Active Adaptation for Novelty Detection in Data Streams with Extreme Latency Using Fuzzy Micro-Clusters

Tiago Carlos Batista [1], **Heloisa Camargo**[1]

[1]Computing Department – Federal University of São Carlos (UFSCar)

tiagodr07@gmail.com, heloisacamargo@ufscar.br

Abstract. *Data Streams are characterized by continuous data generation, evolving distributions, and limited access to labeled data, posing significant challenges for classification and novelty detection. This paper proposes a fuzzy micro cluster-based approach for these tasks using an active adaptation mechanism that triggers model updates based on recent data batches defined by variable-sized windows, improving adaptability to dynamic environments. A partial update strategy, which uses only a fraction of the accumulated data, is also proposed to reduce the number of model updates while maintaining classification performance. Experiments on synthetic datasets show that the proposed approach achieves similar or superior performance in classification and novelty detection compared to its passive adaptation counterpart, while requiring fewer model updates, indicating reduced computational overhead.*

1. Introduction

Data Streams (DS) consist of continuously generated data, often produced at high speed and with evolving distributions over time [Gama et al. 2010]. Unlike traditional machine learning settings, where data are assumed to be static and fully available, learning in DS must handle potentially infinite data, limited memory, and dynamic environments [Ditzler et al. 2015]. These characteristics introduce additional challenges, such as concept drift, where the underlying data distribution changes, and concept evolution, where new classes emerge over time [Fahy et al. 2022].

In this context, classification combined with Novelty Detection (ND) plays a crucial role. ND aims to identify patterns that do not belong to previously known classes, enabling the detection of emerging concepts [Pimentel et al. 2014]. This capability is essential in several real-world applications, such as intrusion detection, fraud detection, and monitoring systems, where new patterns may appear continuously [Din et al. 2021]. Fuzzy clustering has been successfully used in proposals that address DS learning, supervised or unsupervised, as can be seen in [Abdullatif et al. 2018], [Ferdaus et al. 2024] and [Urio-Larrea et al. 2025].

Most previous work, such as EIFuzzCND [Bruzzone 2023], addresses this problem using fuzzy clustering techniques combined with passive adaptation strategies. In passive adaptation, the model is continuously updated as new data arrive, regardless of whether significant changes occur in the data distribution. This approach can lead to unnecessary updates, increased computational cost, and reduced sensitivity to relevant changes.

To address these limitations, this paper proposes a fuzzy-based approach for classification and ND with extreme latency in DS using an active adaptation mechanism. The

proposed method accumulates classified instances in a buffer and performs batch updates only when the buffer reaches its maximum capacity. Additionally, a partial update strategy allows the model to be updated using only a fraction of the accumulated data, further reducing the number of model updates while maintaining classification quality.

The central hypothesis of this work is that accumulating classified instances in a fixed-size buffer and updating the model in batches reduces noise in micro-cluster boundary adjustments compared to instance-by-instance updates, yielding equal or superior classification accuracy while requiring fewer model updates. A secondary hypothesis is that performance can be maintained even when only a fraction of the accumulated instances is used for updating, further reducing the number of updates without significant loss in classification quality.

The main contributions of this work are: (i) the integration of an active adaptation mechanism into a fuzzy-based framework for novelty detection in DS; (ii) a partial update strategy which enables the investigation of the minimum amount of data required for effective model updates; and (iii) an experimental evaluation demonstrating that batch updates can improve accuracy and reduce false positive ND compared to the original instance-by-instance update strategy, under extreme latency conditions.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 describes the proposed method. Section 4 details the experimental setup. Section 5 presents and discusses the results. Finally, Section 6 concludes the paper.

2. Related Works

In recent years, many studies for DS learning have been proposed using fuzzy systems. Some representative research is briefly described below.

In [Wen and Zhao 2026], an online learning framework based on hypersphere granules is proposed, in which the granules can be generated or updated with limited information using neighborhood granules of newly arrived data. The existing granules generate fuzzy rules that are able to perform accurate classification. Conflicting and noise granules are identified and corrected, to ensure that the model remains highly interpretable and robust.

In [Gu 2023], Gu proposes a method to deal with the highly challenging task of semi-supervised learning from data streams, composed of two evolving fuzzy systems. After using a small amount of labelled data during the initial period, the proposed model is able to continuously self-learn and self-expand its knowledge base from unlabelled data on a chunk-by-chunk basis with minimal human expert involvement.

The work proposed in [Samadi et al. 2025] also developed a semi-supervised approach to deal with the scarcity of labels in real-world data stream applications. A method named Weighted Semi-supervised Possibilistic Fuzzy C-Means (WSPFCM-DS) is proposed for DS classification and ND.

Among the approaches more closely related to the proposal presented in this article are those introduced in [Cristiani and Camargo 2021] and [Bruzzone 2023]. In [Cristiani and Camargo 2021], the authors address the problem of classification and ND in data streams under intermediate latency conditions, where true labels become available only after a delay. They proposed the Enhanced Fuzzy Classifier with Multi-class Nov-

elty Detection for Data Streams (EFuzzCND). EFuzzCND maintains two decision models structured as fuzzy micro-clusters (SPFMiCs) that evolve through distinct mechanisms. The Model of Known Classes (MKC) is updated only when a sufficient chunk of true labels becomes available, using a supervised procedure that groups new labeled examples, incorporates them into the MKC, and performs temporal maintenance. Conversely, the Model of Unknown Classes (MUC), which captures novelty patterns, is dynamically updated during the *online* phase in an unsupervised manner: when instances cannot be explained by existing models, they are accumulated, clustered via Fuzzy C-Means (FCM), and validated using fuzzy silhouette to form new novelty patterns that are immediately incorporated into the MUC.

Attempting to overcome limitations related to adaptation, the authors in [Bruzzone 2023] proposed an incremental approach to EFuzzCND named EIFuzzCND based on continuous model updating. The results indicate that EIFuzzCND outperforms established methods, especially in scenarios with extreme latency, demonstrating high adaptability. However, because it adopts a passive update strategy, the model continuously updates itself with each new instance, which can result in a high number of unnecessary updates.

Despite the advances described, existing fuzzy-based approaches for DS classification and ND present a significant gap regarding the efficiency of model adaptation. They typically rely either on static models that fail to adapt over time [Cristiani and Camargo 2021] or on passive adaptation strategies that update the model for every incoming instance [Bruzzone 2023]. This absence of controlled update mechanisms leads to high computational costs and spurious novelty detections during stable periods.

No prior work has systematically investigated whether controlled batch updates can overcome these limitations in fuzzy micro-cluster frameworks. The proposed Partial Active Adaptation (PAA) strategy fills this gap by accumulating instances in a fixed-size window and performing batch updates using only a fraction of the data, significantly reducing the total number of model updates and stabilizing decision boundaries, thus providing an effective balance between computational efficiency and adaptability.

3. Extension Proposed to the EIFuzzCND Method

A Partial Active Adaptation (PAA) strategy is proposed to extend the EIFuzzCND method, giving rise to EIFuzzCND-PAA. In the *offline* phase, a labeled training set is partitioned by class and clustered using Possibilistic Fuzzy C-Means (PFCM) algorithm; the resulting clusters are summarized as SPFMiCs to form the initial MKC. In the *online* phase, each incoming instance is first evaluated by the MKC: if classified, it is added to a fixed-size window buffer; otherwise, it is evaluated by the MUC or stored in an unknown buffer for novelty detection. When the window reaches capacity, a batch update is performed using a fraction of the accumulated instances (controlled by *updatePercent*). The following subsections detail each component.

EIFuzzCND uses two fuzzy clustering algorithms. PFCM [Pal et al. 2005] is a hybrid algorithm robust to noise and outliers that generates both membership and typicality values for each point, eliminating the noise sensitivity of FCM [Bezdek 1981]. FCM is used internally with the standard fuzzification exponent m.

In the *offline* phase, the training data are separated by class and processed by PFCM, generating clusters summarized into micro-clusters of type Supervised Possibilistic Fuzzy Micro-Cluster (SPFMiC) to form the MKC. Let $\vec{x} \in \mathbb{R}^d$ denote a data instance in a d-dimensional feature space. Each SPFMiC maintains the following parameters:

- $N \in \mathbb{N}$: number of assigned instances (scalar);
- $M^e \in \mathbb{R}$: accumulated sum of weighted membership values (scalar);
- $T^e \in \mathbb{R}$: accumulated sum of weighted typicality values (scalar);
- $SSD^e \in \mathbb{R}$: membership-weighted sum of squared Euclidean distances to the centroid (scalar);
- $CF1_\mu \in \mathbb{R}^d$: membership-weighted sum of instances (vector);
- $CF1_T \in \mathbb{R}^d$: typicality-weighted sum of instances (vector);
- $\vec{c} \in \mathbb{R}^d$: centroid of the micro-cluster (vector).

The model generated by EIFuzzCND consists of two components: the MKC, which classifies instances into known classes, and the Model of Unknown Classes (MUC), which models emerging classes. In the original EIFuzzCND, every classified instance immediately triggers a model update (passive adaptation), which may result in high computational cost in high-speed data streams or during stable periods. The PAA strategy addresses this limitation, as described in Section 3.3.

3.1. Classification Criterion and Incremental Update of Micro-Clusters

The classification and incremental update procedures described here are inherited from the original EIFuzzCND [Bruzzone 2023].

During the *online* phase, an incoming instance $\vec{x}$ is covered by a SPFMiC if $d(\vec{x}, \vec{c}) \leq DF$, where $d(\vec{x}, \vec{c})$ is the Euclidean distance between $\vec{x}$ and the centroid $\vec{c}$, and DF is the fuzzy dispersion radius:

$$DF = \sqrt{\frac{SSD^e}{N}} \cdot F \tag{1}$$

with F a scaling factor controlling the coverage area. Among all covering SPFMiCs, the one with the highest typicality is selected as the winner, and its label is assigned to the instance.

The incremental update (line 12 of Algorithm 1) modifies the internal statistics of the winning SPFMiC. Given membership μ and typicality τ of $\vec{x}$, the counter N is incremented and the parameters are updated as:

$$M^e \leftarrow M^e + \mu^\alpha, \quad T^e \leftarrow T^e + \tau^\theta \tag{2}$$

$$CF1_\mu \leftarrow CF1_\mu + \mu \cdot \vec{x}, \quad CF1_T \leftarrow CF1_T + \tau \cdot \vec{x} \tag{3}$$

$$SSD^e \leftarrow SSD^e + \mu \cdot d(\vec{x}, \vec{c})^2 \tag{4}$$

where α and θ are the membership and typicality exponents, respectively. The centroid is then recalculated as:

$$\vec{c} = \frac{\alpha \cdot CF1_\mu + \theta \cdot CF1_T}{\alpha \cdot T^e + \theta \cdot M^e} \tag{5}$$

This formulation computes a weighted combination of the membership-based and typicality-based summaries: the numerator accumulates positional information weighted by both μ and τ, while the denominator normalizes by the corresponding accumulated weights, dynamically balancing the influence of each point according to its fuzzy degrees of belonging.

A minimum weight threshold (*minWeightOnline*) prevents instantiation of insufficiently representative SPFMiCs, and an obsolescence threshold ts discards micro-clusters whose creation and last update times both exceed ts.

3.2. Novelty Detection

The novelty detection mechanism is inherited from EIFuzzCND [Bruzzone 2023]. When the unknown buffer reaches threshold T, FCM is applied to identify candidate clusters, which are validated by the fuzzy silhouette criterion.

For each valid candidate cluster i with centroid $\vec{c}_i$ and radius r_i, the Fuzzy Ratio (FR) against each known micro-cluster j is:

$$FR(i,j) = \frac{r_i + r_j}{d(\vec{c}_i, \vec{c}_j)} \tag{6}$$

where $r_i = \sqrt{SSD_i^e / N_i}$. The minimum FR determines the outcome:

$$\min_j \; FR(i,j) \leq \phi \;\Rightarrow\; \text{known class extension} \tag{7}$$

$$\min_j \; FR(i,j) > \phi \;\Rightarrow\; \text{novelty detected} \tag{8}$$

A known class extension inherits the label of the closest MKC micro-cluster; a novelty receives a new unique label. In both cases, the validated SPFMiC is added to the MUC and its instances are removed from the unknown buffer.

3.3. Partial Active Adaptation Strategy

The PAA strategy is the main contribution of this work. Instead of updating the model after every classified instance, a fixed-size window buffer accumulates classified instances and triggers a batch update only when full, reducing update frequency while preserving adaptability.

The parameter $windowSize$ controls this behavior: when $windowSize = 1$, EIFuzzCND-PAA reduces to EIFuzzCND; for $windowSize > 1$, updates occur in batches, trading responsiveness for lower computational cost. The parameter $updatePercent \in (0, 1]$ further controls what fraction of the window is used: when $updatePercent = 1.0$, all instances contribute; for lower values, a uniformly drawn subset without replacement is used, acting as an unbiased estimator of the window's data distribution and preventing the systematic bias that would arise from selecting only the first or last instances of the window.

The *online* phase of EIFuzzCND-PAA is described in Algorithm 1, with the PAA strategy highlighted in lines 8–15. Classification, SPFMiC update, and novelty detection procedures remain as in the original EIFuzzCND.

4. Experiments and Results

This section describes the experimental setup used to evaluate the proposed EIFuzzCND-PAA algorithm, assessing the impact of the window mechanism and the partial update strategy on classification performance and novelty detection, in comparison with the original EIFuzzCND under extreme latency conditions.

Table 1. *Online* Phase Algorithm of the ElFuzzCND-PAA method

Algorithm: Online Phase Algorithm

Require: *dataStream*, *MKC*, *L*, *tChunk*, *T*, ϕ, *percentLabeled*, *windowSize*, *updatePercent*

Ensure: *LabeledData*

1: *unknownInstances* $\leftarrow$ empty list
2: *labeledInstances* $\leftarrow$ empty list
3: *classifiedWindow* $\leftarrow$ empty list
4: **for** *each instance inst in dataStream* **do**
5: *label* $\leftarrow$ *MKC.classifyWithoutUpdate(inst)*
6: **if** *label* $\neq -1$ **then**
7: *classifiedWindow.append((inst, label))*
8: ***if size(classifiedWindow)*** $\geq$ ***windowSize*** **then**
9: ***nSamples*** $\leftarrow max(1, \lfloor$ ***size(classifiedWindow)*** $\times$ ***updatePercent*** $\rfloor)$
10: ***sampledWindow*** $\leftarrow$ ***randomSample(classifiedWindow, nSamples)***
11: **for** ***each (ex, rot) in sampledWindow*** **do**
12: ***MKC.updateWithExample(ex, rot, currentTime)***
13: **end for**
14: ***clear(classifiedWindow)***
15: **end if**
16: **else**
17: *label* $\leftarrow$ *MUC.predict(inst)*
18: **if** *label* $= -1$ **then**
19: *add inst to unknownInstances*
20: **if** *size(unknownInstances)* $\geq$ *T* **then**
21: *unknownInstances* $\leftarrow$ *multiClassNoveltyDetection(unknownInstances)*
22: **end if**
23: **end if**
24: **end if**
25: **if** *currentTime* $\geq$ *latencyDelay* **then**
26: **if** *Math.random()* $<$ *percentLabeled* **OR** *labeledMem is empty* **then**
27: *labeledMem* $\leftarrow$ *addToList(labeledExample)*
28: **end if**
29: **if** *size(labeledMem)* $\geq$ *tChunk* **then**
30: *MKC* $\leftarrow$ *trainModel(labeledMem, tempo)*
31: *clear(labeledMem)*
32: **end if**
33: **end if**
34: **end for**
35: **return** *testSetLabeled*

Lines 8 through 15 indicate the proposed Partial Active Adaptation strategy.

4.1. Datasets

Two synthetic datasets widely adopted in DS classification and ND studies were used. Both are partitioned into *offline* and *online* sets and present concept drift and concept evolution, with new classes emerging during the *online* phase. Their main characteristics are summarized in Table 2.

Table 2. Dataset characteristics.

Dataset	Attributes	Offline inst.	Online inst.	Known classes	Novel classes	Synthetic
RBF	2	2,000	46,586	3	2	Yes
SyneDC	54	30,000	370,000	6	14	Yes

4.2. Experimental Setup

The experiments evaluate two aspects of the proposed approach: (i) the effect of different window sizes on model performance and (ii) the impact of partial updates using a fraction of the accumulated window data.

The parameter *windowSize* was varied across $\{1, 10, 40, 80, 200\}$, where $windowSize = 1$ replicates the original EIFuzzCND (baseline). The remaining values were selected through preliminary experiments to span a broad range of update frequencies: $windowSize = 10$ represents fine-grained batching, $windowSize = 40$ and $windowSize = 80$ intermediate regimes, and $windowSize = 200$ a coarse-grained regime, allowing investigation of the trade-off between update frequency and model staleness. The parameter *updatePercent* was varied across $\{1.0, 0.8, 0.5\}$, also selected through preliminary experiments to cover three representative regimes: full window utilization, moderate reduction, and aggressive reduction, assessing the minimum data fraction necessary for effective updates.

Performance metrics were computed incrementally every 1,000 instances during the *online* phase, following the evaluation protocol of the original EIFuzzCND. All experiments used fixed random seeds to ensure reproducibility.

4.3. Parameters

Table 3 summarizes the parameter configuration used in all experiments, applied equally to both datasets and both methods. The parameters m, α, θ, K, T, ts, F, $tChunk$, L, and $\phi = 0.2$ were adopted from the original EIFuzzCND to ensure a fair comparison. The parameters $windowSize$ and $updatePercent$ were determined through preliminary experiments as described above.

The latency $L = 10{,}000{,}000$ configures the system under extreme latency, meaning true labels are never available during the *online* phase. EIFuzzCND-PAA is compared with the original EIFuzzCND, which employs passive adaptation (equivalent to $windowSize = 1$, $updatePercent = 1.0$). The comparison aims to show that the window mechanism maintains comparable performance with fewer model updates, and that partial updates do not significantly degrade classification quality.

The following metrics were used:

- **Accuracy** ($Acc = TP/N$): ratio of correctly classified instances, where TP is the count of correct predictions and N the total instances.

Table 3. Parameter configuration for the experiments.

Parameter	Symbol	Value
Fuzzification exponent	m	2.0
Membership exponent	α	2.0
Typicality exponent	θ	1.0
Number of clusters (FCM)	K	4
Novelty detection threshold	ϕ	0.2
Unknown memory threshold	T	40
Minimum weight (online)	–	15
Chunk size (retraining)	$tChunk$	2,000
Obsolescence threshold	ts	200
Latency	L	10,000,000 (extreme)
Percentage of labeled data	–	1.0 (100%)
Radius scaling factor	F	2.0
Window size	$windowSize$	$\{1, 10, 40, 80, 200\}$
Update fraction	$updatePercent$	$\{1.0, 0.8, 0.5\}$

- **Unknown Rate (Unk%)**: percentage of instances not recognized by either the MKC or MUC within each evaluation window of 1,000 instances.

Both metrics are computed via an incremental confusion matrix over consecutive blocks of 1,000 instances, enabling analysis of performance evolution across concept drift and concept evolution.

5. Results and Discussion

This section presents the experimental results obtained with the proposed EIFuzzCND-PAA algorithm across the RBF and SyneDC datasets. The analysis covers Accuracy, Unk, and Novelty Detection performance, comparing the baseline configuration ($windowSize = 1$, $updatedPercent = 1.0$) with representative window configurations ($windowSize \in \{40, 80, 200\}$, $updatedPercent \in \{1.0, 0.8, 0.5\}$).

Tables 4 and 5 present the accuracy values at 10 selected evaluation moments for RBF and SyneDC datasets, respectively.

Table 4. Accuracy (%) at selected evaluation moments - RBF dataset.

Moment	w=1 up=1.0	w=40 up=1.0	w=40 up=0.8	w=40 up=0.5	w=80 up=1.0	w=80 up=0.8	w=80 up=0.5	w=200 up=1.0	w=200 up=0.8	w=200 up=0.5
1	96.90	96.90	96.90	98.00	97.10	97.90	97.80	97.40	96.90	97.00
5	99.58	99.06	99.26	99.42	99.46	99.28	99.62	99.32	99.22	99.12
10	98.87	97.17	98.71	97.27	98.53	97.52	98.87	99.15	98.62	97.82
15	96.81	96.24	97.15	95.81	94.81	97.24	93.15	95.49	98.09	97.01
20	95.07	96.57	96.50	95.34	91.14	97.23	91.01	92.44	97.81	96.68
25	94.14	96.84	96.11	95.09	89.05	97.26	90.64	90.42	97.75	96.66
30	93.79	97.11	96.12	95.18	87.98	97.43	92.25	89.24	97.82	96.85
35	93.51	97.31	96.11	95.26	87.39	97.42	93.35	88.83	97.91	96.99
40	92.78	97.32	95.79	95.04	86.31	97.34	94.19	88.10	97.92	97.13
45	**90.10**	94.93	93.67	92.94	83.76	**96.80**	92.82	85.54	96.08	94.82

For the sake of organization and space constraints, the abbreviation w corresponds to the windowSize parameter, and up corresponds to the updatedPercent parameter.

For RBF, $windowSize = 80$, $updatedPercent = 0.8$ achieves the best sustained accuracy, surpassing the baseline by $\sim$4.8 pp at moment 45 (96.80% vs. 90.10%), while

$windowSize = 80$, $updatedPercent = 1.0$ degrades to 83.76%, the worst result, suggesting that partial updates reduce overcorrection under concept drift.

Regarding novelty detection, Figure 1 shows the behavior of $windowSize = 10$, $updatedPercent = 1.0$ – the only configuration achieving exact novelty detection (2 out of 2). The smaller window size enables the model to react faster to genuine novel classes without accumulating the lag that causes larger windows to generate spurious detections.

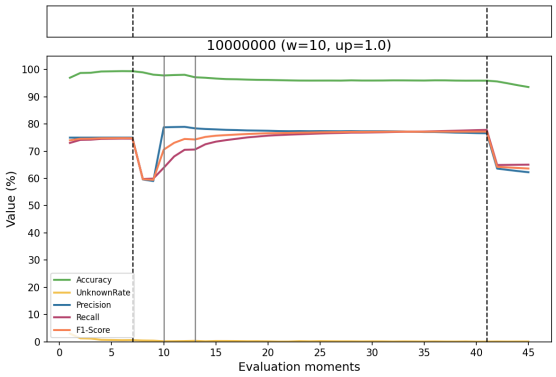


Figure 1. RBF dataset – best novelty detection configuration ($windowSize = 10$, $updatedPercent = 1.0$): the smaller window allows rapid adaptation to novel classes. Note the tighter convergence after each novelty event compared to the baseline.

Table 5. Accuracy (%) at selected evaluation moments - SyneDC dataset.

Moment	w=1 up=1.0	w=40 up=1.0	w=40 up=0.8	w=40 up=0.5	w=80 up=1.0	w=80 up=0.8	w=80 up=0.5	w=200 up=1.0	w=200 up=0.8	w=200 up=0.5
1	88.21	87.21	87.21	93.71	93.01	93.01	90.51	93.41	93.41	88.51
40	91.09	91.80	91.80	91.66	91.71	91.71	90.31	89.53	89.53	89.86
80	95.13	95.22	95.22	95.30	95.03	95.03	94.78	94.29	94.29	94.46
120	96.17	95.95	95.95	96.14	96.01	96.01	96.13	95.68	95.68	95.84
160	**95.96**	96.04	96.04	**96.29**	95.95	95.95	96.08	95.88	95.88	95.88
200	95.53	95.45	95.45	95.69	95.48	95.48	95.50	95.25	95.25	95.54
240	83.44	83.37	83.37	83.57	83.39	83.39	83.41	83.20	83.20	83.45
280	71.61	71.56	71.56	71.73	71.58	71.58	71.59	71.41	71.41	71.62
320	62.66	62.61	62.61	62.76	62.63	62.63	62.64	62.48	62.48	62.67
369	54.34	54.30	54.30	54.43	54.31	54.31	54.32	54.19	54.19	54.35

For the sake of organization and space constraints, the abbreviation w corresponds to the windowSize parameter, and up corresponds to the updatedPercent parameter.

For the SyneDC dataset, the configuration $windowSize = 40$, $updatedPercent = 0.5$ achieves the best accuracy in 6 of the 10 evaluation moments, including the most relevant intermediate ones (160 and 200), slightly surpassing the baseline (96.29% vs. 95.96% at moment 160). The gain is modest in absolute terms (under 1 pp in most moments) but consistent across the entire stream, including the final, most degraded phase (54.43% vs. 54.34% at moment 369). Unlike RBF, where the partial update factor interacts strongly with window size, in SyneDC the differences among $updatedPercent$ values within a fixed window remain small, suggesting a more moderate concept drift rate.

The Unk converges rapidly below 0.5% for all configurations after the warm-up phase, with no systematic increase relative to the baseline. Precision, Recall, F1-Score,

and Unk detailed results are provided in the supplementary material.[1]

Table 6 presents the number of novelties detected by each configuration compared to the actual number of novel classes in each dataset.

Table 6. Comparative Results between RBF and SyneDC

Configuration	RBF			SyneDC		
	Detected	Real	Difference	Detected	Real	Difference
$w = 1, up = 1.0$ (baseline)	3	2	+1	195	14	+181
$w = 10, up = 1.0$	**2**	2	**0**	195	14	+181
$w = 10, up = 0.8$	4	2	+2	195	14	+181
$w = 10, up = 0.5$	7	2	+5	195	14	+181
$w = 40, up = 1.0$	3	2	+1	195	14	+181
$w = 40, up = 0.8$	4	2	+2	**195**	14	**+181**
$w = 40, up = 0.5$	5	2	+3	196	14	+182
$w = 80, up = 1.0$	8	2	+6	196	14	+182
$w = 80, up = 0.8$	5	2	+3	196	14	+182
$w = 80, up = 0.5$	6	2	+4	196	14	+182
$w = 200, up = 1.0$	8	2	+6	196	14	+182
$w = 200, up = 0.8$	4	2	+2	196	14	+182
$w = 200, up = 0.5$	3	2	+1	196	14	+182

For the sake of organization and space constraints, the abbreviation w corresponds to the windowSize parameter, and up corresponds to the updatedPercent parameter.

For the RBF dataset, only $windowSize = 10$, $updatedPercent = 1.0$ achieves exact novelty detection (2 out of 2 real classes). The baseline already overestimates by one false positive (+1), and larger windows tend to amplify this excess, peaking at $windowSize = 80$, $updatedPercent = 1.0$ and $windowSize = 200$, $updatedPercent = 1.0$ (+6 each). This suggests that, for novelty detection specifically – as opposed to classification accuracy – smaller windows with full updates are preferable for this dataset, since they react quickly enough to avoid interpreting transient drift as spurious novel classes.

For SyneDC, novelty detection is essentially unaffected by windowing: all configurations detect either 195 or 196 novelties, against only 14 real ones, a massive and consistent over-detection (+181/+182) regardless of window size or update fraction. This indicates that, unlike RBF, the over-detection in SyneDC is not driven by update granularity but rather by the high cardinality of classes in this dataset, which appears to fragment the class space irrespective of how prototypes are updated.

5.1. Discussion

The results indicate that the effectiveness of the proposed windowing mechanism is dataset-dependent. For RBF, intermediate window sizes improved classification performance, with $windowSize = 80$ and $updatedPercent = 0.8$ achieving the best results. In contrast, SyneDC exhibited only modest classification gains across configurations.

The Unk was largely unaffected by the windowing mechanism, rapidly converging to values below 0.5% for all configurations. This indicates that the observed classification improvements were not achieved at the expense of increased instance rejection.

Novelty detection behaved differently from classification accuracy. For RBF, the best novelty detection result ($windowSize = 10$, $updatedPercent = 1.0$) was obtained

[1]Link for supplementary material:
`https://anonymous.4open.science/r/Article-832B/README.md`

with a smaller window than the best classification configuration, suggesting a trade-off between classification stability and detection responsiveness. For SyneDC, novelty over-detection remained nearly unchanged across configurations, indicating a limitation associated with the dataset characteristics rather than with the update strategy itself.

Overall, no single configuration was optimal for all datasets and objectives. The results suggest that the choice of window size and update fraction should be adapted to the characteristics of the data stream and the target objective, whether classification performance or novelty detection.

6. Conclusions

This work proposed EIFuzzCND-PAA, an extension of EIFuzzCND that replaces the passive instance-by-instance update strategy with a partial active adaptation mechanism based on fixed-size windows and configurable update fractions. The objective is to reduce the frequency of model updates while preserving classification and novelty detection performance under extreme latency conditions.

Experiments on the RBF and SyneDC datasets using 13 configurations ($windowSize \in 1, 10, 40, 80, 200$ and $updatedPercent \in 1.0, 0.8, 0.5$) showed that the impact of windowing is dataset-dependent. For RBF, the configuration $windowSize = 80$, $updatedPercent = 0.8$ achieved the best classification performance, while exact novelty detection was obtained with $windowSize = 10$, $updatedPercent = 1.0$, highlighting a trade-off between classification stability and novelty detection responsiveness. For SyneDC, classification gains were modest and novelty over-detection remained largely unaffected by the update strategy, suggesting a limitation related to dataset characteristics rather than to the proposed mechanism.

The main contributions of this work are: (i) the integration of an active adaptation mechanism into a fuzzy micro-cluster framework for novelty detection in data streams; (ii) the introduction of the partial update fraction (up), demonstrating that effective model updates can be achieved using only a subset of accumulated instances; and (iii) an experimental analysis showing how window size and update frequency influence classification and novelty detection performance under extreme latency.

This work has some limitations. The evaluation was conducted only on synthetic datasets and with a single random seed. In addition, no statistical significance tests were performed, the influence of the novelty detection threshold ϕ was not systematically investigated, and only fixed window configurations under extreme latency were considered. Furthermore, abrupt concept drift may affect the quality of batch updates by introducing heterogeneous instances into the same buffer.

Future work includes evaluating the method on real-world datasets, performing multiple executions with statistical validation, investigating the influence of the novelty detection threshold ϕ, exploring alternative instance selection strategies for partial updates, and developing adaptive window mechanisms to improve robustness under concept drift.

Acknowledgment. This study was conducted with support from the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Funding Code 001.

References

Abdullatif, A., Masulli, F., and Rovetta, S. (2018). Clustering of nonstationary data streams: A survey of fuzzy partitional methods. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1258.

Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.

Bruzzone, L. R. D. (2023). Eifuzzcnd: uma estratégia incremental para classificação multiclasse e detecção de novidades em fluxos de dados. Masters thesis, Federal University of São Carlos.

Cristiani, A. and Camargo, H. (2021). A fuzzy multi-class novelty detector for data streams under intermediate latency. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.

Din, S., Shao, J., Kumar, J., Mawuli, C., Mahmud, S., Zhang, W., and Yang, Q. (2021). Data stream classification with novel class detection: a review. *Knowledge and Information Systems*, 63:2231–2276.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25.

Fahy, C., Yang, S., and Gongora, M. (2022). Scarcity of labels in non-stationary data streams: A survey. *ACM Computing Surveys*, 55(2):40.

Ferdaus, M., Dam, T., Alam, S., and Pham, D. (2024). X-Fuzz: An evolving and interpretable neuro-fuzzy learner for data streams. *IEEE Transactions on Artificial Intelligence*, 5(8):4001–4012.

Gama, J., Rodrigues, P., Spinosa, E., and Carvalho, A. (2010). Knowledge discovery from data streams. In *Web Intelligence and Security*, pages 125–138. Springer.

Gu, X. (2023). A dual-model semi-supervised self-organizing fuzzy inference system for data stream classification. *Applied Soft Computing*, 136:110053.

Pal, N., Pal, K., Keller, J., and Bezdek, J. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13(4):517–530.

Pimentel, M., Clifton, D., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249.

Samadi, N., Tanha, J., and Jalili, M. (2025). A weighted semi-supervised possibilistic fuzzy c-means algorithm for data stream classification and emerging class detection. *Knowledge-Based Systems*, 309:112831.

Urio-Larrea, A., Camargo, H., Lucca, G., Asmus, T., Marco-Detchart, C., Schick, L., Lopez-Molina, C., Andreu-Perez, J., Bustince, H., and Dimuro, G. (2025). Data stream clustering: introducing recursively extendable aggregation functions for incremental cluster fusion processes. *IEEE Transactions on Cybernetics*, 55(3):1421–1435.

Wen, K. and Zhao, T. (2026). Interpretable and robust online self-organizing granule-based fuzzy rule classification for data stream. *International Journal of Fuzzy Systems*, 28:382–396.