

Proof. We apply Chebyshev's inequality [19] to the anomaly score X_t, with empirical mean μ_t and std. σ_t up to time t. For any $k > 0$, Chebyshev's inequality guarantees:

$$\Pr\big(|X_t - \mu_t| \geq k\,\sigma_t\big) \ \leq\ \frac{1}{k^2}. \qquad (24)$$

The upper-tail event $\{X_t > \mu_t + k\,\sigma_t\}$ is contained in the two-sided event $\{|X_t - \mu_t| \geq k\,\sigma_t\}$, hence

$$\Pr\big(X_t > \mu_t + k\,\sigma_t\big) \ \leq\ \Pr\big(|X_t - \mu_t| \geq k\,\sigma_t\big) \ \leq\ \frac{1}{k^2},$$

Therefore, the likelihood of falsely classifying a normal edge is bounded by $\Pr(\text{FPR}) \leq \frac{1}{k^2}$. This bound holds regardless of the distribution, provided it is unimodal with finite variance. □

G. Runtime and Output Tracking

Finally, each detected anomaly is logged as:

$$(u, v, t, \text{Score}_{uvt}), \quad \text{if Score}_{uvt} > \tau_t,$$

Score_{uvt} is the anomaly score of edge (u,v) at t. Let $N = |\mathcal{E}|$ be the total edges processed. The average per-edge runtime is

$$\text{AvgTime} = \frac{T_{\text{exec}}}{N}, \qquad (25)$$

where T_{exec} the total execution time for the entire stream.

Thus, ADAPTIVE-GRAPHSKETCH achieves scalable, low-latency anomaly detection in high-velocity streams.

V. EXPERIMENTS

We evaluate ADAPTIVE-GRAPHSKETCH for real-time detection in dynamic graphs, focusing on accuracy, runtime efficiency, and scalability across diverse graph structures.

TABLE III
DATASET STATISTICS.

Dataset	Nodes ($\|V\|$)	Edges ($\|E\|$)	Timestamps ($\|T\|$)
DARPA	25,525	4,554,344	46,567
ISCX-IDS2012	30,917	1,097,070	165,043
CIC-IDS2018	33,176	7,948,748	38,478
CIC-DDoS2019	1,290	20,364,525	12,224

A. Datasets

We experiment with four intrusion detection benchmarks, each serving as a testbed for evaluating different anomalies. DARPA [20] consists of $4.5M$ ***IP-IP*** communications among $25.5K$ nodes over $46.5K$ timestamps, offering a rich mix of attacks and temporal granularity. ISCX-IDS2012 [21] includes $1.1M$ labeled flows over $165K$ timestamps, capturing stealthy infiltration and brute-force. CIC-IDS2018 [22] contains $7.9M$ edges among $33.1K$ nodes over $38.5K$ timestamps, covering modern attacks such as botnets, DDoS, and port scans. CIC-DDoS2019 [23] contains $20.3M$ edges, $1.29K$ nodes, and $12.2K$ timestamps, characterized by high edge density and burst-heavy traffic. Table III summarizes dataset statistics.

B. Experimental Setup

Experiments are conducted on a 13th Gen Intel Core i9-13900 CPU (24 cores, 5.6 GHz), 32GB RAM, running Ubuntu 22.04. Our model, implemented in C++, is compared against state-of-the-art baselines: DENSESTREAM [2], SEDANSPOT [10], MIDAS-R [1], PENMINER [11], F-FADE [6], and ANOEDGE-G/L [12].

Metrics. We report ROC–AUC (area under the ROC) and wall-clock runtime. Our method uses a temporal tensor sketch with decay and EWMA smoothing. Sketch parameters includes: rows $r \in [2, 10]$; columns $c \in [10, 1300]$ (DARPA 10, ISCX 800, CIC-DDoS2019 1300); decay $\gamma \in [0.95, 0.99]$; step $\delta \in [10, 15]$; EWMA $\alpha \in [0.65, 0.95]$ (lower for bursty traffic like DDoS2019, higher for steadier flows like DARPA).

All experiment is repeated 5 times, and we report mean AUC and I/O runtime to mitigate hash randomness.

C. Accuracy

Table IV presents the ROC-AUC and runtime of ADAPTIVE-GRAPHSKETCH compared to the established baselines across four benchmark datasets.

Detection Performance: ADAPTIVE-GRAPHSKETCH achieves strong and consistent anomaly detection across all four benchmarks, with AUC scores of 0.9568 (DARPA), 0.8761 (ISCX-IDS2012), 0.9923 (CIC-IDS2018), and 0.9993 (CIC-DDoS2019). While not the top performer on DARPA and ISCX-IDS2012, where ANOEDGE-G/L slightly exceed its AUC, our method delivers superior efficiency and surpasses all baselines on large-scale, real-time datasets (CIC-IDS2018, CIC-DDoS2019), balancing high accuracy with low runtime.

Compared to MIDAS-R, our model improves AUC by 1% (DARPA), 22% (ISCX), 12% (CIC-IDS2018), and 4% (CIC-DDoS2019). Against F-FADE, gains are 4.3%, 71.8%, 17.7%, and 566%. Over SEDANSPOT, improvements reach 49.3% (DARPA), 51% (ISCX), 190.7% (CIC-IDS2018), and 76% (CIC-DDoS2019). PENMINER records reasonable AUC (0.872 on DARPA, 0.821 on CIC-IDS2018) but requires extreme runtime (24+ hours on CIC-DDoS2019). Owing to this limitation (as reported in [11] and confirmed in our trials), its AUC values are adopted from the ANOEDGE paper [12].

In contrast, our model outperforms PENMINER in accuracy on all datasets (9.7% on DARPA, 6.9% on IDS2018, 34% on IDS2012) and completes in under 10 seconds. ANOEDGE-G achieves high AUC (e.g., 0.970 on DARPA) but suffers from latency (up to 92s on CIC-DDoS2019), while ANOEDGE-L is faster yet less stably, with lower AUC (e.g., 0.927 on CIC-IDS2018). Overall, our method achieves higher AUC on 3 of 4 datasets, demonstrating both precision and robustness.

Running Time: Table IV reports runtimes. Our method is up to **36× faster** than SEDANSPOT and at least **5× faster** than ANOEDGE-G, while maintaining accuracy. F-FADE is unstable on large datasets, DENSESTREAM is slow on dense graphs, and PENMINER exceeds 24 hours on DDoS2019. Overall, our method delivers a superior speed–accuracy trade-off, enabling real-time detection in high velocity streams.

D. AUC vs. Running Time

To highlight the trade-off between accuracy and efficiency, Figure 4 plots AUC against runtime (log scale, seconds) on four datasets. ADAPTIVE-GRAPHSKETCH consistently achieves the highest AUC with much lower runtime. Compared to traditional baselines (e.g., MIDAS-R, SedanSpot, F-FADE), it is both faster and more accurate, and it outperforms or matches recent methods (e.g., AnoEdge-G, AnoEdge-L), showing a better balance of precision and scalability for real-time edge stream detection in large graphs.

E. Scalability and Robustness

We evaluate ADAPTIVE-GRAPHSKETCH scalability on four datasets. Figure 5 shows runtime (seconds) as edge volume grows from 10 to 10^6 on a log-log scale. A slope-1 line benchmarks linear growth. Our method scales near-linearly, processing 1M edges in under **0.52s** (0.469s DARPA, 0.464s ISCX, 0.519s CIC-IDS2018, 0.486s CIC-DDoS2019), confirming efficiency under both light and heavy volume.

We examine runtime sensitivity to the number of hash functions, a key parameter in sketch models. As shown in Figure 6, increasing hash rows from 2 to 10 yields steady, linear runtime growth. Even at full dataset scale (e.g., 20.3M edges for DDoS2019), processing remains efficient,

TABLE IV
AUC AND RUNNING TIME WHEN DETECTING EDGE ANOMALIES (AVERAGE OVER 5 RUNS)

Algorithm	DARPA	Time (s)	ISCX-IDS2012	Time (s)	CIC-IDS2018	Time (s)	CIC-DDoS2019	Time (s)
DENSESTREAM	0.5323 ±0.000	25.36	0.551 ±0.000	92.54	0.756 ±0.000	5186.9	0.263	99.78
SEDANSPOT	0.6408 ± 0.0025	78.13	0.5807 ± 0.0014	10.29	0.3413 ± 0.0341	110.02	0.5679 ± 0.0022	397.40
MIDAS-R	0.9493 ± 0.0006	0.203	0.7176 ± 0.0696	0.294	0.8834 ± 0.0011	0.361	0.9625 ± 0.0016	0.409
PENminer	0.872	18756	0.530	4680	0.821	36000	—	>86400
F-FADE	0.9173 ± 0.0041	132.29	0.5100 ± 0.0165	42.36	0.8432 ± 0.0038	98.04	0.1499 ± 0.1178	30.3
ANOEDGE-G	**0.970** ± 0.001	21.47	**0.954** ± 0.000	6.6329	0.975 ± 0.001	49.538	0.997 ± 0.001	92.85
ANOEDGE-L	**0.963** ± 0.001	**0.277**	**0.950** ± 0.000	0.58	0.927 ± 0.035	**0.4803**	0.998 ± 0.000	**0.83**
aGRAPHSKETCH	**0.9568 ± 0.000**	**2.15**	**0.8761 ± 0.000**	**0.5168**	**0.9923 ± 0.000**	**4.2006**	**0.9993 ± 0.000**	**8.948**

*All experiments are repeated 5 times. We report the mean ± standard deviation of ROC-AUC and the mean runtime in seconds.

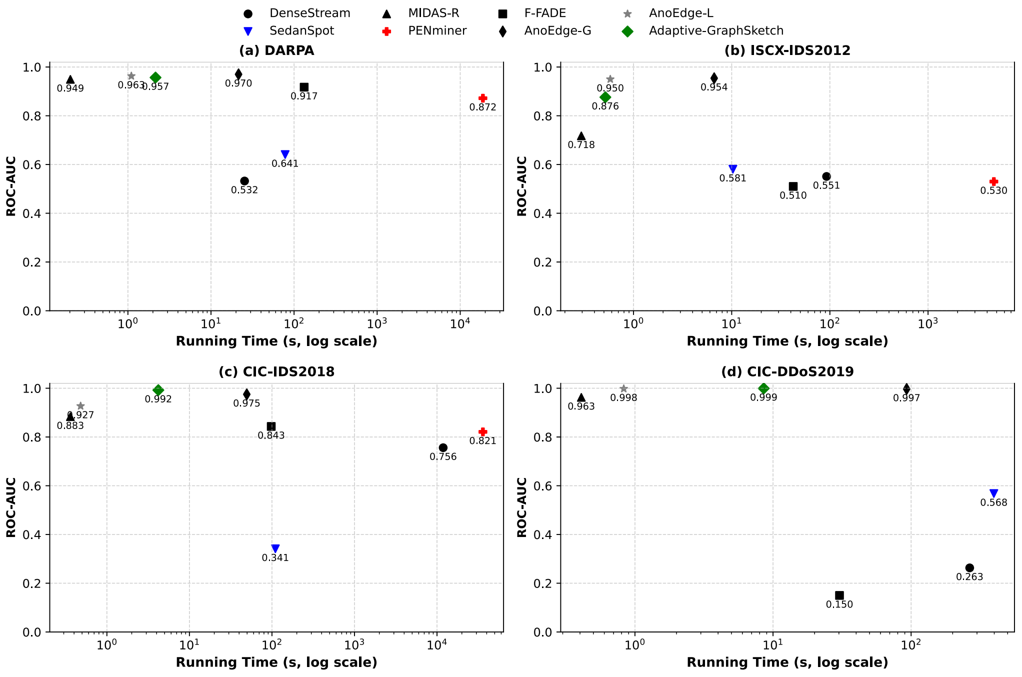


Fig. 4. **AUC vs. Running Time (log scale) across four benchmark datasets.** ADAPTIVE-GRAPHSKETCH achieves the best trade-off between accuracy and efficiency, outperforming state-of-the-art baselines across most settings.

completing in just **3.36s** with 10 hashes. This confirms robustness and scalability under rising sketch complexity.

F. Efficiency Analysis

The efficiency of ADAPTIVE-GRAPHSKETCH comes from its lightweight design that avoids costly operations. Unlike methods such as random walks [3] and matrix factorizations [5], our approach processes each edge in *constant time* using CMS with conservative updates [8]. Multiple sketches are compactly stored as time–protocol *tensors*, preserving temporal granularity while conserving memory. Anomaly scores are further smoothed using *Bayesian exponential moving averages*. Together with CPU-level vectorization and pruning, these elements enable real-time detection with linear scalability, as shown in Figures 5 and 6.

VI. CONCLUSION

In this paeper, we presented ADAPTIVE-GRAPHSKETCH, a real-time edge-anomaly detector for dynamic graphs that leverages tensor-based sketching and Bayesian smoothing. By integrating Count–Min Sketch with conservative updates (CMS-CU), exponential decay, and EWMA-based aggregation, the method achieves robust detection with sub-second latency and low memory. Experiments across four intrusion-detection benchmarks dataset shows that our method outperforms ANOEDGE-G/L, MIDAS-R, PENMINER, and F-FADE, with up to $+6.5\%$ AUC on CIC-IDS2018 and $+15.6\%$ on CIC-DDoS2019, while processing 20M edges in under 3.4s using 10 hashes. Unlike matrix factorization or random-walk methods, it supports $O(1)$ per-edge updates and scales linearly with edge volume and sketch complexity,

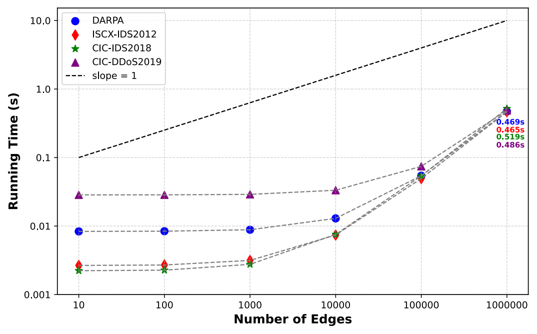


Fig. 5. **Scalability of Adaptive-GraphSketch with Number of Edges** . Runtime increases from 10 to 10^6, with a slope-1 line for linearity comparison. Final values at 10^6 edges are annotated below each marker.

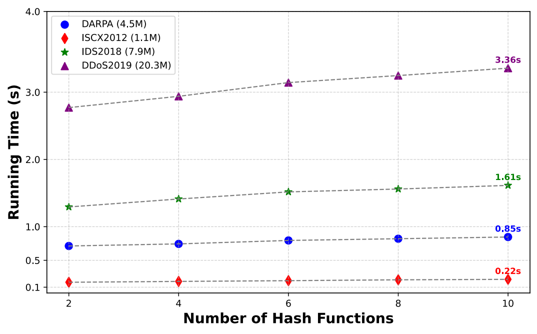


Fig. 6. **Scalability of Adaptive-GraphSketch with Hash Functions.** Runtime on four datasets: DARPA (4.5M), ISCX2012 (1.1M), IDS2018 (7.9M), DDoS2019 (20.3M). Each curve shows processing time as hash rows increase from 2 to 10. Final runtimes at $r = 10$ are annotated beside markers.

making it suitable for edge/real-time deployments. Future work includes extending the framework for multi-modal anomaly detection with content-aware sketches. We also aim to optimize runtime via parallelization and hardware acceleration, while addressing temporal drift and heterogeneity in evolving networks.

Future work: includes extending the framework multimodal, content-aware sketches; accelerate via parallelization/hardware; and address temporal drift and heterogeneity in evolving networks.

Acknowledgments

We thank the CS Department at Tennessee Tech University and the Machine Intelligence and Data Science (MInDS) Center for providing resources to work on this project.

References

[1] S. Bhatia, R. Liu, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, "Real-time anomaly detection in edge streams," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 4, pp. 1–22, 2022.

[2] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, "Densealert: Incremental dense-subtensor detection in tensor streams," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1057–1066.

[3] O. A. Ekle and W. Eberle, "Anomaly detection in dynamic graphs: A comprehensive survey," *ACM Transactions on Knowledge Discovery from Data*, 2024.

[4] O. A. Ekle, W. Eberle, and J. Christopher, "Adaptive decayrank: Real-time anomaly detection in dynamic graphs with bayesian pagerank updates," *Applied Sciences*, vol. 15, no. 6, p. 3360, 2025.

[5] Y. Xie, W. Wang, M. Shao, T. Li, and Y. Yu, "Multi-view change point detection in dynamic networks," *Information Sciences*, vol. 629, pp. 344–357, 2023.

[6] Y.-Y. Chang, P. Li, R. Sosic, M. Afifi, M. Schweighauser, and J. Leskovec, "F-fade: Frequency factorization for anomaly detection in edge streams," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 589–597.

[7] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 239–247.

[8] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Transactions on Computer Systems (TOCS)*, vol. 21, no. 3, pp. 270–313, 2003.

[9] J. M. Lucas and M. S. Saccucci, "Exponentially weighted moving average control schemes: properties and enhancements," *Technometrics*, vol. 32, no. 1, pp. 1–12, 1990.

[10] D. Eswaran and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," in *2018 IEEE International conference on data mining (ICDM)*. IEEE, 2018, pp. 953–958.

[11] C. Belth, X. Zheng, and D. Koutra, "Mining persistent activity in continually evolving networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 934–944.

[12] S. Bhatia, M. Wadhwa, K. Kawaguchi, N. Shah, P. S. Yu, and B. Hooi, "Sketch-based anomaly detection in streaming graphs," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 93–104.

[13] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: a survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, pp. 223–247, 2015.

[14] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," in *Proceedings of the 2016 SIAM international conference on data mining*. SIAM, 2016, pp. 189–197.

[15] O. A. Ekle and W. Eberle, "Dynamic pagerank with decay: A modified approach for node anomaly detection in evolving graph streams," in *The International FLAIRS Conference Proceedings*, vol. 37, 2024.

[16] Y. Wang, A. Chakrabarti, D. Sivakoff, and S. Parthasarathy, "Fast change point detection on dynamic social networks," *arXiv preprint arXiv:1705.07325*, 2017.

[17] Y. B. Mazziane, S. Alouf, and G. Neglia, "A formal analysis of the count-min sketch with conservative updates," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[18] L. Wasserman, *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[19] W. Mendenhall, R. J. Beaver, and B. M. Beaver, *Introduction to probability and statistics*. Cengage Learning, 2012.

[20] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 darpa off-line intrusion detection evaluation," in *Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000 Toulouse, France, October 2–4, 2000 Proceedings 3*. Springer, 2000, pp. 162–182.

[21] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[22] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.

[23] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 international carnahan conference on security technology (ICCST)*. IEEE, 2019, pp. 1–8.