Highlights

Stable Region Enhanced Online Learning Method for Intermediate Verification Latency and Concept Drift

Zixin Zhong, Liyan Song, Fengzhen Tang, Bo Yuan

- Identify and utilize stable region data to address intermediate verification latency.
- Propose CDI, an unsupervised test statistic to identify stable regions.
- Propose DUST, a learning framework that utilizes stable regions identified by CDI.
- Conduct extensive experiments to evaluate the effectiveness of the proposed methods.

Stable Region Enhanced Online Learning Method for Intermediate Verification Latency and Concept Drift

Zixin Zhong[a], Liyan Song[b,c,*], Fengzhen Tang[c], Bo Yuan[a]

[a] *Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China*
[b] *Faculty of Computing, Harbin Institute of Technology, Harbin, China*
[c] *State Key Laboratory of Robotics and Intelligent Systems, Shenyang Institute of automation, Chinese Academy of Sciences, Shenyang, China*

Abstract

In non-stationary data streams, the challenges of concept drift are compounded by Intermediate Verification Latency (IVL), which refers to the delay between the arrival of data features and their corresponding labels. This paper addresses IVL from a novel perspective by examining stable regions within the feature space, where data remain unaffected by concept drift. We propose Centroid-based Drift Index (CDI), an unsupervised metric that quantifies drift to identify these stable regions. Building upon this, we introduce Data Utilization informed by STable regions (DUST), a framework that effectively utilizes both temporarily unlabeled and delayed labeled data by distinguishing stable region data through micro-clusters. Comprehensive experiments conducted on synthetic and real-world datasets validate the effectiveness of CDI and DUST. The results demonstrate that DUST yields accuracy improvements ranging from 0.59% to 2.29% across various base models, with an average accuracy of 84.10% on synthetic streams and 63.18% on real-world streams. The source code and supplementary material is available at `https://github.com/ZeroZill/CDI_DUST`.

Keywords:
online learning, data stream learning, concept drift, verification latency, label delay, stable region

*Corresponding author
Email addresses: `12232431@mail.sustech.edu.cn` (Zixin Zhong), `songly@hit.edu.cn` (Liyan Song), `tangfengzhen@sia.cn` (Fengzhen Tang), `yuanb@sustech.edu.cn` (Bo Yuan)

1. Introduction

While online learning has been widely applied in data stream analysis and extensively studied [1, 2], a significant limitation of most existing approaches is the overly optimistic assumption that true labels are immediately available upon the arrival of data points. In practice, there often exists a *non-negligible delay* in obtaining these true labels. For instance, in web performance optimization, resource prefetching aims to enhance bandwidth efficiency by preloading resources in data-intense network applications; yet, feedbacks on the use of these prefetched resources are often delayed [3]. Similarly, in tool condition monitoring, indirect measurements such as cutting force, vibration, and power signals are utilized to assess tool wear. The substantial cost and time required for manual labeling frequently introduce significant delays in verification, primarily due to the absence of immediate fault status labels [4]. In these contexts, the *finite* delay between receiving a data point and obtaining its corresponding true label is referred to as *intermediate verification latency (IVL)* [5], a factor rarely considered in conventional online learning studies.

IVL significantly complicates online learning, especially coupled with the frequent issue of concept drift. *Concept drift* describes the phenomenon where the underlying data distribution changes over time, a common characteristic of non-stationary data streams [6, 7]. Under IVL, the lack of immediate access to true labels complicates the detection of concept drift and the adaptation of models to new concepts, potentially leading to deteriorating performance. In this scenario, only temporarily unlabeled data and delayed labeled data are available. Therefore, effectively utilizing these two types of data is crucial for addressing the challenge posed by IVL.

A promising strategy for leveraging these data types involves identifying and utilizing data from stable regions within the feature space. As noted by [8], certain regions of the feature space remain unaffected by the drifting concept over extended periods. Data points within these stable regions can retain their characteristics longer, making them reliable indicators for model updates. By concentrating on data from these stable regions in the context of IVL, we can more effectively use temporarily unlabeled data and delayed labeled data, thereby reducing the label noise and inaccuracies introduced

by data already affected by concept drift. Although some studies investigated the understanding and identification of stable regions in conventional online learning settings [9, 10], to the best of our knowledge, we are among the first to consider leveraging stable regions specifically in the presence of IVL, where supervision is limited.

Existing methods for addressing IVL can be broadly classified into three categories: 1) *wait-until-arrival*, 2) *pseudo-labeling with rollback*, and 3) *pseudo-labeling with correction*. The simplest approach, wait-until-arrival, involves passively waiting for true labels to arrive before updating the model [11, 12, 13]. This approach relies solely on delayed labeled data and disregards the potential value of temporarily unlabeled data, probably resulting in slower adaptation to concept drift. The second approach, pseudo-labeling with rollback, assigns pseudo-labels to temporarily unlabeled data and updates the model immediately. If these pseudo-labels are later found incorrect upon receiving true labels, the model state will be rolled back [14, 4]. However, without an effective filtering process, pseudo-labels may introduce significant noise, leading to frequent rollbacks and instability. A more recent approach, pseudo-labeling with correction, restricts the assignment of pseudo-labels to data located in the central regions of each class, which are considered less affected by concept drift. This approach also corrects the model by resampling synthetic data based on delayed labeled data when errors in pseudo-labels are identified [15]. Despite these refinements, this approach still risks exposing the model to noise if concept drift is severe, and may place undue emphasis on outdated data.

None of the three categories explicitly addresses the identification and utilization of stable regions. However, properly utilizing stable regions could help mitigate their limitations and enhance model performance.

To bridge this gap, we propose an unsupervised test statistic called *Centroid-based Drift Index (CDI)*, which quantifies the degree of drift in a given region by measuring the centroid offset between two time steps. This allows us to determine whether a data point lies within a stable region. Building upon this, we introduce a new framework, *Data Utilization informed by STable regions (DUST)*. This framework leverages micro-clusters to efficiently summarize data distribution with minimal space usage and utilizes CDI to distinguish stable region data, thereby enabling more effective manage-

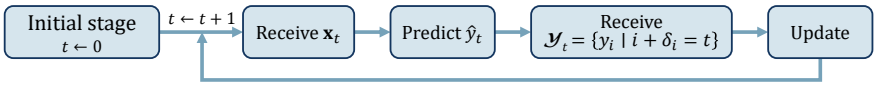


Figure 1: The "test-then-train" process in the context of IVL (a.k.a., wait-until-arrival strategy).

ment of temporarily unlabeled and delayed labeled data.

The main contributions of this work are as follows:

1. We are the first to address the challenge of distinguishing and utilizing stable region data in the IVL context, introducing CDI as a novel test statistic for assessing region stability and identifying stable region data points.

2. We propose DUST, an online learning framework that employs micro-clusters to efficiently summarize data distributions and leverage CDI to identify stable region data. This approach enhances the effective utilization of both temporarily unlabeled and delayed labeled data, thereby mitigating the impact of IVL.

3. We conduct extensive experiments on a variety of synthetic and real-world datasets to validate the effectiveness of CDI and the DUST framework.

2. Problem Formulation

Consider a data stream $S = \{(\mathbf{x}_t, y_t) \mid t = 1, 2, \ldots\}$ with concept drift and IVL, where $\mathbf{x}_t \in \mathcal{X} = \mathbb{R}^d$ represents a d-dimensional feature vector arriving at the t-th time step and $y_t \in \mathcal{Y} = \{0, 1, \ldots, C\}$ represents its true label. Each observation pair $(\mathbf{x}_t, y_t)$ follows a joint probability distribution $P_t(\mathbf{x}, y)$. Due to concept drift, observations at different time steps t_u and t_v may originate from different joint probability distributions, formulated as $P_{t_u}(\mathbf{x}, y) \neq P_{t_v}(\mathbf{x}, y)$ [16].

Let $T(\cdot)$ denote the arrival time of either feature vectors or true labels. The time delay between the arrival of the true label and the feature vector (i.e., verification latency) can be defined as $\delta_t = T(y_t) - T(\mathbf{x}_t)$ [17]. In the context of IVL, this time delay is finite, formulated as $0 < \delta_t < \infty$ [5].

We formulate the problem of online learning with IVL following the conventional "test-then-train" or prequential paradigm [18, 16], as illustrated in Figure 1. Prior to the

online phase, if initial data is available, a pre-training step is performed to warm up the model. At each time step t, the feature vector $\mathbf{x}_t$ is made available first. The classifier $\mathcal{H} : \mathcal{X} \mapsto \mathcal{Y}$ then produces an instant prediction $\hat{y}_t = \mathcal{H}(\mathbf{x}_t)$. Unlike conventional online learning, in the presence of IVL, true labels from previous time steps, denoted as $\mathcal{Y}_t = \{y_i \mid i + \delta_i = t\}$, may also be received at time step t. Subsequently, the model is updated using all available data: the temporarily unlabeled data $\mathcal{U}_t = \{(\mathbf{x}_t, \cdot)\}$ and the delayed labeled data $\mathcal{L}_t = \{(\mathbf{x}_i, y_i) \mid i + \delta_i = t\}$.

3. Background and Related Work

3.1. Intermediate Verification Latency

As discussed in Section 1, research addressing IVL is relatively sparse and can be broadly cast into three main categories as follows:

1) Wait-until-arrival. This is the simplest and most straightforward approach, as employed in [11, 12, 13]. It follows the basic "test-then-train" paradigm under IVL, where the model waits for true labels to become available before updating, as depicted in Figure 1. Consequently, the model relies solely on delayed labeled data, entirely disregarding the potentially valuable temporarily unlabeled data, thereby probably resulting in slower adaptation to new concepts.

2) Pseudo-labeling with rollback. SkipE-RNN [14] and SERMON [4] combine pseudo-labeling temporarily unlabeled data with a rollback mechanism. These methods assign pseudo-labels based on temporal dependencies with historical data, allowing for immediate model updates. To mitigate the noise inherent in pseudo-labels, SkipE-RNN and SERMON incorporate a rollback mechanism. This mechanism introduce a penalty to the loss function when an erroneous update is detected, encouraging the model to revert to a prior state. While this rollback can reduce the impact of pseudo-label noise, the indiscriminate application of pseudo-labels to all temporarily unlabeled data remains overly aggressive. Frequent rollbacks can destabilize the model and erase valuable information learned after an erroneous update. Furthermore, maintaining historical data and model states for rollback operations requires significant storage.

3) Pseudo-labeling with correction. MIPLOSC [15] employs a pseudo-labeling with correction strategy. Unlike the previous methods, MIPLOSC includes a filtering step that only uses pseudo-labeled data from the central regions of each class, which are presumed to be more stable under concept drift. The model is then corrected by resampling synthetic data around identified errors. Although this approach improves the utilization of temporarily unlabeled data by reducing noise through selective pseudo-labeling, it still confines useful data to the central regions of each class. This limitation can be problematic in scenarios of severe concept drift, where even these central regions may become unstable. Additionally, the correction process risks overemphasizing outdated data, potentially hindering the model's adaptation to emerging concepts.

Despite advancements, these methods share a common limitation: a coarse approach to data utilization, which often results in valuable information loss (in Wait-until-arrival and Pseudo-labeling with rollback approaches) or excessive label noise (in Pseudo-labeling with rollback and Pseudo-labeling with correction approaches). Focusing on stable regions in the feature space, where data characteristics remain relatively consistent, offers a promising solution. These regions provide more reliable data for accurate model updates. Therefore, we propose leveraging the concept of stable regions to enhance IVL management in our work.

3.2. Other Techniques Potentially Adaptable to IVL

Beyond methods explicitly designed for the IVL setting, several general online learning techniques, which primarily address label scarcity, can be conceptually adapted or provide useful mechanisms for handling temporarily unlabeled data.

OSNN [19] is an online semi-supervised neural network that employs manifold-based training on a dynamic graph to leverage unlabeled data for learning meaningful representations. It maintains robustness to concept drift through dynamic topology learning. Conversely, WSPFCM-DS [20] utilizes a possibilistic fuzzy c-means framework with micro-clusters. This method handles drift by executing iterative cluster updates and identifying novel class instances based on the typicality of data points within existing cluster structures. Another strategy is presented in GTSS [21], which is a self-training framework enhanced by graph theory. It selects pseudo-labels by combining

the confidence of conformal prediction with graph node centrality and detects emerging classes by analyzing the local density of labeled samples.

Despite these potential advancements, a significant gap remains, as these approaches are not specifically designed to handle the IVL challenge. Consequently, they lack the ability to strategically integrate delayed true labels and cannot benefit from the additional supervision these labels provide for model refinement and error reduction.

3.3. Stable Region Detection

Identifying regions that remain stable or are affected by concept drift – often referred to as drift localization or segmentation in the literature [22] – is of practical significance for model updates in online learning settings where concept drift frequently occurs [10]. It is worth noting that identifying stable regions is conceptually equivalent to identifying drift regions, as the areas outside the stable regions are the drift regions. Thus, we use the term "stable regions" in this paper. By pinpointing data affected by drift, we can preserve valuable data information while discarding obsolete data. Furthermore, distinguishing between drifting and stable regions is essential for fully understanding and explaining the dynamics of concept drifts [16, 22].

Several studies have explored the identification and utilization of stable regions under conventional online learning settings, where verification latency is not considered. For instance, Lu et al. [23] emphasized that detecting stable regions facilitates the identification of outdated data points. In a subsequent study [24], the authors adapted their model by excluding outdated data outside the stable regions. Similarly, Liu et al. [9] introduced a test statistic called Local Drift Degree (LDD) to detect local density changes, which reflect shifts in local distributions. Building upon LDD, they further developed a Density Synchronized Drift Adaptation algorithm (LDD-DSDA), which used stable regions as the basis for an instance selection strategy to construct a training set that continuously tracks new data. Furthermore, in [25], a formal definition of drift localization was presented, along with a theoretical framework that transformed the drift localization problem into a classical probabilistic classification problem. More recently, a feature-based drift detector (FBDD) [26] was developed for batch-processed

data. This approach identifies stable regions by ranking features via LASSO and monitoring their distributional shifts using statistical tests.

Despite the extensive research on stable region detection in conventional online learning, these methods are not readily adaptable to scenarios involving IVL. Therefore, there is a pressing need for an unsupervised stable regions detection method suitable for the context of IVL.

4. The Proposed Test Statistics

This section proposes our method for identifying stable regions, called Centroid-based Drift Index (CDI), which serves as the foundation of our DUST framework (see Section 6). Specifically, Section 4.1 defines CDI; and Section 4.2 presents the approach for determining whether a given data point resides in a stable region.

4.1. Definition of CDI

The core idea behind our proposed CDI is that the centroid of a stable region should remain consistent over time. By analyzing the shift of the centroid within a local area across two time periods, we can assess the stability of the region. Consequently, for any target data point, we can then determine whether it lies within a stable region based on the CDI of its local area. [1]

Consider two windows, A and B, each containing d-dimensional data points sampled from two time periods. Let W denote the target region we wish to inspect for its stability. Both A and B may include data points that fall within the range of W. We define the subsets $A^{(W)} = \{\mathbf{x}_i \mid \mathbf{x}_i \in W \wedge \mathbf{x}_i \in A\}$ and $B^{(W)} = \{\mathbf{x}_i \mid \mathbf{x}_i \in W \wedge \mathbf{x}_i \in B\}$ to be the sets of data points from A and B that fall within W, respectively. The numbers of data points in these subsets are denoted as $n_A^{(W)} = |A^{(W)}|$ and $n_B^{(W)} = |B^{(W)}|$. The centroids of the data points in W from windows A and B are defined respectively as

$$\mathbf{c}_A^{(W)} = \frac{1}{n_A^{(W)}} \sum_{\mathbf{x}_i \in A^{(W)}} \mathbf{x}_i, \quad \mathbf{c}_B^{(W)} = \frac{1}{n_B^{(W)}} \sum_{\mathbf{x}_j \in B^{(W)}} \mathbf{x}_j. \tag{1}$$

[1] The detailed rationale of CDI is discussed in Section 10 of the supplementary material.

Definition 1. *Given two windows of data, A and B, the Centroid-based Drift Index (CDI) for a given region W is defined as:*

$$\Delta^{(W)} = \frac{n_A^{(W)} n_B^{(W)}}{n_A^{(W)} + n_B^{(W)}} (\mathbf{c}_A^{(W)} - \mathbf{c}_B^{(W)})^\top \boldsymbol{\Sigma}^{(W)^{-1}} (\mathbf{c}_A^{(W)} - \mathbf{c}_B^{(W)}). \tag{2}$$

where $\boldsymbol{\Sigma}^{(W)}$ denotes the covariance matrix of all data points in W.

Theorem 1. *If $A^{(W)}$ and $B^{(W)}$ are identically distributed, then $\Delta^{(W)}$ follows a chi-square distribution with d degrees of freedom, i.e., $\Delta^{(W)} \sim \chi_d^2$, where d is the dimension of $A^{(W)}$ and $B^{(W)}$.*

Proof 1. *If the data distributions from which A and B sample their data points are identical, then the mean and covariance of the data points from A and B within W are also identical, denoted as $\mu^{(W)}$ and $\boldsymbol{\Sigma}^{(W)}$, respectively.*

According to the Central Limit Theorem, the centroid of the data points from $A^{(W)}$, denoted as $\mathbf{c}_A^{(W)}$, follows the multivariate Gaussian distribution below:

$$\mathbf{c}_A^{(W)} \sim \mathcal{N}\left(\mu^{(W)}, \frac{\boldsymbol{\Sigma}^{(W)}}{n_A^{(W)}}\right) \tag{3}$$

Similarly, for $\mathbf{c}_B^{(W)}$, we have

$$\mathbf{c}_B^{(W)} \sim \mathcal{N}\left(\mu^{(W)}, \frac{\boldsymbol{\Sigma}^{(W)}}{n_B^{(W)}}\right) \tag{4}$$

The difference between $\mathbf{c}_A^{(W)}$ and $\mathbf{c}_B^{(W)}$ then follows the below distribution:

$$\mathbf{o}^{(W)} = \mathbf{c}_A^{(W)} - \mathbf{c}_B^{(W)} \sim \mathcal{N}\left(\mathbf{0}, \frac{n_A^{(W)} + n_B^{(W)}}{n_A^{(W)} n_B^{(W)}} \boldsymbol{\Sigma}^{(W)}\right) \tag{5}$$

According to the properties of multivariate Gaussian distributions, the value of $\Delta^{(W)}$ follows a chi-square distribution with d degrees of freedom:

$$\begin{aligned} \Delta^{(W)} =& \frac{n_A^{(W)} n_B^{(W)}}{n_A^{(W)} + n_B^{(W)}} (\mathbf{c}_A^{(W)} - \mathbf{c}_B^{(W)})^\top \boldsymbol{\Sigma}^{(W)^{-1}} (\mathbf{c}_A^{(W)} - \mathbf{c}_B^{(W)}) \\ =& \mathbf{o}^{(W)^\top} \left(\frac{n_A^{(W)} + n_B^{(W)}}{n_A^{(W)} n_B^{(W)}} \boldsymbol{\Sigma}^{(W)}\right)^{-1} \mathbf{o}^{(W)} \sim \chi_d^2 \end{aligned} \tag{6}$$

□

The calculation can be interpreted as the Mahalanobis distance between the centroids $\mathbf{c}_A^{(W)}$ and $\mathbf{c}_B^{(W)}$, given the assumption that both are drawn from the same data

Algorithm 1: Local Region's CDI of Target Data

Input:
Data points windows A, B; Target data point $\mathbf{p}$; Neighborhood ratio ρ (default=0.1);
Output: CDI value of $\mathbf{p}$'s local region, $\Delta^{(W)}$.
`// Determine the local region W`
Find the $(\rho \cdot |A|)$-NN of $\mathbf{p}$ in A and record the maximum distance $r^{(W)}$;
Retrieve the data points from A and B within distance $r^{(W)}$ from $\mathbf{p}$, denoted as $A^{(W)}$ and $B^{(W)}$;;
`// Calculate meta-data of W`
Count the number of data points $n_A^{(W)}$ and $n_B^{(W)}$ in $A^{(W)}$ and $B^{(W)}$, respectively;
Compute the centroids $\mathbf{c}_A^{(W)}$ and $\mathbf{c}_B^{(W)}$ for $A^{(W)}$ and $B^{(W)}$, respectively;
Compute the covariance matrix $\mathbf{\Sigma}^{(W)}$ for $A^{(W)} \cup B^{(W)}$;
`// Calculate CDI`
Compute $\Delta^{(W)}$ by Eq. (2) based on meta-data of W;

distribution. We leverage the fact that $\Delta^{(W)}$ follows a chi-square distribution with d degrees of freedom to quantify the stability of the region. For a chosen significance level α, if $\Delta^{(W)} \geq \chi^2_{d,\alpha}$, where $\chi^2_{d,\alpha}$ is the corresponding chi-square critical value, it indicates that the target region is not stable.

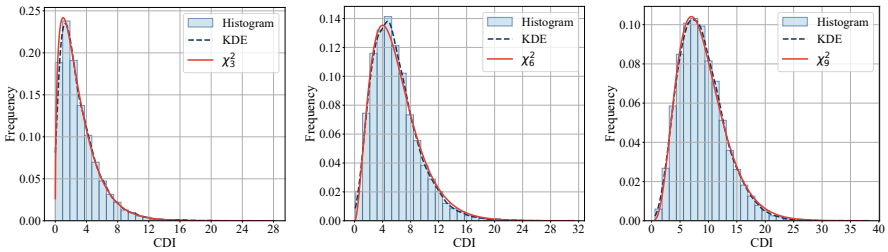
4.2. Calculating Local Region CDI for Target Data

We have introduced a novel test statistic CDI to assess the stability of a given region, without requiring supervision (i.e., true labels). Building upon this, CDI can also be used to determine whether a data point resides in a stable region by setting W as the local region surrounding the target data and calculating the CDI value. This assessment of local region stability can guide decisions about the data's further use.

To identify the local region W surrounding the target data, we employ the k-NN algorithm, chosen for its effectiveness in defining local spaces and its independence from the shape of the feature space [9]. The overall procedure for calculating CDI for a target data point, based on meta-data, is summarized in Algorithm 1.

5. Experimental Validation of CDI

This section evaluates the effectiveness of CDI. We first illustrate the alignment between its experimental and theoretical distributions. Next, we visualize its effect on indicating stable region data and conduct a quantitative assessment, comparing it to LDD [9]. For consistency and fair comparison, we set the hyper-parameters to $k = 100$ and $\alpha = 0.95$ for both CDI and LDD, in line with [9].



(a) $d = 3$ (b) $d = 6$ (c) $d = 9$

Figure 2: Distribution of CDI for different d dimensions. The fitted KDE curves closely align with the theoretical chi-square distributions, demonstrating strong support for Theorem 1.

5.1. Illustrative Support of Theoretical CDI Distribution

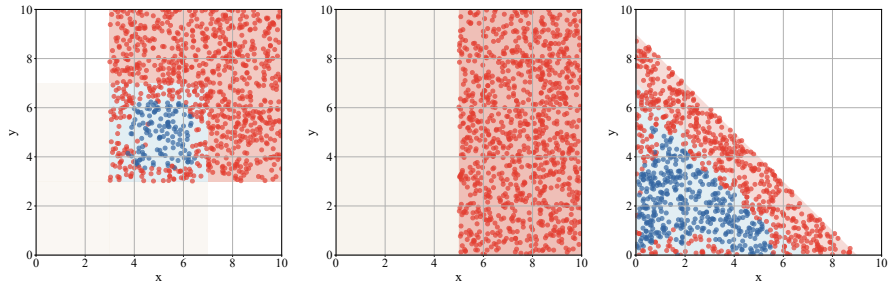
This section experimentally illustrates whether the distribution of CDI values align with a chi-square distribution under identical data distributions.

To this end, we generate two data windows, A and B, each containing 10,000 points drawn from an d-dimensional standard multivariate Gaussian distribution. For each data point in B, we compute its CDI value and fit a distribution curve using Kernel Density Estimation (KDE). We then compare this fitted curve with the theoretical chi-square distribution for d degrees of freedom. The experiment examines the CDI distribution for d values of $\{3, 6, 9\}$.

Figure 2 shows the histogram of CDI values, the fitted KDE curves, and the theoretical chi-square distributions. We can see that the KDE curves closely match the theoretical chi-square distributions, providing strong evidence that CDI aligns well with the expected theoretical distribution. This finding reinforces the validity of the theoretical foundation underlying the proposed CDI in Theorem 1.

5.2. Illustrative Performance Evaluation in Stable Region Detection

This section demonstrates the effectiveness of CDI in identifying stable regions by comparing data windows before and after a deliberately introduced concept drift. We simulate three scenarios, each comprising two data windows with 1,000 points per window: one sampled prior to the drift and the other afterwards. These scenarios are outlined as follows:



(a) (b) (c)

Figure 3: Visualization of CDI results across three scenarios. Red backgrounds represent drift regions, while blue backgrounds represent stable regions. Through CDI values, red points are judged as drifting, while blue points are judged as stable.

(a) Prior to the drift, the data follows a uniform distribution with the ranges $x \in [0, 7]$ and $y \in [0, 7]$. After the drift, the data remains uniformly distributed, but within the ranges $x \in [3, 10]$ and $y \in [3, 10]$. Thus, the stable region should be defined as $x \in [3, 7]$ and $y \in [3, 7]$.

(b) Prior to the drift, the data follows a uniform distribution with $x < 5$. After the drift, the data remains uniformly distributed, but with $x \geq 5$. In this scenario, there is no stable region.

(c) The data is generated within the constraints $x \geq 0$, $y \geq 0$, and $x+y \leq a$, following a uniform distribution. Prior to the drift, $a = 7$, and after the drift, $a = 9$. Thus, the drift region should be defined as $7 < x + y < 9$.

For each data point after the concept drift, we compute CDI values to determine whether it falls within the stable regions. The results, shown in Figure 3, demonstrate that CDI can effectively distinguish between data points in stable and drift regions. Notably, CDI takes a conservative approach, often determining points near drift boundaries as drifting. In the context of IVL, this conservatism does not significantly hinder the use of stable data points, instead, it enhances their reliability. Overall, these results validate the effectiveness of CDI in precisely identifying data within stable regions.

5.3. Quantitative Performance Evaluation in Stable Region Detection

This section quantitatively evaluates the effectiveness of CDI in identifying data points in stable regions during an online learning process with a verification latency δ_t

of 1,000 time steps. We employ two sliding windows of equal length l: the first window contains the most recent l unlabeled data points, reflecting the current data distribution; the second window also spans l data points but ends δ_t time steps earlier than the first window, simulating delayed labeled data. Both windows advance forward by one step as new data arrive. The CDI is computed using the data from these two windows.

The experiments are conducted on two synthetic datasets, SEA [27] and Sine [28], chosen for their simplicity and clear class boundaries, which allow accurate separation of ground truth stable and drift regions, enabling a quantitative evaluation of CDI's performance. Each dataset contains 10,000 data points, with an abrupt drift introduced at the 6,000-th time step. The details of these datasets are as follows:

(a) *SEA.* Three continuous features uniformly sampled from $[0, 10]$. Class labels are determined by whether $x_1 + x_2$ is below a threshold a, which shifts from 7 to 9 at the drift point.
(b) *Sine.* Two numeric features uniformly sampled from $[0, 1]$. Samples below a predefined sine curve are labeled positive, those above are labeled negative. Concept drift is induced by changing the curve from $y = \sin(x)$ to $y = 0.5 + 0.3\sin(3\pi x)$.

The abrupt drift at the 6,000-th time step alters the true class boundaries. Due to verification latency, labels obtained between the 6,000-th and 7,000-th steps do not reflect the latest boundaries. Consequently, during this interval, data points falling between the old and new boundaries are located within drift regions, whereas all other points reside in stable regions.

CDI is compared with LDD [9], a test statistic designed to detect concept drift via regional density changes. Although LDD is primarily intended for conventional online learning contexts, it can be adapted with minor modifications for the IVL scenarios. To investigate the impact of window length, we vary l across $\{500, 1000, \ldots, 3000\}$, and evaluate each configuration across 10 independent runs.

To evaluate the performance of CDI in detecting stable regions during the concept drift from the 6,000-th to the 7,000-th data points, we use the following metrics: $Pre. = \frac{TP}{TP+FP}$, $Rec. = \frac{TP}{TP+FN}$, $Acc. = \frac{TP+TN}{TP+TN+FP+FN}$, and $F1 = \frac{2 \cdot Pre. \cdot Rec.}{Pre. + Rec.}$. Here, TP denotes true positives (correctly identified stable points), TN denotes true negatives

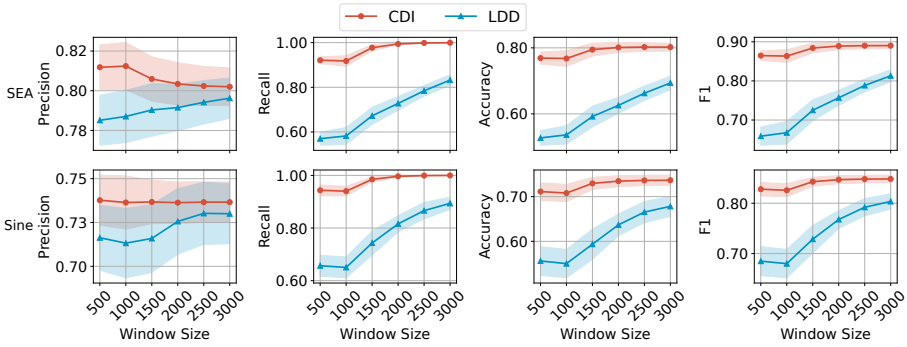


Figure 4: Performance comparison of CDI and LDD across varying window sizes.

(correctly identified drifting points), FP denotes false positives (incorrectly identified stable points), and FN denotes false negatives (incorrectly identified drifting points).

Figure 4 shows the performance of CDI and LDD across the SEA and Sine streams as the window size varies. Key findings include: 1) CDI consistently outperforms LDD in terms of *Pre.*, *Rec.*, *Acc.* and $F1$ at every tested window size. 2) Aside from CDI's precision, all metrics for both methods improve with larger windows until they plateau. This is likely due to larger windows providing more data for reliable estimates, enabling each method to approach its performance ceiling. 3) CDI reaches near-peak performance with relatively small windows (≈1,500), while LDD requires larger windows to reach comparable performance. CDI uses a fixed chi-square critical value determined solely by feature dimension, making it independent of sample size. In contrast, LDD must first gather all regional density values across the window, estimate their standard deviation, and then determine the normal-distribution threshold, necessitating larger sample sizes for reliable estimation, which also increases time consumption. 4) CDI's precision exhibits a slight decline as window size increases, about 1% on SEA and remains nearly flat on Sine. This decline reflects CDI's inclusion of additional borderline points as stable, which boosts true positives but also causes false positives. While CDI is generally conservative, as discussed in Section 5.2, this observation suggests that its conservativeness diminishes slightly with larger window sizes. Nevertheless, the drop in precision is acceptable when balanced against the significant gains in other metrics.

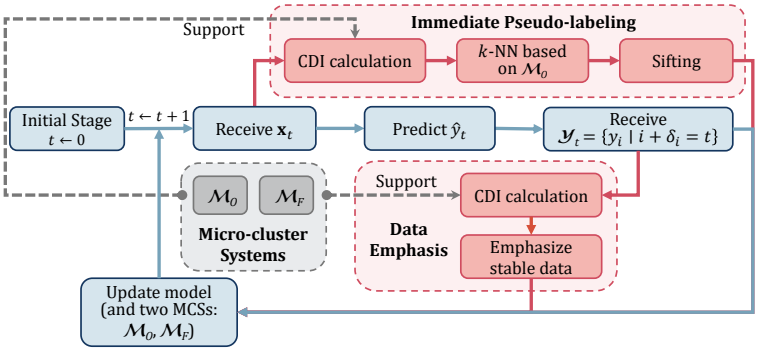


Figure 5: The workflow of proposed DUST framework.

The experiments use simple synthetic streams with a single abrupt drift to illustrate the performance of CDI under varying window sizes. In more complex or real-world streams featuring multiple drifts, excessively large windows may blend distributions from different concepts, thereby degrading performance. In such cases, CDI's strong performance with smaller windows makes it an especially viable option.

6. The Proposed Online Learning Framework for IVL

Building upon CDI, we introduce the DUST framework in the IVL context, extending the wait-until-arrival strategy. The core idea is to identify stable region data using CDI and leverage it through distinct strategies tailored to their specific characteristics. For temporarily unlabeled data, we assign pseudo-labels to sufficiently reliable instances for immediate model updates. For delayed labeled data, we emphasize those within stable regions. The workflow of DUST is illustrated in Figure 5.

6.1. Micro-cluster Systems for Maintaining Meta-data

In online CDI computation, two sliding windows are typically employed: one for fresher data and another for older data. However, sliding windows present a fundamental trade-off: smaller windows may capture limited data distributions, while larger windows incur high storage and computational costs. To address this issue, data points are aggregated into micro-clusters based on geometric proximity, retaining only essential meta-data including counts, centroids, and covariance matrices. This enables

Algorithm 2: Local Region's CDI of Target Data via Micro-cluster Systems

Input:
Micro-cluster systems, $\mathcal{M}_F$, $\mathcal{M}_O$; Target data point $\mathbf{p}$; Neighborhood ratio ρ (default=0.1);
Output: CDI value of $\mathbf{p}$'s local region, $\Delta^{(W)}$.

`// Determine the local region W`

Find the $(\rho \cdot |\mathcal{M}_O|)$-NN of $\mathbf{p}$ in $\mathcal{M}_O$ based on centroids and record the maximum distance $r^{(W)}$;

Retrieve micro-clusters in $\mathcal{M}_O$ and $\mathcal{M}_F$ whose centroids are within distance $r^{(W)}$ from $\mathbf{p}$, denoted as $\mathcal{M}_O^{(W)}$ and $\mathcal{M}_F^{(W)}$;

`// Calculate meta-data of W`

Compute $n_A^{(W)}$, $n_B^{(W)}$ by Eq. (7) based on meta-data of $\mathcal{M}_O^{(W)}$, $\mathcal{M}_F^{(W)}$, respectively;

Compute $\mathbf{c}_A^{(W)}$, $\mathbf{c}_B^{(W)}$ by Eq. (8) based on meta-data of $\mathcal{M}_O^{(W)}$, $\mathcal{M}_F^{(W)}$, respectively;

Compute $\mathbf{\Sigma}^{(W)}$ by Eq. (9) based on meta-data of $\mathcal{M}_O^{(W)} \cup \mathcal{M}_F^{(W)}$;

`// Calculate CDI`

Compute $\Delta^{(W)}$ by Eq. (2) based on meta-data of W;

efficient CDI computation by leveraging micro-cluster statistics.

For region W containing m micro-clusters, where the i-th micro-cluster contains $n_i^{(W)}$ data points with centroid $\mathbf{c}_i^{(W)}$ and covariance matrix $\mathbf{\Sigma}_i^{(W)}$, we compute three key statistics for the entire region: the total number of data points $n^{(W)}$, the centroid $\mathbf{c}^{(W)}$, and the covariance matrix $\mathbf{\Sigma}^{(W)}$, as follows:

$$n^{(W)} = \sum_{i=1}^{m} n_i^{(W)} \tag{7}$$

$$\mathbf{c}^{(W)} = \sum_{i=1}^{m} \frac{n_i^{(W)}}{n^{(W)}} \mathbf{c}_i^{(W)} \tag{8}$$

$$\mathbf{\Sigma}^{(W)} = \sum_{i=1}^{m} \frac{n_i^{(W)}}{n^{(W)}} \left[\mathbf{\Sigma}_i^{(W)} + \left(\mathbf{c}^{(W)} - \mathbf{c}_i^{(W)} \right) \left(\mathbf{c}^{(W)} - \mathbf{c}_i^{(W)} \right)^\top \right] \tag{9}$$

We adopt and enhance the space-efficient micro-clustering algorithm from Din et al. [29] to support covariance matrices. Micro-clusters are initialized via k-Means, with meta-data updated using Eqs. (7)–(9) whenever new points arrive or clusters merge. In DUST, two micro-cluster systems are maintained: $\mathcal{M}_F$ for recent unlabeled data and $\mathcal{M}_O$ for older labeled data, as illustrated in Figure 5. The complete process is formalized in Algorithm 2, which computes the local CDI for a target point $\mathbf{p}$ by determining the region W and calculating the necessary statistics.

6.2. Data Utilization Informed by Stable Regions

1) Temporarily Unlabeled Data. Temporarily unlabeled data is sampled from the most recent data distribution. Due to potential concept drift, this data may not fully align

with the model's knowledge, which is based on earlier obsolete distributions. However, in stable regions unaffected by concept drift, past knowledge remains valid, allowing reliable pseudo-labels to be assigned.

To leverage this characteristic, we introduce an *Immediate Pseudo-labeling (IP)* mechanism for generating and filtering pseudo-labeled data, as illustrated by the red block at the top of Figure 5. Pseudo-labels are generated using k-NN on the micro-clusters in $\mathcal{M}_O$, with majority voting determining the assigned pseudo-label y_t^* and its associated probability $p(y_t^*)$. We then determine a threshold θ to decide whether the pseudo-labeled data should be used for immediate updates, calculated as follows:

$$\theta = C^{\Delta^{(W)}/\chi^2_{d,\alpha}-1} \tag{10}$$

where C is the number of classes, $\Delta^{(W)}$ is the CDI value, $\chi^2_{d,\alpha}$ is the critical value of CDI that indicates whether a data point is in a stable region. Only pseudo-labeled data with $p(y_t^*) \geq \theta$ will be used for immediate updates.

If the local data distribution remains stationary, i.e., $\Delta^{(W)} = 0$, then $\theta = 1/C$. In this case, the pseudo-label is used directly, as $p(y_t^*) \geq 1/C$ is always satisfied. Conversely, if the local data distribution undergoes significant change such that $\Delta^{(W)} \geq \chi^2_{d,\alpha}$, then $\theta \geq 1$. This indicates that the data point is in a drift region, and the pseudo-label will not be used for immediate updates in this case.

2) Delayed Labeled Data. Delayed labeled data is crucial for model updates as they provide ground truth information. However, since these data are collected at earlier times, they may become outdated due to drifts, resulting in misalignment with the current concept. Nonetheless, among all delayed labeled data, those from stable regions remain consistent despite the occurrence of concept drift. Therefore, focusing on stable region data can serve as a bridge, assisting the model's adaptation to new concepts.

In DUST, we emphasize stable region data by training on these data points one additional time. We do not discard data from drift regions because CDI might conservatively classify some stable region data as drifting, as discussed in Section 5.2. Even if a delayed data point is flagged as being in a drift region, its label may still be accurate. Discarding these data points could result in the loss of valuable information. Instead,

Algorithm 3: The Proposed DUST Framework

Input:
Input features $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots\} \in \mathbb{R}^d$; Delayed target labels $\mathbf{Y} = \{y_1, y_2, \ldots\}$ (delayed by $\delta_t \in (0, \infty)$); Micro-cluster systems $\mathcal{M}_F$, $\mathcal{M}_O$; Classifier $\mathcal{H}$; Number of nearest neighbors k; Confidence level of CDI α;
Output: Predicted labels $\widehat{\mathbf{Y}} = \{\hat{y}_1, \hat{y}_2, \ldots\}$

1 Initialize $\mathcal{H}$, $\mathcal{M}_F$, and $\mathcal{M}_O$ with available data;
2 **for** $t \leftarrow 1, 2, \ldots$ **do**
3 Make prediction $\hat{y}_t \leftarrow \mathcal{H}(\mathbf{x}_t)$;
  `// IP mechanism for temporarily unlabeled data`
4 Perform k-NN on $\mathbf{x}_t$ using $\mathcal{M}_O$ to get pseudo-label y_t^* and probability $p(y_t^*)$;
5 Compute CDI value $\Delta^{(W)}$ for $\mathbf{x}_t$ using Algorithm 2;
6 Determine threshold θ from $\Delta^{(W)}$ using Eq. (10);
7 **if** $p(y_t^*) \geq \theta$ **then**
8 Update classifier $\mathcal{H}$ with $(\mathbf{x}_t, y_t^*)$;
9 Update micro-cluster system $\mathcal{M}_F$ with $\mathbf{x}_t$;
  `// DE mechanism for delayed labeled data`
10 Retrieve delayed labeled data $\mathcal{L}_t \leftarrow \{(\mathbf{x}_i, y_i) \mid i + \delta_i = t\}$;
11 **for** $i \leftarrow 1$ **to** $|\mathcal{L}_t|$ **do**
12 Update classifier $\mathcal{H}$ with $(\mathbf{x}_i, y_i)$;
13 Compute CDI value $\Delta^{(W)}$ for $\mathbf{x}_i$ using Algorithm 2;
14 **if** $\Delta^{(W)} \leq \chi^2_{d,\alpha}$ **then**
15 Update classifier $\mathcal{H}$ with $(\mathbf{x}_i, y_i)$ again;
16 Update micro-cluster system $\mathcal{M}_O$ with $\mathbf{x}_i$;

emphasizing stable region data provides a more prudent approach, ensuring more reliable performance improvements. In this paper, we will refer to this mechanism as *Data Emphasis (DE)*, illustrated by the lower red block in Figure 5.

Algorithm 3 outlines the design of DUST, where lines 4∼9 handle temporarily unlabeled data via the IP mechanism, and lines 10∼16 process delayed labeled data via the DE mechanism.

7. Experimental Study of DUST

7.1. Experimental Setup

We conduct experiments on a total of 33 datasets, comprising 22 synthetic and 11 real-world datasets, as summarized in Table 1 and Table 2, respectively. These datasets are widely used in the context of online learning with IVL [30].

To systematically evaluate DUST across various drift scenarios, we construct 22 class-balanced synthetic datasets encompassing four types of concept drift: abrupt,

Table 1: Overview of synthetic datasets. Dataset suffix: *n*: no recurrent drift; *r*: with recurrent drift; *a*: with abrupt drift; *g*: with gradual drift. Drift Types: *Abt.*: Abrupt drift; *Grd.*: Gradual drift; *Rec.*: Recurrent drift. The last column presents the settings that influence the severity of concept drifts. For incremental drifts, parameters controlling the magnitude of change are listed. For other drift types, the sequences of concepts are provided, with the percentage differences between adjacent concepts indicated above the arrows.

Datasets	**#Data**	**#Attr.**	**#Cls.**	**Drift Types**	**Drift Settings**
$\text{Agrawal}_{na/ng}$	16,000	9	2	*Abt. / Grd.*	$f_4 \xrightarrow{50.8\%} f_3 \xrightarrow{51.2\%} f_6 \xrightarrow{59.8\%} f_8$
$\text{Agrawal}_{ra/rg}$	16,000	9	2	*Rec. & Abt. / Rec. & Grd.*	$f_3 \xrightarrow{50.7\%} f_5 \xrightarrow{50.7\%} f_3 \xrightarrow{50.7\%} f_5$
$\text{LED}_{na/ng}$	16,000	24	10	*Abt. / Grd.*	$f_1 \xrightarrow{87.5\%} f_2$
$\text{LED}_{ra/rg}$	16,000	24	10	*Rec. & Abt. / Rec. & Grd.*	$f_1 \xrightarrow{87.5\%} f_2 \xrightarrow{87.5\%} f_1 \xrightarrow{87.5\%} f_2$
$\text{Sine}_{na/ng}$	16,000	4	2	*Abt. / Grd.*	$f_1 \xrightarrow{26.8\%} f_3$
$\text{Sine}_{ra/rg}$	16,000	4	2	*Rec. & Abt. / Rec. & Grd.*	$f_1 \xrightarrow{26.8\%} f_3 \xrightarrow{26.8\%} f_1 \xrightarrow{26.8\%} f_3$
$\text{SEA}_{na/ng}$	16,000	3	2	*Abt. / Grd.*	$f_3 \xrightarrow{7.5\%} f_1 \xrightarrow{8.5\%} f_2 \xrightarrow{4.6\%} f_4$
$\text{SEA}_{ra/rg}$	16,000	3	2	*Rec. & Abt. / Rec. & Grd.*	$f_2 \xrightarrow{4.6\%} f_4 \xrightarrow{4.6\%} f_2 \xrightarrow{4.6\%} f_4$
$\text{Hyperplane}_{1/2/3}$	10,000	10	2	*Inc.*	mag_change=0.001/0.01/0.1
$\text{RBF}_{1/2/3}$	10,000	10	2	*Inc.*	change_speed=0.0001/0.001/0.01

Table 2: Overview of real-world datasets. Drift Types: *Abt.*: Abrupt drift; *Grd.*: Gradual drift; *Rec.*: Recurrent drift; *Unk.*: Unknown.

Datasets	**#Data**	**#Attr.**	**#Cls.**	**Drift Types**	**Class Ratio**
Airlines	539,383	7	2	*Unk.*	1.24 : 1
Asfault	8,564	64	6	*Unk.*	2,335 : 1,002 : 4,761 : 273 : 1 : 192
Covertype	50,000	54	7	*Unk.*	4.70 : 13.33 : 1 : 1 : 1.12 : 1 : 1
Electricity	45,312	8	2	*Unk.*	1 : 1.36
INS-Grad	24,150	33	6	*Grd.*	1 : 1 : 1 : 1 : 1 : 1
INS-Inc-Abt	79,986	33	6	*Inc. & Abt.*	1 : 1 : 1 : 1 : 1 : 1
INS-Inc-Reo	79,986	33	6	*Inc. & Rec.*	1 : 1 : 1 : 1 : 1 : 1
Pokerhand	100,000	10	10	*Unk.*	928.50 : 828.90 : 100.33 : 51.58 : 3.94 : 4.90 : 3.92 : 1
Rialto	82,250	27	10	*Unk.*	1 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : 1
UWave	4,478	315	8	*Unk.*	1 : 1 : 1 : 1 : 1 : 1 : 1 : 1
Weather	18,159	8	2	*Unk.*	1 : 2.19

gradual, recurrent, and incremental, each with varying severities. These datasets are generated using six well-known generators from `scikit-multiflow` [31]: Agrawal [32], LED, Sine [28], SEA [27], Hyperplane [33], and RBF.

Among them, Agrawal, LED, SEA, and Sine are employed to simulate abrupt, gradual, and recurrent drifts. The Agrawal streams represent a loan approval problem with nine features and ten predefined concepts f_1–f_{10}, each corresponding to a distinct decision function. The LED generator creates a digit recognition task with 24 attributes (7 relevant and 17 irrelevant), introducing drift by perturbing three relevant features; the concepts before and after drift are denoted f_1 and f_2. The SEA datasets consist of three continuous features in the range $[0, 10]$, with class labels determined by the condition $x_1 + x_2 \leq a$. The four concepts f_1–f_4 correspond to thresholds $a \in \{8, 9, 7, 9.5\}$. The

Sine streams utilize two numerical features with a decision boundary defined by a sine curve, which shifts across four versions f_1–f_4 to reflect different concepts. We denote drift types through dataset name suffixes: "r" for recurrent drift (where previous concepts reappear), "n" for non-recurrent drift (with each concept seen only once), "a" for abrupt drift (instant concept switches), and "g" for gradual drift (transitions over 2,000 points with instance sampling controlled by a sigmoid function centered at the midpoint). Gradual drift generation is also implemented using `scikit-multiflow`.

Incremental-drift datasets are generated using the Hyperplane and RBF generators. In Hyperplane, the decision boundary is defined by a 10-dimensional rotating hyperplane, where the first five weights are gradually adjusted; drift severity is controlled by the parameter `mag_change`. In RBF, 20 centroids define the data distribution, with three of them continuously moving at a speed determined by the parameter `change_speed`.

To quantify drift severity, we adopt the percentage difference metric from Soares et al. [19]:

$$\text{diff}(f_a, f_b) = \frac{1}{n} \sum_{i=1}^{n} \left| y^i_{f_a} - y^i_{f_b} \right|, \tag{11}$$

where $y^i_{f_a}$ and $y^i_{f_b}$ are the labels produced by functions f_a and f_b on n randomly sampled instances. As shown in Table 1, the values displayed above the transition arrows indicate drift severity, ranging from 4.6% to 87.5%, covering a broad spectrum of concept drift severities. For incremental drift, since changes occur continuously, pairwise percentage differences are not applicable.

To evaluate DUST in more practical scenarios, we also include 11 real-world datasets obtained from the USP-DS repository [30]. To reduce computational cost, Covertype and Pokerhand are truncated to their first 50,000 and 100,000 instances, respectively. Notably, while some of these real-world datasets exhibit class imbalance, we retain them to comprehensively evaluate the performance of DUST under diverse conditions.

Experiments are conducted on Ubuntu 20.04.4 LTS with *Python 3.9*, running on an Intel Xeon(R) Gold 6238R CPU @ 2.20GHz × 112. Unless specified otherwise, the default verification latency δ_t is set to 1,000 for all experiments. Pre-training is performed using the first 500 data points for each dataset to initialize the model.

To thoroughly evaluate the effectiveness and versatility of DUST as an online learn-

ing framework in IVL scenarios, we employ a diverse array of base models and comparison methods. Specifically, DUST is implemented using three representative supervised base models: HBP [34] (a neural network-based model), Hoeffding Tree [33] (HT, a classical tree-based model), and Adaptive Random Forest [35] (ARF, a widely-used ensemble model). For each base model, we also include its corresponding wait-until-arrival variant to serve as a direct baseline. This design helps validate DUST's compatibility with various model families.

To benchmark DUST against existing IVL-handling strategies, we compare it with approaches from different categories: 1) pseudo-labeling with rollback strategies, including SkipE-RNN [14] and SERMON [4]; 2) pseudo-labeling with correction, represented by MIPLOSC [15]; and 3) OSNN [19], a semi-supervised online learning algorithm adapted to utilize temporarily unlabeled data for semi-supervised updates while incorporating delayed labels when available. Additionally, we include a topline setting with no verification latency, serving as an empirical upper bound. In our comparisons, both DUST and the topline setting utilize the best-performing base model (selected from HBP, HT, and ARF) to ensure a fair comparisons across methods.

Parameter tuning is conducted using the first 5,000 data points via grid search. Micro-cluster settings follow the configuration in [29]. For DUST and MIPLOSC, the number of nearest neighbors k is selected from $\{5, 7, 9\}$. For SkipE-RNN and SERMON, learning rate is tuned from $\{0.1, 0.01, 0.001\}$. For OSNN, optimization steps are fixed at 10, and the number of cluster centers is chosen from $\{10, 20, 50\}$.

Accuracy serves as the primary performance metric for evaluating the overall effectiveness of the proposed DUST framework. Additional metrics, including macro-F1 and class-balanced accuracy, are provided in the supplementary material to assess performance under class imbalance. To account for variability, the average results across 10 runs are reported for performance comparisons.

7.2. Effectiveness of DUST

This section evaluates the effectiveness of DUST as an online learning framework in the IVL setting. We investigate whether DUST can consistently enhance predictive performance across various base models. Since DUST is an extension of the wait-until-

Table 3: Comparison of overall accuracy (%) between DUST and corresponding wait-until-arrival variants across datasets and base models. The better performing variant is highlighted in bold. The summary rows reports one-sided Wilcoxon signed-rank p-values, grouped by drift type. Shaded cells indicate statistically significant superiority of DUST. (Complete results can be found in the supplementary material.)

Average accuracy	**DUST|ARF**	**Wait|ARF**	**DUST|HBP**	**Wait|HBP**	**DUST|HT**	**Wait|HT**
Synthetic	**84.10**	83.48	**82.66**	80.37	**81.61**	81.02
Real-world	**63.18**	61.99	**64.33**	63.93	**59.79**	58.16
Summary	**DUST|ARF > Wait|ARF**		**DUST|HBP > Wait|HBP**		**DUST|HT > Wait|HT**	
Abt.	3.9062e-03		3.9062e-03		1.1719e-02	
Grd.	7.4219e-02		3.9062e-03		7.4219e-02	
Rec.	1.9531e-02		3.9062e-03		1.1719e-02	
Inc.	1.5625e-02		1.5625e-02		4.6875e-02	
Synthetic	2.0981e-05		2.3842e-07		3.4690e-04	
Real-world	4.8828e-04		4.1504e-02		2.6855e-02	

arrival strategy, we use the corresponding wait-until-arrival variants of each base model as baselines for a direct and fair comparison of DUST's improvements.

To rigorously assess the significance of DUST's performance gains, we apply one-sided Wilcoxon signed-rank tests at a 0.05 significance level between each pair of DUST and its baseline counterpart. The results are summarized in Table 3.

As shown in Table 3, DUST significantly outperforms its wait-until-arrival counterparts across all three base models on synthetic dataset, with performance improvements of ARF (+0.62%), HBP (+2.29%), and HT (+0.59%). These findings demonstrate the effectiveness of DUST in mitigating IVL-induced degradation and enhancing model adaptivity under varying drift severity levels. When analyzing results in terms of drift type, DUST achieves statistically significant improvements under abrupt, recurrent, and incremental drift scenarios. But significant gains in gradual drift are observed only with the HBP base model. This suggests a limitation of DUST in managing gradual drift, likely stemming from its pseudo-labeling mechanism. In gradual drift scenarios, old and new data distributions coexist and alternate over extended periods. During this transition, CDI may struggle to detect local distributional shifts, mistakenly classifying drifting regions as stable and prompting DUST to generate pseudo-labels in those areas. However, since these pseudo-labels are derived from historical labels of the fading concept, they can therefore be erroneous —- particularly when the two concepts differ substantially —- thereby misleading model updates. Tree-based models, such as HT and ARF (an ensemble method with Hoeffding trees as base learners), are

particularly sensitive to this label noise, as it disrupts the feature–class statistics essential for making splitting decisions. Because these splits are irreversible, initial errors propagate and amplify over time. In contrast, neural models like HBP update their parameters incrementally through gradient-based optimization, enabling them to better tolerate temporary label noise.

On real-world datasets, DUST remains consistent performance gains across all base models, achieving average performance improvements of +1.19%, +0.40%, and +1.63% for ARF, HBP, and HT, respectively. These improvements are statistically significant according to the Wilcoxon signed-rank tests at a significant level of 0.05, highlighting the practical utility of DUST in addressing IVL under real-world data stream conditions.

In summary, DUST demonstrates effectiveness across diverse base models and drift conditions, offering a generalizable and statistically robust solution for addressing IVL in online learning. Given ARF's strong performance across datasets, we adopt it as the default base model in subsequent experiments.

7.3. Performance Evaluation

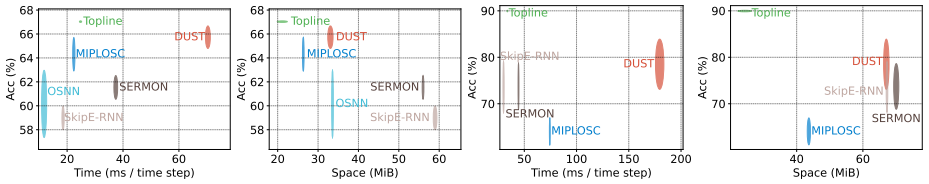
This section presents a comparative evaluation of DUST and other IVL-capable methods in terms of predictive accuracy and computational efficiency.

The results of the performance comparison are presented in Table 4. OSNN is excluded from multi-class datasets due to its applicability being limited to binary-class scenarios. As shown in the table, DUST consistently achieves the highest average accuracy across both synthetic and real-world datasets, with accuracy of 88.07% and 72.90% for binary-class datasets, and 84.10% and 63.18% across all datasets, respectively. These results exclude the topline setting, which represents an idealized scenarios without latency. Statistical significance tests using the Wilcoxon rank-sum test at the significant level of 0.05 confirms that DUST significantly outperforms competing methods across most datasets, including both synthetic and real-world benchmarks. These findings highlight the competing predictive performance of DUST compared to other IVL-handling approaches.

Regarding drift type, DUST consistently demonstrates competitive performance

Table 4: Comparison of overall accuracy (%) between DUST and other IVL-handling methods across datasets. The best-performing method (excluding the topline scenario) is highlighted in bold. Symbols ↑/↓/~ indicate whether DUST significantly outperforms, underperforms, or performs comparably to each method. Summary counts are provided in the bottom rows, grouped by drift type. As OSNN is only applicable to binary-class datasets, entries for multi-class datasets are marked with '–'. (Complete results can be found in the supplementary material.)

Average accuracy	**DUST**	**SkipE-RNN**	**SERMON**	**MIPLOSC**	**OSNN**	**Topline**
Synthetic (binary)	**88.07**	81.65	82.80	86.09	85.52	90.58
Synthetic (all)	**84.10**	74.03	75.20	82.64	-	86.86
Real-world (binary)	**72.90**	68.89	69.52	71.43	70.11	78.15
Real-world (all)	**63.18**	58.99	59.77	61.95	-	78.88
Summary	(↑/↓/~)					
Abt.	-	(6/0/2)	(6/0/2)	(6/1/1)	(4/2/0)	(0/8/0)
Grd.	-	(6/0/2)	(7/0/1)	(6/1/1)	(4/2/0)	(0/8/0)
Rec.	-	(6/0/2)	(7/0/1)	(6/1/1)	(4/2/0)	(0/8/0)
Inc.	-	(3/0/3)	(3/1/2)	(3/3/0)	(3/3/0)	(1/2/3)
Synthetic	-	(15/0/7)	(16/1/5)	(15/5/2)	(11/7/0)	(1/18/3)
Real-world	-	(11/0/0)	(11/0/0)	(9/2/0)	(3/0/0)	(0/11/0)



(a) Airlines: Time v.s. Acc (b) Airlines: Space v.s. Acc (c) Covtype: Time v.s. Acc (d) Covtype: Space v.s. Acc

Figure 6: Comparison of computational efficiency versus prediction accuracy for competing methods on two representative datasets. Due to space limitations, results for more datasets are included in the supplementary material. Panel (a) shows accuracy plotted against time (in milliseconds per time step), while panel (b) displays accuracy versus space (in MiB). The centers of the ellipses represent the mean, and the semi-axis lengths represent the standard deviations. Approaches positioned closer to the top-left corner indicate better efficiency and performance.

under abrupt, gradual, and recurrent drifts. In particular, it outperforms pseudo-labeling with rollback strategies, such as SkipE-RNN and SERMON, without significant losses in any dataset, and is surpassed by OSNN in only two cases. Under incremental drift, DUST performs comparably to state-of-the-art methods such as MIPLOSC and OSNN, exceeding their performance on about half of the datasets. Interestingly, while the performance of all methods declines on the RBF-related datasets as drift severity increases, accuracy on the Hyperplane-related datasets tends to improve with greater drift severity. This suggests that factors such as the nature of the drift and feature distribution may influence predictive performance, warranting further investigation.

Moreover, the computational efficiency of DUST is also important for real-world

applications. A detailed theoretical analysis of its time and space complexity is provided in the supplementary material. Figure 6 illustrates empirical results based on two representative datasets: Airlines and Covtype. The time complexity of DUST is expressed as $O(d^3 + \rho N d^2 + T_{\mathcal{H}})$, where N is the maximum number of micro-clusters, d is the data dimensionality, ρ is the neighborhood ratio (Algorithm 2), and $T_{\mathcal{H}}$ is the runtime of the base model per step. This time complexity indicates that, aside from the model's prediction and update costs, DUST is primarily influenced by two CDI operations: covariance aggregation (Algorithm 2, Line 5) and matrix inversion(Algorithm 2, Line 6), both of which can be more demanding than the $O(Nd)$ cost of k-NN queries as d increases. Even with acceleration techniques like KD-Tree or Ball-Tree acceleration, DUST remains one of the most computationally intensive methods on the Covtype dataset, as shown in Figure 6a and 6c.

The memory requirement is $O(Nd^2 + S_{\mathcal{H}})$, where $S_{\mathcal{H}}$ is the storage needed by the base model. Maintaining a full $d \times d$ covariance matrix for each cluster significantly contributes to this cost. In contrast, MIPLOSC requires only $O(Nd)$ for cluster storage, which is substantially lower than that of DUST, as shown in Figure 6b and 6d. Pseudo-labeling methods with rollback mechanisms, such as SkipE-RNN and SERMON, have to retain historical data and model states for rollback operations, resulting in a heavy space complexity of $O(Mh(d + o))$, where M is the number of saved model states and h and o are hidden and output dimensions. Since M typically exceeds N significantly, and h is comparable to d, DUST exhibits superior overall space efficiency compared to these pseudo-labeling methods, as shown in Figure 6b and 6d.

Overall, despite its relatively high computational overhead, DUST offers substantial improvements in predictive performance, making it a compelling candidate for addressing IVL challenges.

7.4. Ablation Study

To evaluate the contributions of each component in DUST, we conduct an ablation study involving several degraded variants. To investigate the role of stable region identification in utilizing temporarily unlabeled data and delayed labeled data, we remove the two key mechanisms that depend on it: Immediate Pseudo-labeling (IP) and Data

Table 5: Ablation study of DUST in overall accuracy (%) across datasets. The best-performing variant is highlighted in bold. The summary rows report one-sided Wilcoxon signed-rank test p-values, grouped by drift type. Shaded cells denote cases where DUST is statistically superior to the corresponding variant. (Complete results can be found in the supplementary material.)

Average accuracy	**DUST**	**DUST w/o IP**	**DUST w/o DE**	**DUST w_{500}**	**DUST $w_{1,000}$**	**DUST $w_{2,000}$**	**Wait**
Synthetic	**84.10**	83.65	83.79	83.67	83.82	83.91	83.48
Real-world	**63.18**	62.03	62.95	62.42	62.61	62.55	61.99
Summary: DUST > Other		**DUST w/o IP**	**DUST w/o DE**	**DUST w_{500}**	**DUST $w_{1,000}$**	**DUST $w_{2,000}$**	**Wait**
Abt.		7.8125e-03	3.9062e-03	3.9062e-03	3.9062e-03	1.9531e-02	3.9062e-03
Grd.		3.9062e-03	2.7344e-02	3.9062e-02	5.4688e-02	5.4688e-02	7.4219e-02
Rec.		3.9062e-03	1.9531e-02	1.1719e-02	1.1719e-02	2.7344e-02	1.9531e-02
Inc.		1.5625e-02	1.5625e-02	1.5625e-02	1.5625e-02	1.5625e-02	1.5625e-02
Synthetic		4.7684e-07	1.0252e-05	4.5300e-06	7.8678e-06	3.2663e-05	2.0981e-05
Real-world		4.8828e-04	2.4414e-03	4.8828e-04	4.8828e-04	4.8828e-03	4.8828e-04

Emphasis (DE). The variants investigated include DUST without (w/o) IP, DUST w/o DE, and a version that excludes both mechanisms, referred to as "Wait", which corresponds to the wait-until-arrival strategy, where model updates occur only when delayed labels are available. Since both IP and DE rely entirely on stable region identification through CDI, disabling these mechanisms eliminates the influence of CDI.

To further assess the micro-cluster system, we replace it with fixed-size sliding windows of lengths 500, 1,000, and 2,000. In this configuration, the CDI mechanism is adapted to operate on individual data points within the window rather than on micro-clusters. These variants are denoted as DUST w_{500}, DUST $w_{1,000}$, and DUST $w_{2,000}$, respectively. All variants are evaluated under identical conditions, using the same tuned hyperparameters and ARF as the base model to ensure a fair comparison.

The experimental results are presented in Table 5. One-sided Wilcoxon signed-rank tests at a significance level of 0.05 are applied to determine whether DUST significantly outperforms each ablated variant. On synthetic datasets, DUST exhibits statistically significant performance improvements over the variants, maintaining this advantage across various drift severities. When grouped by drift type, DUST demonstrates statistically significant benefits under abrupt, gradual, and recurrent drift. However, in the case of gradual drift, while DUST outperforms variants such as DUST $w_{1,000}$, DUST $w_{2,000}$, and Wait on most datasets, the corresponding p-values (0.05469, 0.05469, and 0.0742) do not meet the 0.05 significance threshold. This observation aligns with findings in Section 7.2 and is likely related to the interaction between the k-NN-based

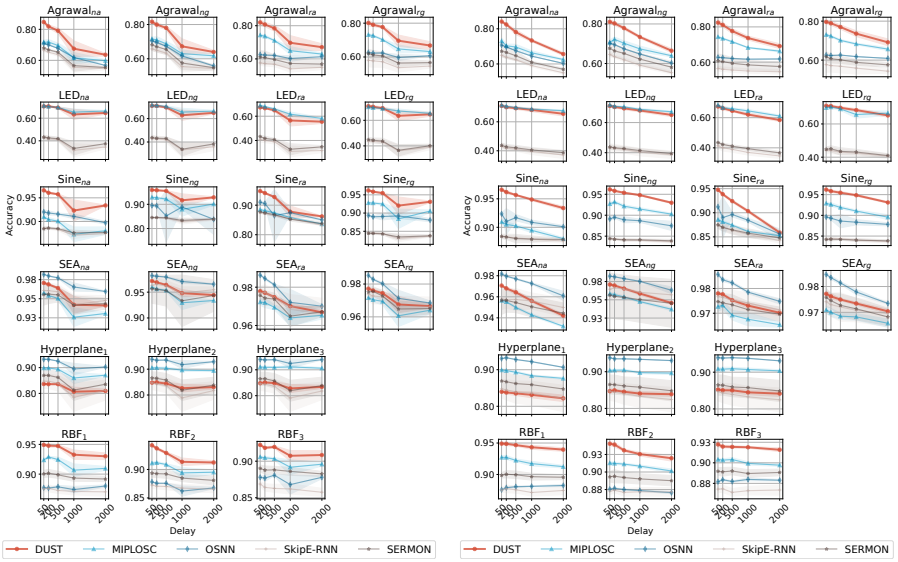
pseudo-labeling and the characteristics of Hoeffding Trees in ARF under gradual drift. On real-world datasets, DUST again significantly outperforms all variants.

Overall, results from the DUST w/o IP and DUST w/o DE variants show that eliminating either mechanism leads to substantial performance degradation, highlighting the crucial roles of IP in leveraging temporarily unlabeled data and DE in utilizing delayed labeled data. Focusing on the sliding window variants (DUST w_{500}, DUST $w_{1,000}$, and DUST $w_{2,000}$), we observe a consistent trend across both synthetic and real-world datasets: larger windows generally yield better accuracy. This is expected, as incorporating more historical data improves CDI's stability. Nevertheless, DUST with micro-clusters outperforms even the largest window variant, demonstrating the superior efficiency and representational power of the micro-cluster systems.

7.5. Performance Under Different Latency Conditions

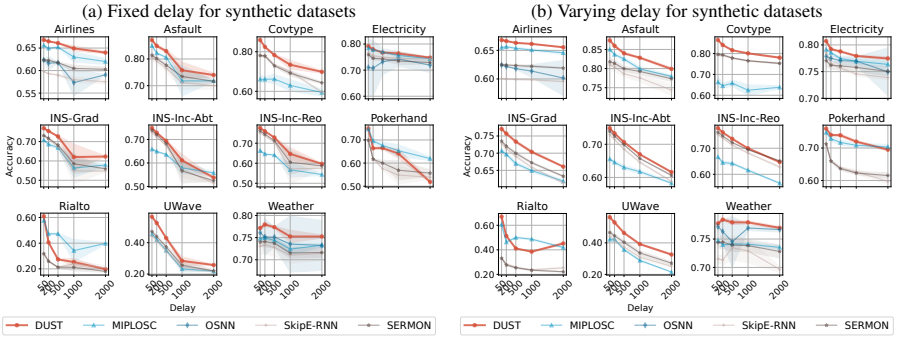
Ensuring robust performance across different latency conditions is essential for effectively addressing IVL. This section investigates DUST's performance under various delay levels and types, assessing how performance evolves with increasing label delays. We test five mean delay levels: $\bar{\delta} \in \{50, 200, 500, 1000, 2000\}$, considering both fixed and varying delay scenarios. For varying delays, we simulate real-world conditions using a Gamma distribution with shape parameter 2 and scale parameter $\bar{\delta}/2$, i.e., $\delta_t \sim \Gamma(2, \bar{\delta}/2)$, where the probability of receiving delayed labels peaks shortly after the event and then declines over time. In the fixed case, delays are set to $\delta_t = \bar{\delta}$. We then compare DUST's performance against other methods under these conditions.

Figure 7 illustrates the accuracy trends across different values of $\bar{\delta}$ for each dataset, capturing how the predictive performance of various methods is affected by increasing label delay. Overall, both fixed and varying delay settings exhibit a consistent pattern: as $\bar{\delta}$ increases, all methods tend to suffer noticeable performance degradation. In some cases, such as INS-Inc-Abt and INS-Inc-Reo, methods like DUST, SkipE-RNN, and SERMON show performance drops of nearly 20 percentage points. This degradation is consistent with intuitive expectations, as longer delays introduce more outdated information, hindering timely adaptation. Despite this, DUST consistently outperforms competing methods across most datasets and delay levels, highlighting its robustness



(a) Fixed delay for synthetic datasets

(b) Varying delay for synthetic datasets



(c) Fixed delay for real-world datasets

(d) Varying delay for real-world datasets

Figure 7: Accuracy trends across various latency values for both fixed and varying delay scenarios. The lighter background areas represent standard deviations.

and effectiveness in adapting to varying degrees of IVL.

8. Discussion

8.1. Runtime Bottleneck of DUST

DUST incurs a relatively high time consumption. As discussed in Section 7.3, its runtime complexity is $O(d^3 + \rho N d^2 + T_{\mathcal{H}})$, where d is the feature dimension, ρ is the neighborhood ratio, N is the maximum number of maintained micro-clusters, and $T_{\mathcal{H}}$ is the cost associated with classifier inference and update. Within DUST, the primary

source of overhead arises from the CDI computation, particularly from two operations: *covariance aggregation* at $O(\rho N d^2)$ and *matrix inversion* at $O(d^3)$.

In contrast, k-NN queries require $O(Nd)$, and with acceleration techniques such as KD-Tree [36] and Ball-Tree [37], their time cost is generally much lower than that of the other two operations. When $\rho d > 1$, the cost of covariance aggregation exceeds that of k-NN queries. To mitigate this, a promising approach is to adopt a parallel, hierarchical merging strategy that reduces the cost to $O(\log(\rho N)d^2)$, albeit at the expense of increased space complexity.

As dimensionality increases, particularly when $d > \rho N$, matrix inversion becomes the dominant bottleneck. Caching and reusing the inverted matrix is impractical due to the continuously evolving nature of the data. Although low-rank approximations and iterative solvers exist, their effectiveness relies on structural assumptions that are difficult to ensure in our context. Therefore, practitioners should exercise caution when opting for our approach (or any existing method), as more specified adaptions will be necessary to effectively address high-dimensional IVL scenarios.

8.2. The k-NN Implementation within DUST

The k-NN algorithm also encounters challenges in high-dimensional spaces. As dimensionality grows, both its computational cost and retrieval quality deteriorate due to the curse of dimensionality, which can adversely affect CDI computation and pseudo-labeling accuracy. To address this challenge, we could explore two complementary approach for future work.

First, we can explore approximate nearest-neighbor techniques, such as Hierarchical Navigable Small World (HNSW) [38] and Locality-Sensitive Hashing (LSH) [39]. These methods remain effective in high-dimensional settings, offer sub-linear query time, and support dynamic index updates, making them well suitable for our evolving micro-cluster scenario.

Second, we can incorporate online dimensionality reduction techniques. For instance, streaming autoencoders such as DEVDAN [40] can be employed to embed incoming data into a lower-dimensional space. This approach can help reduce time consumption, lower memory usage, and mitigate performance degradation caused by

high dimensionality.

These improvements will be explored in future work. Meanwhile, for real-time applications with $d > 100$, we recommend considering other lightweight methods when low latency is essential, while further optimizing DUST's k-NN component to mitigate the curse of dimensionality. In lower-dimensional settings, DUST remains a robust solution for addressing the IVL problem.

9. Conclusion

This paper addresses the challenge of Intermediate Verification Latency (IVL), which often occurs in real-world applications when delayed labeled streaming data necessitates model updates to accommodate concept drift. We introduced a novel online learning approach to enhance the identification and utilization of stable region data.

Specifically, we proposed Centroid-based Drift Index (CDI), an unsupervised test statistic designed to identify stable regions and assess data points residing within them. The CDI method serves as a critical tool for investigating drifting concepts and revealing change patterns from a regional perspective. Building on these stable regions, we developed the Data Utilization informed by STable regions (DUST) framework, which offers a more granular approach than existing IVL methods for utilizing temporarily unlabeled and delayed labeled data.

Systematic experiments on synthetic and real-world datasets demonstrate the efficacy of CDI and DUST. Our results validate the design principle of CDI and indicate its capability to reliably identify stable regions. Moreover, DUST consistently enhances the performance of various base classifiers under IVL conditions. It also outperforms state-of-the-art IVL methods and demonstrates robustness across diverse IVL scenarios. Ablation studies further confirm its effective utilization of both temporarily unlabeled and delayed labeled data.

Despite its advantage, the computational cost of our DUSTis relatively high due to covariance aggregation and matrix inversion operations required for CDI in DUST. Additionally, DUST may struggle in high-dimensional settings due to its cubic space complexity. While alternative solutions may be preferable in such cases, the superior

predictive performance of DUST makes it a strong candidate for addressing IVL.

The applicability of CDI extends beyond IVL, offering potential contributions to other online learning contexts related to concept drift, which presents an avenue for further research. Future work should focus on reducing the computational burden of CDI and improving its scalability to high-dimensional data, which entails systematically investigating performance optimization strategies for DUST and carefully balancing accuracy with computational cost.

Declaration of Competing Interests

The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 62572154 and 62002148, the Heilongjiang Province Natural Science Foundation under Grant No. LH2024F016, the State Key Laboratory of Robotics under Grant No. 2023-O11, and the Harbin Institute of Technology Talent Start-up Project under Grant No. AUGA5710010924.

References

[1] E. Yu, J. Lu, B. Zhang, G. Zhang, Online boosting adaptive learning under concept drift for multistream classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2024.

[2] T. T. T. Nguyen, T. T. Nguyen, A. W.-C. Liew, S.-L. Wang, Variational inference based bayes online classifiers with concept drift adaptation, Pattern Recognition 81 (2018) 280–293.

[3] T. Buyuktanir, M. S. Aktas, A deep learning-based prefetching approach to enable scalability for data-intensive applications, in: IEEE International Conference on Big Data, 2022, pp. 2716–2721.

[4] M. Das, M. Pratama, T. Tjahjowidodo, A self-evolving mutually-operative recurrent network-based model for online tool condition monitoring in delay scenario, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2775–2783.

[5] V. Souza, T. Pinho, G. Batista, Evaluating stream classifiers with delayed labels information, in: Proceedings of 7th Brazilian Conference on Intelligent Systems, 2018, pp. 408–413.

[6] H. Yu, W. Liu, J. Lu, Y. Wen, X. Luo, G. Zhang, Detecting group concept drift from multiple data streams, Pattern Recognition 134 (2023) 1–11.

[7] Y. Guo, J. Pu, J. He, B. Jiao, J. Ji, S. Yang, Adaptive stochastic configuration network based on online active learning for evolving data streams, Information Sciences 711 (2025) 122113.

[8] A. Tsymbal, M. Pechenizkiy, P. Cunningham, S. Puuronen, Dynamic integration of classifiers for handling concept drift, Information Fusion 9 (1) (2008) 56–68.

[9] A. Liu, Y. Song, G. Zhang, J. Lu, Regional concept drift detection and density synchronized drift adaptation, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 2280–2286.

[10] A. Liu, J. Lu, F. Liu, G. Zhang, Accumulating regional density dissimilarity for concept drift detection in data streams, Pattern Recognition 76 (2018) 256–272.

[11] M. Masud, J. Gao, L. Khan, J. Han, B. M. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE Transactions on Knowledge and Data Engineering 23 (6) (2011) 859–874.

[12] A. L. Cristiani, T. P. da Silva, H. de Arruda, Camargo, A fuzzy approach for classification and novelty detection in data streams under intermediate latency,

in: Proceedings of 9th Brazilian Conference on Intelligent Systems, 2020, pp. 171–186.

[13] A. L. Cristiani, H. de Arruda Camargo, A fuzzy multi-class novelty detector for data streams under intermediate latency, in: Proceedings of IEEE International Conference on Fuzzy Systems, 2021, pp. 1–6.

[14] M. Das, M. Pratama, J. Zhang, Y. Ong, A skip-connected evolving recurrent neural network for data stream classification under label latency scenario, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3717–3724.

[15] Z. Zhong, L. Song, F. Tang, B. Yuan, Addressing intermediate verification latency in online learning through immediate pseudo-labeling and oriented synthetic correction, in: Proceedings of International Joint Conference on Neural Networks, 2024, pp. 1–10.

[16] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Transactions on Knowledge and Data Engineering 31 (12) (2019) 2346–2363.

[17] H. M. Gomes, M. Grzenda, R. Mello, J. Read, M. H. Le Nguyen, A. Bifet, A survey on semi-supervised learning for delayed partially labelled data streams, ACM Computing Surveys 55 (4) (2022) 1–42.

[18] J. a. Gama, R. Sebastião, P. P. Rodrigues, Issues in evaluation of stream learning algorithms, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 329–338.

[19] R. G. F. Soares, L. L. Minku, Osnn: An online semisupervised neural network for nonstationary data streams, IEEE Transactions on Neural Networks and Learning Systems 34 (9) (2023) 6029–6041.

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

[21] N. Samadi, J. Tanha, M. Jalili, Graph theory-based semi-supervised self-training for data stream classification and emerging class detection, Information Sciences 698 (2025) 121762.

[22] F. Hinder, V. Vaquet, J. Brinkrolf, B. Hammer, Model-based explanations of concept drift, Neurocomputing 555 (2023) 1–15.

[23] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artificial Intelligence 209 (2014) 11–28.

[24] N. Lu, J. Lu, G. Zhang, R. Lopez de Mantaras, A concept drift-tolerant case-base editing technique, Artificial Intelligence 230 (2016) 108–133.

[25] F. Hinder, V. Vaquet, J. Brinkrolf, A. Artelt, B. Hammer, Localization of concept drift: Identifying the drifting datapoints, in: Proceedings of International Joint Conference on Neural Networks, 2022, pp. 1–9.

[26] P. Porwik, K. Wrobel, T. Orczyk, R. Doroz, Fbdd: feature-based drift detector for batch processing data, Cluster Computing 27 (5) (2024) 6805–6822.

[27] W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 377–382.

[28] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: A. L. C. Bazzan, S. Labidi (Eds.), Advances in Artificial Intelligence, 2004, pp. 286–295.

[29] S. Ud Din, J. Shao, J. Kumar, W. Ali, J. Liu, Y. Ye, Online reliable semi-supervised learning on evolving data streams, Information Sciences 525 (2020) 153–171.

[30] V. M. A. Souza, D. M. dos Reis, A. G. Maletzke, G. E. A. P. A. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Mining and Knowledge Discovery 34 (6) (2020) 1805–1858.

[31] J. Montiel, J. Read, A. Bifet, T. Abdessalem, Scikit-multiflow: A multi-output streaming framework, Journal of Machine Learning Research 19 (72) (2018) 1–5.
URL https://scikit-multiflow.github.io/

[32] R. Agrawal, T. Imielinski, A. Swami, Database mining: a performance perspective, IEEE Transactions on Knowledge and Data Engineering 5 (6) (1993) 914–925.

[33] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 97–106.

[34] D. Sahoo, Q. Pham, J. Lu, S. C. H. Hoi, Online deep learning: learning deep neural networks on the fly, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 2660–2666.

[35] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Machine Learning 106 (9–10) (2017) 1469–1495.

[36] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[37] S. M. Omohundro, Five balltree construction algorithms (1989).

[38] Y. A. Malkov, D. A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (4) (2020) 824–836.

[39] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, 1998, p. 604–613.

[40] A. Ashfahani, M. Pratama, E. Lughofer, Y.-S. Ong, Devdan: Deep evolving denoising autoencoder, Neurocomputing 390 (2020) 297–314.