Enhancing Model Adaptation with Stable Region Data to Address Intermediate Verification Latency: Supplementary Material

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

[a]*Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, 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

This supplementary material complements the paper entitled "Enhancing Model Adaptation with Stable Region Data to Address Intermediate Verification Latency" submitted to *Pattern Recognition*, which cannot be included in the main paper due to page limitations.

Section 1 presents detailed descriptions of the datasets used in this study. Section 2 provides a thorough analysis and comprehensive results of the computational efficiency experiments. Sections 3, 4, and 5 present complete experimental findings. Section 6 explores DUST's performance under varying degrees of label availability, including scenarios with partially missing ground truth. Finally, Section 7 provides a case study on Just-In-Time Software Defect Prediction, demonstrating the practical applicability of DUST in real-world contexts.

1. Datasets

We conduct experiments on a total of 33 datasets—22 synthetic and 11 real-world—as summarized in Table 1 and Table 2, respectively. These benchmarks are widely adopted in online learning with intermediate verification latency (IVL) studies [1].

1.1. Synthetic datasets

The 22 synthetic data streams span four concept-drift patterns—abrupt, gradual, recurrent, and incremental—with varying severity levels (see Table 1). All streams are generated using the six built-in generators of the `scikit-multiflow` framework [2]:

- *Agrawal.* This generator simulates a binary classification problem related to loan approval, using nine numerical features. It includes ten distinct concepts, denoted as f_1–f_{10}, as defined in [3], each corresponding to a different logical function that determines whether a loan should be approved.
- *LED.* This generator models digit recognition on a seven-segment LED display using 24 attributes—seven relevant and seventeen irrelevant. Concept drift is introduced by randomly perturbing three of the relevant features. We use f_1 and f_2 to represent the pre-drift and post-drift concepts, respectively.
- *SEA.* This generator produces instances with three continuous features, each uniformly distributed in the range [0, 10]. The class label is assigned based on whether the inequality $x_1 + x_2 \leq a$ holds. Concept drift is simulated by varying the threshold value a. Specifically, for concepts f_1–f_4, the values of a are 8, 9, 7, and 9.5, respectively.

*Corresponding author. Email address: songly@hit.edu.cn (L. Song).

- *Sine.* This generator creates data with two numerical features. The class label is determined by whether a point lies above or below a sine wave. Concept drift is introduced by modifying the sine function across concepts f_1 to f_4, following the approach described in [4].
- *Hyperplane.* This generator produces data from a rotating hyperplane in a ten-dimensional space, where each feature value is drawn uniformly from the range [0, 1]. The decision boundary is defined by the equation $\sum_{i=1}^{10} w_i x_i = w_0$, where $w_0 = \sum_{i=1}^{10} w_i$ and the weights w_i are randomly initialized. Incremental drift is simulated by gradually changing the weights of the first five features over time. The degree of change is controlled by the parameter `mag_change`.
- *RBF.* The Radial Basis Function (RBF) generator creates instances based on 20 randomly distributed centroids, each associated with a position, standard deviation, weight, and class label. Incremental concept drift is introduced by continuously shifting the positions of three randomly selected centroids at a fixed speed, governed by the parameter `change_speed`.

Drift patterns. The type of concept drift in each dataset is indicated by a suffix in its name:

- `*r`: recurrent drift—at least one revisit of a previous concept.
- `*n`: non-recurrent—each concept appears exactly once.
- `*a`: abrupt drift—instantaneous switch between two concepts.
- `*g`: gradual drift—a smooth transition between two concepts over 2,000 instances, where the probability of sampling from the new concept increases following a sigmoid function centered at the midpoint of the transition window. This implementation follows the design provided by the `scikit-multiflow` framework [2].

Severity quantification. We adopt the percentage-difference metric from Chiu et al.[5] and Soares et al.[6]:

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

where $y^i_{f_a}$ and $y^i_{f_b}$ are the labels assigned by functions f_a and f_b, respectively, on n uniformly sampled points. Values above 50% denote severe drift; otherwise it is mild. And for incremental drift datasets, since the drift occurs continuously, pairwise percentage differences are not applicable.

1.2. Real-world datasets

The 11 real-world streams are drawn from the USP-DS repository [1]. To limit computational expense, we truncate Covertype to its first 50,000 instances and Poker-Hand to 100,000 instances.

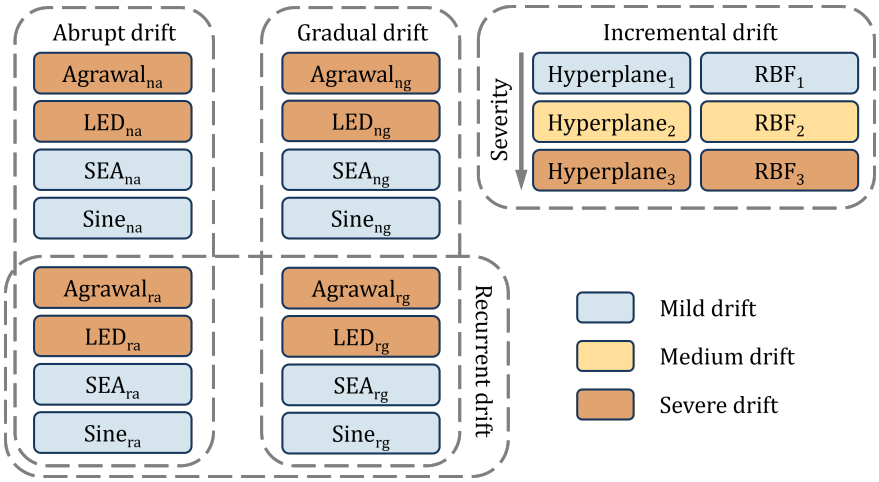


Figure 1: Illustration of synthetic datasets.

Table 1: Overview of synthetic datasets. Suffix of datasets: *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, the parameters controlling the changing magnitude 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**
Agrawal$_{na/ng}$	16,000	9	2	*Abt. / Grd.*	$f_4 \xrightarrow{50.8\%} f_3 \xrightarrow{51.2\%} f_6 \xrightarrow{59.8\%} f_8$
Agrawal$_{ra/rg}$	16,000	9	2	*Rec. & Abt. / Rec. & Grd.*	$f_3 \xrightarrow{50.7\%} f_5 \xrightarrow{50.7\%} f_3 \xrightarrow{50.7\%} f_5$
LED$_{na/ng}$	16,000	24	10	*Abt. / Grd.*	$f_1 \xrightarrow{87.5\%} f_2$
LED$_{ra/rg}$	16,000	24	10	*Rec. & Abt. / Rec. & Grd.*	$f_1 \xrightarrow{87.5\%} f_2 \xrightarrow{87.5\%} f_1 \xrightarrow{87.5\%} f_2$
Sine$_{na/ng}$	16,000	4	2	*Abt. / Grd.*	$f_1 \xrightarrow{26.8\%} f_3$
Sine$_{ra/rg}$	16,000	4	2	*Rec. & Abt. / Rec. & Grd.*	$f_1 \xrightarrow{26.8\%} f_3 \xrightarrow{26.8\%} f_1 \xrightarrow{26.8\%} f_3$
SEA$_{na/ng}$	16,000	3	2	*Abt. / Grd.*	$f_3 \xrightarrow{7.5\%} f_1 \xrightarrow{8.5\%} f_2 \xrightarrow{4.6\%} f_4$
SEA$_{ra/rg}$	16,000	3	2	*Rec. & Abt. / Rec. & Grd.*	$f_2 \xrightarrow{4.6\%} f_4 \xrightarrow{4.6\%} f_2 \xrightarrow{4.6\%} f_4$
Hyperplane$_{1/2/3}$	10,000	10	2	*Inc.*	mag_change=0.001/0.01/0.1
RBF$_{1/2/3}$	10,000	10	2	*Inc.*	change_speed=0.0001/0.001/0.01

2. Computation Efficiency Analysis and Results

In this section, we provide a detailed, per-time-step analysis of both time and space complexity on DUST. For the reader's convenience, the complete algorithm is provided here in Algorithm 1 and Algorithm 2.

For clarity, we introduce the following notation:

- d: dimensionality of each data point;
- N: max number of micro-clusters in $\mathcal{M}_O$ or $\mathcal{M}_F$ (1,000 in DUST);
- ρ: neighborhood ratio for CDI calculation;
- k: number of nearest neighbors for pseudo-labeling, $k \ll N$;
- $|\mathcal{L}_t|$: number of delayed labels arriving at time step t, averaging at most one per step;
- $T_{\mathcal{H}}$: time cost of a single classifier prediction or update (depends on the base model).
- $S_{\mathcal{H}}$: space cost of a single classifier prediction or update (depends on the base model).

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
Airlines	539,383	7	2	*Unk.*
Asfault	8,564	64	6	*Unk.*
Covertype	50,000	54	7	*Unk.*
Electricity	45,312	8	2	*Unk.*
INS-Grad	24,150	33	6	*Grd.*
INS-Inc-Abt	79,986	33	6	*Inc. & Abt.*
INS-Inc-Reo	79,986	33	6	*Inc. & Rec.*
Pokerhand	100,000	10	10	*Unk.*
Rialto	82,250	27	10	*Unk.*
UWave	4,478	315	8	*Unk.*
Weather	18,159	8	2	*Unk.*

Algorithm 1: Local Region's CDI of Target Data based on 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, Δ_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 $\mathcal{M}_O^{(W)} \cup \mathcal{M}_F^{(W)}$;

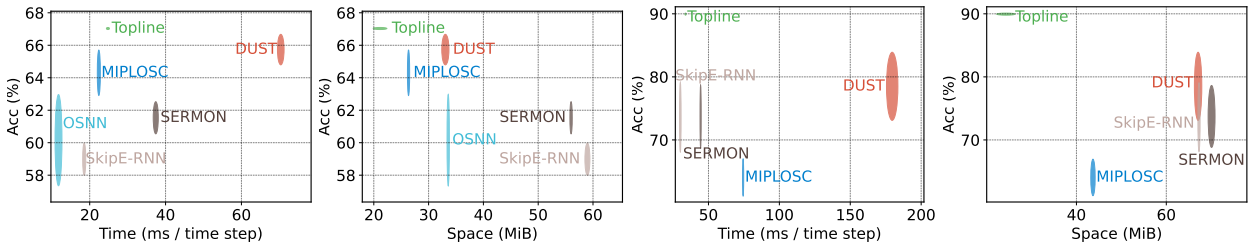
`// Calculate CDI`

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

Algorithm 2: 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 $\widehat{\mathbf{Y}} = \{\hat{y}_1, \hat{y}_2, \ldots\}$

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



(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 2: 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.

Time complexity. In this analysis we assume a naive k-NN implementation. According to Algorithm 2, at each time step the following operations occur:

- **Classifier inference** (Line 3 of Algorithm 2): $O(T_{\mathcal{H}})$.
- **k-NN pseudo-labeling** (Line 4 of Algorithm 2): $O(Nd)$.
- **CDI computations** (Lines 5 and 13 of Algorithm 2), performed $1 + |\mathcal{L}_t|$ times. Each evaluation (Algorithm 1) comprises:
  - Nearest-neighbor search in $\mathcal{M}_O$ and recording the maximum distance (Line 1 of Algorithm 1): $O(Nd)$.
  - Radius-based retrieval from $\mathcal{M}_O$ and $\mathcal{M}_F$ (Line 2 of Algorithm 1): scan cost $O(Nd)$ (the expected number of clusters retrieved is k, to be used below).
  - Meta-data aggregation (Lines 3–5 of Algorithm 1): $O(2\rho Nd^2 + 2\rho Nd + 2\rho N) = O(\rho Nd^2)$.
  - Final CDI score via Eq. (2) (Line 6 of Algorithm 1), dominated by a $d \times d$ matrix inversion: $O(d^3)$.

  Summing these costs yields
  $$O(Nd + Nd + \rho Nd^2 + d^3) = O(d^3 + \rho Nd^2).$$
  Hence CDI runs in
  $$O((1 + |\mathcal{L}_t|)(d^3 + \rho Nd^2)).$$
- **Classifier updates** (Lines 8 and 15 of Algorithm 2), up to $1 + |\mathcal{L}_t|$ times: $O((1 + |\mathcal{L}_t|)\, T_{\mathcal{H}})$.
- **Micro-cluster updates** (Lines 9 and 16 of Algorithm 2), up to $1 + |\mathcal{L}_t|$ times. Each round of update involves identifying the micro-cluster to which the new data point belongs, with a cost of $O(Nd)$, followed by updating the micro-cluster's centroid and covariance matrix in $O(d^2)$. Thus, the total cost is $O((1 + |\mathcal{L}_t|)(d^2 + Nd))$.

Therefore, the total time per step is

$$\begin{aligned} &O(T_{\mathcal{H}}) + O(Nd) + O((1 + |\mathcal{L}_t|)(d^3 + \rho Nd^2)) + O((1 + |\mathcal{L}_t|)\, T_{\mathcal{H}}) + O((1 + |\mathcal{L}_t|)(d^2 + Nd)) \\ =&O(T_{\mathcal{H}} + Nd + (1 + |\mathcal{L}_t|)(d^3 + \rho Nd^2 + T_{\mathcal{H}} + d^2 + Nd)). \end{aligned}$$

Since $|\mathcal{L}_t| \leq 1$, we simplify to

$$\boxed{O(d^3 + \rho Nd^2 + T_{\mathcal{H}}).}$$

Our analysis indicates that, aside from the prediction and update costs of the base classifier, the overall runtime of DUST is primarily dominated by two components within the CDI computation: the covariance aggregation step (Algorithm 1, Line 5) and the matrix inversion step (Algorithm1, Line 6). Notably, the computational cost of these two CDI components could outweigh that of k-NN operations, which only require $O(Nd)$ time. This explains why

DUST, despite leveraging KD-Tree and Ball-Tree structures for nearest-neighbor acceleration, remains among the most computationally intensive methods (see Figure 2a and 2c).

A promising direction for improving efficiency is to accelerate the covariance aggregation step by adopting a parallelized, bottom-up hierarchical merging scheme. Such an approach could potentially reduce the cost from $O(\rho N d^2)$ to $O(\log(\rho N) d^2)$ by aggregating local covariances in a tree-like fashion. As for the matrix inversion step, viable acceleration techniques are more limited. While approximate matrix inversion methods (e.g., using low-rank approximations or iterative solvers like Conjugate Gradient) exist, they typically require strong assumptions about the structure or sparsity of the covariance matrix. Additionally, caching or reusing inverted matrices is infeasible due to the dynamic nature of streaming data and evolving cluster statistics.

While these optimizations lie beyond our current scope, they provide critical pathways for enhancing DUST's efficiency in future work.

Space complexity. The main sources of memory consumption in DUST are the storage of micro-clusters and the base model:

- **Micro-cluster storage:** Each micro-cluster in the two micro-cluster systems maintains the following:
  - the number of data points it contains ($O(1)$);
  - the centroid ($O(d)$);
  - the local covariance matrix ($O(d^2)$);
  - auxiliary meta-data for updates, including the linear sum ($O(d)$), squared sum ($O(d)$), class label ($O(1)$), weight ($O(1)$), and timestamp ($O(1)$).

  Among these components, the storage required for the covariance matrix dominates all other terms in asymptotic complexity. Therefore, the space complexity per micro-cluster is $O(d^2)$, and for $2N$ clusters in total, it becomes $O(2Nd^2) = O(Nd^2)$.
- **Base model storage:** $O(S_{\mathcal{H}})$.

Hence, the overall space complexity of DUST is:

$$\boxed{O(Nd^2 + S_{\mathcal{H}}).}$$

From the space complexity analysis, it is evident that maintaining full covariance matrices within each micro-cluster considerably increases memory consumption. This accounts for the significantly higher storage cost of DUST compared to MIPLOSC, which only requires $O(Nd)$ for storing its micro-cluster system, as illustrated in Figure 2b and 2d. In contrast, pseudo-labeling methods with rollback mechanisms, SkipE-RNN and SERMON, must retain historical data and model states to enable rollback operations, resulting in a space complexity of $O(Mh(d + o))$, where M denotes the number of stored states, typically much larger than N, and the hidden layer size h is often comparable to the input dimension d. Therefore, despite its higher per-cluster memory overhead, DUST exhibits better overall space efficiency than these rollback-based methods, as shown in Figure 2b and Figure 2d, particularly on the Covtype dataset, which has 54 input dimensions—a level already considered relatively high. However, as the input dimensionality further increases, the memory cost of DUST may eventually exceed that of SERMON and SkipE-RNN.

The full results on computational efficiency for real-world datasets are provided in Table 3, which were omitted from the main paper due to space constraints.

Table 3: Comparison of overall computational efficiency between DUST and other IVL-handling methods across datasets. As OSNN is only applicable to binary-class datasets, entries for multi-class datasets are marked with '–'.

(a) Time (ms / time step)

Datasets	**DUST**	**SkipE-RNN**	**SERMON**	**MIPLOSC**	**OSNN**	**Topline**
Airlines	70.37±0.87	18.55±0.09	37.41±0.67	22.41±0.05	11.78±0.91	24.80±0.32
Asfault	225.08±17.45	26.73±0.53	35.27±1.72	123.81±1.73	-	53.84±2.89
Covtype	179.57±4.03	30.14±0.56	44.46±0.41	74.43±0.42	-	33.82±0.31
Electricity	71.66±0.77	18.95±0.04	32.75±0.91	24.53±0.20	11.14±0.68	24.85±0.56
INS-Grad	145.21±2.32	28.58±0.32	45.03±2.00	81.41±0.16	-	47.17±0.72
INS-Inc-Abt	130.80±1.04	29.48±0.37	48.86±0.97	77.94±0.87	-	46.98±0.72
INS-Inc-Reo	131.91±1.45	28.95±0.53	45.26±1.16	76.73±0.78	-	45.84±0.64
Pokerhand	66.57±0.80	18.67±0.21	39.76±1.27	30.13±0.27	-	22.97±0.60
Rialto	117.05±2.09	28.68±0.15	67.90±0.46	126.93±1.08	-	52.97±0.43
UWave	353.69±8.19	32.93±2.52	37.87±2.46	492.15±1.92	-	184.84±3.41
Weather	75.96±1.04	17.95±0.12	32.06±0.86	22.12±0.04	10.20±0.21	29.18±0.84
JIT-SDP_django	102.94±0.81	26.35±0.23	35.20±1.76	35.93±0.36	13.92±2.37	30.60±1.49
JIT-SDP_pandas	110.85±1.42	26.76±0.19	37.94±0.32	39.69±0.40	12.77±3.13	28.73±0.32
Average (binary)	86.36	21.71	35.07	28.94	11.96	27.63
Average (all)	137.05	25.59	41.52	94.48	-	48.20

(b) Space (MiB)

Datasets	**DUST**	**SkipE-RNN**	**SERMON**	**MIPLOSC**	**OSNN**	**Topline**
Airlines	33.06±0.65	58.98±0.45	56.00±0.21	26.34±0.15	33.58±0.22	21.14±1.25
Asfault	124.08±1.03	74.08±0.22	64.24±0.24	51.30±0.16	-	19.44±1.48
Covtype	67.08±0.82	67.28±0.25	70.08±0.77	43.68±0.49	-	24.30±2.05
Electricity	45.88±0.49	89.66±0.42	85.52±0.39	41.10±0.51	42.74±0.21	26.34±0.32
INS-Grad	65.80±1.16	82.64±0.15	78.46±0.26	43.40±0.48	-	29.94±1.05
INS-Inc-Abt	124.00±0.89	102.54±0.30	96.32±0.28	88.18±0.59	-	41.52±2.13
INS-Inc-Reo	105.62±1.36	102.60±0.37	97.02±0.19	65.28±0.41	-	47.82±3.35
Pokerhand	52.22±0.41	80.92±0.56	76.78±0.59	52.86±0.23	-	18.10±0.34
Rialto	74.84±0.21	103.00±0.57	98.98±0.22	71.90±0.41	-	33.08±0.30
UWave	1036.42±0.34	264.20±1.59	253.12±1.09	21.92±0.04	-	11.52±0.47
Weather	39.16±1.04	73.22±0.31	69.92±0.11	22.02±0.18	38.00±0.14	24.28±1.00
JIT-SDP_django	38.92±0.65	60.76±0.18	58.74±0.05	22.64±0.15	35.52±0.33	29.94±3.79
JIT-SDP_pandas	47.58±4.42	61.74±0.17	60.62±0.40	23.52±0.13	36.34±0.38	23.72±1.04
Average (binary)	40.92	68.87	66.16	27.12	37.24	25.08
Average (all)	142.67	93.97	89.68	44.16	-	27.01

3. Effectiveness of DUST: Complete Results

Table 4: Comparison of overall accuracy (%) between DUST and corresponding wait-until-arrival variants across datasets and base models. For each dataset, the better performing variant is highlighted in bold. The bottom rows reports one-sided Wilcoxon signed-rank p-values, grouped by drift type. Shaded cells indicate statistically significant superiority of DUST.

(a) Synthetic datasets

Datasets	**DUST|ARF**	**Wait|ARF**	**DUST|HBP**	**Wait|HBP**	**DUST|HT**	**Wait|HT**
Agrawal_{na}	**73.87±1.46**	72.75±0.71	**68.73±0.63**	64.84±1.46	**65.78±0.65**	63.30±0.00
Agrawal_{ng}	72.85±1.09	**73.20±0.98**	**68.51±1.12**	65.31±1.24	**66.32±0.76**	64.56±0.00
Agrawal_{ra}	**74.72±1.51**	74.03±3.12	**66.12±1.12**	63.75±1.55	**68.17±0.59**	68.12±0.00
Agrawal_{rg}	73.49±1.31	**73.80±2.65**	**65.23±1.02**	63.10±1.12	**67.27±0.34**	66.91±0.00
LED_{na}	**68.23±0.89**	67.08±1.46	**68.62±0.51**	66.29±0.70	**65.40±0.15**	64.95±0.00
LED_{ng}	**67.71±0.73**	67.09±1.11	**69.10±0.59**	66.87±0.74	65.57±1.34	**67.75±0.00**
LED_{ra}	**61.37±1.27**	60.99±1.67	**63.42±0.53**	61.12±0.69	**62.23±0.62**	61.37±0.00
LED_{rg}	**67.53±1.17**	66.98±1.52	**68.59±0.38**	65.99±0.64	**66.29±0.13**	65.76±0.00
Sine_{na}	**94.95±0.15**	94.35±0.12	**87.19±2.45**	83.99±5.47	**88.76±0.78**	87.72±0.00
Sine_{ng}	**94.80±0.10**	94.08±0.25	**87.66±2.54**	81.09±4.18	**88.44±0.30**	87.55±0.00
Sine_{ra}	**90.54±0.27**	90.15±0.26	**84.72±2.37**	82.25±4.34	85.55±0.17	**85.67±0.00**
Sine_{rg}	**94.78±0.31**	94.03±0.39	**83.43±3.01**	80.90±4.40	**89.93±0.40**	87.01±0.00
SEA_{na}	**95.66±0.21**	95.50±0.20	**96.76±0.28**	95.78±1.34	**93.97±0.13**	93.66±0.00
SEA_{ng}	**95.79±0.23**	95.75±0.16	**97.07±0.29**	96.12±1.39	**94.24±0.11**	93.94±0.00
SEA_{ra}	**97.18±0.10**	97.00±0.19	**97.70±0.39**	96.77±1.39	**96.57±0.11**	96.09±0.00
SEA_{rg}	**97.22±0.08**	97.14±0.12	**97.55±0.41**	96.49±1.45	**96.64±0.10**	96.21±0.00
Hyperplane_1	**82.74±0.29**	80.97±1.10	**90.62±0.67**	89.07±1.15	**88.17±0.18**	87.46±0.00
Hyperplane_2	**84.26±0.46**	82.44±0.61	**91.44±0.65**	89.84±1.35	**88.93±0.25**	88.83±0.00
Hyperplane_3	**84.65±0.68**	83.14±0.88	**91.55±0.62**	90.10±0.96	89.35±0.37	**89.65±0.00**
RBF_1	**94.16±0.23**	93.65±0.41	**92.51±0.45**	90.25±1.07	**90.26±0.93**	89.92±0.00
RBF_2	**92.16±0.28**	91.25±0.34	**91.40±0.37**	89.37±1.21	**88.79±0.37**	88.03±0.00
RBF_3	**91.43±0.37**	91.13±0.34	**90.50±0.51**	88.85±1.23	**88.86±0.56**	87.92±0.00
Average	**84.10**	83.48	**82.66**	80.37	**81.61**	81.02
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	
All	2.0981e-05		2.3842e-07		3.4690e-04	

(b) Real-world datasets

Datasets	**DUST|ARF**	**Wait|ARF**	**DUST|HBP**	**Wait|HBP**	**DUST|HT**	**Wait|HT**
Airlines	**65.59±0.12**	65.39±0.11	**64.95±0.09**	64.90±0.12	**63.97±0.29**	63.58±0.00
Asfault	**79.06±0.29**	75.67±0.70	**80.56±0.20**	79.04±0.31	**75.29±0.23**	73.72±0.00
Covtype	**75.50±0.27**	73.18±0.49	**75.31±0.38**	73.60±0.69	**71.74±0.70**	70.05±0.00
Electricity	**76.44±0.29**	76.31±0.21	75.26±0.32	**75.36±0.16**	**73.51±0.50**	71.65±0.00
INS-Grad	**69.27±0.26**	66.56±0.34	**68.32±0.18**	67.64±0.17	55.93±1.60	**57.48±0.00**
INS-Inc-Abt	**63.94±0.05**	63.78±0.21	**66.75±0.08**	66.46±0.06	**55.98±0.29**	55.58±0.00
INS-Inc-Reo	**68.36±0.21**	68.11±0.13	**70.86±0.10**	70.54±0.06	52.36±0.57	**53.75±0.00**
Pokerhand	**65.84±0.35**	65.22±0.60	**69.88±1.33**	69.60±1.15	**66.19±1.99**	62.28±0.00
Rialto	**22.29±0.29**	21.04±0.21	**24.32±0.95**	22.71±0.72	**39.47±0.28**	38.37±0.00
UWave	**32.04±0.55**	31.03±0.65	34.51±0.36	**37.33±0.15**	**30.78±0.36**	21.54±0.00
Weather	**76.66±0.34**	75.55±0.17	**76.87±0.40**	76.05±0.40	**72.49±0.42**	71.75±0.08
Average	**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**	
All	4.8828e-04		4.1504e-02		2.6855e-02	

This section evaluates the effectiveness of DUST as an online learning framework under the IVL setting. We assess whether DUST can consistently enhance predictive performance across different base models. Since DUST is an extension of the wait-until-arrival strategy, we use the corresponding wait-until-arrival variants of each base model as baselines to enable a direct and fair evaluation of DUST's improvements. One-sided Wilcoxon signed-rank tests (at a 0.05 significance level) are conducted to assess the significance of improvements. Due to page limits, the complete results are shown in Table 4.

As shown in Table 4, 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 handling gradual drift, likely due to the pseudo-labeling mechanism based on k-NN: in gradual drift scenarios, conflicting concepts in the local neighborhood may undermine the reliability of majority voting, resulting in noisy or misleading pseudo-labels. Compared to the HBP neural model, tree-based models like HT and ARF (an ensemble approach with Hoeffding trees as base learners) are more vulnerable to this label noise, potentially resulting in incorrect split decisions and irreversible structural changes.

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.

4. Performance Evaluation: Complete Results

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

The complete results of the accuracy-based comparison are presented in Table 5. Since OSNN is only applicable to binary-class datasets, its results are omitted for multi-class datasets.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 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.

Table 5: Comparison of overall accuracy (%) between DUST and other IVL-handling methods across datasets. For each dataset, 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 '–'.

(a) Synthetic datasets

Datasets	**DUST**	**SkipE-RNN**	**SERMON**	**MIPLOSC**	**OSNN**	**Topline**
Agrawal_{na}	**73.87±1.46**	60.14±1.05 (↑)	61.54±0.33 (↑)	64.89±1.15 (↑)	63.61±0.05 (↑)	83.11±0.81 (↓)
Agrawal_{ng}	**72.85±1.09**	60.42±0.59 (↑)	62.11±2.62 (↑)	65.87±0.95 (↑)	63.36±0.11 (↑)	81.88±0.71 (↓)
Agrawal_{ra}	**74.72±1.51**	55.93±1.43 (↑)	58.84±2.50 (↑)	67.56±0.58 (↑)	61.88±0.08 (↑)	81.86±2.72 (↓)
Agrawal_{rg}	**73.49±1.31**	55.84±2.82 (↑)	58.79±1.59 (↑)	67.14±0.48 (↑)	61.87±0.04 (↑)	81.15±2.26 (↓)
LED_{na}	**68.23±0.89**	39.44±1.87 (↑)	40.93±1.69 (↑)	68.20±0.39 (~)	-	70.69±0.75 (↓)
LED_{ng}	67.71±0.73	40.67±2.50 (↑)	41.69±1.25 (↑)	**69.17±0.28 (↓)**	-	70.69±0.81 (↓)
LED_{ra}	61.37±1.27	36.85±1.61 (↑)	38.94±0.93 (↑)	**63.04±0.67 (↓)**	-	67.96±1.07 (↓)
LED_{rg}	67.53±1.17	41.94±1.77 (↑)	42.36±1.47 (↑)	**68.06±0.30** (~)	-	71.08±0.89 (↓)
Sine_{na}	**94.95±0.15**	87.97±0.21 (↑)	88.22±0.12 (↑)	89.23±0.49 (↑)	91.17±0.28 (↑)	96.21±0.11 (↓)
Sine_{ng}	**94.80±0.10**	84.19±0.28 (↑)	84.31±0.22 (↑)	91.89±0.31 (↑)	88.85±1.23 (↑)	95.68±0.21 (↓)
Sine_{ra}	**90.54±0.27**	85.49±0.28 (↑)	85.77±0.54 (↑)	86.05±0.39 (↑)	87.85±0.53 (↑)	94.66±0.23 (↓)
Sine_{rg}	**94.78±0.31**	84.16±0.34 (↑)	84.27±0.33 (↑)	92.34±0.28 (↑)	88.55±0.63 (↑)	95.63±0.25 (↓)
SEA_{na}	95.66±0.21	95.45±0.97 (~)	95.08±1.00 (~)	94.17±0.10 (↑)	**97.04±0.06 (↓)**	97.05±0.11 (↓)
SEA_{ng}	95.79±0.23	95.95±0.97 (~)	94.94±3.03 (~)	94.69±0.11 (↑)	**97.51±0.09 (↓)**	97.04±0.10 (↓)
SEA_{ra}	97.18±0.10	97.30±0.43 (~)	97.00±0.33 (~)	96.93±0.10 (↑)	**97.76±0.07 (↓)**	97.86±0.10 (↓)
SEA_{rg}	97.22±0.08	96.91±0.55 (~)	96.92±0.45 (↑)	96.67±0.10 (↑)	**97.75±0.05 (↓)**	97.74±0.06 (↓)
Hyperplane_1	82.74±0.29	83.23±3.12 (~)	85.21±2.28 (↓)	88.79±0.35 (↓)	**91.64±0.09 (↓)**	82.72±0.84 (~)
Hyperplane_2	84.26±0.46	83.17±2.85 (~)	85.12±2.78 (~)	90.03±0.29 (↓)	**93.20±0.11 (↓)**	83.78±0.62 (~)
Hyperplane_3	84.65±0.68	83.19±2.86 (~)	85.12±2.87 (~)	90.84±0.42 (↓)	**94.10±0.07 (↓)**	83.98±0.59 (↑)
RBF_1	**94.16±0.23**	87.14±0.67 (↑)	89.62±0.51 (↑)	91.86±0.19 (↑)	87.92±0.13 (↑)	94.30±0.25 (~)
RBF_2	**92.16±0.28**	86.87±0.37 (↑)	88.70±0.71 (↑)	90.66±0.24 (↑)	87.18±0.14 (↑)	93.66±0.16 (↓)
RBF_3	**91.43±0.37**	86.33±0.64 (↑)	88.82±0.48 (↑)	90.10±0.08 (↑)	88.03±0.14 (↑)	92.08±0.42 (↓)
Average (binary)	**88.07**	81.65	82.80	86.09	85.52	90.58
Average (all)	**84.10**	74.03	75.20	82.64	-	86.86
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)
All	-	(15/0/7)	(16/1/5)	(15/5/2)	(11/7/0)	(1/18/3)

(b) Real-world datasets

Datasets	**DUST**	**SkipE-RNN**	**SERMON**	**MIPLOSC**	**OSNN**	**Topline**
Airlines	**65.59±0.12**	58.57±0.31 (↑)	61.23±0.33 (↑)	64.54±0.28 (↑)	60.92±0.04 (↑)	67.03±0.07 (↓)
Asfault	**79.06±0.29**	73.68±1.13 (↑)	76.29±0.67 (↑)	76.65±0.34 (↑)	-	88.46±0.22 (↓)
Covtype	**75.50±0.27**	70.77±0.39 (↑)	70.37±0.30 (↑)	64.65±2.31 (↑)	-	89.95±0.18 (↓)
Electricity	**76.44±0.29**	75.59±0.63 (↑)	74.58±0.36 (↑)	75.94±0.30 (↑)	73.14±2.69 (↑)	88.79±0.06 (↓)
INS-Grad	**69.27±0.26**	62.59±0.50 (↑)	63.88±0.44 (↑)	63.06±0.27 (↑)	-	77.60±0.41 (↓)
INS-Inc-Abt	**63.94±0.05**	61.57±0.20 (↑)	62.42±0.19 (↑)	60.38±0.50 (↑)	-	75.68±0.11 (↓)
INS-Inc-Reo	**68.36±0.21**	66.09±0.32 (↑)	66.64±0.16 (↑)	60.80±0.38 (↑)	-	78.45±0.20 (↓)
Pokerhand	65.84±0.35	60.29±1.11 (↑)	60.49±1.37 (↑)	**67.53±1.08 (↓)**	-	83.89±0.29 (↓)
Rialto	22.29±0.29	19.48±0.31 (↑)	18.58±0.33 (↑)	**47.20±0.87 (↓)**	-	79.73±0.16 (↓)
UWave	**32.04±0.55**	27.71±0.61 (↑)	30.26±0.90 (↑)	26.88±0.20 (↑)	-	59.42±0.53 (↓)
Weather	**76.66±0.34**	72.51±1.74 (↑)	72.76±1.30 (↑)	73.82±0.22 (↑)	76.26±0.44 (↑)	78.64±0.16 (↓)
Average (binary)	**72.90**	68.89	69.52	71.43	70.11	78.15
Average (all)	**63.18**	58.99	59.77	61.95	-	78.88
Summary			(↑/↓/~)			
All	-	(11/0/0)	(11/0/0)	(9/2/0)	(3/0/0)	(0/11/0)

5. Ablation Study: Complete Results

Table 6: Ablation study of DUST in overall accuracy (%) across datasets. For each dataset, the best-performing variant is highlighted in bold. The bottom 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.

(a) Synthetic datasets

Datasets	DUST	DUST w/o IP	DUST w/o DE	DUST w_{500}	DUST $w_{1,000}$	DUST $w_{2,000}$	Wait
$Agrawal_{na}$	**73.87±1.46**	73.42±0.76	73.57±1.82	72.93±0.82	73.12±0.69	73.34±0.76	72.75±0.71
$Agrawal_{ng}$	72.85±1.09	72.83±0.71	72.83±0.86	73.02±0.67	72.94±0.56	72.86±0.80	**73.20±0.98**
$Agrawal_{ra}$	**74.72±1.51**	74.68±1.19	74.34±2.38	74.34±1.58	74.62±0.76	74.45±0.98	74.03±3.12
$Agrawal_{rg}$	73.49±1.31	72.84±2.16	**73.80±1.84**	73.59±1.90	73.54±1.16	72.78±1.75	73.80±2.65
LED_{na}	**68.23±0.89**	66.85±1.39	68.00±1.47	67.49±0.84	67.62±0.55	68.10±0.90	67.08±1.46
LED_{ng}	**67.71±0.73**	67.10±1.47	67.22±0.67	66.93±0.50	67.16±0.67	67.39±0.71	67.09±1.11
LED_{ra}	**61.37±1.27**	60.31±1.78	60.99±1.27	61.03±1.14	61.07±1.27	61.36±0.59	60.99±1.67
LED_{rg}	**67.53±1.17**	67.28±1.54	67.01±1.02	67.24±0.98	67.47±0.62	67.42±0.76	66.98±1.52
$Sine_{na}$	94.95±0.15	**94.96±0.18**	94.56±0.20	94.44±0.25	94.87±0.13	94.75±0.12	94.35±0.12
$Sine_{ng}$	**94.80±0.10**	94.29±0.20	94.07±0.23	94.46±0.21	94.44±0.13	94.66±0.27	94.08±0.25
$Sine_{ra}$	**90.54±0.27**	90.36±0.28	89.78±0.37	90.27±0.20	90.34±0.18	90.16±0.10	90.15±0.26
$Sine_{rg}$	**94.78±0.31**	94.27±0.42	94.13±0.16	94.36±0.14	94.63±0.26	94.76±0.30	94.03±0.39
SEA_{na}	**95.66±0.21**	95.53±0.14	95.34±0.15	95.53±0.07	95.61±0.10	95.64±0.11	95.50±0.20
SEA_{ng}	95.79±0.23	95.71±0.07	95.65±0.12	95.48±0.11	95.40±0.16	**95.81±0.14**	95.75±0.16
SEA_{ra}	97.18±0.10	96.98±0.16	97.11±0.09	97.04±0.06	97.10±0.07	**97.21±0.11**	97.00±0.19
SEA_{rg}	**97.22±0.08**	97.13±0.05	97.19±0.05	97.16±0.06	97.19±0.12	97.21±0.09	97.14±0.12
$Hyperplane_1$	**82.74±0.29**	82.08±1.00	82.35±0.66	81.96±0.54	82.09±0.64	82.51±0.35	80.97±1.10
$Hyperplane_2$	**84.26±0.46**	83.32±0.40	83.91±0.63	83.13±0.83	83.83±0.53	83.80±0.25	82.44±0.61
$Hyperplane_3$	**84.65±0.68**	83.63±0.69	84.24±0.58	84.08±0.56	84.11±0.70	84.26±0.49	83.14±0.88
RBF_1	**94.16±0.23**	94.04±0.21	94.11±0.31	93.68±0.18	93.86±0.42	94.11±0.22	93.65±0.41
RBF_2	**92.16±0.28**	91.59±0.48	91.94±0.32	91.40±0.28	91.74±0.21	91.97±0.16	91.25±0.34
RBF_3	**91.43±0.37**	91.16±0.50	91.31±0.29	91.16±0.31	91.28±0.25	91.38±0.14	91.13±0.34
Average	**84.10**	83.65	83.79	83.67	83.82	83.91	83.48
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
All		4.7684e-07	1.0252e-05	4.5300e-06	7.8678e-06	3.2663e-05	2.0981e-05

(b) Real-world datasets

Datasets	DUST	DUST w/o IP	DUST w/o DE	DUST w_{500}	DUST $w_{1,000}$	DUST $w_{2,000}$	Wait
Airlines	**65.59±0.12**	65.56±0.08	65.49±0.08	64.60±0.08	64.80±0.07	64.82±0.10	65.39±0.11
Asfault	79.06±0.29	75.22±0.59	**79.11±0.59**	77.88±0.28	78.19±0.31	77.00±0.37	75.67±0.70
Covtype	**75.50±0.27**	72.97±0.53	75.39±0.27	73.60±0.09	74.06±0.18	75.00±0.31	73.18±0.49
Electricity	**76.44±0.29**	76.42±0.22	76.40±0.36	76.30±0.24	76.31±0.27	76.22±0.28	76.31±0.21
INS-Grad	**69.27±0.26**	66.61±0.33	68.80±0.28	68.56±0.33	67.26±0.27	66.15±0.18	66.56±0.34
INS-Inc-Abt	63.94±0.05	63.86±0.17	63.89±0.18	63.62±0.08	63.80±0.08	**63.95±0.05**	63.78±0.21
INS-Inc-Reo	**68.36±0.21**	68.36±0.10	68.34±0.09	68.18±0.13	68.26±0.07	68.31±0.07	68.11±0.13
Pokerhand	**65.84±0.35**	65.22±0.45	65.44±0.31	65.39±0.10	65.51±0.14	65.83±0.15	65.22±0.60
Rialto	22.29±0.29	20.88±0.26	22.17±0.29	20.86±0.16	21.84±0.11	**22.32±0.16**	21.04±0.21
UWave	**32.04±0.55**	31.91±0.86	31.66±0.43	31.35±0.44	31.98±0.77	31.78±0.77	31.03±0.65
Weather	**76.66±0.34**	75.28±0.32	75.81±0.23	76.28±0.31	76.65±0.20	76.64±0.27	75.55±0.17
Average	**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**
All		4.8828e-04	2.4414e-03	4.8828e-04	4.8828e-04	4.8828e-03	4.8828e-04

To further evaluate the contribution of each component in DUST, we conduct an ablation study involving several degraded variants. Specifically, 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 Emphasis (DE). The variants studied include DUST w/o IP, DUST w/o DE, and a version without both mechanisms called Wait, which corresponds to the wait-until-arrival strategy where model updates occur only when delayed labels are available. Since both IP and DE fully rely on stable region identification via CDI, disabling these

mechanisms removes CDI's influence. Additionally, to assess the impact of the micro-cluster system, we replace it with fixed-size sliding windows of lengths 500, 1,000, and 2,000. Correspondingly, 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, with the same tuned hyperparameters and ARF as the base model, to ensure a fair comparison.

The complete results of the ablation experiments are shown in Table 6. One-sided Wilcoxon signed-rank tests at a significance level of 0.05 were 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 **??** 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.

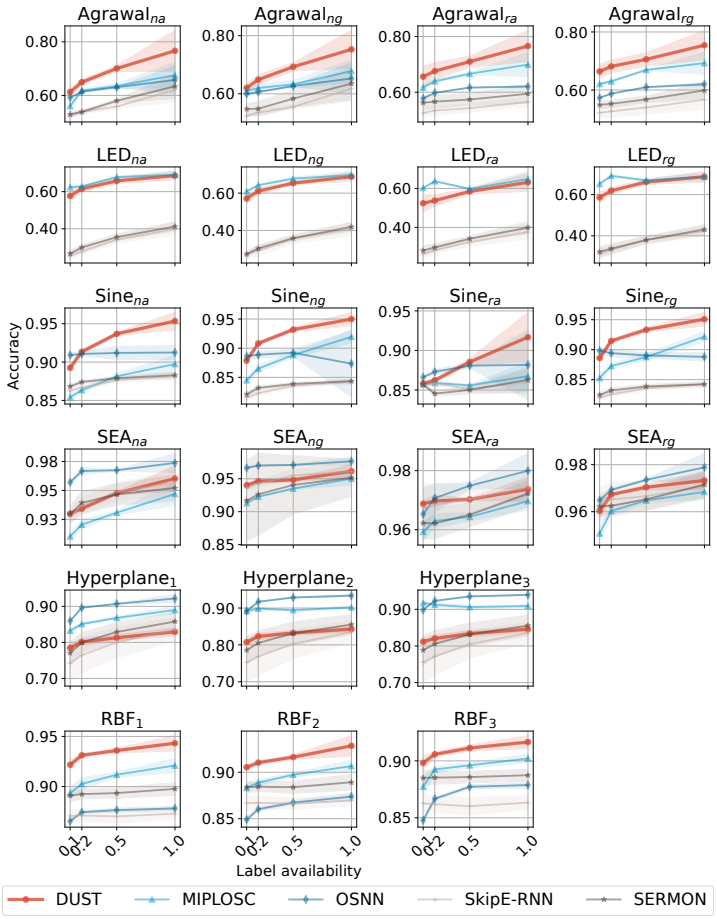
6. Performance under Different Label Availability

While our work primarily addresses the IVL problem, where ground truth labels eventually become available, it is also of practical significance to explore scenarios in which only a portion of data labels is accessible after a delay. This exploration serves as a complementary investigation into the practical utility of the DUST framework under conditions of limited label availability.

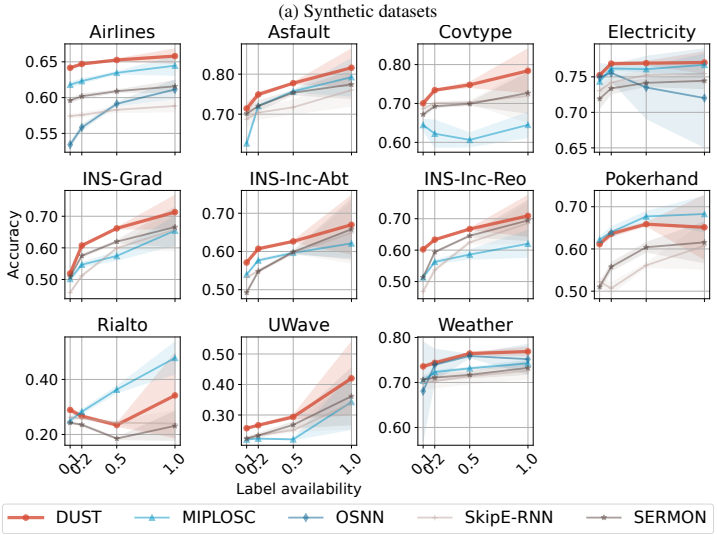
This section evaluates the performance of the proposed DUST framework across varying levels of label availability. To simulate restricted label access, we assign a label delay of $\delta_t = \infty$ to a proportion $1 - p$ of data instances, where $p \in \{0.1, 0.2, 0.5, 1.0\}$ denotes the proportion of available labels. For the remaining p proportion of instances, labels are delayed but ultimately become available, aligning with the standard IVL setting.

Figure 3 presents the performance fluctuations of all investigated approaches as the label availability p increases, across both synthetic and real-world datasets. We can see that most methods exhibit improved predictive performance as label availability rises. Furthermore, the relative rankings of the methods remain largely consistent across various p values, with the exception of the OSNN method. This consistency suggests that the comparative findings made under full label availability in Section 7.3 of the main paper also apply to scenarios with partial labeling.

Notably, DUST consistently outperforms other IVL methods, except for OSNN, which achieves better predictive performance on the Sine-related datasets (including Sine_{na}, Sine_{ng}, Sine_{ra}, Sine_{rg}) under low label availability. When label availability is extremely limited ($p = 0.1$), OSNN attains higher accuracy than DUST; however, as p increases, DUST gradually surpasses OSNN. This suggests that OSNN may serve as a viable alternative to DUST in binary classification tasks when very few labels are available; In most other scenarios, DUST remains a highly competitive and reliable option, even in partially labeled setting.



(a) Synthetic datasets



(b) Real-world datasets

Figure 3: Accuracy trends across various label availabilities. The lighter background areas represent standard deviations.

7. Case Study: Just-In-Time Software Defect Prediction (JIT-SDP)

Just-in-Time Software Defect Prediction (JIT-SDP) is an emerging research area in software engineering that aims to predict whether a code change is likely to induce defects within a more immediate timeframe, referred to as "just-in-time" [7, 8]. This task can be modeled as an online binary classification problem, where the objective is to predict at commit time whether a code change is defect-inducing (class 1) or clean (class 0) [9]. JIT-SDP faces two primary challenges. First, concept drift occurs as the codebase, development process, and team structures evolve over time, altering the relationship between commit features and defect likelihood. Second, verification latency further complicates the task, as defect labels are typically available only after the corresponding bugs are identified or fixed, which may take weeks or even years. Consequently, commits (data instances) remain unlabeled unless they are confirmed as defect-inducing or clean, making this scenario particularly relevant for our investigation.

Table 7: JIT-SDP commit features (Brief version of Table 1 in [7]

Dimension	Feature	Description
Diffusion	NS	# modified subsystems
	ND	# modified directories
	NF	# modified files
	Entropy	distribution of changes across files
Size	LA	lines of code added
	LD	lines of code deleted
	LT	lines of code before change
Purpose	FIX	indicator if change is a bug fix
History	NDEV	# developers contributing to modified files
	AGE	time since last change (avg.)
	NUC	# unique changes to those files
Experience	EXP	overall developer experience
	REXP	recent developer experience
	SEXP	experience on the affected subsystem

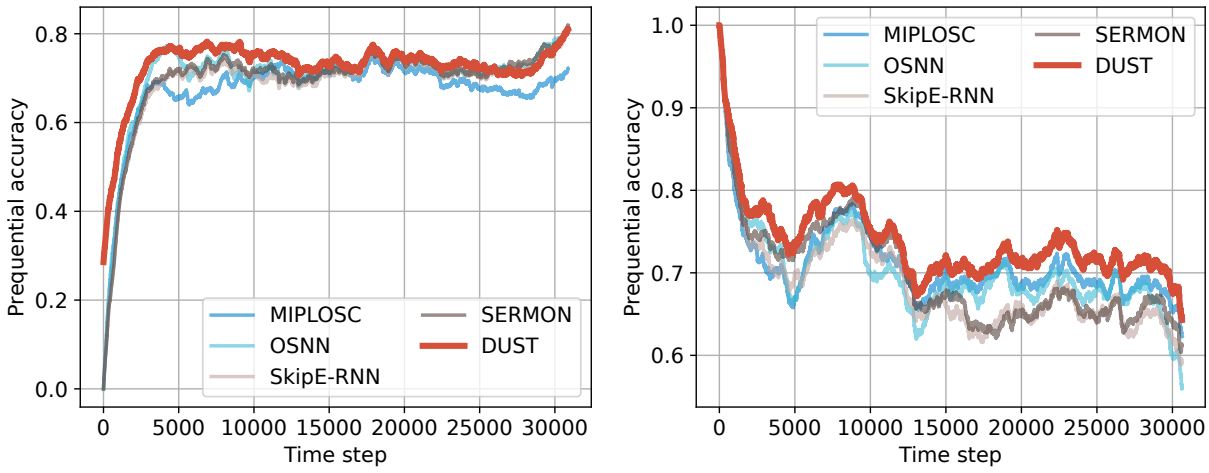
In this context, we evaluate DUST on two public JIT-SDP datasets, `django` and `pandas`, originally introduced in [9, 10]. Both datasets were constructed using `Commit Guru` [11] by mining GitHub histories and labeling each commit via the SZZ algorithm [12], supplemented by expert reviews to ensure annotation quality. Each commit is represented by 14 metrics organized into five dimensions, as shown in Table 7. The `django` dataset consists of 31,376 commits from 2005-07-03 to 2024-03-11, with a defect ratio of 40.95%, and the `pandas` dataset contains 31,138 commits from 2009-07-31 to 2024-03-10, with a defect ratio of 54.58%.

In these JIT-SDP datasets, only defect-inducing commits can eventually receive labels once they are confirmed to be linked to actual defects. As a result, only a subset of the true defect-inducing commits is labeled, while the rest remains unlabeled. To enable effective training of a binary classification model, we adopt the waiting time strategy [13, 14, 10], which assumes that if a commit is not identified as defect-inducing within a period (15 days in this case), it can be considered clean for practical purposes.

Table 8: Comparison of overall accuracy (%) between DUST and other IVL-handling methods across JIT-SDP datasets. For each dataset, the best-performing method is highlighted in bold.

Datasets	DUST	SkipE-RNN	SERMON	MIPLOSC	OSNN
Django	**74.43±0.28**	71.58±1.26	72.42±0.86	69.67±0.75	73.12±0.19
Pandas	**72.69±0.15**	67.27±0.22	68.51±1.20	69.74±0.46	68.71±2.47

Experimental results for the Django and Pandas datasets are summarized in Table 8, where DUST achieves the highest predictive accuracy on both datasets. Figure 4 further illustrates the prequential accuracy of each method over time, using a decay factor of 0.999. This evaluation protocol, following Wang et al. [15], provides a real-time view of model performance throughout the data stream.



(a) Django

(b) Pandas

Figure 4: Continuous performance comparison over time on JIT-SDP datasets in terms of prequential accuracy with decay factor of 0.999.

In the `django` dataset, the accuracy of all methods remains relatively stable throughout the stream, suggesting a low degree of concept drift. In this relatively stationary environment, DUST performs consistently well, quickly converging to high accuracy within the first 2,500 time steps. In contrast, the `pandas` dataset show frequent and substantial fluctuations in performance across all methods. Notable drops occur around time steps 3,000–5,000, 8,000–10,000, and 11,000–13,000, suggesting significant concept drifts during these periods. Despite this, DUST maintains strong performance throughout. The latter part of the stream (starting around time step 15,000) is marked by particularly volatile performance across all methods, reflecting a highly dynamic environment with frequent and potentially overlapping concept drifts. In such scenarios, models often struggle to fully recover from one drift before encountering the next. Under theses circumstances, pseudo-labeling with rollback methods, such as SERMON and SkipE-RNN, demonstrate significantly degraded performance. Their reliance on uncertain pseudo-labels during rapid drift can trigger frequent rollback operations, amplifying instability and reducing predictive accuracy. In contrast, DUST leverages stable region identification to filter unreliable information and mitigate noise introduced by outdated data, facilitating more robust model updates.

Overall, the evaluation of the JIT-SDP datasets demonstrate that DUST is highly capable of effectively addressing IVL under real-world, dynamic conditions.

References

[1] 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.

[2] 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/`

[3] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th International Conference on Very Large Data Bases, 2003, pp. 81–92.

[4] 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.

[5] C. W. Chiu, L. L. Minku, A diversity framework for dealing with multiple types of concept drift based on clustering in the model space, IEEE Transactions on Neural Networks and Learning Systems 33 (3) (2022) 1299–1309.

[6] 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.

[7] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, N. Ubayashi, A large-scale empirical study of just-in-time quality assurance, IEEE Transactions on Software Engineering 39 (2013) 757–773.

[8] M. Tan, L. Tan, S. Dara, C. Mayeux, Online defect prediction for imbalanced data, in: Proceedings of the 2015 IEEE International Conference on Software Engineering, Vol. 2, 2015, pp. 99–108.

[9] L. Song, L. L. Minku, C. Teng, X. Yao, A practical human labeling method for online just-in-time software defect prediction, in: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2023, p. 605–617.

[10] C. Teng, L. Song, X. Yao, Online cross-project approach with project-level similarity for just-in-time software defect prediction, Empirical Software Engineering 29 (158) (2024) 1–53.

[11] C. Rosen, B. Grawi, E. Shihab, Commit guru: analytics and risk prediction of software commits, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, p. 966–969.

[12] J. Śliwerski, T. Zimmermann, A. Zeller, When do changes induce fixes?, in: Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Vol. 30, 2005, pp. 1–5.

[13] G. G. Cabral, L. L. Minku, Towards reliable online just-in-time software defect prediction, IEEE Transactions on Software Engineering 49 (3) (2023) 1342–1358.

[14] L. Song, L. L. Minku, A procedure to continuously evaluate predictive performance of just-in-time software defect prediction models during software development, IEEE Transactions on Software Engineering 49 (2) (2023) 646–666.

[15] S. Wang, L. L. Minku, X. Yao, A systematic study of online class imbalance learning with concept drift, IEEE Transactions on Neural Networks and Learning Systems 29 (10) (2018) 4802–4821.