Autonomous Concept Drift Threshold Determination

Pengqian Lu[1], Jie Lu[1]*, Anjin Liu[1], En Yu[1], Guangquan Zhang[1]

[1]Australian Artificial Intelligence Institute (AAII)
University of Technology Sydney
Ultimo, NSW 2007, Australia
{Pengqian.Lu@student., Jie.Lu@, Anjin.Liu@, En.Yu-1@, Guangquan.Zhang@}uts.edu.au

Abstract

Existing drift detection methods focus on designing sensitive test statistics. They treat the detection threshold as a fixed hyperparameter, set once to balance false alarms and late detections, and applied uniformly across all datasets and over time. However, maintaining model performance is the key objective from the perspective of machine learning, and we observe that model performance is highly sensitive to this threshold. This observation inspires us to investigate whether a dynamic threshold could be provably better. In this paper, we prove that a threshold that adapts over time can outperform any single fixed threshold. The main idea of the proof is that a dynamic strategy, constructed by combining the best threshold from each individual data segment, is guaranteed to outperform any single threshold that apply to all segments. Based on the theorem, we propose a Dynamic Threshold Determination algorithm. It enhances existing drift detection frameworks with a novel comparison phase to inform how the threshold should be adjusted. Extensive experiments on a wide range of synthetic and real-world datasets, including both image and tabular data, validate that our approach substantially enhances the performance of state-of-the-art drift detectors.

Code — https://github.com/AAII-DeSI/concept-drift-RocStone/tree/main/AAAI2026-DTD

Introduction

In many applications, including network intrusion detection (Park et al. 2018) and solar forecasting (Wojtkiewicz, Katragadda, and Gottumukkala 2018), data is generated as a continuous stream whose underlying distribution is non-stationary and may change over time (Lu et al. 2018a). This phenomenon is termed as *concept drift*, which can significantly degrade model performance. A user-defined threshold is central to handling this drift. Typically, a hypothesis test statistic is monitored: when it crosses this threshold, a drift is signaled. This triggers an adaptation procedure, such as retraining (Gama et al. 2004; Baena-Garcia et al. 2006), to update the model for the new concept.

Traditionally, threshold selection has been seen as a trade-off: a lenient threshold risks delayed detection (leaving the model mismatched with new data), while a stricter threshold risks frequent false alarms (leading to excessive adaption and possible drops in accuracy). This view of the threshold is reflected in the design of many drift detectors (Frias-Blanco et al. 2014; Bifet and Gavalda 2007).

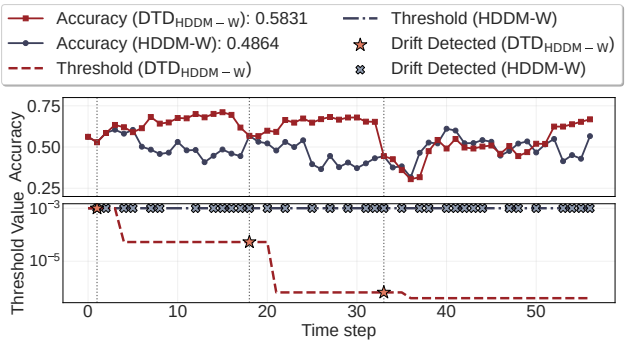

Figure 1: A case study on the Airline dataset shows the classic HDDM-W detector is overly sensitive, raising 36 alarms for a low 48.64% accuracy. By applying our DTD algorithm, the enhanced $DTD_{HDDM\text{-}W}$ detector dynamically adapts its threshold and trigger only three alarms, significantly boosting mean accuracy to 58.31%.

A recent study shows that by calibrating thresholds for sensitivity, different statistical tests can achieve similar model performance, i.e., online prediction accuracy (Liu et al. 2022). This inspired us to question the conventional view of the threshold as merely a tool for balancing statistical trade-offs. **Can we achieve better model performance by dynamically adjusting the threshold?**

In this paper, we answer this question affirmatively. We first prove that simply balancing delayed detection against false alarms does not guarantee optimal performance under concept drift. We then prove that no single fixed threshold can be universally optimal. Finally, we prove that dynamic threshold can significantly outperform any fixed threshold.

Motivated by these insights, we propose a dynamic threshold determination algorithm (DTD) to adjust the threshold in response to the current data, the model's state, and the chosen adaptation method. The main idea is that when a drift is detected, our algorithm runs three models

*Corresponding author.

in parallel for several time steps to test three hypotheses: that the detection was too late, correctly timed, or a false alarm. The model with best performance during this comparison directly informs how to adjust the threshold for future detections. By placing threshold design at the forefront of drift handling, our work offers a concrete direction for practitioners seeking to bridge the gap between theoretical ideals and the practical challenges of real-world data streams. As shown in Figure 1, we provide a case study on the real world Airline dataset. This case study highlights the effectiveness of our propose algorithm. By dynamically adjusting the detection threshold, our method reduced 36 drift alarms to just 3, boosting mean accuracy from a low 48.64% to 58.31%. The contributions of the paper are summarized below.

1. We first prove that the conventional goal of balancing detection tradeoff does not guarantee optimal model performance. We argue for shifting the focus from statistical trade-off to maintaining model performance.
2. We prove that no single, fixed threshold can be universally optimal. Furthermore, we prove that a dynamic threshold strategy is strictly superior to any static one, providing a firm theoretical basis for online adaptation.
3. Based on these insights, we propose the Dynamic Threshold Determination (DTD) algorithm. DTD introduces a novel comparison phase after a drift signal, using the performance of candidate models to intelligently adjust the threshold for future use.
4. We conduct extensive experiments on a wide range of synthetic and real-world datasets. The results demonstrate that our DTD algorithm enhances the performance of drift detectors in online data stream scenarios.

Related Work

Concept drift detection is typically addressed through two dominant paradigms: those that analyze the statistical properties of the data stream, and those that monitor model performance. One family of techniques quantifies dissimilarity between data samples, for instance, through statistical density estimation (Song et al. 2007). Histograms are a popular tool for representing distributions, especially in high-dimensional feature spaces (Liu et al. 2017), with innovations including the use of hierarchical structures (Boracchi et al. 2018) and dynamically adjustable binning strategies (Yonekawa, Saito, and Kurokawa 2022). Alternative partitioning methods like QuadTree (Coelho, Torres, and de Castro 2023) and K-means clustering (Liu, Lu, and Zhang 2020) have also been utilized. Some approaches extend beyond direct statistical comparisons by incorporating contextual factors (Lu et al. 2018b), using context-aware functions like CoDiTE (Cobb and Van Looveren 2022; Park et al. 2021), or forecasting future distributions (Li et al. 2022).

The second major paradigm, based on error rates, is often favored for its computational efficiency. Well-established detectors such as DDM (Gama et al. 2004), EDDM (Baena-Garcia et al. 2006), and HDDM (Frias-Blanco et al. 2014) function by monitoring fluctuations in the model's error rate. Refinements to this approach include adaptive window resizing (Bifet and Gavalda 2007) and forgetting mechanisms that dynamically weight classifiers (Jiao et al. 2022). More recent strategies have incorporated Gaussian Mixture Models or Fourier transform for comparing data windows (Yu, Lu, and Zhang 2024; Yu et al. 2025), implemented reactive states that activate upon alarm detection (Tahmasbi et al. 2021), detecting concept drift based on fine-grained error rate (Lu et al. 2025), monitoring the change of loss value (Zhou et al. 2024; Zhou and Lu 2025), or setting a threshold for true positive rate (Yang, Lu, and Yu 2025).

Existing drift detectors often rely on predefined thresholds or p-values to manage the trade-off between false alarms and detection speed. We will prove that such fixed settings prevent optimal performance in the next section. Importantly, we also demonstrate that dynamic thresholds are superior to static ones. We believe that this is the first study to introduce a strategy for automatically adjusting thresholds to maximize a model's overall effectiveness when data changes.

Methodology

Problem Setup

Let us denote a stream as D, which includes some labeled samples $\{(x_t, y_t)\}_{t=1}^{T}$. Here, each $x_t \in \mathcal{X}$ is the instance collected at time t, and $y_t \in \mathcal{Y}$ is its corresponding label. The size of stream T may be large or potentially unbounded. If the stream is collected in chunks, we denote the chunk collect at time t as $C_t = \{(x_i, y_i) | i \in [1, |C_t|]\}$ where $|C_t|$ is the size of the chunk. The joint distributions on $\mathcal{X} \times \mathcal{Y}$ is denoted as $\{P_t\}_{t=1}^{T}$, where P_t generates (x_t, y_t) at time t. If P_t remains identical for all t, there is no concept drift. Otherwise, if there is at least one time step t such that $P_t \neq P_{t+1}$, we claim the concept drift occurs at time $t+1$. We then consider P_t as the old concept and P_{t+1} as the new concept.

To detect whether a concept drift occurs at time t, we define a data window Ω_t as

$$\Omega_t \;=\; \{\,(x_k, y_k) \;\mid\; k \in [t - W + 1, \ldots, t]\},$$

where W is the size of the window and $t \geq W$. Drift detectors often split Ω_t into several sub-windows, compare distributions, or apply hypothesis tests to detect drift. Let S_t denote the test statistic computed on Ω_t, which can be considered as a function of Ω_t:

$$S_t \;=\; f\big(\Omega_t\big),$$

where $f(\cdot)$ is any statistic designed to signal a possible distribution change (e.g., an error-increase measure).

Drift Alarm. Given a fixed threshold θ at each time t, we pose a hypothesis test:

$$H_0 : \text{No drift at time } t \quad \text{vs.} \quad H_1 : \text{Drift at time } t.$$

If $S_t > \theta$, we reject null hypothesis and raise a drift detection alarm at t. Different methods define $f(\cdot)$ differently, but almost all compare a final statistic S_t to θ.

False Alarm. A false alarm arises when no actual drift is present but an alarm is raised. Formally, we define the probability of an false alarm at t as

$$\Pr[\text{false alarm at } t] = \Pr\big(S_t > \theta \;\mid\; P_t = P_{t-1}\big).$$

In practice, the design of drift detector seeks to keep this probability low to avoid frequent drift adaption.

Detection Delay. Assume a true concept drift occurs at time t (i.e., $P_t \neq P_{t-1}$). The detection delay, denoted by $\Delta(t)$, is the number of time steps required for the detector to raise an alarm. Formally, it is the smallest non-negative integer d such that the test statistic S_{t+d} exceeds a predefined threshold θ. A delay of $\Delta(t) = 0$ indicates an immediate detection. The probability of a specific delay d is given by:

$$\Pr[\Delta(t) = d] = \Pr\big(S_t \leq \theta, \ldots, S_{t+d-1} \leq \theta, \\ S_{t+d} > \theta \mid P_t \neq P_{t-1}\big).$$

Any $d > 0$ constitutes a delayed detection, as the alarm is raised only after additional data points have been observed.

Perfect Detection. Without loss of generality, we assume the statistic S_t is a measure of dissimilarity and a drift alarm will be raised when $S_t > \theta$. The threshold θ thus governs the critical trade-off between detection delay and false alarm. A lower threshold enhances sensitivity, enabling fast detection but at the cost of more frequent false alarms. A higher threshold ensures robustness against false alarms but at the expense of detection latency for actual drifts. We define a perfect, idealized detector as one that, for all t, simultaneously achieves zero false alarms and zero delay. Formally, it is a detector satisfies

$$\Pr\big[\text{false alarm at } t\big] = 0 \quad \text{and} \quad \Pr\big[\Delta(t) = 0\big] = 1.$$

Model Performance Drift adaptation is triggered when a test statistic S_t crosses a threshold. The rule for setting this threshold is the threshold strategy, denoted by $\boldsymbol{\theta}$. A strategy can be: 1) A fixed threshold $\boldsymbol{\theta} = \theta_{\text{const}}$. 2) A sequence of thresholds varying over time $\boldsymbol{\theta} = \{\theta_t\}_{t=1}^{T}$. The performance of a strategy $\boldsymbol{\theta}$ on a stream D is its online accuracy, defined using a 0-1 loss function $\ell(\cdot, \cdot)$:

$$A(\boldsymbol{\theta}; D) = \frac{1}{T} \sum_{t=1}^{T} (1 - \ell(\hat{y}_t, y_t)).$$

Note that the predictions $\{\hat{y}_t\}$ depend on $\boldsymbol{\theta}$, as it determines when the model conducts drift adaption. For brevity, we denote the performance on a stream D as $A(\cdot; D)$.

Theoretical Analysis

This section establishes the theoretical foundation that motivates the development of dynamic thresholding algorithms. We present three theorems that formalize the limitations of conventional fixed-threshold approaches and prove the superiority of a dynamic strategy. Due to space constraints, all proofs are deferred to the Appendix. First, we challenge the notion that perfect detection is always optimal.

Theorem 1 (Perfect Detection May Not Be Optimal)**.** *Perfect detection of concept drift may fail to yield optimal model performance in a streaming setting.*

This implies that even a statistically perfect detection, with zero delay and no false alarms, does not necessarily maximize model performance. For instance, detecting a very subtle drift might trigger an unnecessary adaptation, causing the model to forget valuable prior knowledge and ultimately harming its overall accuracy. This insight suggests that drift detection should focus more on preserving model performance rather than just achieving statistical perfection. Our second theorem challenges the notion that the threshold should be treated as a predefined, fixed value.

Theorem 2 (No Single Threshold is Universally Optimal)**.** *No single drift-detection threshold guarantees optimal performance on every dataset, model, and adaptation method.*

These limitations motivate our final theorem, which establishes the formal superiority of a dynamic approach.

Theorem 3 (Dynamic Thresholds Outperform Stationary Thresholds)**.** *Consider a data stream D. Let Θ_{const} be the set of all stationary thresholds and Θ_{dyn} be the set of all dynamic-threshold strategies. Let $A(\cdot; D)$ be the model performance on a stream D. Then:*

$$\max_{\{\theta_t\} \in \Theta_{\text{dyn}}} A(\{\theta_t\}; D) \geq \max_{\theta \in \Theta_{\text{const}}} A(\theta; D).$$

This result provides the theoretical justification for designing algorithms that adapt the detection threshold over time, which is the core contribution of this work.

Dynamic Threshold Determination Algorithm

Our proposed Dynamic Threshold Determination (DTD) algorithm, detailed in Algorithm 1, adaptively adjusts the threshold of a concept drift detector. At time step t, its core mechanism will be triggered if the detector's statistic S_t exceeds the current threshold θ. Instead of immediately conduct drift adaption, DTD initiates three candidate models enter a comparison phase. The threshold is then adjusted based on the relative performance of these models.

1. **Early Drift Model** (`EDM`): This model represents an aggressive strategy, assuming that the drift was detectable *before* the current time step t. Consequently, it initiates adaptation based on the data collected at last time step $t-1$. If `EDM` performs best among all of the candidate models, it suggests the initial detection was delayed. DTD then sets the threshold to the detector statistic from the previous time step, $\theta \leftarrow S_{i-1}$, to enhance sensitivity for the earlier detection of future drifts.
2. **Reactive Drift Model** (`RDM`): This model embodies a standard strategy and assumes the current sensitivity is appropriate. If `RDM` excels, it indicates that the detection timing and current threshold are appropriate. Accordingly, θ remains unchanged.
3. **Previous Model** (`PM`): This model holds the assumption that the drift signal at t is a false alarm. It thus refuse to conduct drift adaption at time step t. If `PM` demonstrates superior performance, it implies the system was overly sensitive and the signal at t was likely a false alarm. Therefore, DTD increases the threshold to $\theta \leftarrow S_i + \eta$, where η is a small positive constant. This adjustment aims to prevent similar false signals from triggering drift detections in the future.

Specifically, the algorithm operates in two primary phases: a Normal Operation Phase and a Comparison Phase.

Algorithm 1: Dynamic Threshold Determination Algorithm

Input: Data Stream $\mathcal{D} = \{C_t\}$; Drift Detector ψ; Initial θ_0; Initial Model M_0; Leading candidate model M_l = RDM; Prediction model $M = M_0$; Threshold $\theta = \theta_0$; List of accuracy $\Lambda = []$; Comparison phase flag Γ = **false**; Comparison steps $k = K$; Last model $M' = M_0$; A extreme small constant η.
Output: Avg(Λ).
List of candidate models $\mathcal{M} = \emptyset$.
List of accuracy of candidate models $\Psi = \emptyset$.
List of candidate drift detectors $\Pi = \emptyset$.
for each chunk $C_t \in \mathcal{D}$ **do**
  if Γ = **false then**
    a_i, S_t = Evaluate(M, C_t, ψ)
    if $S_t > \theta$ **then**
      $\mathcal{M}, \Pi, \Psi$ = CreateCandidates$(M, M', C_t, C_{t-1}, a_t, S_t, S_{t-1})$ # See Appendix
      Γ = **true**; $k = K$; M_l = RDM
    end if
    M' = copy(M);
    Train(M, C_t) # If continual training
  else
    A = EvalCandidates$(\mathcal{M}, C_t, \Pi)$ # See Appendix
    $k = k - 1$; $a_t = A[M_l]$; $M_l = \arg\max_{\texttt{name}}[A]$
    if $k = 0$ **then**
      $\Pi = \{$EDM : Avg(Π[EDM]), PM : Avg(Π[PM]), RDM : Avg(Π[RDM])$\}$
      $M_l = \arg\max_{name}[\Pi]$
      $M = \mathcal{M}[M_l]$; $\psi = \Psi[M_l]$
      θ = threshold of $\Psi[M_l]$;
      Γ = **false**; $\mathcal{M} = \emptyset$; $\Pi = \emptyset$; M_l = RDM;
    end if
  end if
  Add a_t to Λ
end for

Normal Operation Phase. During this phase, the system employs a primary predictive model M to process incoming data chunks C_t. For each chunk, M is evaluated, yielding an accuracy a_t, and a detector-specific statistic S_t is computed. If this statistic S_t exceeds the current threshold θ (or falls below it, depending on the nature of the detector), a potential concept drift is signaled. At this time, the system records S_t, the statistic from the previous chunk S_{t-1}, and instantiates three candidate models for the comparison phase.

As detailed in Appendix Algorithm 2, three distinct candidate models are instantiated, each representing a different hypothesis regarding the suspected drift:

1. EDM assumes that the true drift occurred at the preceding chunk C_{t-1}. It is therefore constructed by adapting the predictive model at last time step M' using data C_{t-1}.
2. RDM is constructed by adapting the prediction model M using data from the current chunk t_i.
3. PM is a direct copy of the primary model M, embodying the hypothesis that the drift signal was a false alarm.

To monitor these models during the comparison phase, DTD also initializes three corresponding drift detectors. The threshold for the detector associated with EDM is set to S_{t-1}. The threshold remains unchanged for RDM. The drift detector for PM is a copy of the primary drift detector used in the normal operation phase with the threshold set as $S_t + \eta$.

Comparison Phase. Upon a drift detection at time step t, DTD initiates the comparison phase to ascertain the nature of the detected change and to inform the threshold adjustment. This phase spans K subsequent data chunks, during which the candidate models are evaluated and incrementally updated using these new chunks if the training strategy is set as continual learning. The three corresponding drift detectors continuously monitor the performance of respective candidate models. If a detector signals a drift for its associated model, that model undergoes adaptation accordingly.

Upon completion of the K-chunk comparison phase, DTD compares the accumulated performance of the candidate models during this period as described in Appendix Algorithm 3. The winning model and its corresponding drift detector are then selected to become the new primary predictive model and its associated drift detector for the subsequent normal operation phase.

Experiment

Datasets and Baselines

Our experimental evaluation is conducted on a diverse set of datasets, comprising 3 real-world datasets (airline (Ikonomovska 2011), elec2 (Harries 1999), powersupply (Dau et al. 2019)) and 4 synthetic datasets (sine (Gama et al. 2004), mixed (Gama et al. 2004), CIFAR-10-CD (Lu et al. 2025), sea variants (Bifet et al. 2010)). Dataset details are provided in Appendix.

We establish comprehensive comparisons against 11 baseline methods: 5 classic concept drift detectors and 6 state-of-the-art (SOTA) approaches. The classic detectors include DDM (Gama et al. 2004), HDDM-A (Frias-Blanco et al. 2014), HDDM-W (Frias-Blanco et al. 2014), KSWIN (Raab, Heusinger, and Schleif 2020), and PH (Sebastião and Fernandes 2017). The SOTA methods are MCDD (Wan, Liang, and Yoon 2024), AMF (Mourtada, Gaïffas, and Scornet 2021), IWE (Jiao et al. 2022), NS (Wang et al. 2021), ADLTER (Wang et al. 2022), and PUDD (Lu et al. 2025). The threshold of these methods are set according to their original papers. The original PUDD paper proposes three options for the initial threshold. To distinguish between them, we denote a PUDD detector with an initial threshold of 10^{-x} as PUDD-x. Specific details of all baselines are described in Appendix due to page limit.

For baseline methods that are inherently classifier-agnostic, namely DDM, HDDM-A, HDDM-W, KSWIN, PH, and PUDD, we evaluate our proposed method using different base classifiers to ensure a comprehensive comparison. Specifically, our method is configured with a Gaussian Naive Bayes classifier (GNB) , a Very Fast Decision Tree (VFDT) (Hulten, Spencer, and Domingos 2001), and a Deep Neural Network (DNN) as its base learners. We also test all methods with two different training scenario for comprehensive evaluation. In the continual scenario, the classifier

Dataset	Training	KSWIN		DDM		PH		HDDM-A	
		Baseline	DTD_{KSWIN}	Baseline	DTD_{DDM}	Baseline	DTD_{PH}	Baseline	DTD_{HDDM-A}
Airline	continual	50.21±1.95	**57.29±4.44**	52.94±0.00	**53.60±0.00**	49.35±0.00	**52.69±0.00**	52.80±0.00	**52.98±0.00**
	sporadic	48.69±0.95	**53.49±2.36**	**52.43±0.00**	51.78±0.00	49.02±0.00	**51.19±0.00**	**55.62±0.00**	52.17±0.00
Elec2	continual	67.85±0.24	**69.26±0.54**	67.75±0.00	**71.83±0.00**	70.12±0.00	**71.28±0.00**	67.73±0.00	**70.19±0.00**
	sporadic	67.59±0.17	**68.21±0.34**	**67.60±0.00**	66.44±0.00	70.04±0.00	**70.48±0.00**	67.73±0.00	**68.23±0.00**
PS	continual	**71.21±0.13**	71.11±0.22	69.63±0.00	**71.88±0.00**	70.36±0.00	**71.88±0.00**	70.87±0.00	**72.01±0.00**
	sporadic	68.57±1.85	**70.98±0.35**	67.52±0.00	**70.74±0.00**	68.67±0.00	**70.12±0.00**	71.24±0.00	**72.09±0.00**
SEA0	continual	91.03±0.96	**91.66±1.08**	94.03±0.56	**94.75±0.51**	94.35±0.31	**94.84±0.23**	94.27±0.36	**94.75±0.31**
	sporadic	89.96±2.35	**90.50±2.11**	93.49±0.85	**94.21±0.66**	94.01±0.52	**94.60±0.24**	93.94±0.40	**94.51±0.29**
SEA10	continual	84.20±0.66	**85.05±0.99**	85.28±0.99	**87.08±0.41**	87.14±0.18	**87.61±0.18**	86.84±0.23	**87.27±0.28**
	sporadic	83.18±1.32	**84.92±1.62**	84.79±1.01	**86.35±0.70**	86.56±0.27	**87.08±0.28**	86.07±0.62	**86.75±0.41**
SEA20	continual	76.12±0.60	**77.13±0.61**	76.24±0.63	**77.36±0.77**	77.80±0.20	**78.18±0.20**	77.50±0.27	**77.85±0.33**
	sporadic	74.81±1.00	**76.94±0.70**	74.88±1.28	**76.76±0.75**	76.84±0.51	**77.61±0.27**	76.32±0.59	**77.14±0.58**
Sine	continual	81.73±1.01	**82.50±1.24**	82.19±1.44	**83.53±1.28**	83.55±1.36	**84.39±1.17**	83.09±1.26	**83.97±1.25**
	sporadic	80.84±1.81	**82.06±1.40**	81.05±3.81	**83.59±2.15**	83.16±2.23	**84.50±1.30**	83.28±1.27	**83.36±2.63**
Mixed	continual	83.87±0.11	**84.23±0.11**	83.80±0.14	**84.21±0.11**	83.87±0.11	**84.23±0.11**	83.87±0.11	**84.23±0.11**
	sporadic	83.51±0.24	**83.92±0.20**	82.69±2.21	**83.82±0.59**	83.51±0.24	**83.92±0.20**	83.52±0.24	**83.92±0.20**

Dataset	Training	HDDM-W		PUDD-1		PUDD-3		PUDD-5	
		Baseline	DTD_{HDDM-W}	Baseline	DTD_{PUDD-1}	Baseline	DTD_{PUDD-3}	Baseline	DTD_{PUDD-5}
Airline	continual	48.66±0.00	**58.31±0.00**	**53.57±0.27**	53.11±0.75	53.03±0.29	**53.71±0.42**	**52.16±0.88**	51.65±0.12
	sporadic	48.62±0.00	**49.88±0.00**	**51.05±0.01**	50.66±0.19	49.45±0.43	**52.77±0.09**	**54.37±0.77**	53.22±0.54
Elec2	continual	67.73±0.00	**70.11±0.00**	70.85±0.52	**72.12±0.14**	70.85±0.50	**72.16±0.20**	70.69±0.77	**72.24±0.25**
	sporadic	67.73±0.00	**67.95±0.00**	62.76±0.78	**69.04±0.15**	59.32±0.75	**69.04±0.15**	59.44±0.85	**69.04±0.15**
PS	continual	71.06±0.00	**71.90±0.00**	71.88±0.76	**71.98±0.04**	71.59±0.28	**71.98±0.04**	71.59±0.83	**71.80±0.00**
	sporadic	69.53±0.00	**70.04±0.00**	**71.13±0.69**	70.65±0.00	**71.20±0.49**	70.65±0.00	70.40±0.10	**70.52±0.00**
SEA0	continual	91.90±1.07	**92.73±1.07**	94.61±0.07	**94.96±0.29**	94.81±0.58	**94.97±0.20**	94.85±0.76	**94.97±0.19**
	sporadic	91.63±0.97	**92.53±1.05**	94.25±0.73	**94.66±0.26**	94.60±0.56	**94.65±0.25**	**94.62±0.22**	**94.62±0.28**
SEA10	continual	85.78±0.72	**86.75±0.60**	87.24±0.94	**87.67±0.16**	87.28±0.13	**87.67±0.15**	87.18±0.18	**87.60±0.19**
	sporadic	85.31±0.50	**86.68±0.45**	86.53±0.07	**87.22±0.25**	86.61±0.04	**87.08±0.32**	86.55±0.41	**87.01±0.28**
SEA20	continual	77.70±0.23	**77.92±0.25**	78.08±0.73	**78.33±0.19**	77.86±0.28	**78.32±0.17**	77.68±0.85	**78.19±0.22**
	sporadic	76.86±0.33	**77.41±0.42**	76.89±0.34	**77.55±0.36**	77.02±0.58	**77.48±0.48**	76.67±0.78	**77.19±0.65**
Sine	continual	82.46±1.08	**83.56±1.35**	83.12±0.19	**84.30±1.23**	83.39±0.35	**84.31±1.10**	83.43±0.51	**84.15±1.05**
	sporadic	82.61±1.87	**84.02±1.40**	81.48±0.11	**83.61±2.90**	83.80±0.16	**84.39±1.24**	83.38±0.82	**84.16±1.18**
Mixed	continual	83.87±0.11	**84.23±0.11**	82.99±0.13	**83.87±0.46**	83.92±0.41	**83.99±0.38**	84.12±0.60	**84.13±0.23**
	sporadic	83.51±0.24	**83.92±0.20**	79.23±0.67	**82.49±2.49**	83.58±0.19	**83.93±0.20**	**83.96±0.30**	83.92±0.20

Table 1: Performance comparison with classic drift detector and PUDD using the **GNB** classifier. We compare each baseline against our proposed method $DTD_{Baseline}$. The results are presented as mean accuracy (%) ± standard deviation (multiplied by 100 for space efficiency). The best performance in each pair is highlighted in **bold**. PS is short for powersupply.

learns at every time step. In the sporadic scenario, the classifier is trained only upon a drift alarm.

For the remaining SOTA methods (MCDD, AMF, IWE, NS, ADLTER), which primarily raise adaption without adaption, we compare them directly against our proposed ensemble method. Our method is presented as an ensemble version of PUDD, further enhanced with our novel DTD mechanism, considering that these baselines are also ensemble-based approaches. More implementation details are provided in Appendix due to page limit.

Comparison with Baselines and Ablation Studies

We conduct a comprehensive set of experiments to validate our claims and evaluate the performance of our proposed Dynamic Threshold Determination (DTD) Algorithm. Our evaluation is threefold: first, we apply DTD to a wide range of established drift detectors to demonstrate its general compatibility and effectiveness. Second, we compare a DTD-enhanced detector against state-of-the-art (SOTA) concept drift handling methods. Third, we test its applicability on complex image data streams. The results are summarized in Table 1, 2, 3, and Figure 2. Other results are provided in Appendix due to page limit.

Observation 1: DTD generally improves the performance of existing drift detectors. Experimental results demonstrate that DTD significantly enhances the performance of existing drift detectors. As shown in Tables 1 and 2, detectors equipped with DTD achieve higher predictive accuracy in the vast majority of scenarios. For instance, on the Sine dataset with a DNN classifier (Table 2), DTD boosts KSWIN's accuracy from 77.74% to 94.14%. While several exceptions e.g., PUDD-1 on Airline, show a baseline detector performs slightly better, this typically occurs when a candidate model's performance is high by chance during the

Dataset	Training	KSWIN		DDM		PH		HDDM-A	
		Baseline	DTD_{KSWIN}	Baseline	DTD_{DDM}	Baseline	DTD_{PH}	Baseline	DTD_{HDDM-A}
Airline	continual	61.05±2.22	**64.36±0.55**	61.49±2.41	**65.70±0.29**	60.47±2.24	**65.65±0.29**	61.72±2.15	**65.43±0.72**
	sporadic	60.90±0.65	**61.80±1.38**	57.88±1.67	**61.07±0.90**	60.19±0.49	**61.77±0.81**	59.17±0.79	**61.81±0.81**
Elec2	continual	73.45±1.30	**74.17±0.48**	73.41±1.20	**76.31±0.40**	74.15±0.84	**75.43±0.56**	73.29±1.59	**75.77±0.46**
	sporadic	72.54±1.21	**73.40±0.47**	**72.81±1.32**	71.99±0.54	72.18±0.93	**72.86±0.71**	72.71±1.42	**72.97±0.58**
PS	continual	70.92±2.03	**72.24±0.18**	71.34±0.55	**72.26±0.16**	71.09±1.90	**72.22±0.25**	69.62±2.74	**72.23±0.17**
	sporadic	67.52±2.72	**71.99±0.34**	65.58±3.74	**71.14±0.38**	69.07±0.65	**70.18±0.26**	68.74±3.57	**71.83±0.54**
SEA0	continual	97.96±0.20	**98.77±0.10**	97.19±0.70	**98.73±0.08**	97.96±0.28	**98.70±0.15**	97.98±0.22	**98.73±0.11**
	sporadic	91.12±1.95	**92.08±2.53**	93.66±3.83	**97.73±0.44**	97.04±0.44	**97.87±0.16**	96.78±0.63	**97.77±0.18**
SEA10	continual	88.18±0.28	**89.13±0.09**	88.23±0.25	**89.06±0.08**	87.64±2.17	**88.86±0.20**	87.70±2.88	**88.98±0.16**
	sporadic	82.80±1.20	**85.12±1.23**	85.04±1.28	**86.54±0.47**	86.79±0.42	**87.08±0.25**	86.44±0.38	**86.79±0.28**
SEA20	continual	77.51±1.47	**79.17±0.14**	78.03±0.38	**79.16±0.17**	77.75±1.12	**78.93±0.17**	77.49±1.56	**79.09±0.12**
	sporadic	74.55±0.77	**76.32±0.78**	74.47±1.27	**75.54±0.93**	76.56±0.65	**76.88±0.26**	75.86±0.67	**76.40±0.43**
Sine	continual	77.74±9.97	**94.14±0.82**	87.36±6.07	**93.56±1.20**	77.03±9.75	**93.96±1.02**	81.34±9.84	**94.39±0.67**
	sporadic	84.67±1.72	**87.35±2.04**	82.70±4.13	**89.02±5.26**	88.64±1.44	**91.28±0.97**	88.38±1.60	**91.32±0.95**
Mixed	continual	78.72±9.05	**88.85±0.15**	84.74±1.10	**89.97±0.03**	82.51±7.07	**88.02±0.24**	81.42±8.43	**89.99±0.03**
	sporadic	85.26±0.51	**86.87±0.11**	83.95±2.79	**86.07±2.14**	85.26±0.57	**86.83±0.15**	85.44±0.47	**86.47±1.58**

Dataset	Training	HDDM-W		PUDD-1		PUDD-3		PUDD-5	
		Baseline	DTD_{HDDM-W}	Baseline	DTD_{PUDD-1}	Baseline	DTD_{PUDD-3}	Baseline	DTD_{PUDD-5}
Airline	continual	60.55±1.65	**64.59±0.39**	**63.31±0.52**	62.49±0.31	**63.21±0.36**	62.60±0.29	**63.35±0.50**	62.53±0.34
	sporadic	61.81±0.42	**62.28±0.39**	**60.90±0.08**	60.73±0.91	**60.16±0.57**	59.99±1.21	**60.19±0.86**	59.77±0.89
Elec2	continual	73.33±1.36	**75.30±0.34**	74.92±0.18	**76.57±0.50**	74.93±0.90	**76.64±0.29**	74.92±0.41	**76.58±0.34**
	sporadic	72.82±1.66	**73.05±0.52**	69.35±0.19	**71.47±0.76**	68.98±0.67	**71.68±0.69**	68.68±0.24	**71.54±0.80**
PS	continual	70.20±3.05	**72.19±0.16**	**72.25±0.74**	72.12±0.17	**72.23±0.57**	72.14±0.19	**72.24±0.10**	72.21±0.19
	sporadic	68.58±3.42	**71.89±0.33**	**71.47±0.31**	71.02±0.79	70.37±0.92	**70.85±0.55**	70.20±0.72	**70.69±0.70**
SEA0	continual	97.96±0.21	**98.77±0.11**	97.94±0.89	**98.34±0.17**	98.04±0.15	**98.35±0.17**	98.23±0.03	**98.35±0.18**
	sporadic	92.21±1.07	**93.91±1.76**	94.89±0.37	**97.90±0.16**	95.99±0.79	**97.87±0.31**	96.29±0.22	**97.92±0.17**
SEA10	continual	88.05±0.43	**89.11±0.10**	87.86±0.85	**88.02±0.24**	87.80±0.15	**88.02±0.24**	87.80±0.61	**88.04±0.29**
	sporadic	85.30±0.55	**86.39±0.51**	85.91±0.26	**86.93±0.39**	86.02±0.20	**87.01±0.31**	86.24±0.07	**86.89±0.37**
SEA20	continual	77.50±1.58	**79.14±0.11**	**77.50±0.57**	77.45±0.38	77.33±0.06	**77.38±0.31**	**77.38±0.30**	77.36±0.34
	sporadic	**76.85±0.34**	76.73±0.37	76.00±0.79	**76.72±0.30**	76.35±0.17	**76.64±0.40**	76.38±0.28	**76.49±0.38**
Sine	continual	87.56±7.46	**94.19±0.74**	86.19±0.88	**93.02±1.07**	85.12±0.82	**93.26±0.91**	82.51±0.48	**93.23±0.92**
	sporadic	87.07±2.44	**88.57±2.70**	83.39±0.34	**90.63±1.79**	84.97±0.17	**91.36±0.97**	85.09±0.15	**91.33±0.96**
Mixed	continual	80.95±8.86	**89.98±0.02**	77.39±0.92	**88.79±0.13**	80.05±0.70	**88.79±0.12**	82.81±0.02	**88.79±0.12**
	sporadic	85.21±0.48	**86.92±0.16**	82.65±0.04	**86.34±1.64**	84.65±0.02	**86.92±0.18**	84.90±0.39	**86.92±0.18**

Table 2: Performance comparison with classic drift detector and PUDD using the **DNN** classifier. We compare each baseline against our proposed method $DTD_{Baseline}$. The results are presented as mean accuracy (%) ± standard deviation (multiplied by 100 for space efficiency). The best performance in each pair is highlighted in **bold**. PS is short for powersupply.

brief comparison phase. Nevertheless, the evidence strongly indicates that our method is more robust and effective on average than relying on a fixed threshold.

Observation 2: The benefits of dynamic thresholding are more significant in complex scenarios. A closer analysis of the results reveals a notable trend regarding the performance of DTD. The gains are often more significant when it is paired with complex models. While DTD provides benefits with simpler models like GNB and Hoeffding Trees, the improvements are particularly pronounced with the DNN classifier, as shown in Table 2. On the Mixed and Sine datasets, DTD provides a substantial accuracy improvement to nearly every detector. This suggests that the limitations of a static threshold become more evident as data and model complexity increase, making an adaptive approach like DTD more critical. The experiment results on hallenging CIFAR10-CD image dataset (Figure 2) also support our observation. It reinforces our argument that a dynamic, performance-aware thresholding mechanism is essential for handling complex data streams.

Observation 3: DTD is highly competitive with state-of-the-art methods. To evaluate its competitiveness, we compare our best configuration, DTD_{PUDD}, against several SOTA methods in Table 3. Our approach achieves the highest accuracy on all datasets, and consistently outperforms all rivals on the most cases, underscoring its robustness. Note the result on Sine dataset is provided in Appendix due to page limit. However, no single method dominates across all scenarios. For instance, ADLTER performs best on Airline, while a baseline PUDD-3 excels on PowerSupply and Sine. This observation reinforces our central thesis: no single configuration is universally optimal. Nevertheless, the Wilcoxon-Holm analysis in Figure 4 shows DTD_{PUDD} outperforms all SOTA competitors.

Method	Airline	Elec2	Mixed	PS	SEA0	SEA10
AMF	38.56	66.24	49.49	69.63	93.67	83.70
IWE	38.02	68.90	49.47	64.10	93.14	84.73
NS	67.91	76.42	81.09	72.39	93.54	84.39
ADLTER	**70.00**	76.10	87.63	72.48	93.40	85.89
MCD-DD	63.65	69.81	86.68	71.66	97.66	87.22
PUDD-1	63.78	77.28	89.51	72.68	98.47	87.72
PUDD-3	64.62	76.77	89.47	**72.79**	98.44	87.67
PUDD-5	64.45	76.92	89.37	72.74	98.49	87.74
$\text{DTD}_{\text{PUDD-1}}$	65.59	**77.30**	89.92	72.33	98.70	**88.88**
$\text{DTD}_{\text{PUDD-3}}$	65.59	77.16	89.92	72.34	**98.71**	**88.88**
$\text{DTD}_{\text{PUDD-5}}$	65.62	**77.30**	**89.98**	72.33	**98.71**	**88.88**

Table 3: Test accuracy (%) comparison of DTD VS SOTA methods on real-world and synthetic datasets. The best method is in bold. PS is short for powersupply. Results on Sine and SEA20 datasets are provided in the Appendix.

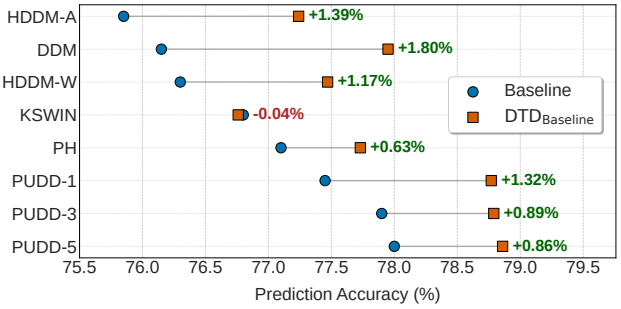

Figure 2: Comparison of accuracy on CIFAR10-CD dataset.

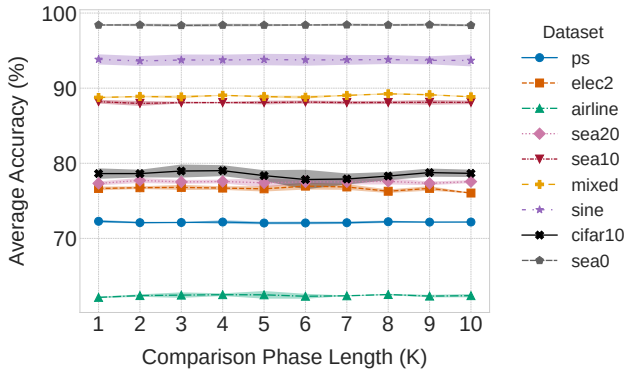

Figure 3: Ablation study of the DTD_{PUDD} algorithm's hyperparameter K, which indicates the length of comparison phase. Lines indicate the mean accuracy for each dataset, while shaded regions show the standard deviation calculated from multiple trials. PS is short for the powersupply dataset.

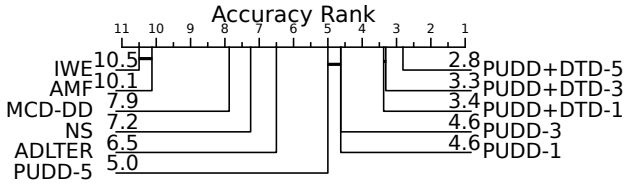

Figure 4: The critical difference diagram shows statistically significant superiority of DTD_{PUDD} over SOTA methods.

Observation 4: DTD is robust to its comparison phase duration, K. To assess DTD's sensitivity to its main hyperparameter K, we performed a dedicated ablation study. As DTD is a threshold adaptation algorithm, it must be paired with a base detector to monitor for drift. For this analysis, we therefore selected the combination where DTD proved most effective: the PUDD detector with an initial threshold of 10^{-3} and a DNN classifier. The only exception was the CIFAR10-CD dataset, for which a ResNet-18 classifier was used. This configuration was evaluated across 9 diverse datasets. The results in Figure 3 reveal remarkable stability for K values in the range of $[1, 10]$. On most datasets, like powersupply, accuracy remains nearly constant around 72.1%, showing the choice of K has a negligible impact. Even on complex datasets such as mixed and CIFAR10-CD, performance variation is minimal, with accuracy on mixed fluctuating only between 88.7% and 89.2%. This result shows our method is insensitive to the choice of K. A small default value (e.g., $K = 3$) thus provides a reliable and efficient configuration. This experiment validates the stability of our threshold adaptation mechanism.

Conclusion

This paper argues that conventional static thresholds for concept drift detection are suboptimal because they fail to maximize overall model performance. We theoretically prove that no single threshold is universally optimal and that dynamic strategies are inherently superior. We demonstrate this by constructing a superior dynamic strategy from a sequence of locally optimal thresholds and proving that no single, static threshold can match its overall performance. To address this, we propose a Dynamic Threshold Determination Algorithm (DTD), which dynamically adjusts the detection threshold by empirically evaluating the performance of different online adaptation strategies. Our extensive experiments confirm that DTD consistently improves a wide range of existing detectors. Our DTD-enhanced detectors are highly competitive with SOTA methods. Our future work will focus on two main objectives. First, we will try to include the threshold threshold in a loss function to build an end-to-end framework for dynamic threshold determination. Second, we plan to extend our algorithm to determine when to fine-tune large pre-trained language models. This would bring a cost-effective strategy that preserves model performance by avoiding unnecessary retraining.

Acknowledgements

This work was supported by the Australian Research Council through the Laureate Fellow Project under Grant FL190100149.

References

Baena-Garcia, M.; del Campo-Avila, J.; Fidalgo, R.; Bifet, A.; Gavalda, R.; and Morales-Bueno, R. 2006. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, 77–86. Citeseer.

Bifet, A.; and Gavalda, R. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, 443–448. SIAM.

Bifet, A.; Holmes, G.; Pfahringer, B.; Kranen, P.; Kremer, H.; Jansen, T.; and Seidl, T. 2010. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, 44–50. PMLR.

Boracchi, G.; Carrera, D.; Cervellera, C.; and Maccio, D. 2018. QuantTree: Histograms for change detection in multivariate data streams. In *International Conference on Machine Learning*, 639–648. PMLR.

Cobb, O.; and Van Looveren, A. 2022. Context-aware drift detection. In *International conference on machine learning*, 4087–4111. PMLR.

Coelho, R. A.; Torres, L. C. B.; and de Castro, C. L. 2023. Concept Drift Detection with Quadtree-based Spatial Mapping of Streaming Data. *Information Sciences*.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305.

Frias-Blanco, I.; del Campo-Ávila, J.; Ramos-Jimenez, G.; Morales-Bueno, R.; Ortiz-Diaz, A.; and Caballero-Mota, Y. 2014. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3): 810–823.

Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. 2004. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, 286–295. Springer.

Harries, M. 1999. Splice-2 comparative evaluation: electricity pricing (technical report unsw-cse-tr-9905). *Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney*, 2052.

Hulten, G.; Spencer, L.; and Domingos, P. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 97–106.

Ikonomovska, E. 2011. Airline dataset. *URL http://kt. ijs. si/elena_ikonomovska/data. html.(Accessed on 02/06/2020)*.

Jiao, B.; Guo, Y.; Yang, C.; Pu, J.; Zheng, Z.; and Gong, D. 2022. Incremental Weighted Ensemble for Data Streams with Concept Drift. *IEEE Transactions on Artificial Intelligence*.

Li, W.; Yang, X.; Liu, W.; Xia, Y.; and Bian, J. 2022. Ddgda: Data distribution generation for predictable concept drift adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 4092–4100.

Liu, A.; Lu, J.; Song, Y.; Xuan, J.; and Zhang, G. 2022. Concept drift detection delay index. *IEEE Transactions on Knowledge and Data Engineering*, 35(5): 4585–4597.

Liu, A.; Lu, J.; and Zhang, G. 2020. Concept drift detection via equal intensity k-means space partitioning. *IEEE transactions on cybernetics*, 51(6): 3198–3211.

Liu, A.; Song, Y.; Zhang, G.; and Lu, J. 2017. Regional concept drift detection and density synchronized drift adaptation. In *IJCAI International Joint Conference on Artificial Intelligence*.

Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G. 2018a. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12): 2346–2363.

Lu, J.; Xuan, J.; Zhang, G.; and Luo, X. 2018b. Structural property-aware multilayer network embedding for latent factor analysis. *Pattern Recognition*, 76: 228–241.

Lu, P.; Lu, J.; Liu, A.; and Zhang, G. 2025. Early concept drift detection via prediction uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19124–19132.

Mourtada, J.; Gaïffas, S.; and Scornet, E. 2021. AMF: Aggregated Mondrian forests for online learning. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(3): 505–533.

Park, J.; Shalit, U.; Schölkopf, B.; and Muandet, K. 2021. Conditional distributional treatment effect with kernel conditional mean embeddings and U-statistic regression. In *International Conference on Machine Learning*, 8401–8412. PMLR.

Park, S.; Seo, S.; Jeong, C.; and Kim, J. 2018. Network intrusion detection through online transformation of eigenvector reflecting concept drift. In *Proceedings of the first international conference on data science, E-learning and information systems*, 1–4.

Raab, C.; Heusinger, M.; and Schleif, F.-M. 2020. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416: 340–351.

Sebastião, R.; and Fernandes, J. M. 2017. Supporting the page-hinkley test with empirical mode decomposition for change detection. In *International Symposium on Methodologies for Intelligent Systems*, 492–498. Springer.

Song, X.; Wu, M.; Jermaine, C.; and Ranka, S. 2007. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 667–676.

Tahmasbi, A.; Jothimurugesan, E.; Tirthapura, S.; and Gibbons, P. B. 2021. Driftsurf: Stable-state/reactive-state learning under concept drift. In *International Conference on Machine Learning*, 10054–10064. PMLR.

Wan, K.; Liang, Y.; and Yoon, S. 2024. Online Drift Detection with Maximum Concept Discrepancy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Wang, K.; Lu, J.; Liu, A.; Song, Y.; Xiong, L.; and Zhang, G. 2022. Elastic gradient boosting decision tree with adaptive iterations for concept drift adaptation. *Neurocomputing*, 491: 288–304.

Wang, K.; Lu, J.; Liu, A.; Zhang, G.; and Xiong, L. 2021. Evolving gradient boost: A pruning scheme based on loss improvement ratio for learning under concept drift. *IEEE Transactions on Cybernetics*.

Wojtkiewicz, J.; Katragadda, S.; and Gottumukkala, R. 2018. A concept-drift based predictive-analytics framework: Application for real-time solar irradiance forecasting. In *2018 IEEE International Conference on Big Data (Big Data)*, 5462–5464. IEEE.

Yang, X.; Lu, J.; and Yu, E. 2025. Adapting Multi-modal Large Language Model to Concept Drift From Pre-training Onwards. In *The Thirteenth International Conference on Learning Representations*.

Yonekawa, K.; Saito, K.; and Kurokawa, M. 2022. RIDEN: Neural-based Uniform Density Histogram for Distribution Shift Detection. In *Proceedings of the Second International Conference on AI-ML Systems*, 1–9.

Yu, E.; Lu, J.; Yang, X.; Zhang, G.; and Fang, Z. 2025. Learning Robust Spectral Dynamics for Temporal Domain Generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yu, E.; Lu, J.; and Zhang, G. 2024. Fuzzy Shared Representation Learning for Multistream Classification. *IEEE Transactions on Fuzzy Systems*.

Zhou, M.; and Lu, J. 2025. Continuous Graph Learning-Based Self-Adaptation for Multi-Stream Concept Drift. *IEEE Transactions on Cybernetics*, 1–14.

Zhou, M.; Lu, J.; Lu, P.; and Zhang, G. 2024. Dynamic Graph Regularization for Multi-Stream Concept Drift Self-Adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 36(11): 6016–6028.