Early Concept Drift Detection via Prediction Uncertainty

Pengqian Lu, Jie Lu*, Anjin Liu, Guangquan Zhang

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

Abstract

Concept drift, characterized by unpredictable changes in data distribution over time, poses significant challenges to machine learning models in streaming data scenarios. Although error rate-based concept drift detectors are widely used, they often fail to identify drift in the early stages when the data distribution changes but error rates remain constant. This paper introduces the Prediction Uncertainty Index (PU-index), derived from the prediction uncertainty of the classifier, as a superior alternative to the error rate for drift detection. Our theoretical analysis demonstrates that: (1) The PU-index can detect drift even when error rates remain stable. (2) Any change in the error rate will lead to a corresponding change in the PU-index. These properties make the PU-index a more sensitive and robust indicator for drift detection compared to existing methods. We also propose a PU-index-based Drift Detector (PUDD) that employs a novel Adaptive PU-index Bucketing algorithm for detecting drift. Empirical evaluations on both synthetic and real-world datasets demonstrate PUDD's efficacy in detecting drift in structured and image data.

Code — https://github.com/RocStone/PUDD
Extended version — https://arxiv.org/abs/2412.11158

Introduction

In real-world applications, such as medical triage (Huggard et al. 2020) or time series forecasting tasks (Miyaguchi and Kajino 2019), the distribution of data may unpredictably change over time. This phenomenon, termed concept drift (Yuan et al. 2022), significantly degrades model performance. Moreover, drift also manifests between clients and servers in federated learning tasks (Jiang, Wang, and Dou 2022), or decision making process (Lu et al. 2020) further complicating the learning process. Error rate-based drift detection is one of the most popular approaches to handling concept drift due to its efficiency (Lu et al. 2018a). It continuously monitors the classifier's error rate, issuing an alarm when this rate exceeds a preset threshold (Raab, Heusinger, and Schleif 2020; Frias-Blanco et al. 2014).

However, in the early stages of concept drift, a model's error rate may remain stable. In such a case, error rate-based drift signals are unable to indicate the changes in data distribution. To address this, we explore alternative methods to detect distribution changes before a drop in error rate occurs. Upon rethinking the logic behind error rate-based drift signals, we realize that the distribution of prediction probability from a given model will change prior to the error rate itself. In addition, if the error rate does change, then the distribution of prediction probability must change too.

Prediction probability is just an example of a quantitative measure of predictive uncertainty in models. We believe that by clearly defining predictive uncertainty and demonstrating that it is a superior alternative to error rates, we can significantly enhance the sensitivity and robustness of drift detection. Therefore, we introduce the Prediction Uncertainty Index (PU-index), which measures the probability assigned by a classifier that an instance does not belong to the true class. Fig. 1 shows an illustrative example.

The objective of this paper is to address two key questions: (1) Can we identify a more effective drift detection signal that captures changes in data distribution when the error rate remains stable? (2) Can we theoretically prove that if the drift detection signal shows no significant change, then the model's error rate will also remain stable? In other words, we seek to develop a superior drift detection signal that can detect drift when the error rate cannot, and to prove that if the new signal fails to detect drift, the error rate will also fail to do so. We believe that these criteria are essential for evaluating and comparing different drift detection signals.

The theoretical results provided in this paper show strong evidence for the superiority of the PU-index as a metric for concept drift detection. It exhibits at least equivalent sensitivity to error-based metrics and potentially higher sensitivity in certain scenarios, rendering it a more robust and comprehensive measure for identifying concept drift.

However, it could be argued that a more sensitive detection method might result in a higher false alarm rate, potentially overreacting to minor fluctuations in model performance. If there is no significant drift in the error rate, why should we still seek to detect it? To mitigate such concern, we employ the Chi-square test, a robust statistical significance test, to the PU-index for drift detection. This approach helps distinguish between meaningful distributional shifts and inconsequential variations, thereby maintaining the benefits of increased sensitivity while minimizing false alarms. The p-value obtained

*Corresponding author.

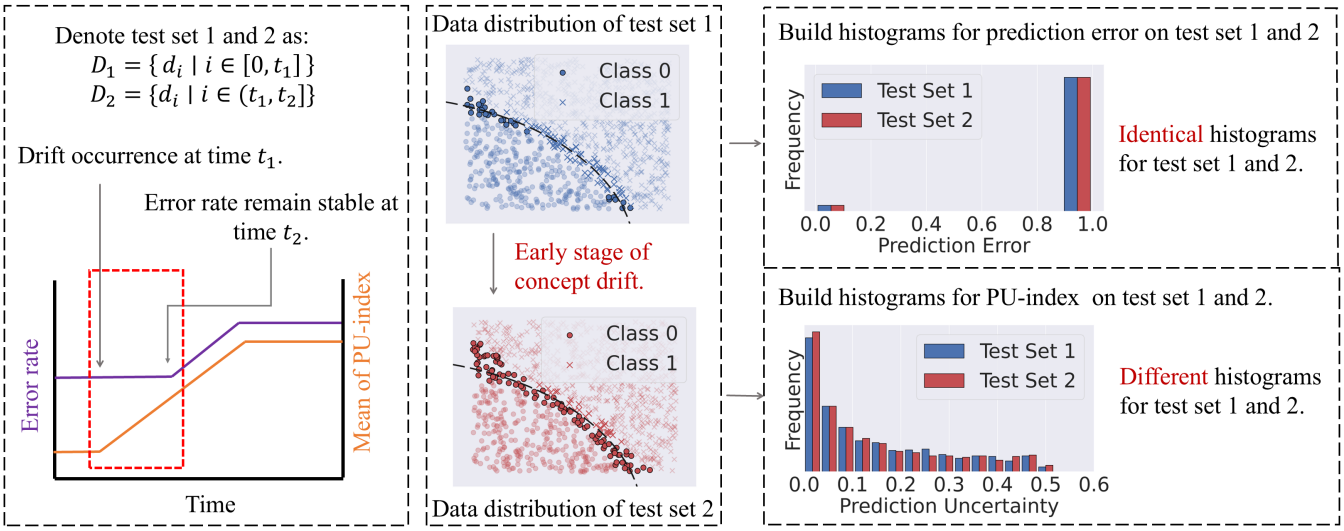

Figure 1: Illustrative example of the early stage of concept drift when an error rate-based detector fails to detect concept drift, but a prediction uncertainty-based detector can. The data around decision boundaries have been highlighted in the middle of the figure, showing the distribution gap between test sets 1 and 2. Such a gap implies concept drift occurrence. However, in this case, the error rates of the two test sets are the same. We also provide a theoretical proof in the Appendix to demonstrate the existence of such a case. By contrast, the distribution of prediction uncertainty has changed. The example implies that a prediction uncertainty-based detector can detect drift when an error rate-based detector fails.

from the Pearson's Chi-square test serves as a precise control mechanism for our tolerance to false alarms. By adjusting the significance level (α), we can directly modulate the trade-off between sensitivity and false positive rate.

In this paper, we introduce PUDD, a drift detector that uses the Prediction Uncertainty (PU) index to identify concept drift. PUDD employs a sliding window approach to remove outdated data and split the historical stream into two samples. It then applies an Adaptive PU-index Bucketing algorithm to automatically construct histograms that meet our theoretical conditions. Using these histograms, we apply Pearson's Chi-square test to determine if drift has occurred. Our main contributions are:

1. To the best of our knowledge, this is the first systematic study that compares two different drift detection signals with theoretical analysis.
2. We propose a novel drift detection metric called the PU-index, which is theoretically proven always to outperform error rate-based drift measurements. This provides crucial insight into the PU-index as a more sensitive and robust alternative for concept drift detection.
3. To identify concept drift in streaming data through the PU-index, we propose a Prediction Uncertainty index base drift detector. It comprises an Adaptive PU-index Bucketing algorithm to build a histogram for the PU-index, which meets the condition of our theoretical analysis, to conduct the Pearson's Chi-square test and detect drift.

Literature Review

In this section, we examine two approaches to concept drift detection: data distribution-based and error rate-based methods. The former directly addresses the root cause of drift—changes in the data distribution—while the latter focuses on variations in the model's performance, often achieving higher computational efficiency.

Data Distribution-based Methods

Data distribution-based methods measure shifts in the underlying distribution. For instance, a statistical density estimation approach is proposed in (Song et al. 2007), enabling the quantification of differences between two samples. Histogram-based techniques frequently serve to represent distributions in high-dimensional feature spaces (Liu et al. 2017). For example, (Boracchi et al. 2018) and (Yonekawa, Saito, and Kurokawa 2022) introduce hierarchical and dynamically adjustable strategies to construct histograms, respectively. Interval formation can also rely on methods like QuadTree (Coelho, Torres, and de Castro 2023) and K-means clustering (Liu, Lu, and Zhang 2020). Beyond direct histogram or density estimation, some methods incorporate contextual factors (Lu et al. 2018b) or anticipate future distributions. For example, (Cobb and Van Looveren 2022) uses a context-based CoDiTE function (Park et al. 2021) to detect drift, while (Li et al. 2022) exploits a predictive model for future distributions. There are also approaches that leverage Graph Neural Networks to track and adapt to distribution changes directly (Zhou et al. 2023). Although effective, these strategies can be computationally expensive in high-dimensional data streams (Souza et al. 2021). To date, no existing work uses histograms constructed from prediction uncertainty for drift detection, leaving a notable gap in the literature. To the best of our knowledge, there is no previous work that proposes building a histogram of prediction uncertainty to detect concept drift.

Error Rate-based Methods

Error rate-based detectors are well-studied and computationally efficient. Approaches such as (Gama et al. 2004), (Baena-Garcıa et al. 2006), and (Frias-Blanco et al. 2014) monitor variations in model error rates to detect drift. Adaptive window resizing is explored in (Bifet and Gavalda 2007), and forgetting mechanisms are introduced in (Jiao et al. 2022) to weight classifiers dynamically. More recent strategies apply Gaussian Mixture Models to compare windows (Yu et al. 2024) or enter reactive states upon detecting alarms (Tahmasbi et al. 2021). Despite their efficiency, error rate-based detectors struggle to identify drift when accuracy remains stable, particularly during its early stages (see Fig. 1). To address this limitation, we propose a prediction uncertainty-based approach, capable of detecting shifts even before error rates degrade. This method enhances early detection capabilities and complements existing drift detection strategies.

Preliminaries

Pearson's Chi-Square Test

The Pearson's Chi-square test assesses whether two categorical variables are independent. Its null hypothesis assumes independence, and if the computed p-value falls below a chosen significance level, the null hypothesis is rejected, indicating potential dependence between the variables.

The test's reliability depends on having sufficiently large observed and expected frequencies. As noted in (EP 1978), the Chi-square test produces valid results when each observed count exceeds 50 and every expected count exceeds 5. Under these conditions, the distribution of the test statistic closely approximates a normal distribution, enhancing the test's validity. The test relies on a contingency table. For the cell in the i-th row and j-th column, O_{ij} denotes the observed frequency, and its expected frequency is given by:

$$E_{ij} = \frac{n_i \times n_j}{N}, \tag{1}$$

where n_i and n_j are the cumulative frequencies of the respective row and column, and N is the sum of the table. The Chi-square test statistic is derived as:

$$\chi^2 = \sum_i \sum_j \left(\frac{O_{ij}^2}{E_{ij}} \right) - N. \tag{2}$$

Correspondingly, the p-value associated with χ^2 is:

$$p = 1 - \int_0^{\chi^2} \frac{x^{\frac{w}{2}-1} \cdot e^{-\frac{x}{2}}}{2^{\frac{w}{2}} \cdot \Gamma\left(\frac{w}{2}\right)} dx, \tag{3}$$

where w indicates the degrees of freedom, calculated as:

$$w = (\text{number of columns} - 1) \times (\text{number of rows} - 1). \tag{4}$$

Space Partitioning Algorithms

Space partitioning algorithms are widely studied to build histograms to establish density estimators (Silverman 2018). The key is to split feature space into partitions and count the instances falling into it to build a histogram. The partitions can be built by QuanTree (Boracchi et al. 2018), Kernel QuantTree (Stucchi et al. 2023), or Neural Network (Yonekawa, Saito, and Kurokawa 2022). Particularly, we introduce the Ei-kMeans space partitioning algorithm (Liu, Lu, and Zhang 2020), which can automatically determine the number and size of partitions.

Given two samples A and B, Ei-kMeans initializes centroids by iteratively selecting N/K points from a copy of A, where $N = |A|$ and K is the hyperparameter of the kMeans algorithm. Each iteration: (1) Select the point in A with the largest 1-NN distance as z_i. (2) Removes z_i and its N/K-nearest neighbors from A. This process repeats N/K times, yielding N/K initial centroids. Then the kMeans algorithm is applied to A with the initial centroids to derive K clusters denoted as $\{C_i | i \in [1, \frac{N}{K}]\}$.

To ensure that the number of examples in each cluster is larger than 5 to be able to conduct the Chi-square test, an amplify-shrink algorithm is proposed to adjust the number of examples in each cluster. Let us denote the number of instances in the clusters as $V = \{C_i || i \in [1, \frac{N}{K}]\}$. The distance matrix between the examples in A and the centers of each cluster is denoted as $M_{dist} \in \mathbb{R}^{N \times K}$, which is amplified by: $M_{dist} = M_{dist} \odot \left(\mathbf{1} \cdot e^{\theta \cdot \left(\frac{V}{N-1}\right)} \right)$, where $\mathbf{1} \in \mathbb{R}^{N \times 1}$ is an all-ones matrix, θ denotes the hyperparameter controlling the shape of the coefficient function, . Let M_{dist}^{ij} denote the amplified distance between i-th data A_i and j-th center c_j, the assigned cluster for A_i is defined as $y_i = \arg\min_{j=1\ldots K} M_{dist}^{ij}$.

After the amplify-shrink algorithm, the final clusters are derived and can be considered as subspaces in the feature space of A. The numbers of examples of A and B in the clusters are counted to form a histogram.

Methodology

This section sets up the problem and provides a theoretical analysis demonstrating the advantages of the PU-index over error rate-based methods. We then introduce a novel sliding window strategy and an Adaptive PU-index Bucketing algorithm for concept drift detection and adaptation.

Problem Setup

Formally, we represent the streaming data collected during the period $[1, t]$ as $D_{1,t} = \{(x_j, y_j) | j \in [1, t]\}$. If the data is collected in chunks, then the stream includes a set of chunks $D_{1,t} = \{\bar{D}_j | j \in [1, t]\}$, where each chunk $\bar{D}_j = \{(x_{jk}, y_{jk}) | k \in [1, M]\}$ includes M examples. Here, x_{jk} represents an instance with d dimensional attributes, y_{jk} denotes the corresponding label, and M denotes the chunk size. In this paper, we focus only on the data collected in chunks. If the stream $D_{1,t}$ follows a distribution $P_{1,t}(x, y)$, following (Lu et al. 2018a), we claim that a drift occurs at time $t + 1$ if

$$P_{1,t}(x, y) \neq P_{t,\infty}(x, y). \tag{5}$$

The goal of concept drift detection is to raise an alarm at time $t + 1$ when the distribution of data changes. The most popular metric for detecting the distribution change is prediction error. For a classifier f, assuming the number of

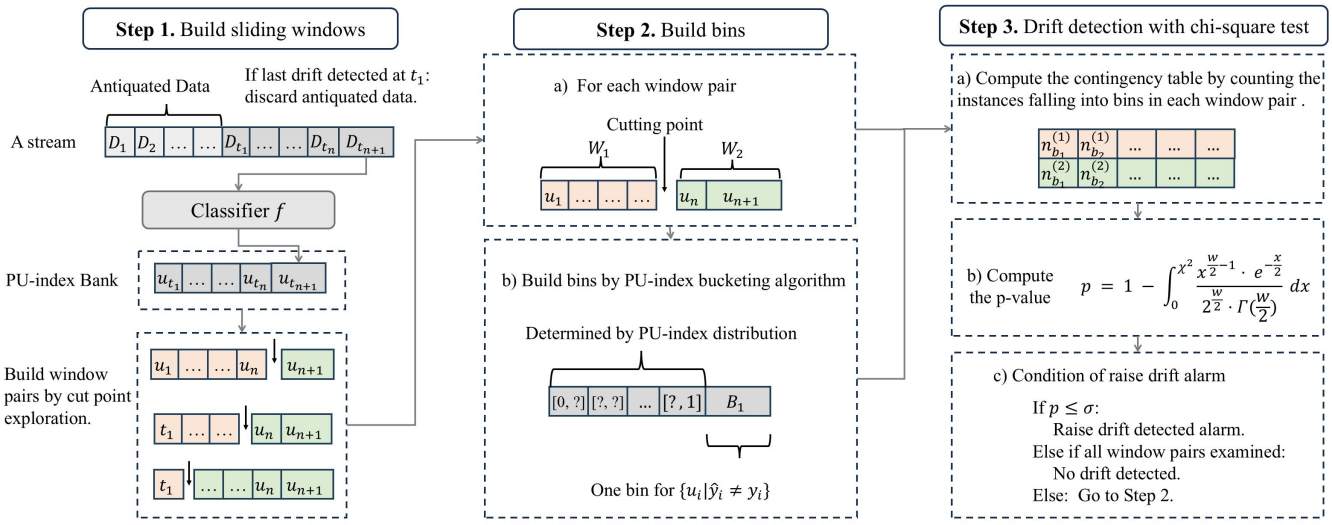

Figure 2: The framework of our proposed algorithm. The sliding window strategy has two components, i.e., antiquated data discard and cutting point exploration as shown on the left. The Adaptive PU-index Bucketing algorithm is shown in the middle. The drift detection process is shown on the right.

classes is n, the classifier will output a prediction probability for each class $\hat{y} \in \mathbb{R}^n$ and $\sum_{i=1}^{n} \hat{y} = 1$. The prediction error of an instance x_i is defined as:

$$e_i = \mathbb{I}(\hat{y}_j = \arg\max_j f_j(x_i) \neq y_i), \tag{6}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

As we mentioned earlier, our motivation is that the prediction probability will intuitively change before the prediction error when drift occurs. In this paper, we measure the prediction probability by the PU-index which is defined as:

$$u_i = 1 - f_{y_i}(x_i), \tag{7}$$

where $f_{y_i}(x_i)$ denotes the probability predicted by the classifier that x_i belongs to the ground truth class y_i.

Theoretical Analysis of Error Rate and PU-index

To rigorously evaluate the efficacy of these two metrics for concept drift detection, we conduct a theoretical comparison from two complementary perspectives. (1) When the PU-index distribution remains stable, potentially failing to detect concept drift, we investigate whether the error rate distribution exhibits changes that could indicate drift. (2) Conversely, when the error rate distribution remains constant, we examine whether the PU-index distribution demonstrates changes that might reveal underlying changes in the data stream.

Theorem 1. *Let W_1 and W_2 be two windows of a data stream in a multi-class classification problem. If their respective PU-index histograms H_1 and H_2 are identical, where the histograms are constructed such that the first bin contains all misclassified instances and the remaining bins partition the misclassified instances, then the error rates and error standard deviations of W_1 and W_2 are equal.*

Theorem 2. *Given a multi-class classification problem, if two windows have equal error standard deviations or error rates, their PU-index histograms, where the first bin contains all correctly classified instances and the remaining bins partition the misclassified instances, may not have identical bin proportions.*

Due to the page limit, the proofs are provided in the Appendix. These theorems lead to the following conclusions: (1) Theorem 1 demonstrates that when the PU-index distribution remains stable, the error rate and the error standard deviation also remain constant. This implies that if the PU-index fails to detect concept drift, error-based metrics will also fail to detect it. (2) Theorem 2 establishes that even when error rates and error standard deviations are equal between two windows, the PU-index distributions may differ. This suggests that the PU-index has the potential to detect subtle changes in the data distribution that are not captured by traditional error-based metrics. These findings show that the PU-index offers at least the same sensitivity as error-based metrics and potentially higher sensitivity in certain scenarios, making it a more robust and comprehensive measure for detecting concept drift in streaming data environments.

Sliding Window Strategy and Adaptive PU-index Bucketing Algorithm

To detect concept drift using the PU-index without making it overly sensitive, we apply the Chi-square test to examine the PU-index distribution. The hypotheses are:

Null Hypothesis ($\mathcal{H}_0$): The PU-index distribution does not change over time, indicating no concept drift.

Alternative Hypothesis ($\mathcal{H}_1$): The PU-index distribution changes over time, indicating concept drift.

If the Chi-square statistic exceeds the critical value at the chosen significance level, we reject $\mathcal{H}_0$ and conclude that

	Plane	**Car**	**Bird**	Cat	$\cdots$	Truck
Plane	0.98	0.00	0.00	0.00	$\cdots$	0.01
Car	0.01	0.89	0.01	0.01	$\cdots$	0.01
Bird	0.01	0.01	0.91	0.01	$\cdots$	0.01
Cat	0.00	0.02	0.01	0.92	$\cdots$	0.01
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$
Truck	0.00	0.01	0.01	0.01	$\cdots$	0.92

Table 1: Transition matrix of generating CIFAR-10-CD through the Markov process. The classes marked in bold represent the user's initial interest and are considered positive labels. All other classes are considered negative labels.

concept drift has occurred. Otherwise, we detect no significant drift. Thus, detecting concept drift using Chi-square involves two key steps: (1) partitioning the data stream into two windows and (2) constructing a histogram of the collected PU-index values.

We adopt a sliding window strategy to handle online streaming data. Let $D_{1,t} = \{\bar{D}_j | j \in [1, t]\}$ denote PU-index chunks collected from the start of the stream, potentially from different distributions. For instance, suppose D_{1,t_1} and $D_{t_1,t}$ differ in distribution. If a new chunk $\bar{D}_{t+1}$ matches the distribution of $D_{t_1,t}$, no drift should be detected. However, keeping antiquated data D_{1,t_1} could trigger a false alarm, since D_{1,t_1} and $D_{t_1,t+1}$ differ in distribution.

To solve this "antiquated distribution" problem, we discard outdated data after detecting a drift at time t_1. Subsequent drift detection uses only $D_{t_1,t+1}$, avoiding false alarms caused by old data. After discarding antiquated data, we must determine how to form two windows on the current substream. We do this by exploring all possible cutting points $r \in [t_1, t+1]$. Thus, $D_{t_1,t+1}$ is split into $D_{t_1,r}$ and $D_{r,t+1}$ for the Adaptive PU-index Bucketing algorithm. The sliding window is illustrated on the left side of Fig. 2.

Theoretical analysis shows that misclassified instances' PU-indices must be grouped into the same bin. For counterpart, we use Ei-kMeans to form bins that meet the Chi-square test requirements. We call this the Adaptive PU-index Bucketing algorithm, illustrated in the middle of Fig. 2.

PU-Index based Drift Detector

In this subsection, we introduce the overall of our method. Firstly, given a substream containing the recent chunks' PU-index $u_{t_1,t}$, we explore all cutting points $r \in [t_1, t]$. Based on the cutting points, we have $t - t_1$ window pairs, denoted as $\boldsymbol{u}_{t_1,r}$, and $\boldsymbol{u}_{r,t}$. Then we defined the PU-index pairs for correctly and wrongly classified instances as:

$$\boldsymbol{u}^C_{t_1,r} = \{u | u_i \in \boldsymbol{u}_{t_1,r} \wedge \hat{y}_i = y_i\}, \tag{8}$$

$$\boldsymbol{u}^C_{r,t} = \{u | u_i \in \boldsymbol{u}_{r,t} \wedge \hat{y}_i = y_i\}, \tag{9}$$

$$\boldsymbol{u}^M_{t_1,r} = \{u | u_i \in \boldsymbol{u}_{t_1,r} \wedge \hat{y}_i \neq y_i\}, \tag{10}$$

$$\boldsymbol{u}^M_{r,t} = \{u | u_i \in \boldsymbol{u}_{r,t} \wedge \hat{y}_i \neq y_i\}. \tag{11}$$

These four equations represent the PU-index for correctly and misclassified instances in the first and second windows, respectively. Therefore, $\boldsymbol{u}_{t_1,r} = \{\boldsymbol{u}^C_{t_1,r}, \boldsymbol{u}^M_{t_1,r}\}$ and $\boldsymbol{u}_{r,t} = \{\boldsymbol{u}^C_{r,t}, \boldsymbol{u}^M_{r,t}\}$. Next, we compute the contingency table $T \in \mathbb{R}^{2\times(K+1)}$, where K is a hyperparameter in Ei-kMeans. To calculate T, we apply the Adaptive PU-index Bucketing algorithm on $\boldsymbol{u}^C_{t_1,r}$ to build a histogram. Then we count the instances of $\boldsymbol{u}^C_{t_1,r}$ that fall into the histogram bins and fill them in T_{1i} where $i \in [1, K]$. Likewise, we count the examples of $\boldsymbol{u}^C_{r,t}$ falling into the previously obtained bins and fill them in T_{2i}. Finally, we fill in $T_{1,K+1}$ and $T_{2,K+1}$ with the size of $\boldsymbol{u}^M_{t_1,r}$ and $\boldsymbol{u}^M_{r,t}$. The expected frequency of T_ij is defined as:

$$E_{ij} = \frac{\sum_{j=1}^{K+1} T_{ij} \times \sum_{i=1}^{2} T_{ij}}{\sum_{ij} T_{ij}}. \tag{12}$$

The Chi-square test statistic is defined as:

$$\chi^2 = \sum_i \sum_j \left(\frac{T_{ij}^2}{E_{ij}} \right) - \sum_{ij} T_{ij}. \tag{13}$$

Finally, the p-value is computed by:

$$p = 1 - \int_0^{\chi^2} \frac{x^{\frac{K}{2}-1} \cdot e^{-\frac{x}{2}}}{2^{\frac{K}{2}} \cdot \Gamma\left(\frac{K}{2}\right)} \, dx, \tag{14}$$

Based on the Equation (12-14), we can compute the p-value for each window pair $D_{t_1,r}$, and $D_{r,t}$. If the minimum p-value among all window pairs is smaller than a predefined threshold σ, we raise a drift detected alarm. It is important to clarify that the specified threshold controls the Type I error rate for each individual window pair test rather than the overall Type I error across the entire substream. Consequently, we do not employ multiple comparison adjustments since our statistical guarantees apply at the single-test level rather than the family-wise level. The pseudo-code and time complexity analysis is provided in the Appendix.

Experiments

In this section, we introduce the settings and results of the experiments in our paper. The details of implementation, datasets, baselines, and the critical difference diagrams (Ismail Fawaz et al. 2019) for the experiments in this paper are introduced in the Appendix.

Datasets and Baselines

We propose CIFAR-10-CD, a synthetic concept drift image dataset with transition matrix shown in Table 1, to simulate user interests changing via a Markov process. Initially, three CIFAR-10 classes are marked positive, with interest shifts occurring probabilistically (e.g. 1% chance of Plane to Horse transfer). Our experiments utilize 3 real-world datasets (airline(Ikonomovska 2011), elec2(Harries 1999), powersupply(Dau et al. 2019)) and 4 synthetic sets (sine(Gama et al. 2004), mixed(Gama et al. 2004), CIFAR-10-CD, sea variants(Bifet et al. 2010)). We compare against 7 classic detectors (ADWIN(Bifet and Gavalda 2007), DDM(Gama et al. 2004), EDDM(Baena-Garcıa et al. 2006), HDDM-A(Frias-Blanco et al. 2014), HDDM-W(Frias-Blanco et al. 2014), KSWIN(Raab, Heusinger, and Schleif 2020), PH(Sebastião

		Incremental Training						Training only at Initialization or Adaptation					
Classifier	ddm name	airline-I	elec2-I	mixed-I	ps-I	sea0-I	sine-I	airline-O	elec2-O	mixed-O	ps-O	sea0-O	sine-O
DNN	ADWIN	61.65	71.94	78.45	71.12	97.70	79.27	59.36	69.71	84.41	65.95	95.18	86.59
	DDM	61.29	71.02	**80.72**	71.37	96.86	**86.41**	56.79	69.51	83.03	69.35	93.55	80.32
	EDDM	62.13	70.39	78.01	65.21	96.89	75.34	**61.17**	69.22	71.16	67.57	92.18	76.54
	HDDM-A	62.70	71.16	76.70	68.97	97.84	81.65	59.13	68.98	84.27	67.81	95.11	**86.82**
	HDDM-W	61.50	71.23	77.07	68.74	97.84	**85.43**	**62.42**	68.48	84.32	66.89	92.24	86.57
	KSWIN	63.02	70.56	78.58	70.80	97.88	79.00	**61.34**	69.08	**84.43**	66.46	91.60	83.66
	PH	62.15	72.25	79.49	70.86	97.73	78.07	60.36	68.24	84.36	68.83	**95.21**	**86.88**
	PUDD-1	**63.31**	**74.92**	77.39	**72.25**	**97.94**	**86.19**	60.90	69.35	82.65	**71.47**	94.89	83.39
	PUDD-3	**63.21**	**74.93**	**80.05**	**72.23**	**98.04**	85.12	60.16	68.98	**84.65**	**70.37**	**95.99**	84.97
	PUDD-5	**63.35**	**74.92**	**82.81**	**72.24**	**98.23**	82.51	60.19	68.68	**84.90**	**70.20**	**96.29**	85.09
GNB	ADWIN	50.17	68.90	**83.95**	70.06	94.18	82.49	**54.66**	**68.30**	83.62	68.78	93.93	82.45
	DDM	52.94	67.75	83.82	69.63	93.97	82.07	52.43	67.60	**83.59**	67.52	93.48	81.90
	EDDM	**62.72**	67.73	83.19	70.04	94.06	83.12	54.11	67.64	74.65	**70.04**	**94.06**	73.88
	HDDM-A	52.80	67.73	83.92	70.87	94.28	83.25	**55.62**	**67.73**	83.66	**71.24**	93.96	**83.36**
	HDDM-W	48.66	67.73	83.91	71.06	92.11	82.83	48.62	67.71	83.60	69.53	91.63	83.25
	KSWIN	49.84	67.87	83.92	71.23	91.72	81.88	48.83	67.63	83.59	67.99	90.02	81.32
	PH	49.35	70.12	83.88	70.36	94.34	**83.54**	49.02	**70.04**	83.61	68.67	94.12	83.10
	PUDD-1	**53.57**	**70.85**	82.99	**71.88**	**94.61**	83.12	51.05	62.76	79.23	**71.13**	**94.25**	81.48
	PUDD-3	**53.03**	**70.85**	**83.92**	**71.59**	**94.81**	**83.39**	49.45	59.32	83.58	**71.20**	**94.60**	**83.80**
	PUDD-5	52.16	**70.69**	**84.12**	**71.59**	**94.85**	**83.43**	**54.37**	59.44	**83.96**	70.40	**94.62**	**83.38**
VFDT	ADWIN	60.39	**73.98**	84.30	71.69	94.77	87.11	61.22	**74.29**	83.54	68.78	92.96	**85.74**
	DDM	60.16	**74.82**	84.14	70.68	94.86	86.50	59.28	**74.75**	82.31	67.53	93.63	82.23
	EDDM	61.19	73.81	83.15	70.06	94.17	85.59	**62.31**	73.81	73.73	70.06	93.60	77.72
	HDDM-A	60.95	73.90	**84.40**	70.84	**95.20**	**87.53**	60.29	73.83	**83.66**	**71.24**	93.93	85.21
	HDDM-W	61.11	73.80	**84.40**	70.99	93.50	**87.45**	**61.92**	73.73	83.59	69.54	91.76	85.28
	KSWIN	**61.30**	**74.10**	**84.42**	71.27	93.40	86.22	**62.06**	**74.10**	83.57	67.51	89.84	82.60
	PH	60.95	73.70	83.56	70.88	94.69	87.15	60.97	73.99	83.30	68.67	93.85	85.19
	PUDD-1	61.38	73.86	84.25	**71.77**	95.13	87.33	61.16	69.79	82.13	**71.13**	**94.10**	82.21
	PUDD-3	**61.57**	73.84	83.94	**71.79**	**95.21**	87.42	59.90	69.79	84.04	**71.20**	**94.63**	**85.81**
	PUDD-5	**61.57**	73.64	84.01	**71.79**	**95.24**	**87.63**	57.04	71.83	**84.16**	70.40	**94.56**	**86.01**

Table 2: Comparative analysis against classic drift detectors across 3 synthetic and 3 real-world datasets. The top 3 results are highlighted in bold and the top 1 results are in both bold and underlined. PUDD-x represents the threshold set as 10^{-x} for our method. The ps is short of powersupply dataset. Results for dataset sea10 and sea20 is provided in Appendix.

and Fernandes 2017)) and 5 SOTA methods (MCDD(Wan, Liang, and Yoon 2024), AMF(Mourtada, Gaïffas, and Scornet 2021), IWE(Jiao et al. 2022), NS(Wang et al. 2021), and ADLTER(Wang et al. 2022)).

Comparison with Baselines and Ablation Studies

In this subsection, we compare our method with 7 classic drift detectors and 5 SOTA methods on 9 datasets (including a variant of the SEA dataset). Due to page constraints, results for SEA10 and SEA20 appear in the Appendix. We evaluate all methods using three classifiers—DNN (architecture detailed in the Appendix), Gaussian Naive Bayes (GNB) (Virtanen et al. 2020), and VFDT (Hulten, Spencer, and Domingos 2001)—under two training regimes: incremental (dataset-I) and one-time training at an alarm (dataset-O).

Results for the comparison with classic detectors are presented in Table 2, and those for SOTA methods are given in Table 3. For CIFAR-10-CD, due to its learning complexity, we only use incremental training and report results in Fig. 3. Our method is denoted as **PUDD-X**, where X represents the exponent in 10^{-X}. Based on these experiments, we derive 6 observations. We introduce 4 of them here and leave the remaining 2 in the Appendix.

Observation 1: our method shows stronger performance compared to classic drift detectors as evidenced by the results presented in Table 2, Fig. 3, and additional results in Appenidx. In incremental learning settings, PUDD ranks first in 17 out of 24 cases across different datasets and classifiers, and it is in the top 3 in 20 out of 24 cases. When trained only initially or with adaptation, it still performs well. It achieves first rank in 15 cases and top 3 in 19 cases. This shows that PUDD is particularly effective with incremental training. Results in Fig. 3 show PUDD outperformed all the baselines, which demonstrates the superiority of our method in detecting the concept drift in the image dataset. The critical difference diagram of the experiment in the Appendix shows that the PUDD is statistically significantly outperforms SOTA methods.

Observation 2: PUDD performs better with a smaller threshold as revealed in Table 2 and additional results in Ap-

ddm name	airline	elec2	mixed	ps	sea0	sine
AMF	38.56	66.24	49.49	69.63	93.67	49.52
IWE	38.02	68.90	49.47	64.10	93.14	49.51
NS	**67.91**	76.42	81.09	72.39	93.54	91.01
ADLTER	**70.00**	76.10	87.63	72.48	93.40	**92.18**
MCD-DD	63.65	69.81	86.68	71.66	97.66	90.21
PUDD-1	63.78	**77.28**	**89.51**	**72.68**	**98.47**	**94.52**
PUDD-3	**64.62**	**76.77**	**89.47**	**72.79**	**98.44**	**94.76**
PUDD-5	64.45	**76.92**	**89.37**	**72.74**	**98.49**	90.90

Table 3: Comparison with SOTA methods. The dataset ps is short for powersupply. The results for dataset sea10 and sea20 is provided in Appnedix. The table shows that our methods PUDD in achieved top-1 in 5 out of 6 datasets, implying the effectiveness of PUDD compared with SOTA methods.

pendix. In incremental learning scenarios, PUDD-1, PUDD-3, and PUDD-5 achieve top 1 in 5, 5, and 8 cases respectively. When tested in training only once until alarm way, PUDD-1, PUDD-3, and PUDD-5 achieved top 1 in 2, 6, and 8 cases respectively. PUDD consistently shows improved performance at lower thresholds in both scenarios. As detailed in the Sensitivity of PU-index section, a drift alarm triggers when the p-value is below the threshold, with lower thresholds indicating stricter conditions for alarm detection. Therefore, PUDD's better performance with smaller thresholds suggests a high sensitivity to drift.

Observation 3: PUDD shows very competitive performance compared to SOTA methods. As shown in Table 3 and additional results in Appendix, our method attains the top rank in 7 out of 8 cases. On certain datasets, this improvement is particularly pronounced. For instance, PUDD-5 achieves a 98.49% accuracy, which is 2.8% higher than the best SOTA method. The only exception occurs in the airline dataset, where NS and ADLTER outperform PUDD.

This discrepancy can be explained by the airline dataset's tabular nature and its numerous attributes, which are more effectively modeled through tree-based ensemble learning utilized by these SOTA methods. Moreover, these methods adapt to drift by adjusting ensembles rather than discarding and retraining them. In contrast, PUDD relies on retraining classifiers solely on recent data, which may not be suitable for attribute-rich datasets like the airline dataset. Nevertheless, for all other datasets, the results confirm that PUDD surpasses SOTA methods, thereby underscoring its overall superiority.

Observation 4: The Adaptive PU-index Bucketing algorithm outperforms Ei-kMeans. Figure 4 shows that PUDD surpasses Ei-kMeans across various datasets, classifier training methods, and threshold settings. The critical difference diagram in the Appendix shows that improvements at thresholds 10^{-3} and 10^{-5} are statistically significant.

In summary, these results confirm the theoretical benefits of the PU-index for drift detection. PUDD outperforms both classic and SOTA detectors, and the Adaptive PU-index Bucketing algorithm shows significant improvements over Ei-kMeans. This validates the PU-index as a sensitive, robust indicator capable of detecting drift even when error rates remain unchanged, thereby overcoming a major shortcoming of error rate-based approaches.

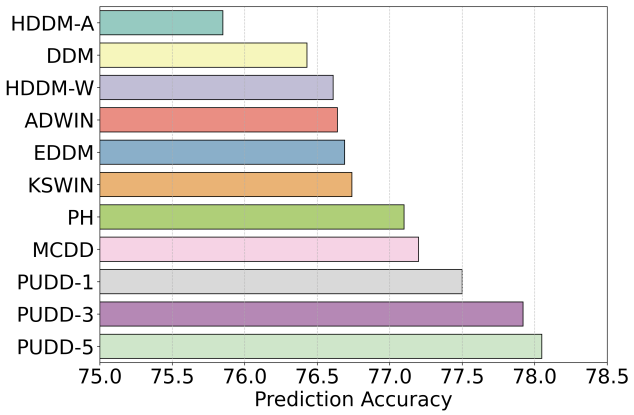

Figure 3: Comparison with baselines on CIFAR-10-CD, excluding methods unable to detect drift in image datasets.

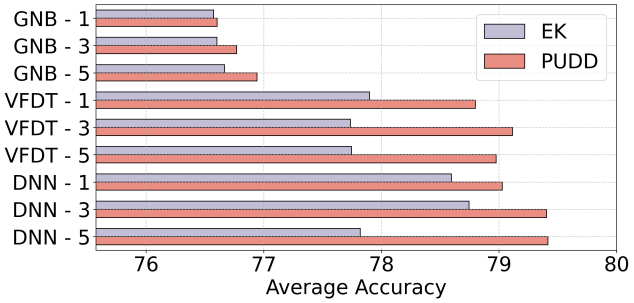

Figure 4: Accuracy comparison between PUDD (using Adaptive PU-index Bucketing) and Ei-kMeans (EK). We show average accuracy across 9 datasets using 3 classifiers.

Conclusion and Future Work

In our study, we demonstrated that the PU-index, as opposed to the error rate, is a more effective measure for detecting concept drift in machine learning models. We utilized the Adaptive PU-index bucketing algorithm to partition the PU-index and the Chi-square test to detect concept drift. We also introduced a technique for inducing concept drift in image datasets by simulating changes in user interest. We validated our method through experiments on both synthetic and real-world datasets. Future work should focus on automating drift alarm threshold determination, as current methods rely on manual settings that may not remain optimal over time. Our research also uncovers a method for generating multi-stream concept drift in image datasets by emulating shifts in user interests using Markov matrices, offering valuable insights for research in multistream concept drift learning.

Acknowledgments

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

References

Baena-Garcıa, M.; del Campo-Ávila, 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*, volume 6, 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.

EP, B. G. 1978. Statistics for Experimenters: An Introduction to Design. *Data Analysis, and Model Building*, 43–48.

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.

Huggard, H.; Koh, Y. S.; Dobbie, G.; and Zhang, E. 2020. Detecting concept drift in medical triage. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1733–1736.

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)*.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4): 917–963.

Jiang, M.; Wang, Z.; and Dou, Q. 2022. Harmofl: Harmonizing local and global drifts in federated learning on heterogeneous medical images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 1087–1095.

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.; 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.; Liu, A.; Song, Y.; and Zhang, G. 2020. Data-driven decision support under concept drift in streamed big data. *Complex & intelligent systems*, 6(1): 157–163.

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.

Miyaguchi, K.; and Kajino, H. 2019. Cogra: Concept-drift-aware stochastic gradient descent for time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4594–4601.

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.

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.

Silverman, B. W. 2018. *Density Estimation for Statistics and Data Analysis*. Monographs on statistics and applied probability. Boca Raton, FL: CRC Press. ISBN 1-351-45617-2.

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.

Souza, V. M.; Parmezan, A. R.; Chowdhury, F. A.; and Mueen, A. 2021. Efficient unsupervised drift detector for fast and high-dimensional data streams. *Knowledge and Information Systems*, 63: 1497–1527.

Stucchi, D.; Rizzo, P.; Folloni, N.; and Boracchi, G. 2023. Kernel quanttree. In *International Conference on Machine Learning*, 32677–32697. PMLR.

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.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

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*.

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.; Zhang, B.; and Zhang, G. 2024. Online Boosting Adaptive Learning under Concept Drift for Multistream Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16522–16530.

Yuan, L.; Li, H.; Xia, B.; Gao, C.; Liu, M.; Yuan, W.; and You, X. 2022. Recent Advances in Concept Drift Adaptation Methods for Deep Learning. In *IJCAI*, 5654–5661.

Zhou, M.; Lu, J.; Song, Y.; and Zhang, G. 2023. Multi-Stream Concept Drift Self-Adaptation Using Graph Neural Network. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12828–12841.