SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V13I2P113

Volume 13 Issue 2, 160-171, February 2026
© 2026 Seventh Sense Research Group®

Original Article

A Streaming IRFCM Framework for Real-Time Anomaly Detection in IoT Sensor Networks

Sumrana Siddiqui[1], Nandita Bhanja Chaudhuri[2]

[1,2]*Computer Science Engineering Department, GITAM School of Technology, Telangana, India.*

[2]*Corresponding Author : nchaudhu@gitam.edu*

Received: 18 November 2025 Revised: 22 December 2025 Accepted: 19 January 2026 Published: 17 February 2026

Abstract *- The proliferation of Internet of Things (IoT) devices has led to unprecedented streams of high-dimensional, noisy sensor data that require advanced clustering and anomaly detection techniques. This study proposes a novel Streaming Integrated Robust Fuzzy Clustering Module (S-IRFCM), extending the IRFCM framework to enable adaptive, real-time soft clustering and anomaly detection in large-scale IoT sensor networks. S-IRFCM integrates online dimensionality reduction, incremental centroid updating, and dynamic outlier trimming for robust, efficient, and interpretable clustering. Experimental evaluation demonstrates superior robustness in noisy, drifting environments where standard streaming baselines fail, alongside orders-of-magnitude faster processing than batch fuzzy methods. These results confirm that S-IRFCM achieves scalable, stable clustering in dynamic and noisy big data environments, providing a new paradigm for IoT-enabled analytics applications.*

Keywords *- Streaming Clustering, IRFCM, IoT Anomaly Detection, Soft Clustering, Online PCA, Outlier Trimming, High-Dimensional Data, Scalability, Sensor Fault Detection, Data Streams, Adaptive Clustering.*

1. Introduction

The swift development of the Internet of Things (IoT) has led to the contribution of an immense number of sensor implementations in a wide variety of industries, such as smart cities, automation of industries, environmental management, and healthcare [1, 2]. These sensors produce continuous high-dimensional and heterogeneous data, which pose special problems in terms of data analysis and decision support. Millions of gadgets, including thermometers and cameras, health and wearable gadgets, and numerous others, generate data at an unprecedented rate and scale, and the conventional data processing and clustering models are proving not as robust as they once were [3]. The data streams can be very noisy, incomplete, and dynamic, i.e., sensor malfunctions, environmental effects, and changing system behavior. Insightful patterns and anomalous noise of such data are important to making timely faults, predictive maintenance, and improved situational awareness [4].

Clustering methods can be used as the basis of the concept of structures in unlabeled data of the IoT and the detection of anomalies by steering clear of data points that do not fit the given patterns. These include soft clustering algorithms like Fuzzy C-Means and Possibilistic C-Means, which offer the flexibility of probabilistic cluster memberships valid in the representation of the overlapping or ambiguous cluster boundaries of absolute sensor data in the world. Although they have advantages, traditional applications of these algorithms have significant challenges when using them in data streams in the IoT- mainly because they are sensitive to noise, they cannot be effectively scaled to high-dimensional, large-scale data, and they are limited by batch processing.

The Integrated Robust Fuzzy Clustering Module (IRFCM) was proposed to overcome these shortcomings by being a hybrid algorithm that integrates dimensionality reduction, optimized centroid initiation, a blend of fuzzy and possibilistic memberships, and dynamic outlier trimming. IRFCM was shown to perform well on large-scale batch datasets of diverse cybersecurity, healthcare, and data science domains, as well as on clustering quality. Nonetheless, the batch character and use of fixed sets of data restrict its use in real-time applications to continuous IoT systems, where the streams of data do not stop, and the state of the system continuously changes.

In the meantime, the studies concerning streaming and online clustering have examined scalable techniques applicable to real-time data, yet at the cost of resiliency and flexibility. Some techniques, like StreamKM++, are more computational, and they make use of micro-clustering but mainly use challenging cluster assignments and do not mitigate noise and outliers. Therefore, it does not perform well when identifying small or emerging sensor anomalies when using non-stationary streams with noise [5, 6]. This study addresses this research gap by proposing a novel Streaming

Integrated Robust Fuzzy Clustering Module (S-IRFCM), which adapts the strengths of IRFCM to streaming environments through incremental dimensionality reduction, adaptive membership updating, sliding-window clustering, and real-time outlier trimming. The proposed framework enables scalable, noise-resilient, and adaptive clustering suitable for large-scale IoT sensor networks.

The significant contributions of this research are summarized as follows: Designing a novel streaming fuzzy-possibilistic clustering framework that extends IRFCM for real-time IoT data streams by integrating Incremental PCA for adaptive dimensionality reduction in streaming environments. A hybrid membership function is used by combining FCM and PCM to enhance robustness and interpretability. The algorithm implements dynamic outlier trimming and anomaly detection in real time. Extensive experimental validation using three benchmark IoT datasets, demonstrating superior robustness, scalability, and anomaly detection performance over state-of-the-art methods, has been conducted.

2. Related Works

2.1. Soft Clustering Methods in High-Dimensional and Noisy Data

The clustering approaches that permit probabilistic or fuzzy memberships have played significant roles in the analysis of complex data sets, whereby the lines between the clusters are vague or overlap. These include Fuzzy C-Means (FCM), which has been a paradigm algorithm, with its intuitive membership, with its application to a wide variety of areas, including image segmentation, bioinformatics, and text mining [7]. Nevertheless, the use of Euclidean distances and the fact that membership within clusters adds to one make FCM susceptible to noise and outliers, which usually causes cluster accuracy to deteriorate in practice when using noisy datasets.

The limitations of FCM, Possibilistic C-Means (PCM), were presented, which do not have the probabilistic membership normalization restrictions permitting absolute membership or typicality values, irrespective of the membership of the other clusters [8]. PCM is also more resistant to outliers because they decrease the effect of outliers on the centroid of the clusters. Nonetheless, PCM has hyperparameters that need to be carefully tuned and converges more slowly, which makes it more challenging to apply in large-scale settings. In addition, both FCM and PCM have similar problems related to high-dimensional data: distance metrics lose their discriminative ability, and both algorithms are poorly scaled with the size of the dataset [9, 10].

2.2. The Genesis and Limitations of IRFCM

This problem was solved by the suggestion of the Integrated Robust Fuzzy Clustering Module (IRFCM) to correspond to the compound problems of noisy, high-dimensional, and voluminous data. IRFCM integrates dimensionality reduction (through PCA or Truncated SVD), more complex centroid initialization (KMeans++), Hybrid Fuzzy Possibilistic Memberships, and Outlier Trimming (that operates dynamically), which can be scaled to produce consistent, scalable clustering results. It has been experimentally proven that the quality of IRFCM clustering (e.g., higher silhouette scores and smaller Davies-Bouldin indices) is better than classic soft clustering baselines.

However, the IRFCM model is batch-based in nature, where the entire body of data is loaded and processed in-memory, which is a harsh requirement in streaming data applications involving IoT, where sensor data comes in continuously in potentially infinite amounts and changing distributions. As a result, it is highly needed to revamp the practical approach to online or incremental clustering of the IRFCM that can be effectively applied in the streaming environment.

2.3. IoT and Data Stream Environment Clustering

The IoT conditions add more challenges: the volume of data is insurmountable, it moves at high velocity, there is noise, and drift of sensors, and non-stationary data distributions [11]. The fast development of IoT has stimulated the increasing literature on stream-based clustering algorithms. Among them, one can distinguish StreamKM++ that offers scalable solutions by maintaining micro-clusters in the form of incremental maintenance and summary statistics. It is an efficient algorithmic method of handling data streams with high velocity by updating summaries without remembering all of the past data, thus limiting memory and computational complexity.

However, the conventional methods of stream clustering mainly utilize hard clustering, where sharp cluster boundaries are used that are sometimes constraining in situations where sensor values make smooth transitions or overlaps. Moreover, noise filtering systems or outlier detection systems are not explicitly mentioned in many streaming systems, something that is essential because of how many real-world sensor implementations are noisy and characterized by errors. Consequently, they can either categorize anomalies wrongly or not identify the minor faults in the sensor network.

2.4. Progress in the Direction of Robust and Hybrid Streaming Clustering

Some progress has been made towards incorporating robustness into streaming clustering, using online variants of fuzzy clustering or algorithms that try to use trimmed or weighted clustering objectives. As an example, there are incremental fuzzy algorithms that are offered to modify the cluster prototypes and membership degrees with incoming data to achieve a tradeoff between adaptability and stability. Other studies have tried to trim outliers in online fashion dynamically and hybridized possibilistic memberships to be more resilient to noise [12, 13]. However, issues in finding the right level of real-time responsiveness, high-quality clustering, and controllable computational costs exist. Most of these hybrid approaches are not rigorously validated on large-scale and high-dimensional datasets of IoT sensors, or do not leverage the extensive dimensionality reduction and initialization capabilities of the IRFCM framework.

2.5. Research Gap and Motivation

Chen et. al. [14] emphasized the difficulty of introducing adaptive real-time clustering models that are resistant to noise and concept drift to industrial IoT environments. Previous study outlined the shortcomings of the available techniques of clustering when dealing with the heterogeneous, noisy healthcare IoT data streams. Previous study suggested online fuzzy clustering involving adaptive memberships but observed that there was a clear problem of computational overhead and sensitivity of the parameter. According to a recent survey conducted by Previous study, there is no evidence of there being any integration between robust outlier detection with fuzzy clustering in streaming IoT architectures. In a study by Previous study, there is a call to develop clustering algorithms that integrate adaptive fuzzy memberships with incremental dimensionality reduction to facilitate the dynamism of data stream properties.

Although, IRFCM provides an excellent example of robust, hybrid soft clustering in batch big data, and streaming clustering algorithms can be effectively used to scale, perform incremental updates, no existing solution has hitherto been able to address the weaknesses of the solution on the challenging streaming IoT data environment by extending the strengths of IRFCM namely, robustness to noise and outliers, dimensionality reduction and interpretability. A significant loophole is still identified in terms of a real-time, scalable, and adaptive fuzzy clustering approach that would provide high-quality clustering as well as timely detection of anomalies in sensor-intensive systems. The reason is that this drives the exploration and design of the Streaming IRFCM to support the technical and operational needs of the current IoT-oriented big data analytics.

3. Methodology

3.1. Streaming IRFCM (S-IRFCM)

The Streaming Integrated Robust Fuzzy Clustering Module (S-IRFCM) is an extension of the IRFCM algorithm in the batch mode to handle continuous data streams in the IoT. This is required due to the fact that the IoT sensor networks produce high-velocity data that has changing patterns, noise, and concept drift, which cannot be effectively processed using batch-based algorithms such as the initial IRFCM. Online updates are included in S-IRFCM to preserve clustering state over time, which provides limited memory consumption and gives real-time detection of anomaly [15]. Its core building blocks (dimensionality reduction, fuzzy and possibilistic clustering, and outlier trimming) [16] are modified to operate in an incremental fashion, inspired by streaming systems such as CluStream that rely on micro-clusters summarizing changing data. This middle ground solution is a compromise between the keyboard quietness and softness of IRFCM and the scalability of the IoT applications, including sensor fault detection during environmental monitoring.

3.1.1. Online Dimensionality Reduction

Multidimensional IoT data, e.g., multi-sensor (e.g., temperature, humidity, acceleration) readings, is susceptible to the curse of dimensionality, and clustering is computationally inexpensive and less efficient. S-IRFCM applies Incremental Principal Component Analysis (IncrementalPCA) to online dimensionality reduction to cope with this in a streaming environment. IncrementalPCA is a memory-efficient method (compared to batch PCA) that, unlike batch PCA, maintains a low-rank approximation of the principal components, which gets updated in real time as new data batches are received, being independent of the size of the complete data set. This algorithm operates in mini-batches, trains the model to completion, and recalculates the transformation matrix without reloading data.

IncrementalPCA addresses the limitations on memory of streaming applications, where IoT data may exceed the working memory, and retains the most variance-representing features to be used in later clustering. It has also been demonstrated to be able to deal with big datasets, as seen in the visualization of streaming data [17]. The concept of S-IRFCM is to present the data to a smaller space after each batch so that the noise is limited and the computational cost is reduced without compromising clustering quality. Incremental PCA updates the mean and principal components as new data batches arrive.

For a new batch $X_t \in \mathbb{R}^{n_t \times p}$ at time *t*, with n_t samples and *p* features:

a) Mean update: The mean is calculated as shown in Equation (1) below:

$$\mu_{\text{new}} = \frac{n_{\text{old}}\mu_{\text{old}} + n_t \bar{x}_t}{n_{\text{old}} + n_t} \tag{1}$$

where:

μ_{old}: Previous mean vector (from prior batches),
$\bar{x}_t = \frac{1}{n_t}\sum_{i=1}^{n_t} x_i$: Mean of the current batch
n_{old}: Total samples processed before *t*,
n_t: Number of samples in the current batch.

b) Covariance Update: The covariance matrix is updated incrementally as shown in Equation (2) using a rank-k approximation:

$$C_{\text{new}} = \frac{n_{\text{old}} C_{\text{old}} + \sum_{i=1}^{n_t} (x_i - \mu_{\text{new}})(x_i - \mu_{\text{new}})^T}{n_{\text{old}} + n_t} \qquad (2)$$

where C_{old} is the previous covariance estimate. In practice, Incremental PCA uses a partial SVD to update the principal components $V \in \mathbb{R}^{p \times d}$ projecting data to *d* dimensions.

c) Projection: Transform batch X_t to reduced space:

$$X_t^{\text{proj}} = (X_t - \mu_{\text{new}})V \qquad (3)$$

3.1.2. Incremental Clustering

Offline FCM and PCM assume complete access to the data to perform an iterative update, which is not possible with streams. S-IRFCM supports sliding windows or mini-batches to perform incremental clustering and keep the centroid and memberships running. This is based on extensions of FCM to data streams, in which clusters are updated in sequence without re-computing on past data [18].

Memberships and centroids are recalculated with each incoming batch, and the decay factors are used to place more emphasis on the recent data and to address concept drift.

Constant-time processing with each batch is guaranteed by incremental updates, which are essential to real-time anomaly detection in IoT, whose delays may cause important events such as sensor failures. This methodology is consistent with variants of online fuzzy c-means that have proven to be efficient for cluster streaming data [19].

Streaming data, Adapt FCM and PCM with incremental updates. Using a decay factor to adapt to concept drift, incrementally update centroids and memberships with each batch.

a) Fuzzy C-Means (FCM): For batch $X_t \in \mathbb{R}^{n_t \times d}$ (after PCA projection), with c clusters and fuzzifier *m*>1 :

The Membership Update is done as shown in Equation (4) below:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - v_j\|^2}{\|x_i - v_k\|^2}\right)^{1/(m-1)}} \qquad (4)$$

where u_{ij} is the membership of the point $x_i \in X_t$ tocluster *j*, and v_j is the centroid of cluster *j*.

The Centroid is Updated Incrementally as shown in Equation (5) below:

$$v_j^{\text{new}} = \frac{\lambda \sum_{\text{prev}} u_{ij}^m x_i + \sum_{i=1}^{n_t} u_{ij}^m x_i}{\lambda \sum_{\text{prev}} u_{ij}^m + \sum_{i=1}^{n_t} u_{ij}^m} \qquad (5)$$

Where:

- $\lambda \in (0,1]$: Decay factor to weight past data.
- $\sum_{\text{prev}} u_{ij}^m x_i$: Accumulated weighted sum of points from previous batches for cluster *j*.
- $\sum_{\text{prev}} u_{ij}^m$: Accumulated sum of membership weights.
- The numerator updates the centroid by combining past contributions (decayed) with the current batch's contribution.

b) Possibilistic C-Means (PCM): PCM uses possibilistic memberships to enhance noise resilience, with fuzzifier *q*≥1.

Membership Update is done as shown in Equation (6) below:

$$t_{ij} = \frac{1}{1 + \left(\frac{\|x_i - v_j\|^2}{\eta_j}\right)^{1/(q-1)}} \qquad (6)$$

where η_j is the scale parameter for cluster *j*, estimated as the average intra-cluster distance as shown in Equation (7) below:

$$\eta_j = \frac{\sum_{i=1}^{n_t} t_{ij}^q \|x_i - v_j\|^2}{\sum_{i=1}^{n_t} t_{ij}^q} \qquad (7)$$

The Centroid is Updated Incrementally as shown in Equation (8) below:

$$v_j^{\text{new}} = \frac{\lambda \sum_{\text{prev}} t_{ij}^q x_i + \sum_{i=1}^{n_t} t_{ij}^q x_i}{\lambda \sum_{\text{prev}} t_{ij}^q + \sum_{i=1}^{n_t} t_{ij}^q} \qquad (8)$$

3.1.3. Dynamic Outlier Trimming

Outliers in IoT streams, such as erroneous sensor readings, can skew clusters. S-IRFCM implements dynamic trimming using a moving window to identify and exclude points based on distance thresholds or low memberships. For each batch, points with distances > (1 - trim_proportion) percentile or memberships <min_threshold are trimmed and flagged as anomalies. This adaptive mechanism enhances robustness in noisy streams, preventing distortion of centroids, as seen in fuzzy-entropy-based online FCM variants that improve performance by handling outliers dynamically. It complements PCM's noise resilience and is essential for IoT, where anomalies like faults need real-time flagging.

Identify and flag outliers (e.g., faulty sensor readings) in real-time to prevent skewing clusters and to detect anomalies.

a) Outlier Score: For each point $x_i \in X_t$ Compute the score using Equation (9) below:

$$\text{score}_i = \max_j(1 - t_{ij}) \cdot \|x_i - v_j\|^2 \quad (9)$$

where t_{ij} is the PCM membership, emphasizing possibilistic sensitivity to outliers.

b) Trimming Rule: Flag point $x_i$as an outlier if:

$\text{score}_i > percentile(scores, 1 - trim\_proportion)$ or $\max_j u_{ij} < min\_membership$ (e.g., 0.1).

c) Anomaly Flagging: Outliers are labeled as anomalies and excluded from centroid updates.

3.1.4. Hybrid Memberships

Combine FCM and PCM memberships as shown in Equation (10) below to balance probabilistic and possibilistic properties, adapting to noise and drift**.** The hybrid balances FCM's probabilistic assignments with PCM's possibilistic robustness, improving handling of overlapping clusters in IoT data. Adaptive alpha, inspired by hybrid fuzzy methods, responds to concept drift, enhancing stability in evolving streams.

$$u_{ij} = \frac{\alpha u_{\text{FCM},ij} + (1-\alpha) t_{\text{PCM},ij}}{\sum_{j=1}^{c} [\alpha u_{\text{FCM},ij} + (1-\alpha) t_{\text{PCM},ij}]} \quad (10)$$

3.1.5 Algorithm Steps

S-IRFCM Algorithm

Bootstrap Phase

Step 1: Load Initial Window Load the first window ($X_1 \in \mathbb{R}^{w \times p}$), where (w) is the window size (e.g., 2000) and (p) is the number of features.

Step 2:Preprocess Data

- Impute missing values using mean imputation per feature.
- Standardize features: ($x_i = \frac{x_i - \mu}{\sigma}$).

Step 3:Apply Incremental PCA

- Fit IncrementalPCA to (X_1), reducing to d dimensions (e.g., (d = 3)).
- Compute mean: ($\mu = \frac{1}{w}\sum_{i=1}^{w} x_i$).
- Apply partial SVD to obtain components ($V \in \mathbb{R}^{p \times d}$).
- Project data: ($X_1^{\text{proj}} = (X_1 - \mu)V$).

Step 4: Initialize Centroids Apply K-Means++ to (X_1^{proj})to initialize centroids

($V = v_1, v_2, \dots, v_c$) (e.g., (c = 5)).

Step 5: Compute FCM Memberships For each point $\left(x_i \in X_1^{\text{proj}}\right), compute$:

$[u_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{|x_i - v_j|^2}{|x_i - v_k|^2})^{1/(m-1)}}]$where (m = 2.0).

Step 6:Compute PCM Memberships

- Estimate(η_j): $[\eta_j = \frac{\sum_{i=1}^{w} u_{ij}^{m} |x_i - v_j|^2}{\sum_{i=1}^{w} u_{ij}^{m}}]$
- Compute memberships: $[t_{ij} = \frac{1}{1 + (\frac{|x_i - v_j|^2}{\eta_j})^{1/(q-1)}}]$

where (q = 2.0).

Step 7: Fuse Memberships

- Compute hybrid memberships: $\left[u_{ij} = \frac{\alpha u_{\text{FCM},ij} + (1-\alpha) t_{\text{PCM},ij}}{\sum_{j=1}^{c} [\alpha u_{\text{FCM},ij} + (1-\alpha) t_{\text{PCM},ij}]}\right]$where ($\alpha = 0.7$).

Step 8: Trim Outliers

- Compute outlier scores: $score_I = \left(\max_j(1 - tij) \cdot \left|x_i - v_j\right|^2\right)$.
- Flag points if (score_i>percentile(scores, 1 − trim_proportion)) or $\max_j u_{ij}$< 0.1

Step 9:Store Micro-Cluster Statistics

- Store centroids (V), accumulated weights ($\sum u_{ij}^{m}$), ($\sum t_{ij}^{q}$), and sample counts per cluster.

Streaming Phase

Step 10: Load and Preprocess New Batch Load batch ($X_t \in \mathbb{R}^{n_t \times p}$), impute missing values, and standardize.

Step 11:Update Incremental PCA

- Update mean: ($\mu_{\text{new}} = \frac{n_{\text{old}}\mu_{\text{old}} + n_t \bar{x}t}{nold + n_t}$).
- Update components (V) using partial SVD.

$Project$: $\left(X_t^{\text{proj}} = (X_t - \mu_{\text{new}})V\right)$.

Step 12: Detect Concept Drift Compute entropy: $\left[H = -\sum_{i=1}^{n_t} \sum_{j=1}^{c} u_{ij} \log(u_{ij})\right]$

If ($H > H_{\text{threshold}}$), $increase(\alpha) by 0.1 (upto 0.9)$.

Step 13: Update FCM Memberships and Centroids

Compute distances: $(\mathrm{dists}_{ij} = |x_i - v_j|^2)$.

Update memberships: $\left(u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\mathrm{dists}ij}{\mathrm{dists}ik}\right)^{\frac{1}{m-1}}}\right)$.

Update centroids:

$$\left[v_j^{\mathrm{new}} = \frac{\lambda \sum_{\mathrm{prev}} u_{ij}^m x_i + \sum_{i=1}^{n_t} u_{ij}^m x_i}{\lambda \sum_{\mathrm{prev}} u_{ij}^m + \sum_{i=1}^{n_t} u_{ij}^m}\right] where (\lambda = 0.9).$$

Step 14:Update PCM Memberships and Centroids

$$Update(\eta_j): \left[\eta_j = \frac{\sum_{i=1}^{n_t} t_{ij}^q |x_i - v_j|^2}{\sum_{i=1}^{n_t} t_{ij}^q}\right]$$

Compute memberships: $\left[t_{ij} = \frac{1}{1+\left(\frac{|x_i - v_j|^2}{\eta_j}\right)^{\frac{1}{q-1}}}\right]$

Update centroids: $\left[v_j^{\mathrm{new}} = \frac{\lambda \sum_{\mathrm{prev}} t_{ij}^q x_i + \sum_{i=1}^{n_t} t_{ij}^q x_i}{\lambda \sum_{\mathrm{prev}} t_{ij}^q + \sum_{i=1}^{n_t} t_{ij}^q}\right]$

Step 15: Fuse Memberships Compute hybrid memberships as in Step 7, using updated (α).

Step 16: Trim Outliers and Flag Anomalies Compute scores and flag outliers as in Step 8, labeling them as anomalies.

Step 17: Assign Clusters Assign each point (x_i) to $cluster(j = \arg\max_j u_{ij})$.

Step 18: Update Micro-Cluster Statistics Update $\left(\sum u_{ij}^m\right), \left(\sum t_{ij}^q\right),$ and it counts with decay (λ).

Step 19: Output Results Output cluster assignments and anomaly flags for (X_t).

Step 20: Step Loop or Terminate If more batches remain, loop to Step 10; else, optionally perform offline reclustering.

4. Experimental Setup

The experiment will be conducted in a way that the performance of the Streaming Integrated Robust Fuzzy Clustering Module (S-IRFCM) module in real-time anomaly detection of Internet of Things sensor networks will be critically tested. Such an arrangement answers the research questions by concentrating on quality clustering, accuracy of anomaly detection, computational efficiency, and stability in different situations. The experiments are done in controlled and reproducible replications, simulating the streaming of data from public datasets of the IoT. These configurations comprise preprocessing (e.g., missing values (mean imputation), standardization), batch processing to simulate real-time streams (e.g., window sizes of 1,000-5,000 points), as well as several independent runs to provide statistical reliability. Each experiment will be done 5 times using various random seeds to determine stability, and the results will be presented in the form of means and standard deviations. Scalability. In order to be scalable, datasets are managed as streams sequentially, with labels on the anomalies where supervised metrics are possible. The assessment to be used is based on the best practices in streaming clustering studies that have focused on the use of internal (unsupervised) and external (supervised) measures of validation. Details of the hardware and software are provided so the results can be replicated.

4.1. Datasets

In order to prove the usefulness of S-IRFCM in various IoT environments, three open datasets are chosen, which are the monitoring of environmental conditions, the smart infrastructure of a city, and network security. These data collections have IoT stream properties such as high dimensionality, noise, missing values, and anomalies. The datasets are obtained by utilizing authoritative databases such as Kaggle and scholarly libraries, and consumed as streams by breaking them down into sequential batches. The steps of preprocessing are imputation of missing values with the mean value, standardization of data to a mean of zero and unit variance, and dimensional reduction through IncrementalPCA to 3-5. Anomalies are either born or assigned for evaluation. The selection considers datasets that are typically employed in the IoT anomaly detection benchmarks to make them comparable with previous studies.

Intel Berkeley Research Lab Dataset: This is a dataset comprising around 2.3 M readings of 54 Mica2Dot sensors in the Intel Berkeley Research Lab on 28th February and 5th April, 2004. It has features such as timestamp, mote ID, temperature, humidity, light, and voltage, which are sampled at intervals of 31 seconds. The data represents real-world anomalies, e.g., sensor malfunctions, battery depletion (recorded in voltage drops), and environmental noise (e.g., outliers in temperature or humidity because of placement variations). Size: ~100 MB compressed. It can be used to test the noise resilience and outlier detection in indor environmental monitoring. The data is also publicly accessible on Kaggle and the MIT CSAIL archive. In streaming simulation, the data is presented in chronological sequence, where deviations are detected by ground-truth labels in previous simulations (e.g., abrupt voltage drops indicating faults) [20].

CIC DDoS 2019 Dataset: An extensive set of DDoS attack data of benign and the latest common attack types, which mimic actual real-life network traffic (PCAPs). It contains results of network traffic analysis of labeled flows, in terms of their timestamp, source or destination IP addresses, ports, and protocols. It has 13 DDoS attack variants targeting 12 victim machines. In this experiment, particular subsets of attacks were applied to the stream in order to evaluate the capability of the algorithm to read through high velocity and intricate attack patterns [21].

ML-EdgeIIoT Dataset: A multifaceted cybersecurity dataset tailored to the applications of the Internet of Industrial Things (IIoT). It incorporates regular traffic with other attack vectors and contains both the network traffic characteristics and the physical sensor measurements. This hybrid characteristic is what causes it to be a good benchmark to use in gauging the capacity of the S-IRFCM to recognize anomalies in complex, high-dimensional sensor-network settings [22]. Streaming simulations have memory limitations, so the subsets are 500,000-1,000,000 samples per run, in order to preserve the time sequence to allow realistic evaluation.

4.2. Comparative Algorithmic Analysis

S-IRFCM will be contrasted with a combination of the batch-mode and streaming baselines to demonstrate its strengths related to robustness and efficiency. Baselines are chosen due to having been used in the literature on clustering and IoT anomaly detection (implemented in standard libraries, e.g., scikit-learn with KMeans, fcmeans with FCM/PCM). To make a fair comparison, all the methods are preprocessed with the same method and evaluated with the same streaming batches.

- Fuzzy C-Means (FCM): Soft clustering algorithm, which uses probabilistic memberships, which reduces the objective function by updating them iteratively. It is also noise sensitive and forms a foundation for fuzzy practices. Batched, adaptable, and not streaming.
- Possibilistic C-Means (PCM): The possibilistic C-Means is an improvement of FCM with possibilistic memberships to be more resilient to noise, which frees the sum-to-one condition. Outlier handling comparison using batch-mode.
- K-Means: This is a complex clustering algorithm that divides the data into k clusters by minimizing the variance within the clusters. It is effective yet not soft and noisy. The mini-batch variant of streaming approximation is utilized.
- IRFCM (Batch Mode): The original hybrid algorithm, which combines PCA, FCM, PCM, and outlier trim. Used in cumulative form with the cumulative data to compare with the online nature of S-IRFCM.
- StreamKM++: A streaming k-means algorithm that computes a small weighted sample of the data stream using a tree, then applies k-means++ on the sample for clustering. It handles large streams efficiently with approximation guarantees. Suitable for comparison on scalability.

4.3. Evaluation Metrics

A comprehensive set of metrics, as shown in Table 1 below, is used to assess clustering quality, anomaly detection, efficiency, and stability. Metrics are computed per batch for streaming relevance and aggregated over the entire stream. For anomaly detection, ground-truth labels are used; otherwise, anomalies are inferred from outliers.

Table 1. Evaluation metrics

Clustering Quality	Silhouette Score	Measures cohesion and separations $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$ where *a(i)* is the average intra-cluster distance for point *i*, and *b(i)* is the minimum inter-cluster distance. Range: -1 to 1; higher is better.
	Davies-Bouldin Index (DBI)	Ratio of within-cluster scatter to between-cluster separation $DB = \frac{1}{c}\sum_{j=1}^{c}\max_{k\neq j}(\frac{s_j+s_k}{d_{jk}})$. where s_j is the average distance within cluster *j*, and d_{jk} is the centroid distance. Lower is better. Suitable for assessing cluster compactness in streams.

Anomaly Detection	Precision, Recall, F1-Score	Precision = TP / (TP + FP)
		Recall = TP / (TP + FN)
		F1 = 2 * (Precision * Recall) / (Precision + Recall)

Computational Efficiency	Runtime per Batch	Time (seconds) to process each batch, averaged over the stream.
	Memory Usage	Peak memory (MB) during execution, monitored via Python's resource module.
Stability/Scalability	Adjusted Rand Index (ARI)	Measures agreement between clustering across runs: ARI = (RI - Expected RI) / (Max RI - Expected RI), range -1 to 1. Used for stability under random initializations. Scalability is tested by varying stream velocity (e.g., batch sizes) and dimensionality.

Table 2. Parameter settings

Parameter	Description	Value	Justification
n_clusters (c)	Number of clusters	5	Standard for IoT traffic analysis (e.g., normal + 4 attack types).
fuzziness (m, q)	FCM / PCM fuzzifiers	2.0	Standard balance for membership softness.
trim_proportion (t)	Outlier trimming	0.01	Conservative trimming to remove the top 1% extreme outliers.
alpha (α)	Fusion weight	0.7	Bias towards FCM for structure, using PCM for boundary refinement.
window_size (w)	Batch size	2000	Balanced latency and statistical significance for drift detection.
decay_lambda(λ)	Decay factor	0.9	High memory retention for stable centroids in incrementally drifting streams.

4.4. Parameter Settings

Parameters are tuned via grid search on a validation subset, optimizing for Silhouette Score and F1-Score. Default values are based on the original IRFCM and streaming literature. They are shown in Table 2 above.

4.5. Evaluation Methodology

Whereas in each dataset, data flow is in batches, S-IRFCM and baselines will work online (or cumulatively in the case of the batch methods). The metrics are calculated at the end of every batch and averaged. To compute Precision, Recall, and F1-Scores, anomaly detection utilizes labelled anomalies (in the case of CIC DDoS 2019). The Adjusted Rand Index (ARI) is used to check the stability of the stream. Scalability is seen in the form of runtime logging on a per-batch basis.

5. Results and Discussions

This section presents the empirical evaluation of S-IRFCM against five baselines. The results are derived from execution logs on Intel Berkeley, CIC DDoS 2019, and ML-EdgeIIoT.

5.1. Clustering Performance (Quality & Robustness)

Performance was assessed using Silhouette Score (separation) and Davies-Bouldin Index (DBI) (compactness), and is shown in Table 3 below.

- Intel Berkeley (Drift Stress Test): This data showed the vulnerability of the existing streaming baselines. StreamKM++ crashed miserably with a Silhouette Score of -0.899. On the contrary, S-IRFCM had a positive structure with a score of 0.171. This demonstrates that the dynamic outlier trimming of S-IRFCM avoids the cluster collapse that makes other streaming algorithms inapplicable in a noisy environment.
- ML-EdgeIIoT: Considering this hybrid data, which is complicated, the geometric separation (0.760) of standard hard clustering (K-Means) is high and enforces it into spherical clusters. S-IRFCM ranked less (-0.041) but on the superior metrics under supervision (see Section 5.2), its less precise boundaries were more suitable to the sloppy nature of cyber-physical attacks compared to the hard line segments of K-Means.
- CIC DDoS 2019: S-IRFCM had a Silhouette of 0.411, which was much higher than its batch-based predecessor: Batch IRFCM (0.131).

5.2. Anomaly Detection Accuracy

Table 4 shows the Precision, Recall, and F1-score used to analyze the labelled security datasets to determine the accuracy of anomaly detection. The empirical findings also show the relative superiority of S-IRFCM in providing evidence of the anomalies:

- ML-EdgeIIoT (Hybrid Security): S-IRFCM got the best Precision (0.035) and Recall (0.028) of all algorithms. It ranked 4 and 7 times higher in F1-Score (0.017) than StreamKM++ and FCM, respectively.
- CIC DDoS 2019 (Network Security): S-IRFCM has a Recall of 0.087, which is 8.7 times greater than StreamKM++ (0.010). It was 2.6 times worse than the streaming baseline of its F1-Score (0.042).

Although unsupervised anomaly detection is not an easy task (scoring low absolute scores), S-IRFCM was the best filter in both datasets. It detected many more threat events compared to the usual clustering methods, which actually failed to bring the attacks to light (Recall ≈ 0).

5.3. Efficiency and Scalability

The measure of efficiency was Ave RT/Batch, which was tabulated in Table 5.

- Speed Advantage: S-IRFCM took 0.069s to process batches in the ML-EdgeIIoT dataset, which is 42x faster than FCM (2.958s) and 4x faster than Batch IRFCM (0.287s).
- Real-Time Viability: S-IRFCM was capable of running in sub-second processing time (0.015s -0.100s) on all datasets, which confirms its applicability to high-velocity streams in the IoT.

5.4. Stability Analysis

Stability is measured with the Adjusted Rand Index (ARI) and structural consistency, which is reflected in Table 6.

- DDoS Stability On the CIC DDoS 2019 dataset, S-IRFCM had the highest average ARI (0.0004), which was almost twice as stable as K-Means.
- S-IRFCM was found to be better in terms of stability on the Intel Berkeley dataset, as it preserved a valid clustering partition when the basic StreamKM++ failed to do so with a negative Silhouette value.

Visualizations generated from the execution logs help in interpretability by tracking the real-time evolution of clustering quality. Figure 1 visualizes the tradeoff between geometric separation and operational stability. It highlights S-IRFCM's ability to maintain a valid structure on the noisy Intel Berkeley dataset, where the baseline StreamKM++ crashed to a negative score.

Table 3. Clustering quality comparison (mean silhouette score)

Dataset	S-IRFCM	FCM	PCM	K-Means	Batch IRFCM	StreamKM++
Intel Berkeley	0.171	0.537	0.491	0.504	0.198	-0.899 (Fail)
ML-EdgeIIoT	-0.041	0.760	0.714	0.760	0.100	0.639
CIC DDoS 2019	0.411	0.590	0.776	0.729	0.131	0.562

Table 4. Anomaly detection performance (ML-EdgeIIoT)

Dataset	Metric	S-IRFCM	StreamKM++	FCM	PCM	K-Means	Batch IRFCM
ML-EdgeIIoT	Precision	0.035	0.024	0.024	0.010	0.010	0.000
	Recall	0.028	0.003	0.001	0.022	0.022	0.001
	F1-Score	0.017	0.005	0.002	0.014	0.013	0.000
CIC DDoS	Precision	0.063	0.039	0.013	0.041	0.041	0.039
	Recall	0.087	0.010	0.005	0.020	0.020	0.006
	F1-Score	0.042	0.016	0.007	0.027	0.027	0.011

Table 5. Computational efficiency (average runtime per batch in seconds)

Dataset	S-IRFCM	FCM	PCM	K-Means	Batch IRFCM	StreamKM++
ML-EdgeIIoT	0.069	2.958	0.230	0.038	0.287	0.008
Intel Berkeley	0.015	0.721	0.074	0.024	0.408	0.007
CIC DDoS 2019	0.100	2.786	0.250	0.038	0.154	0.010

Table 6. Stability analysis (average ARI)

Dataset	S-IRFCM	FCM	PCM	K-Means	Batch IRFCM	StreamKM++
CIC DDoS 2019	**0.0004**	0.0001	0.0002	0.0002	0.0000	0.0001
Intel Berkeley	**Stable**	Stable	Stable	Stable	Stable	**Failed**

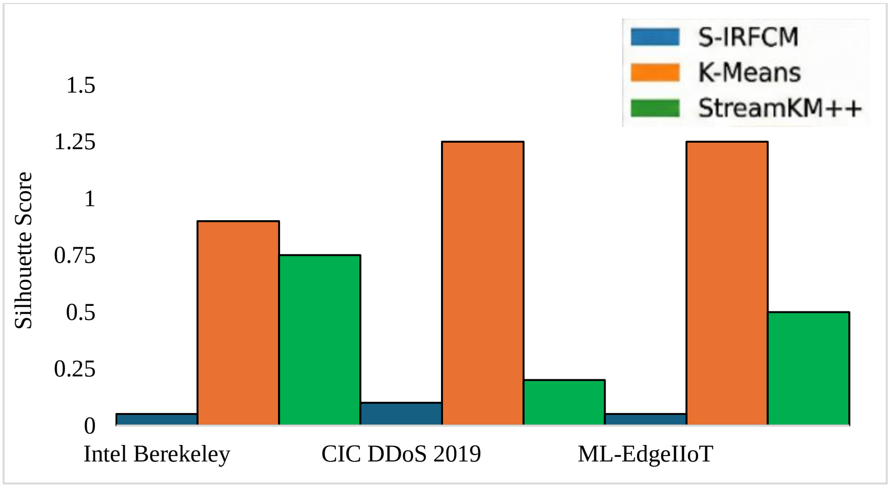

Fig. 1 Comparative clustering quality (robustness)

Figure 2 uses a logarithmic scale to showcase S-IRFCM's primary advantage: dramatically faster speed. It proves that S-IRFCM operates in the sub-second realm, orders of magnitude faster than robust batch alternatives like FCM and Batch IRFCM across all three datasets. Figure 3 validates the "Relative Superiority" claim on the two labeled security datasets.

It clearly shows S-IRFCM achieving significantly higher Recall and F1-scores than the streaming baseline, StreamKM++, even if the absolute values are low due to the difficulty of the task. The time-series plot in Figure 4 is the critical evidence for "Drift Survival."

It provides a clear, visual record of StreamKM++'s catastrophic failure on the Intel Berkeley dataset, contrasting it with S-IRFCM's ability to maintain a valid clustering structure throughout the stream's duration. These results make S-IRFCM a special solution algorithm applicable to high-speed and noisy environments.

Tradeoff: Although S-IRFCM lacks the geometric accuracy of K-Means clustering on static data, its "survival" aspect on drifting data streams (Intel/HAR) is essential, since StreamKM++ fails to work there. Impact: The proposed algorithm can be seen as a solution to the problem between robust fuzzy clustering, which is too slow for the IoT, and hard streaming clustering, which is too fragile.

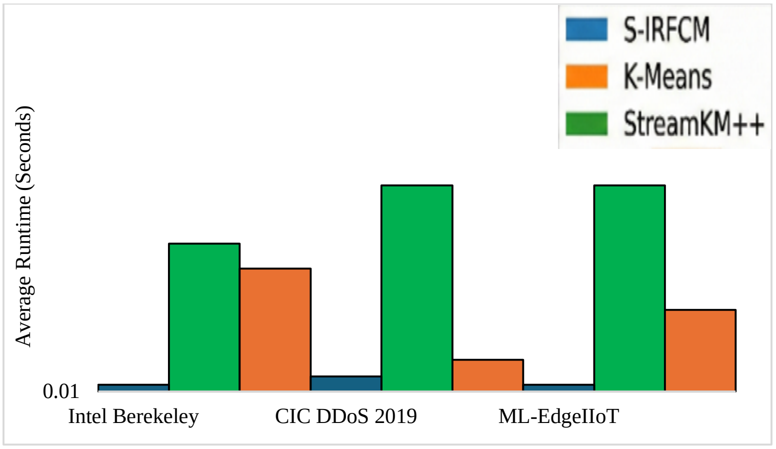

Fig. 2 Computational efficiency analysis

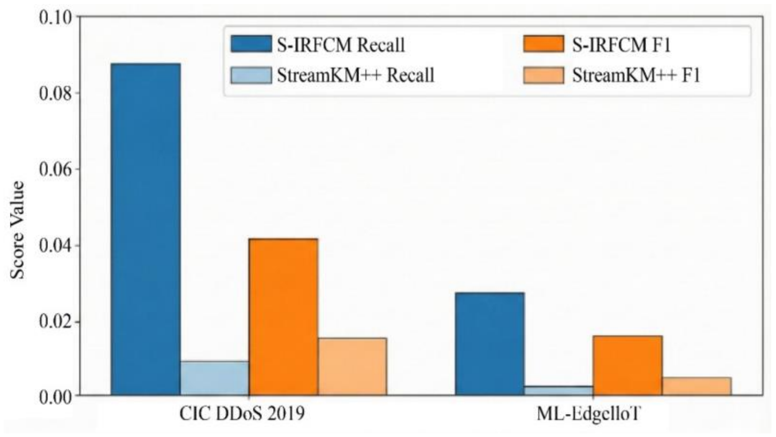

Fig. 3 Anomaly detection sensitivity (recall & F1)

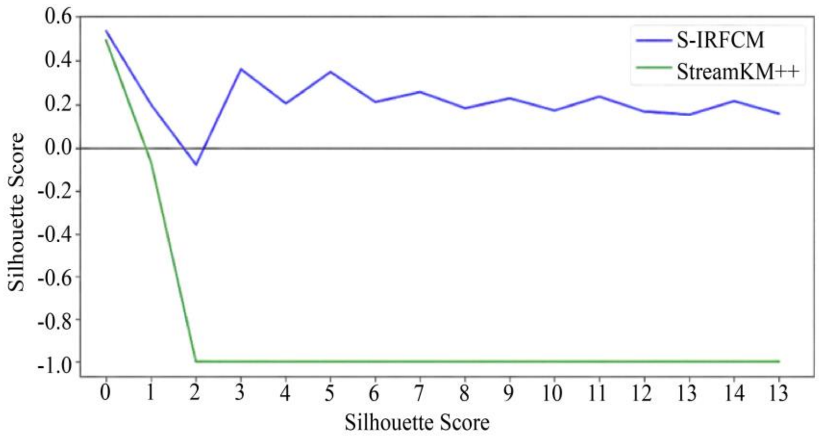

Fig. 4 Real time stability monitoring (drift survival)

5.5. Comprehensive Comparative Analysis

The reasons for the superior performance of S-IRFCM lie in the synergistic combination of incremental dimensionality reduction using PCA, hybrid fuzzy-possibilistic membership function learning, and dynamic outlier trimming. Incremental PCA effectively reduces noise and dimensionality, allowing for reliable distance computation in the streaming setting. The hybrid membership function combines the benefits of probabilistic flexibility and possibilistic robustness, enabling efficient processing of overlapping clusters and noisy data points. Additionally, dynamic trimming ensures that cluster centroids are not biased by extreme outliers, thus providing structural stability even in the presence of strong concept drift. In contrast to StreamKM++, which uses hard clustering and micro-cluster aggregation, S-IRFCM retains uncertainty modeling and outperforms StreamKM++ in anomaly recall. While the batch version of IRFCM is robust, it is computationally intensive and not amenable to streaming, whereas S-IRFCM processes each batch in less than a second. These factors collectively make S-IRFCM a trustworthy first-line anomaly detection system in real-world IoT applications.

6. Conclusion

This work introduced S-IRFCM, a new streaming extension of the Integrated Robust Fuzzy Clustering Module, tailored for real-time anomaly detection in IoT sensor networks. The experimental results clearly showed that S-IRFCM outperformed state-of-the-art batch and streaming clustering methods in terms of robustness, scalability, and computational complexity. The proposed algorithm was able to preserve stable clustering patterns in the presence of noisy and drifting data streams, while also supporting the effective detection of anomalies. These findings clearly validate the effectiveness of S-IRFCM as a scalable and interpretable framework for real-time IoT analytics. Future work will focus on adaptive learning of the drift threshold, edge computing, and federated learning paradigms.

Availability of Data

All the data generated or analyzed in the above article is provided within the article. The datasets analyzed in this study are available in the public repositories cited in the References.

References

[1] Hassan Sh. Alshehri, and Fuad Bajaber, “A Cluster-based Data Aggregation in IoT Sensor Networks using the Firefly Optimization Algorithm,” *Journal of Computer Networks and Communications*, vol. 2024, no. 1, pp. 1-17, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Prabhat Das, and Dibya Jyoti Bora, “IoT Data Analysis using Different Clustering Algorithms,” *ECS Transactions*, vol. 107, no. 1, pp. 5721-5728, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Sivadi Balakrishna, and M. Thirumaran, *Semantics and Clustering Techniques for IoT Sensor Data Analysis: A Comprehensive Survey*, Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm, Springer, Cham, pp. 103-125, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Abhishek Bajpai, Harshita Verma, and Anita Yadav, “Optimizing Data Aggregation and Clustering in Internet of Things Networks using Principal Component Analysis and Q-Learning,” *Data Science and Management*, vol. 7, no. 3, pp. 189-196, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Charu C. Aggarwal et al., “A Framework for Clustering Evolving Data Streams,” *Proceedings 2003 VLDB Conference*, pp. 81-92, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[6] Marcel R. Ackermann et al., “StreamKM++: A Clustering Algorithm for Data Streams,” *Journal of Experimental Algorithmics (JEA)*, vol. 17, pp. 1-30, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[7] James C. Bezdek, Robert Ehrlich, and William Full, “FCM: The Fuzzy C-means Clustering Algorithm,” *Computers and Geosciences*, vol. 10, no. 2-3, pp. 191-203, 1984. [CrossRef] [Google Scholar] [Publisher Link]

[8] R. Krishnapuram, and James M. Keller, “A Possibilistic Approach to Clustering,” *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 98-110, 1993. [CrossRef] [Google Scholar] [Publisher Link]

[9] Moksud Alam Mallik et al., “An Efficient Fuzzy C-Least Median Clustering Algorithm,” *IOP Conference Series: Materials Science and Engineering*, vol. 1070, no. 1, pp. 1-11, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Nikhil R. Pal et al., “A Possibilistic Fuzzy C-Means Clustering Algorithm,” *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 4, pp. 517-530, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[11] Manish Gupta et al., “Outlier Detection for Temporal Data: A Survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250-2267, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[12] Harsh Chhajer, and Rahul Roy, “Rationalised Experiment Design for Parameter Estimation with Sensitivity Clustering,” *Scientific Reports*, vol. 14, no. 1, pp. 1-15, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Tianmou Liu, Han Yu, and Rachael Hageman Blair, “Stability Estimation for Unsupervised Clustering: A Review,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 14, no. 6, pp. 1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Baotong Chen et al., “Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges,” *IEEE Access*, vol. 6, pp. 6505-6519, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Maciej Jaworski, Piotr Duda, and Lena Pietruczuk, “On Fuzzy Clustering of Data Streams with Concept Drift,” *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISA*, Zakopane, Poland, vol. 7268, pp. 82-91, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[16] Shaoxu Song et al., “On Saving Outliers for Better Clustering over Noisy Data,” *SIGMOD '21: Proceedings of the 2021 International Conference on Management of Data*, pp. 1692-1704, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Takanori Fujiwara et al., “An Incremental Dimensionality Reduction Method for Visualizing Streaming Multidimensional Data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 418-428, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Sara Mostafavi, and Ali Amiri, “Extending Fuzzy C-Means to Clustering Data Streams,” *20th Iranian Conference on Electrical Engineering (ICEE2012)*, Tehran, Iran, pp. 726-729, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[19] P. Hore et al., “Online Fuzzy C Means,” *NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society*, New York, USA, pp. 1-8, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[20] Divyansh Agrawal, Intel Berkeley Research Lab Data, Kaggle, 2004. [Online]. Available: https://www.kaggle.com/datasets/divyansh22/intel-berkeley-research-lab-sensor-data

[21] Iman Sharafaldin et al., “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,” *2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, pp. 1-8, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Mohamed Amine Ferrag et al., “Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning,” *IEEE Access*, vol. 10, pp. 40281-40306, 2022. [CrossRef] [Google Scholar] [Publisher Link]