IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 November 2023, accepted 17 January 2024, date of publication 26 January 2024, date of current version 13 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3358817

SURVEY

Roadmap of Concept Drift Adaptation in Data Stream Mining, Years Later

OSAMA A. MAHDI[1], NAWFAL ALI[2], ERIC PARDEDE[3], (Senior Member, IEEE), AMMAR ALAZAB[4], TAHSIEN AL-QURAISHI[5], AND BHAGWAN DAS[1], (Senior Member, IEEE)

[1]School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, VIC 3000, Australia
[2]Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia
[3]School of Computing, Engineering and Mathematical Sciences, La Trobe University, Melbourne, VIC 3086, Australia
[4]Centre for Artificial Intelligence and Optimization, DCT, Torrens University, Sydney, NSW 2007, Australia
[5]Higher Education Department, Victorian Institute of Technology (VIT), Melbourne, VIC 3000, Australia

Corresponding author: Osama A. Mahdi (omahdi@mit.edu.au)

This work was supported by the Melbourne Institute of Technology (MIT).

ABSTRACT As machine learning models are increasingly applied to real-world scenarios, it is essential to consider the possibility of changes in the data distribution over time. Concept drift detection and adaptation refers to the process of identifying and tracking these changes and updating the model accordingly. Researchers have devoted significant efforts to develop various techniques and tools for concept drift detection and adaptation, as this paper provides a generic roadmap and review of the field. In this paper, we begin by reviewing the background of data stream classification and its assumptions and requirements. Then, we explore the historical development of concept drift detection and adaptation and highlight the key points of approaches that have emerged over time. Next, we summarize the major findings, challenges, and limitations of past research, and provide insights into potential future directions of the field. The paper can benefit researchers and practitioners who seek to navigate the challenges and opportunities in concept drift detection and adaptation.

INDEX TERMS Concept drift, data stream, non-stationary environments.

I. INTRODUCTION

Concept drift in machine learning refers to a scenario in which the statistical characteristics of the target variable, which the model aims to predict, change over time [1], [2], [3]. This implies that the meaning of the input data on which the model was originally trained has significantly altered. However, the model remains unaware of these changes and, as a result, may not produce accurate predictions.

The sudden changes in human behavior caused by the COVID-19 pandemic represent a prime example of concept drift in action. To understand how the pandemic-related concept drift may affect predictive models, let's consider an example from Melbourne City, where the COVID-19 quarantine has impacted various aspects such as shopping behavior, electricity usage, and motor vehicle collisions. As an illustration, let's examine the number of future motor vehicle collisions prediction. VicRoads is responsible for reporting all accidents involving injuries, deaths or damages of at least \$1,000. During the pandemic, the annual collision count was significantly lower compared to previous years, which was a result of the lockdown and business closures. Furthermore, if we consider electricity usage, the data patterns before and after the pandemic are not the same (as life returns to normal). Therefore, any predictive model created before the pandemic that assumes a specific relationship between inputs and outputs will perform poorly due to changes in the underlying data patterns. To further illustrate the phenomenon of concept drift, consider the following scenario in **Table 1a** and **Table 1b**. Suppose we are attempting to predict whether a customer will purchase a product based on their age and salary, and we train a logistic regression model using historical customer data from January to June (Table 1a). This example highlights how changes in data over time can lead to concept drift, potentially impacting the accuracy of the model's predictions. Imagine a model that initially performs well on the training data, achieving an accuracy of 80%. However, suppose in July, there's a shift in the demographics of

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

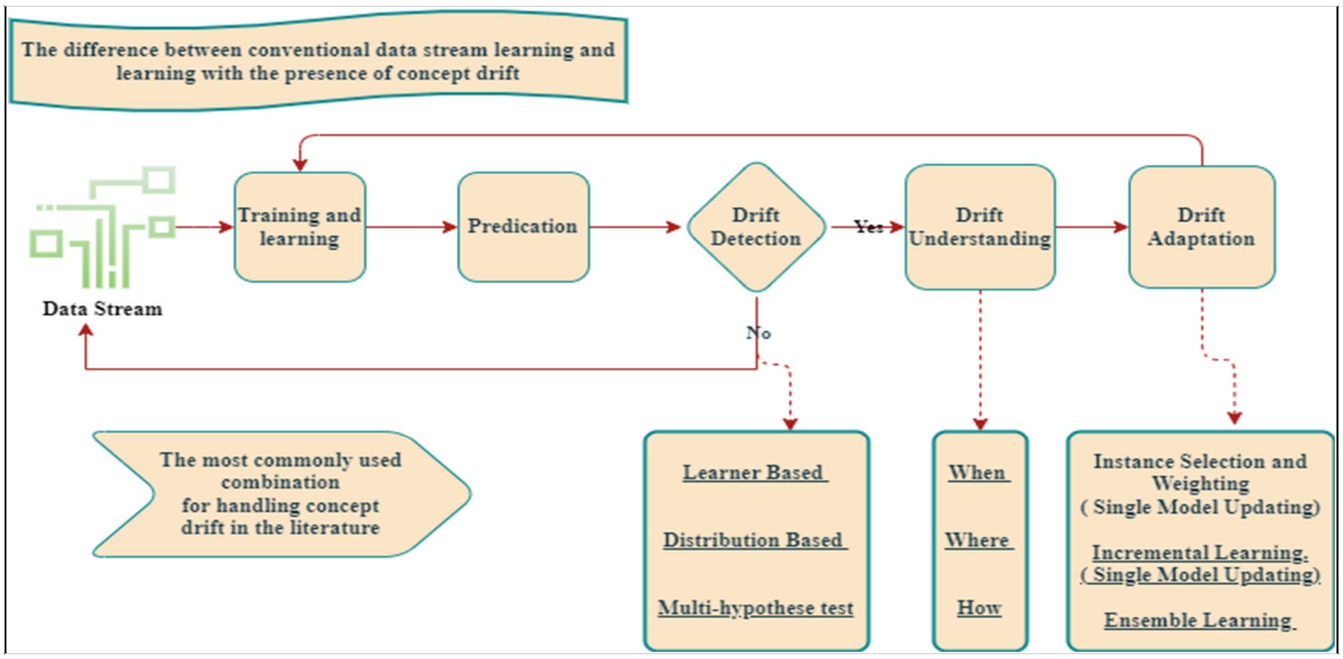

The difference between conventional data stream learning and learning with the presence of concept drift
Data Stream
Training and learning
Predication
Drift Detection
Yes
No
Drift Understanding
Drift Adaptation
The most commonly used combination for handling concept drift in the literature
Learner Based
Distribution Based
Multi-hypothese test
When
Where
How
Instance Selection and Weighting (Single Model Updating)
Incremental Learning. (Single Model Updating)
Ensemble Learning

FIGURE 1. A general framework of concept drift handling [1].

TABLE 1. Illustration of concept drift.

(a)

Age	Salary	Purchased
35	50000	Yes
42	60000	No
29	80000	Yes
36	55000	No

(b)

Age	Salary	Purchased
40	65000	No
45	75000	No
33	90000	Yes
38	68000	No

our customer base, leading to an increase in both the average age and salary (as shown in Table 1b). If we were to use the same model, which was trained on the prior data, to predict purchases for this new customer profile, we would likely see a drop in performance. This is an example of concept drift - the statistical properties of the target variable have changed over time, causing the model to become outdated and perform poorly on new data.

Concept drift is a widely occurring phenomenon that affects many different applications. It has been identified and addressed in various fields, including medicine, industry, education, and business. In the medical field, the effectiveness of antibiotics can decrease over time as microorganisms become resistant. If an unnecessary use of antibiotics leads to resistance, it may render antibiotics useless when they are needed. Changes in the drug being used can trigger changes in disease progression. In finance, bankruptcy prediction or credit scoring are usually considered to be stable problems, but latent factors such as social demands and movements can cause concept drift. In industrial monitoring applications, changes in production or service monitoring and user behavior can also result in concept drift. In transportation, traffic management systems use data mining to monitor traffic states like car density and accidents, which are subject to change due to seasonal or permanent changes in traffic patterns. Thus, these systems must be able to handle concept drift [1]. The phenomenon of concept drift has been explored in numerous disciplines such as data mining, machine learning, statistics, and information retrieval [4]. Notably, this phenomenon might be identified by different terminologies depending on the field. **Table 2** lists the diverse terms associated with the concept drift phenomenon across different research fields.

TABLE 2. Concept drift terminology.

Field	Terminology
Data Mining	Concept Drift
Machine Learning	Concept Drift, Covariate Shift
Evolutionary Computation	Changing Environment
AI and Robotics	Dynamic Environment
Statistics, Time Series	Non-Stationary
Databases	Concept Drift, Load Shedding
Information Retrieval	Temporal Evolution

II. CONTRIBUTION AND INNOVATION IN THE RESEARCH COMMUNITY OF CONCEPT DRIFT

Traditional machine learning involves two key elements: training and prediction. However, research on machine learning with concept drift has added three new aspects: detecting the presence of drift (concept drift detection), comprehending when, where, and how the drift occurred (drift understanding), and adapting to the drift (drift adaptation). The most prevalent methods/ techniques for handling streaming data with concept drift are depicted in **Figure 1**. Over the past decade, many scientific studies have focused on the topic of concept drift, with recent research specifically exploring *how to detect it accurately* [5], [6], [7], *understand it effectively*, and *adapt related knowledge in response to* [8] and [9], in order to ensure that prediction and decision-making are adaptable in an environment with concept drift. However, stream learning poses additional challenges due to time and storage limitations and balancing computational cost with learning accuracy is crucial for practical applications. Thus, it is important to find ways to efficiently handle concept drift in real-world situations and applications [10], [11], [12] that have limited computational resources and time, as accuracy is not the only factor in determining the effectiveness of learning models.

The groundbreaking findings significantly advance studies in artificial intelligence and data science, especially in the realms of pattern recognition and data stream mining. Furthermore, a recent technical paper from Berkeley [13] identifies acting in continuous learning and dynamic settings as one of the nine pivotal research avenues to tackle present AI research obstacles. Therefore, this study centers on examining the primary research issues associated with detection methods and adaptive ensembles for evolving data streams.

III. DATA STREAM CLASSIFICATION

Data stream classification involves building a model using accessible data (namely, the training data) to forecast the labels of previously unobserved instances. The formal description of data stream classification is presented as follows:

Consider a stream S that consists of a series of instances like (x_1, y_1), (x_2, y_2), $\ldots$,(x_t, y_t), appearing in sequence over time. Each pair (x_t, y_t) represents an instance at a specific moment t. Here x_t is a vector containing values for k attributes, represented $x = (x_1, x_2, \ldots, x_k)$, and y_t denotes a class label from a set of m class labels, indicated as $y_t \in \{c^1, c^2, \ldots, c^m\}$. Imagine a target function $y_t = f(X_t)$ that associates an input vector with its respective class label. Typically, the objective in learning is to successively develop a mode $\tilde{f}$ that comes close to mirroring function f as each instance is handled. Achieving an approximation that heightens classification accuracy is crucial. Regarding data stream classification, there are specific assumptions regarding the behavior of data streams. Alongside these assumptions, we also touch upon four core prerequisites every data stream classification endeavor should adhere to, which are detailed further in subsequent subsections.

A. ASSUMPTIONS

Broadly speaking, there are six primary assumptions commonly accepted in the realm of data stream classification as indicated in the literatures [1] and [14]. These assumptions include:

- A data stream is characterized by a consistent set of attributes. Too many attributes can hinder the learning procedure and elevate memory consumption.
- The overall count of instances or records is significantly larger compared to the number of attributes. Most learning algorithms are believed to handle potentially limitless data without exhausting memory resources.
- It's preferable to have a limited number of class labels. An increased number of class labels necessitates more statistics to craft a classification model. Given that these statistical values are constantly refreshed over time, processing a data stream with numerous classes becomes computationally demanding. This demand escalates linearly with the class count in terms of computational complexity.
- Typically, the volume of the data surpasses available memory. As a result, it's im-practical to load the entirety of the data into memory.
- Learning algorithms are expected to navigate both the training and testing stages al-most instantaneously, given the rapid influx of data stream instances.
- The learning paradigm is either perceived as stable or ever-changing. When there's a shift in the foundational data distribution, it's termed concept drift.

The initial trio of assumptions delves into the behavior and characteristics of data streams. In contrast, the final three highlight the attributes and requirements learning algorithms should exhibit when classifying against these data streams.

B. REQUIREMENTS

The main obstacles in data stream classification arise from the limited computational resources and the occurrence of concept drift. To make learning from data streams viable, classification algorithms need to satisfy these four essential criteria as outlined in [1] and [15]:

- **Requirement One (R1): Process each instance individually and review it a single time** – Recall which instances of a data stream arrive one after another and they are handled only once in the order of appearance. In other words, random access to the instances is not doable. An instance is thrown away once it has been handled. While this is a vital necessity for data stream mining, a learning algorithm can internally store instances for a short time for additional usage without violating Requirement 2.
- **Requirement Two (R2): Conserve memory usage** - The rationale behind incrementally training classification models is that data size often surpasses the capacity of accessible memory. In essence, vast volumes of data can't be accommodated within constrained memory space. Therefore, it's crucial to set a maximum limit on memory consumption to prevent possible memory depletion. Typically, a learning algorithm may utilize the primary memory to keep the existing model.

- **Requirement Three (R3): Process an instance within a limited timeframe -**Learning algorithms need to address instances almost immediately upon their arrival. This means they must process data faster than the rate at which it comes in. Failing to keep up will inevitably lead to missed information. Therefore, a maximum limit should be established concerning the time dedicated to handling each instance.
- **Requirement Four (R4): Always be ready to make predictions -**An ideal learning algorithm creates a classification model that can swiftly determine the label of new, unseen instances. This readiness to predict signifies that a class label can be provided at any given moment, a crucial aspect for data stream classification.

C. THE CYCLE OF DATA STREAM CLASSIFICATION

The data stream classification cycle is depicted in **Figure** 2. This cycle comprises three phases: processing, learning, and utilizing [16], [17]. The details of these phases are as follows:

- **Processing**: This phase aligns completely with Requirement 1, wherein instances from the data stream are readied and then forwarded to the learning phase.
- **Learning**: The learning algorithm refines its prediction model by training with every new instance. It also ensures that it stays within the constraints of Requirements 2 and 3 by not surpassing the memory or processing time limits.
- **Utilizing**: The model is employed to determine the class labels of previously unobserved instances. Fulfilling Requirement 4 involves ensuring the model is always primed for prediction.

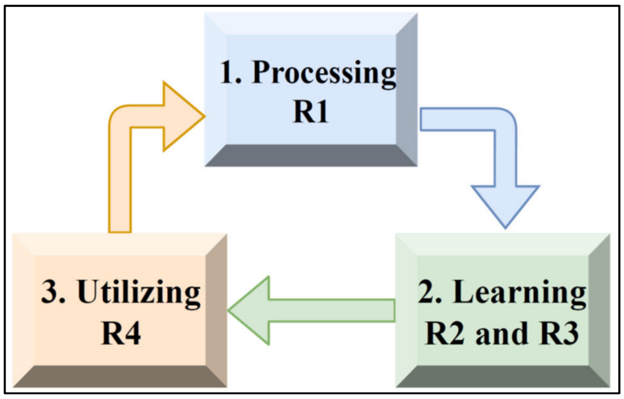


FIGURE 2. The cycle of data stream classification [1].

IV. ADAPTIVE CLASSIFICATION

A. CONCEPT DRIFT PHENOMENON

In non-stationary environments, the distribution of data may shift over time, resulting in $D_i \neq D_j$ for any two time points i and j. This makes the notion of two points unstable and the model unable to accurately capture the latest data distribution. Therefore, a central obstacle in analyzing streaming data is identifying notable shifts in the incoming data. Moreover, variations in the definitions and distributions of incoming data can influence the classification precision of a model trained on past data. The detailed explanation of concept drift will be addressed in the following subsections.

Formal Definition

An important feature of data streams is their dynamic nature, which requires classifiers to anticipate, detect, and adapt to changes in concepts. To accomplish this, changes in type, frequency, source, predictability, and impact must be assessed [18].

Bayesian decision theory [19] proposes that a classification model is defined by prior probabilities of classes $p(y)$ and class conditional probabilities $p(y|x)$ for all predefined classes $y \in K_1, \ldots, K_c$ (where c is the number of classes). The nature of data streams is characterized by changes in these probability distributions, which is known as concept drift, occurring after a period of stability [2]. In various domains, concept drift might also be known as temporal evolution, population drift, covariate shift, or non-stationarity. The majority of research indicates that concept drift is not easily predictable, in contrast to seasonal variations. However, certain adaptive strategies can foresee shifts tied to environmental elements. The formal definition of concept drift is as follows:

Definition: For a specified data stream S, if there's a change in concept between the time points t and t $+\Delta$, *it happens if and only if there's some x such that* $p^t(x,y)$ *is not equal to* $p^{t+\Delta}(x, y)$. *Here* p^t *denotes the joint distribution at time t linking the collection of input attributes to the class label.*

Taking this into account, modifications in incoming data can be identified as alterations in the constituents of Bayesian decision theory [4], [20], [21]:

- **Prior probabilities** $p(y)$ are susceptible to alterations.
- **Probabilities** $p(x|y)$ of class conditional are likewise susceptible to alterations.
- As a **result**, **posterior probabilities** $p(y|x)$ might either change or remain the same.

Considering the reasons and outcomes of these shifts, two primary forms of drift are recognized: real drift and virtual drift [18], as depicted in **Figure 3**.

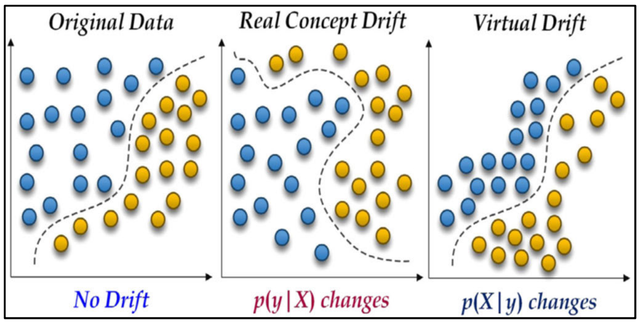


FIGURE 3. Two variations of drift: where instances are depicted as circles with distinct colors [1].

Real drift refers to alterations in $p(y|x)$. It's important to note that such changes can occur irrespective of changes in p(i), making them undetectable from the data distribution without knowledge of the true class labels. This differentiation is crucial, as some techniques aim to detect concept drift by solely examining attribute values [22]. Real drift is also known as concept shift [23] or conditional change [20]. Example of Real drift, suppose a financial institution is using a machine learning model to detect fraudulent transactions, and over time, the characteristics of fraudulent transactions change, such that the model's predictions become less accurate. In this scenario, the changes in the conditional probability distribution $p(y|x)$ can be classified as real drift.

Virtual drift typically denotes shifts in the distribution of attribute-value $p(x)$ or class $p(y)$ that do not influence $p(y|x)$ [24], [25], [26]. However, the root and understanding of such modifications differ among scholars. Widmer and Kubat [26] suggested that virtual drift stems from an incomplete representation of data, rather than genuine conceptual shifts. In contrast, Tsymbal [24] characterized virtual drift as shifts in data distribution that modify the decision boundary, while Delany et al. [25] viewed it as a drift that doesn't influence the core concept. Moreover, virtual drifts have also been labeled as transient drifts [27], sampling shifts [23], and feature changes [20]. To illustrate virtual drift, consider a model trained to recognize cats in photos, using a dataset featuring only black and white cats. In real-world scenarios, there might be pictures of cats with varied fur colors. If the model struggles with identifying these newly introduced cats, this discrepancy can be traced back to virtual drift in the distribution $p(x)$ or class distribution $p(y)$, which doesn't impact the conditional probability $p(y|x)$.

To better elucidate the distinction between real and virtual drift, let's delve into an example, which is elaborated upon in Table 3. Suppose a company has built a machine learning model to predict customer churn based on customer demographics and purchasing behavior. After a few months, the company introduces a new product line, which results in changes to the purchasing behavior of some customers. The model's accuracy starts to decline as a result, and the company wants to know whether this is due to real or virtual drift.

Ultimately, when a real concept drift is detected, it becomes necessary to adapt the decision model to the new incoming data as the current decision boundary becomes outdated. This adaptation process involves updating the classification model to maintain high classification accuracy for the new distribution.

B. CONCEPT DRIFT PATTERNS

Aside from variations in the causes and consequences of concept changes, several methods have been identified by researchers to further characterize such changes. These methods include analyzing the permanence, severity, predictability, and frequency of drifts. However, the manifestation of drifts over time is the most extensively analyzed aspect [2], [24], [28], [29]. **Figure 4** illustrates three fundamental structural patterns of changes that could occur over time.

TABLE 3. Difference between real and virtual drift.

Type of Drift	Example	Description
Real Drift	Changes in purchasing behavior	In this scenario, changes in the purchasing behavior of some customers can be classified as real drift. This is because the change in purchasing behavior affects the relationship between the customer demographics and the likelihood of churn. In other words, the conditional probability distribution $p(y \mid x)$.
Virtual Drift	Change in feature space	If, on the other hand, the company decided to add a new demographic feature (e.g., occupation) to the model, but the new feature had no relationship to customer churn, this would be an example of virtual drift in the feature space. The attribute-value distribution $p(x)$ would change, but the underlying relationship between the customer demographics and churn would remain the same. As a result, the conditional probability distribution $p(y \mid x)$.

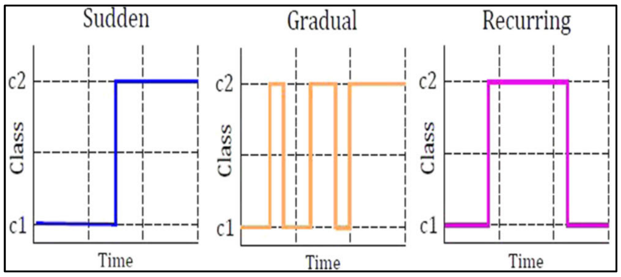


FIGURE 4. Concept drift patterns.

To start with, sudden or abrupt drift is characterized by a sudden substitution of the source distribution in S_t at time t with another distribution S_{t+1}. This causes the classification accuracy of a classifier to reduce as the new distribution is used for training. On the other hand, gradual drift happens at a slower rate and involves a transition stage where examples from two different distributions P_j and P_{j+1} are mixed. As time passes, the likelihood of monitoring P_j examples decreases, while that of P_{j+1} examples increases. Recurrent drift refers to a situation where previous concepts may reappear after some time. Recent approaches to detect these three types of drifts include abrupt drift, gradual drift, and recurring drift. It is worth mentioning that the proposed concept drift detection method primarily addresses sudden or abrupt drift.

V. CONCEPT DRIFT DETECTION METHODS

In this section, a thorough evaluation of current methods for detecting concept drift is presented.

The objective of this thorough evaluation is to offer a comprehensive overview of Concept Drift Detection Methods. Our literature search was conducted during my PhD candidature, utilizing established academic databases. The search process was steered by keywords such as "concept drift", "non-stationary environments", and "data stream mining". From this effort, pertinent papers were analyzed to trace the evolution of methodologies and techniques, establishing a framework for learning under concept drift. These papers were then organized into three principal categories: Statistical-based Methods, Windows-based Methods, and Ensemble-based Methods, as depicted in **Figure 5**. Our analysis encompasses the fundamentals of data stream classification, its underlying assumptions, and requirements. We delved into the historical progression of concept drift detection and adaptation, emphasizing the salient aspects of the methods that have surfaced over the years. Subsequently, we collate the significant findings, challenges, and constraints observed in previous research and offer perspectives on the potential trajectories for future endeavors in the field.

A. STATISTICAL-BASED METHODS

The Statistical-based Methods are employed to observe the progress of the learning process by tracking the online error rate changes of base learners. If the model's decline in performance goes beyond the level of significance testing, it is believed that concept drift has occurred. The basic procedure of using Statistical-based Methods is presented in Algorithm 1.

Algorithm 1 Generic Schema of Statistical-Based Methods

```
    Input: $S$, Data Stream of Examples
    Drift Test (DT): Using Statistical Tests or
    Mathematical Inequalities.
    C: Classifier
    Output: Drift ∈ {TRUE, FALSE}
1.  Initialize (Parameters)
2.  For each example $x^t$ ∈ $S$ do
3.      Measure DT;
4.      If drift detected,
        then
5.          Return TRUE
6.      Else
7.          Return TRUE
8.      End if
9.  End for
```

Algorithm 1 describes a data stream learning process that utilizes a drift test to detect concept drift in the data stream. The algorithm takes a data stream of examples S as input and a pre-trained classifier C. The output of the algorithm is a Boolean value indicating whether drift has been detected or not. The steps of the algorithm are as follows:

1) Initialize Parameters: The algorithm initializes the parameters required for the drift detection process.
2) For each example x^t ∈ S do: The algorithm iterates over the data stream examples.
3) Measure DT: The algorithm measures the drift test to check if the current example is consistent with the concept the classifier has learned so far. This can be done using statistical tests or mathematical inequalities. If the current example is consistent with the learned concept, the drift test will return a value indicating that no drift has been detected.
4) If drift detected, then: If the drift test returns a value indicating that drift has been detected; otherwise, return false.

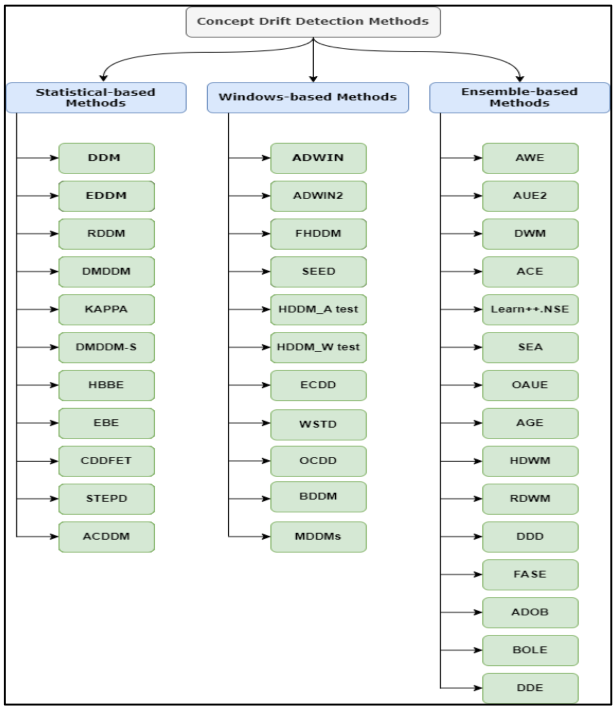


FIGURE 5. Concept drift detection methods.

In summary, the algorithm processes the data stream examples one by one, measures the drift test for each example, and outputs a Boolean value indicating whether drift has been detected or not. If drift is detected, the algorithm can trigger re-training of the classifier or other actions to handle the concept drift.

There are several techniques available in the literature for detecting concept drift in data streams, each with their own strengths and weaknesses. DDM (Drift Detection Method) is a widely used method that monitors the classifier's error rate [30]. EDDM (Early Drift Detection Method) is a modification of DDM that uses the distance error rate instead of the classifier's error rate [31]. RDDM (Reactive Drift Detection Method) was introduced to address the issue of performance degradation in DDM by incorporating a mechanism to discard older examples and periodically update the statistics used for drift detection [32], [33]. DMDDM (Diversity Measure as a new Drift Detection Method) combines the disagreement measure with the Page-Hinkley test to quickly detect concept drift with less memory and time consumption [34]. KAPPA measures the level of agreement among different classifiers to detect concept drifts while minimizing computational resources [35]. DMDDM-S (Diversity Measure as a new Drift Detection Method for

Streaming Data without Class Labels) was designed to detect sudden drifts in the absence of class labels, monitoring the diversity of a pair of classifiers instead of error estimates [36]. HBBE (Hybrid Block-Based Ensemble) is a hybrid block-based ensemble for multi-class classification in evolving data streams, which integrates the strengths of an online drift detector for a k-class problem and the concept of block-based weighting to respond effectively to different types of drifts [37]. EBE (Entropy Based Ensemble) uses information entropy to detect concept drifts, incorporating entropy as a drift detector into the evolving ensemble to improve performance [38]. CDDFET (Concept Drift Detection based on Fisher's Exact Test) is a modification of STEPD that uses Fisher's Exact test to calculate the p-value, with three modified versions proposed for detecting drifts [39]. ACDDM (Accurate Concept Drift Detection Method) uses Hoeffding's inequality to analyze prequential error rate consistency for detecting concept drift in dynamic data streams [40].

B. WINDOWS-BASED METHODS

Typically, a static reference window is used to summarize previous information, while a dynamic sliding window is used to summarize the latest information. If there is a notable difference between the distributions of these two windows, it indicates a concept drift has taken place. The basic procedure of using Statistical-based Methods is presented in Algorithm 2, and different window-based methods are described below.

Algorithm 2 Generic Schema of Windowing-Based Methods

```
    Input: $S$, Data Stream of Examples
      $W$, Window of Examples
    Output: C: a classifier built on examples in window $W$
1.  Initialize window $W$
2.  For each example $x^t \in S$ do
3.      W ← W U {$x^t$}
4.      If necessary, remove outdated examples from $W$
5.      rebuild/update $C$ using $W$
6.  End for
```

Algorithm2 is designed to build a classifier C using a sliding window of examples in a data stream S. The algorithm takes S and a window size W as input and outputs a classifier C built on the examples in the window. The steps of the algorithm are as follows:

1) Initialize window W: The algorithm initializes a window W to store a fixed number of the most recent examples from the data stream.
2) For each example $x^t \in S$ do: The algorithm iterates over the data stream examples.
3) W ← W U {x^t }: The algorithm adds the current example to the window W.
4) If necessary, remove outdated examples from W: If the size of W exceeds the maximum window size, the algorithm removes the oldest examples from W to maintain a fixed window size.
5) Rebuild/update C using W: The algorithm rebuilds or updates a classifier C using the examples in the window W.

In summary, the algorithm maintains a fixed window of the most recent examples from the data stream and rebuilds or updates a classifier using the examples in the window. This allows the classifier to adapt to changes in the data over time.

Many methods have been proposed for detecting drift in data streams under this category. ADWIN (Adaptive Sliding Window) [41] is a technique that compares the means of two sub-windows, w_0 and w_1, of a larger window, w, to detect drift. If there is a significant difference in the means of the two sub-windows, ADWIN signals a drift and removes the tail of the window until the means become similar again. ADWIN2 [42] is an upgrade to ADWIN that uses a window of variable size to adapt to changes in the data distribution. FHDDM (Fast Hoeffding Drift Detection Method) [43] uses Hoeffding's inequality and a window of size n to detect drift by comparing the current probabilities to the highest level of correct predictions. SEED [44] also uses Hoeffding's inequality, but it applies Bonferroni correction and block compression to compare two sub-windows and remove the older portion of the window if there is a noticeable difference in the average. HDDM_A test and HDDM_W test [45] are two methods that use Hoeffding's bounds to detect drift. HDDM_A examines moving averages, while HDDM_W examines the weight of moving averages with weighting based on the exponentially weighted moving average (EMWA) forgetting scheme. ECDD (EWMA for Concept Drift Detection) [46] adapts the EWMA technique to detect drift by using weights to distinguish between recent and older instances. STEPD (Statistical Test of Equal Proportions) [47] uses two windows to evaluate the accuracy of a learner's performance and issues a warning if there is a substantial difference in accuracy within the recent window. WSTD (Wilcoxon Rank Sum Test Drift Detector) [32], [48] is similar to STEPD but uses the Wilcoxon rank sum statistical test to detect drifts. OCDD (One Class Drift Detector) [49] is an implicit concept drift detector that uses a sliding window approach to approximate the distribution of a new concept. It estimates the percentage of outliers to detect drift within the sliding window. BDDM (Bhattacharyya Distance Drift Detector) [50] monitors changes in the mean and variance over time using the Bhattacharyya distance to identify changes in distribution. MDDMs (McDiarmid Drift Detection Methods) [51] leverage McDiarmid's inequality to detect drift by monitoring the prediction results in a window of size n. Whenever a prediction is correct, a 1 is added to the window; otherwise, a 0 is added.

C. ENSEMBLE-BASED METHODS

Block-based approaches process data streams in portions called blocks. These methods often replace the weakest ensemble member with a new classifier after periodically evaluating their components. This strategy is effective at detecting gradual concept drift and maintaining high accuracy. Additionally, ensemble-based approaches combine the outcomes of multiple diverse base learners, and the overall performance can be monitored by assessing the accuracy of all the ensemble members or each individual base learner. The majority of ensemble-based detectors utilize the Weighted Majority Algorithm (WMA) [52] as their foundation. WMA selects the top-performing learners in the ensemble by assigning a weight to each learner based on its performance. The basic procedure of using the Ensemble-Based Method is presented in Algorithm 3, the subsequent sections elaborate on different block-based ensemble methods.

Algorithm 3 outlines a generic schema of an ensemble-based method for building a weighted classifier ensemble from examples of a data stream split into blocks. The algorithm takes as input the data stream split into blocks, the size of the block, the number of components in the ensemble, and a measure of classifier quality (Q). The output of the algorithm is a k weighted classifier ensemble. The steps of the algorithm are as follows:

Algorithm 3 Generic Schema of Ensemble-Based Method

```
   Input: $S$, Data Stream divided into blocks
   $d$, block's size
   $k$, number of the elements within an ensemble.
   (Q): a measure for classifier quality
   Output: $E$: $k$ weighted classifier (ensemble)
1. For each blocks $B^t \in S$ do
2.     using $B^t$ and $Q()$ to construct and assign weights to each proposed classifier $C'$
3.     Weight $C^i$ using $B^i$ and $Q(\ )$
4.     If $|E| < k$ then
5.         $E \leftarrow E \cup \{C'\}$
6.     Else
7.         Replace the least robust element in the ensemble with C
8.     End if
9. End for
```

1) The algorithm iterates over each block in the data stream (S).
2) For each block, the algorithm builds and weights each nominated classifier (C') using the block (B^t) and the measure of classifier quality (Q).
3) The algorithm weights each nominated classifier (C') using the block (Bi) and the measure of classifier quality (Q).
4) If the number of components in the ensemble ($|E|$) is less than k, then the algorithm adds the nominated classifier (C') to the ensemble (E).
5) If the number of components in the ensemble ($|E|$) is equal to or greater than k, then the algorithm substitutes the weakest component in the ensemble with the nominated classifier (C).
6) The algorithm continues iterating over each block in the data stream (S) until all blocks have been processed.

In summary, Algorithm 3 describes a generic schema of an ensemble-based method for building a weighted classifier ensemble from examples of a data stream split into blocks. The algorithm builds and weights nominated classifiers using each block in the data stream and a measure of classifier quality. If the number of components in the ensemble is less than k, the algorithm adds the nominated classifier to the ensemble. If the number of components in the ensemble is equal to or greater than k, the algorithm substitutes the weakest component in the ensemble with the nominated classifier. The algorithm iterates over each block in the data stream until all blocks have been processed.

Ensemble methods have become increasingly popular in handling concept drift in data streams. A number of different ensemble methods have been proposed, each with its own strengths and weaknesses. In this section, we discuss some of the most well-known and effective ensemble methods for concept drift handling.

The accuracy-weighted ensemble (AWE) method, proposed by [53], operates by first training a new classifier for each incoming data block, using a standard static learning algorithm. Once the new classifier is trained, it is evaluated along with all existing classifiers in the ensemble using the most recent block of data. Following the evaluation, the algorithm uses mean square error to select the top n classifiers and update the ensemble accordingly. The Accuracy Updated Ensemble (AUE2) method, proposed by [54], differs from AWE in that it employs an online classifier which can update individual learning models directly, rather than just adjusting their weights. If the system does not anticipate any concept drift, then the classifiers improve as if they were trained on a single large dataset, allowing the block size to be reduced without impacting the ensemble's accuracy. The Dynamic Weighted Majority (DWM) algorithm, proposed by [55], consists of a set of incremental classifiers, each of which is given a weight based on its accuracy after processing each incoming example. If a classifier makes a mistake, its weight is decreased by a user-defined factor β. Additionally, after a specified period of predictions p, the entire ensemble is evaluated and a new classifier is added to the ensemble if necessary. When trained on a large data stream, DWM can generate a large number of components, which is why ensemble pruning is often employed as an extension [2].

The Adaptive Classifier Ensemble (ACE), proposed by [56], utilizes a combination of an online learner and an ensemble of batch classifiers. It consists of one online classifier, several batch classifiers, and a drift detector. As each new data example arrives, the online classifier is trained incrementally, and the batch classifier is extended. Additionally, the drift detector assesses the average accuracy of each batch classifier (on the current block), and if the best-performing component falls outside a $100(1-\alpha)\%$ confidence interval (where α is a user-defined parameter), a change is detected.

Once a concept drift is detected, or the number of buffered examples exceeds the block size, a new batch classifier is created, and the online learner is reset. Learn++.NSE, proposed by [57], is an ensemble method based on human learning theory, which incorporates aspects of schema theory [58] that explains how humans learn and organize knowledge. This algorithm adapts its knowledge retention based on changes in the stream, similar to how humans construct, or discard knowledge based on new information. In addition, Learn++.NSE assigns weights to examples based on their difficulty in terms of ensemble performance. The Streaming Ensemble Algorithm (SEA) [59], one of the earliest works that uses ensembles to deal with concept drift, processes the data stream by dividing it into blocks of examples. Each block of data is used to train a new classifier, which is later compared with the existing ensemble members. If the candidate classifier outperforms any ensemble member, the latter is dropped and the former takes its place.

The Online Accuracy Updated Ensemble (OAUE) [16], makes use of a drift detector that is incorporated into an online learning system. This detector sends a signal to the learner to reweight the classifiers when a drift is detected. The Accuracy and Growth Rate updated Ensemble (AGE) [60] has expanded on the OAUE approach to address different types of concept drift. Heterogeneous Dynamic Weighted Majority (HDWM) [61] enhances Dynamic Weighted Majority (DWM) [55] by creating a mixed ensemble that selects the most appropriate learners at different points in time. This prevents a decline in performance when the data distribution changes.

Recurring Dynamic Weighted Majority (RDWM) [62] is designed to handle recurring concept drift. It includes two ensembles: a primary online ensemble (EO) and a secondary ensemble (EB). The primary online ensemble represents the current concept, while the secondary ensemble represents the old concept since the beginning of the learning process. The best learner is selected from the secondary ensemble and copied into the primary ensemble. Dynamic Drift Detection (DDD) [63] is a modified version of Online Bagging that employs four classifier ensembles with different levels of diversity before and after a concept drift is detected. DDD identifies the optimal weighted majority of ensembles before and after the detection of concept drifts, which are identified by a configurable auxiliary drift detector (by default, EDDM is used). Fast Adaptive Size Estimation (FASE) [64] is a variation of the OzaBag algorithm that employs a meta-classifier to merge the predictions of the base adaptive learners and employs the HDDMA approach to detect concept drifts. When drift is detected, the worst classifier is removed and a new one is added, and the voting strategy used is weighted based on the error rates of the components. The authors claim that FASE can process input data with constant time and space complexity.

Accelerated Drift Online Boosting (ADOB) [17] is a boosting ensemble that builds on OzaBoost and aims to accelerate the recovery of experts after concept drifts. ADOB sorts the experts according to their accuracy before processing each instance, affecting the distribution of diversity among the classifiers and leading to a slight improvement in the accuracy of the ensemble just after concept drifts. A configurable auxiliary drift detector is also included. Boosting Online Ensemble with Less Restrictions (BOLE) [32], [65] is an algorithm derived from ADOB with some heuristic changes. These modifications allow the experts to vote with fewer restrictions, enhancing the accuracy of the ensemble in various scenarios, particularly when there are frequent and sudden concept drifts. BOLE shows exceptional performance across most datasets, regardless of the auxiliary drift detection technique employed. e-Detector and Drift Detection Ensemble (DDE), are discussed in this paragraph. e-Detector [66] is an ensemble-based approach that can identify both sudden and gradual concept drifts. It organizes candidate detectors into clusters and selects the best component from each cluster based on a Coefficient of Failure (CoF). The ensemble generates drift and warning signals when any of its components identifies a drift, following an early-find-early-report rule. According to the authors, this approach enhances recall and reduces false negatives without significantly increasing false positives and shows better generalization capabilities than individual detectors. DDE [67] is a lightweight ensemble of detectors consisting of three configurable components designed to improve the detection of drifts, with minimal impact on execution time and better final accuracy. Its detections depend on a sensitivity parameter, which determines the minimum number of components required to signal warnings and drifts. If the sensitivity is set to 1, the DDE's strategy is similar to the early-find-early-report rule of e-Detector. DDE also offers two other sensitivity values (2 and 3) that make it more robust to false positives but may delay the true detections.

D. ANALYSIS, DISCUSSION, AND FUTURE DIRECTIONS

This section delves into and deliberates on the pertinent studies we've examined to derive insights and highlight the prevailing patterns in this domain of research. Our analysis centers around emphasizing the key aspects that define the methods outlined in this paper. First,Table 4, provides an overview of the key points of all the works mentioned above. However, our observation / findings can be discussed as follow:

DDM, EDDM, RDDM, DMDDM, DMDDM-S, and ADWIN/ADWIN2 are capable of detecting abrupt and gradual concept drift. DDM is a popular and widely used approach, while EDDM is an extension of DDM that aims to detect drift earlier. RDDM is a relatively new algorithm that employs a robust approach to handle noise and outliers in the data. DMDDM and DMDDM-S are variants of DDM, while ADWIN/ADWIN2 uses a sliding window approach to monitor changes in the mean of the data.

KAPPA, HBBE, EBE, CDDFET, ACDDM, FHDDM, SEED, ECDD, STEPD, WSTD, OCDD, BDDM, AUE2, ACE, OAUE, HDWM, RDWM, ADOB, BOLE, and

TABLE 4. Key points of the literature.

Method	Year	Key Points	Type of Drift
DDM [30]	2004	• **Mechanism**: Utilizes statistical process control, monitoring the standard error of the online learning model's predictions to detect drift. • **Effectiveness**: Effective in detecting abrupt drifts but can remain at the warning level too long, potentially slowing down processing. • **Limitations**: Struggles with detecting gradual drift as it relies on the binomial distribution and may not detect such drifts in a timely manner.	Abrupt, gradual
EDDM [31]	2005	• **Mechanism**: Enhances DDM by examining the distance between errors, searching for significant changes to signal drift. • **Effectiveness**: Outperforms DDM in detecting slow gradual drifts but can be sensitive to noise. • **Limitations**: The method may initiate the search for drift too frequently when noise levels are high, potentially leading to false positives.	Abrupt, gradual
RDDM [32, 33]	2017	• **Mechanism:** Focuses on reducing the number of instances of stable concepts in memory to identify drifts at regular intervals. • **Effectiveness:** Shows superior accuracy compared to ECDD, STEPD, and DDM but requires more memory. • **Limitations:** Can perform poorly with very large concept drifts due to increased memory demand.	Abrupt, gradual
DMDDM [34]	2020	• **Mechanism**: Proposes the integration of a diversity measure, specifically the disagreement among classifiers, to detect drifts. • **Effectiveness**: Reported to quickly adapt to changes in concepts using less time and memory than existing methods. • **Limitations**: Does not fully support slow-gradual drifts and noisy data streams.	Abrupt
KAPPA [35]	2021	• **Mechanism**: Designed to quickly detect concept drift using a metric based on the degree of consensus among various classifiers. • **Effectiveness**: Claims to identify drifts swiftly and with computational efficiency. • **Limitations**: Does not provide comprehensive performance metrics on initial research datasets, making it difficult to evaluate against other methods.	Abrupt
HBBE [37]	2021	• **Mechanism**: Introduces a hybrid block-based ensemble framework designed for multi-class classification in continuously changing data streams. • **Effectiveness**: Aims to harness the advantages of an online drift detector for k-class issues, combined with the concept of block-based weighting, to respond to various drift types. • **Limitations**: While it offers thorough testing on synthetic and real-world data sets, the algorithm's performance in highly dynamic or noisy environments may not be fully explored.	Abrupt, gradual
DMDDM-S [36]	2020	• **Mechanism**: Focuses on addressing the challenge of detecting concept drift without class labels by utilizing two classifiers and measuring the diversity between their predictions. • **Effectiveness**: Offers a solution to detect drifts even without class labels, enhancing the method's applicability in unsupervised learning scenarios. • **Limitations**: The reliance on classifier diversity may not fully accommodate every type of drift, especially in scenarios with complex or subtle changes.	Abrupt
EBE [38]	2018	• **Mechanism**: Utilizes the discrepancy between two classifiers to detect drifts, eliminating the need for a true label to ascertain if components are in disagreement. • **Effectiveness**: Designed for labeled data streams, this approach may facilitate the concurrent identification of abrupt and gradual shifts. • **Limitations**: The absence of true labels in the evaluation may pose challenges in verifying the accuracy of the detected drifts.	Abrupt, gradual
ACDDM [40]	2020	• **Mechanism**: Employs Hoeffding's inequality to gauge the difference between the observed error rate and its anticipated value, aiming to identify concept drifts by observing fluctuations in the error rate. • **Effectiveness**: The method focuses on quick identification of drifts, potentially offering more timely updates to models in streaming scenarios. • **Limitations**: May require further validation to establish its reliability, especially in data streams with complex or subtle drift patterns.	abrupt, gradual, incremental, recurring
CDDFET [39]	**2018**	• **Mechanism:** Advises against the use of the STEPD test for limited data samples, suggesting that it is unsuitable for sparse or imbalanced datasets. • **Effectiveness:** Evaluated for both sudden and gradual drift variations, but with a cautionary note on its application scope. • **Limitations:** The approach may not be ideal for all data stream sizes and types, particularly those that are imbalanced or thinly populated.	Abrupt, gradual

TABLE 4. ***(Continued.)*** **Key points of the literature.**

ADWIN / ADWIN2 [41, 42]	2007	• **Mechanism**: ADWIN2 improves on ADWIN by adjusting the window size dynamically based on the detected changes in the data stream, utilizing statistical tests like Fisher's Exact Test. • **Effectiveness**: Offers an improvement over ADWIN with better time and memory management, though still with some limitations on large-scale data. • **Limitations**: The method's efficiency decreases with multi-dimensional data and larger window sizes can cause memory issues.	Abrupt, gradual
FHDDM [43]	2016	• **Mechanism**: Leverages a sliding window in combination with Hoeffding's inequality to detect discrepancies in prediction likelihoods. • **Effectiveness**: Provides a higher likelihood of accurate prediction compared to several other drift detection techniques. • **Limitations**: Its application is limited to one-dimensional data, which may not be suitable for complex, multi-dimensional tasks.	Abrupt, gradual
SEED [44]	2014	• **Mechanism**: Simultaneously examines the pace of alterations in the data stream and identifies drift through volatility detection. • **Effectiveness**: Touts a reduced rate of false positives and is more efficient than ADWIN2, but it may not detect every form of drift. • **Limitations**: There are shortcomings in detecting recurring drifts and the model is not entirely independent of the data distribution.	Abrupt, gradual
HDDM_$_{Atest}$ and HDDM_$_{W\ test}$ [45]	2015	• **Mechanism**: These methods use supervised incremental learning to detect drifts, applying probability inequalities as the basis for detection. • **Effectiveness**: Claimed to be algorithm-agnostic, meaning it can be applied to various machine learning algorithms. • **Limitations**: While they aim to reduce error over classifiers, their performance in variable contexts has not been fully validated.	A= abrupt, W = gradual
ECDD [46]	2012	• **Mechanism**: Utilizes classification error to detect drift, offering control over false positive rates with O(1) complexity. • **Effectiveness**: Adds minimal overhead to the classifier and is praised for its efficiency. • **Limitations**: Does not require storage of data instances in memory, which might limit its applicability in certain real-world scenarios.	Abrupt, gradual
STEPD [47]	2008	• **Mechanism**: Evaluates two types of accuracies: recent and overall, to determine the presence of drift. • **Effectiveness**: Noted for its capability in detecting sudden drift, potentially outperforming methods like DDM, EDDM, PHT, and ECDD. • **Limitations**: Its performance in the presence of noise and its ability to detect gradual drift are not fully explored.	Abrupt, gradual
WSTD [32, 48]	2018	• **Mechanism**: Employs just two rank values to detect drift, aiming for a simple and straightforward approach. • **Effectiveness**: Though simple, it requires significant computational resources. • **Limitations**: The method's effectiveness is not thoroughly demonstrated across varying data stream types and sizes.	Abrupt, gradual
OCDD **[49]**	2020	• **Mechanism**: Combines sliding window techniques with a one-class classifier for drift adaptation in dynamic environments. • **Effectiveness**: Offers a tiered approach with varying windows for different stability conditions. • **Limitations**: Lacks adaptive parameter tuning, which may limit its efficacy in evolving data stream conditions.	Abrupt, gradual
BDDM [50]	2012	• **Mechanism:** Applies the Bhattacharyya distance to monitor the mean and variance of a sequence of values over time, identifying significant changes indicative of drifts. • **Effectiveness:** Tailored to identify abrupt variations in the distribution and has a monitoring system for changes, but its effectiveness in incremental and recurring drifts is not explicitly stated. • **Limitations:** The method's reliance on specific distance metrics may not capture all types of drift, especially those that evolve incrementally.	Abrupt, gradual

TABLE 4. *(Continued.)* Key points of the literature.

AWE [53]	2003	• **Mechanism**: Uses an ensemble approach with a novel trajectory for classifier selection, adjusting to drift by re-weighting classifiers based on their performance. • **Effectiveness**: Yields more precise results than a standard classifier for sudden drift, though may be less suited for gradual or minimal volume changes. • **Limitations**: It faces challenges in optimizing chunk size and handling minimal data volumes, which can impede its adaptability.	Abrupt, gradual
MDDMs [51]	2018	• **Mechanism**: Employs a weighting mechanism within a sliding window to prioritize recent data, which helps in the quicker identification of concept drift. • **Effectiveness**: Offers three versions with arithmetic, geometric, and Euler weighting for versatility in different environment conditions. • **Limitations**: The detailed impact of these weighting mechanisms on the accuracy and efficiency of drift detection is not fully discussed.	Abrupt, gradual
AUE2 [54]	2015	• **Mechanism**: Optimized for stable and changing environments, AUE2 delivers classification precision by applying an updated weighting system to the data. • **Effectiveness**: Claims the highest average classification precision with less memory use compared to other techniques. • **Limitations**: The scope of drift types that AUE2 can effectively handle is not comprehensively detailed, leaving some ambiguity regarding its adaptability.	Abrupt, gradual
DWM [55]	2003	• **Mechanism**: Responds to changes in data stream performance by updating base learners and their weights, facilitating drift adaptation. • **Effectiveness**: Particularly effective for recurring drifts, though maintaining a significant number of base learners might impact efficiency. • **Limitations**: The balance between the number of learners and the timely detection of drifts could pose a challenge in real-time scenarios.	Abrupt, gradual, recurring
ACE [56]	2005	• **Mechanism**: Combines a single online classifier, multiple batch classifiers, and a drift detection mechanism to evaluate average accuracy and detect shifts. • **Effectiveness**: ACE identifies shifts by merging the forecasts of both online and batch learners through a weighted majority consensus. • **Limitations**: While the ensemble approach is innovative, the time taken to collect and process comprehensive data could delay drift detection.	Abrupt, gradual
Learn++.NSE [57]	2011	• **Mechanism**: Adapts to various types of concept drift and is efficient in changing environments through its specialized drift detection technique. • **Effectiveness**: Its computational efficiency is a strength, but the method's adaptability to all forms of drift is not fully explored. • **Limitations**: The technique's reliance on batch processing might not align with the needs of real-time data stream analysis.	Adapt regardless of the type
SEA [59]	2001	• **Mechanism**: Utilizes a single classifier for concept drift detection, aiming to be computationally more efficient than other approaches. • **Effectiveness**: Shows promise in certain scenarios but may lag in adapting quickly to new concepts or recurring drifts. • **Limitations**: The system's effectiveness against previously unseen problems with data streams was not addressed.	Abrupt
RDWM [62]	2019	• **Mechanism**: RDWM (Robust Drift Window Model) addresses the challenge of identifying previously unseen problems within data streams. • **Effectiveness**: It aims to improve predictive accuracy for new and recurring concept drifts, with an emphasis on robustness. • **Limitations**: The method's success in addressing new and previously unseen problems has not been fully validated, and it may require further testing to confirm its effectiveness across various types of data streams.	recurring
DDD [63]	2011	• **Mechanism**: Focuses on detecting concept drift with a higher accuracy compared to earlier methods like EDDM and has strong resilience against false alarms. • **Effectiveness**: Shows improvement in accuracy but suggests the need for further enhancements to process recurring concept drifts effectively. • **Limitations**: There is an acknowledgment that the method requires improvements for better processing of recurring drifts.	Abrupt, gradual

TABLE 4. *(Continued.)* Key points of the literature.

OAUE [16]	2014	• **Mechanism**: Enhances online learning with an algorithm that adjusts to concept drifts by updating classifier weights based on error rates. • **Effectiveness**: Noted for providing high classification accuracy regardless of drift type and boasts a low memory and time footprint. • **Limitations**: Challenges include handling minimal data volumes and the potential for underperformance in non-abrupt drift scenarios.	Abrupt, gradual, incremental, recurring
HDWM [61]	2020	• **Mechanism**: Introduces a heterogeneous approach to model selection in data streams, aiming to enhance predictive accuracy by incorporating human insights into the model selection process. • **Effectiveness**: Demonstrates potential for improved accuracy through the strategic use of heterogeneity in model selection. • **Limitations**: The study suggests that more work is needed to fully understand the impact of heterogeneity on performance, especially in diverse problem settings.	Abrupt, gradual, recurring
FASE [64]	2016	• **Mechanism**: Employs a meta-classifier that aggregates predictions from base classifiers in an ensemble to adapt to various drift types. • **Effectiveness**: Improves predictive accuracy in the presence of different drift types, including gradual, sudden, and recurring. • **Limitations**: The complexity of the ensemble approach and its computational demands are not thoroughly discussed in the context of real-world application.	Abrupt
ADOB [17]	2014	• **Mechanism**: Efficiently distributes instances among base learners to manage frequent drifts, employing a confidence level for change detection. • **Effectiveness**: Designed for quick recovery from drifts with an emphasis on minimizing the impact on runtime and memory usage. • **Limitations**: The potential impact of diversity and accuracy on runtime performance has not been fully explored.	Abrupt
BOLE [32, 65]	2016	• **Mechanism**: Aims to increase sensitivity of detectors to detect concept drift by aggressively parameterizing them. • **Effectiveness**: The method's aggressive parameterization may help in quick detection of drifts. • **Limitations**: Comprehensive analysis on how changing boundary parameters and classifier weight impacts diversity distribution and final accuracy of the base classifier is not provided.	Abrupt
e-Detector [66]	2014	• **Mechanism**: Combines multiple techniques to identify concept drifts of varying velocities, leveraging an ensemble approach. • **Effectiveness**: Offers quicker detection and reduction in time delays than single methods, and is capable of discerning drifts that begin suddenly and transition to gradual. • **Limitations**: Further validation is required to confirm the effectiveness of the ensemble approach, especially in early-and-often detection scenarios.	Abrupt, gradual
DDE [67]	2015	• **Mechanism**: Integrates outcomes from distinct concept drift detection techniques to offer a more nuanced and precise identification of drifts. • **Effectiveness**: The tiered detection strategy is aimed at learning from data streams using a foundational concept drift adaptation strategy. • **Limitations**: Performance across a wide range of data streams and drift types, including the ability to operate concurrently with multiple detection methods, is an area for further research and validation.	Abrupt, gradual

e-Detector are also capable of detecting both abrupt and gradual concept drift. These algorithms differ in their approach and methods for detecting concept drift. KAPPA is a newer algorithm based on the Kappa statistic used in inter-rater reliability. HBBE is a hybrid algorithm that combines Bayesian estimation and decision trees. EBE uses an ensemble of base detectors to improve detection accuracy. CDDFET is a variant of the popular CUSUM algorithm. ACDDM uses hierarchical clustering to monitor the change in data distribution. FHDDM is a method that combines multiple detectors to improve detection accuracy. SEED, ECDD, STEPD, WSTD, and OCDD are based on sliding window approaches. BDDM and AUE2 are extensions of DDM, while ACE is a variant of CUSUM. OAUE is capable of detecting incremental and recurring drift in addition to abrupt and gradual drift. HDWM is capable of detecting recurring drift, while RDWM is focused on detecting only recurring drift. ADOB and BOLE are both focused on detecting only abrupt drift.

In terms of future directions, there are several areas where more research is needed to improve the detection of concept drift, we list these directions as follow:

1) The literature that was examined primarily deals with sudden and gradual drifts as the main types of drift. There are not as many studies on incremental drift which can be difficult to differentiate from the system's natural evolution and continuous changes. Incremental drift occurs when changes in the data distribution are gradual and occur over an extended period. It is often caused by gradual changes in the environment, such as changes in the economy or changes in consumer preferences. Detecting incremental drift is challenging because it requires monitoring the data distribution over time and detecting small changes in the data. When drift occurs repeatedly, the system needs to handle it in a specific manner by having a buffer to store previous behaviors and reusing the knowledge learned from past observations when it occurs again in the system.
2) Another area that needs more focus is the detection of recurring drift. Recurring drift is a type of drift that occurs periodically in a dataset. It is often caused by external factors, such as seasonal changes or changes in consumer behavior. Detecting recurring drift is challenging because it requires identifying patterns in the data that repeat over time.
3) There is a need for algorithms that can handle high-dimensional data and online learning tasks. High-dimensional data refers to datasets with a large number of features or variables. Online learning tasks refer to situations where the data arrives continuously and in real-time. Handling high-dimensional data and online learning tasks is challenging because it requires processing large amounts of data quickly and efficiently while maintaining high accuracy.
4) Drift detection systems need frequent updates after deployment, so it's crucial for drift detection methods to support incremental learning and dynamic adaptation. Hoeffding Trees and Naive Bayes are popular base learners for most performance-based concept drift detectors due to their ability to handle large data streams. However, recent approaches have incorporated neural networks, which may present challenges for deployment within big data stream systems because updating the neural network architecture dynamically can be difficult. Another significant drawback of neural networks is their lack of transparency and interpretability, which adds a burden to concept drift handling systems since understanding drift is essential for detecting and adapting to it.

Overall, the development of more robust and accurate algorithms for detecting concept drift is an active area of research, and further developments in this area are essential for the continued success of machine learning and data science applications.

VI. CONCEPT DRIFT TOOLKIT AND DATASETS

We now have a better understanding of the various categories of drift detectors and the pros and cons associated with each one. This section of the paper will discuss both current and potential tools that can be employed to detect drifts. In recent years, there has been a surge in the popularity of monitoring for Concept Drift. Most major cloud providers offer their own drift monitoring capabilities.

- **Vertex AI Model Monitoring** [68]: In May 2022, Vertex AI Model Monitoring was available on Google Cloud Platform, and it offered only Univariate Drift Detection methods without labels, such as the Jensen-Shannon divergence and L-infinity distance. Additionally, it was possible to create a monitoring system that included alerts and warnings.
- **Amazon SageMaker Model Monito** [69], [70]**:** Amazon SageMaker Model Monitor offers a sophisticated monitoring system for Machine Learning models that are deployed in production and generates reports. However, there is limited information available re-garding the specific methods for detecting drift that it employs.
- **Watson OpenScale Drift Monitor** [70], [71]: IBM Cloud's Watson OpenScale Drift Monitor offers a monitoring dashboard to detect drifts, utilizing a variety of methods to detect inconsistencies and accuracy drops. This includes some drift detectors with labels, such as DDM.
- **Azure Machine Learning** [72]: Microsoft Azure's Azure Machine Learning includes drift monitoring functionality among its features, but there is a lack of transparency regarding the specific metrics it uses. It is known that the platform is unable to address real concept drifts.

Open-source tools are also available for detecting drifts:

- **MOA (Massive Online Analysis)** [73] is a freely available open-source software designed for data stream mining with a focus on detecting concept drift. It includes a prequential evaluation method, as well as the EDDM concept drift methods, and can read ARFF real datasets, along with generating artificial stream data such as SEA concepts, STAGGER, rotating hyperplane, random tree, and random radius-based functions. MOA also supports bi-directional interaction with Weka.
- **River** [74] is a machine learning library designed for dynamic data streams and continual learning. It offers a range of state-of-the-art learning methods, data generators/transformers, performance metrics, and evaluators tailored for various stream learning challenges. Drawing on insights from seminal packages, River boasts a refined architecture.

Also, a variety of dataset types are also accessible for assessments: have been widely used in above works [75].

Real Datasets include:

- **The USP Data Stream Repository** [40]**:** contains 27 real-world datasets with concept drift, compiled by Souza et al. in 2020. The repository can be accessed online.
- **The Airline dataset** [76], which includes approximately 116 million flight arrival and departure records that have been cleaned and sorted, was compiled by E. Ikonomovska and can be accessed through the Data Expo 2009 Competition.
- **The Elec2 dataset** [77], It comprises 45,312 data points, with 8 input features, collected every 30 minutes over two years from the New South Wales Electricity in Australia. The objective is to forecast whether the electricity price will increase (Up) or decrease (Down). Concept drift might occur due to shifts in consumer behavior, unforeseen incidents, and seasonal variations.
- **Forest CoverType** [78], It features 54 attributes and 581,012 data points that depict 7 types of forest cover for areas measuring 30 X 30 meters. This information was sourced from the US Forest Service (USFS) Region 2 Resource Information System (RIS) and pertains to four wilderness zones situated in the Roosevelt National Forest in northern Colorado.
- **Poker hand** [79], It consists of 1,000,000 data points, with each data point representing a hand of five playing cards taken from a standard 52-card deck. Every card has two characteristics (suit and rank), resulting in ten predictive features. The classification determines the type of poker hand.
  Synthetic Datasets includes:
- **Sine1 (with abrupt concept drift)** [80]: The dataset contains two features, x and y, which are evenly distributed within the range [1, 0]. The classification is determined by the function $y = \sin(x)$. Data points below the curve are labeled as positive, while those above are deemed negative. When a drift occurs, the class labels switch.
- **Sine2 (with abrupt concept drift)** [80]: The dataset features two variables, x and y that are evenly spread within the [1, 0]. The classification is determined by the function $0.5 + 0.3 * \sin(3 * \pi * x)$. Data points that fall beneath the curve are labeled as positive, and those above it are deemed negative. When a drift takes place, the labeling system flips.
- **Stagger (with abrupt concept drift)** [81]: The dataset includes three categorical attributes: size (with values {small, medium, large}), color (with values {red, green}), and shape (with values {circular, non-circular}). Before the initial drift point, entries are labeled positive when both conditions (color = red) and (size = small) are met. Following this and up to the second drift, entries are marked positive if either (color = green) or (shape = circular). After the second drift, entries are deemed positive if the size is either medium or large.
- **Mixed (with abrupt concept drift)** [30]: The dataset features two numerical variables, x and y, which lie in the range [1, 0], along with two boolean attributes, v and w. Data points are labeled as positive when a minimum of two out of these three conditions hold true: v, w, or y is less than $0.5 + 0.3 * \sin(3 * \pi * x)$. When drift events arise, the classification system is flipped.
- **Circles (with gradual concept drift)** [80]: It has two attributes x and y that are distributed in [1, 0]. The function of circle <(x_c,y_c), r_c> is (x - x_c)^2 + (y - y_c)^2 = r_c^2 where (x_c, y_c) is its center and r_c is the radius. Four circles of <(0.2, 0.5), 0.15>, <(0.4, 0.5), 0.2>, <(0.6, 0.5), 0.25>, and <(0.8, 0.5), 0.3> classify instances in order. Instances inside the circle are classified as positive. A drift happens whenever the classification function, i.e. circle function, changes.
- **LED (with gradual concept drift)** [80]: The aim of this dataset is to identify the number shown on a seven-segment display, with each digit having a 10% probability of being presented. The dataset comprises 7 attributes directly tied to the class and an additional 17 that are irrelevant. Concept drift is emulated by swapping the pertinent attributes.

VII. FUTURE TRENDS

Regarding future research, there are numerous intriguing directions that can be pursued.

- **Intrusion** detection systems (IDS): are an important component of cybersecurity in the Internet of Things (IoT) ecosystem, as they help detect and respond to potential security threats [82], [83], [84]. However, IDS in IoT environments face significant challenges due to the unique characteristics of IoT data, such as the large volume, high velocity, and heterogeneity of data. Concept drift is one of the major challenges that IDS in IoT face. The dynamic nature of IoT data and the complex relationships between devices and the environment can cause concept drift to occur rapidly and frequently. To address this challenge, future directions for concept drift in IDS in IoT could include:
  1. Development of specialized algorithms: Traditional machine learning algorithms may not be suitable for handling concept drift in IoT data. There is a need for specialized algorithms that can detect and adapt to concept drift in IoT data.
  2. Integration of context-awareness: Context-awareness can help improve the accuracy of IDS in IoT by considering the context in which the data is collected. Future research could focus on developing context-aware IDS that can adapt to changes in the data distribution and the environment.
- **Federated Learning**: Federated learning is a machine learning technique that allows multiple edge devices to collaborate on model training without sharing their data. This approach can be used to detect concept drift in

IoT data, allowing IDS to adapt to new threats without compromising the privacy of the data.

- **Large-Scale Deployment**: IoT systems can generate massive amounts of data, and concept drift detection methods need to be scalable to handle this data. Future work can focus on developing distributed and parallel algorithms that can handle large-scale IoT data and detect concept drift in real-time.
- **Semi-supervised or unsupervised approaches to drift detection and adaptation:** Present methods for identifying and adjusting to concept drift operate under the assumption that correct labels for data instances are instantly accessible post-prediction, rendering these processes supervised. Yet, in practical settings, actual labels might not be immediately obtainable. This necessitates the creation of enhanced drift detection and adaptation strategies suited for semi-unsupervised situations, leveraging either semi-supervised or unsupervised techniques [85].
- **Focusing on concept-driven data filtering:** Concept drift challenges can arise not just in data stream learning but also in batch learning, especially when data for training and testing is gathered over a span of time rather than a fixed moment. In such scenarios, knowledge trends might shift over different timeframes, and the most pertinent predictive data might be missing from the training and testing sets collected concurrently. Relying on cross-validation might fall short in mitigating overfitting and underfitting problems. A potential remedy for these concept drift issues is the strategy of concept-driven data filtering. This approach centers on excluding data that doesn't pertain to the specific concept drift.
- **Analyzing concept drift in video streams:** The exploration of concept drift within video streams can gain from the innovative methods suggested for video-centric applications. Moreover, understanding the connections and distinctions between concept drift adjustment and visual domain adaptation methods is essential. Addressing the challenge of concept drift is vital for the development of adaptive systems, marking it as a pressing and significant concern. Future research holds immense promise in honing the adaptability of machine learning methods and frameworks to confront concept drift.
- **Class Imbalance issue:** The issue of Class Imbalance is a recognized difficulty in machine learning, characterized by one class appearing much less frequently than the other. Consequently, identifying drift in the underrepresented class becomes a formidable endeavor.
- **Data streams in big data:** Identifying different types of concept drift within extensive data streams can be challenging. It's vital to enhance precision while also reducing both computational time and memory consumption.
- **Distributed Learning:** With the rise of edge computing, machine learning models are being trained and deployed in distributed environments. Techniques for detecting and handling concept drift will need to be adapted to work with these distributed systems, ensuring that they remain accurate and robust.

VIII. CONCLUSION

This paper provides a concise overview of the developments in concept drift detection methods since their introduction. We have summarized the existing techniques, their applications, and future trends in the field of concept drift detection. As new sources of data and types of drift continue to emerge, the problem of concept drift detection remains open-ended. This offers many opportunities for researchers and tool developers to advance the field. To improve the accuracy, speed, and robustness of the models, future work can focus on more advanced machine learning algorithms, edge computing, federated learning, explainable AI, and hybrid methods.

Moreover, research can focus on addressing specific challenges such as online and incremental learning, handling multimodal data, robustness to adversarial attacks, and large-scale deployment of concept drift detection methods. The future work in these areas can lead to the development of more effective and efficient concept drift detection techniques.

REFERENCES

[1] O. A. Mahdi, "Diversity measures as new concept drift detection methods in data stream mining," Ph.D. dissertation, Dept. CS. IT., La Trobe Univ., Melbourne, 2020.

[2] J. Gama, *Knowledge Discovery From Data Streams*. Boca Raton, FL, USA: CRC Press, 2010.

[3] B. R. Prasad and S. Agarwal, "Stream data mining: Platforms, algorithms, performance evaluators and research trends," *Int. J. Database Theory Appl.*, vol. 9, no. 9, pp. 201–218, Sep. 2016.

[4] I. Žliobaitė, "Learning under concept drift: An overview," 2010, *arXiv:1010.4784*.

[5] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognit.*, vol. 76, pp. 256–272, Apr. 2018.

[6] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras, "A concept drift-tolerant case-base editing technique," *Artif. Intell.*, vol. 230, pp. 108–133, Jan. 2016.

[7] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artif. Intell.*, vol. 209, pp. 11–28, Apr. 2014.

[8] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2280–2286.

[9] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2017, pp. 1–6.

[10] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 59–68.

[11] J. Gama, R. Sebastiao, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, 2013.

[12] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Mach. Learn.*, vol. 98, no. 3, pp. 455–482, Mar. 2015.

[13] I. Stoica, D. Song, R. Ada Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez, K. Goldberg, A. Ghodsi, D. Culler, and P. Abbeel, "A Berkeley view of systems challenges for AI," 2017, *arXiv:1712.05855*.

[14] P. M. Domingos and G. Hulten, "Catching up with the data: Research issues in mining data streams," in *Proc. DMKD*, 2001, pp. 1–5.

[15] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun. 2009, pp. 329–338.

[16] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, May 2014.

[17] S. G. T. C. Santos, P. M. G. Júnior, G. D. S. Silva, and R. S. M. de Barros, "Speeding up recovery from concept drifts," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2014, pp. 179–194.

[18] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Apr. 2014.

[19] P. E. Hart, D. G. Stork, and R. O. Duda, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2000.

[20] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 3–14.

[21] M. G. Kelly, D. J. Hand, and N. M. Adams, "The impact of changing populations on classifier performance," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 1999, pp. 367–371.

[22] W. Fan, Y.-A. Huang, H. Wang, and P. S. Yu, "Active mining of data streams," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 457–461.

[23] M. Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching," in *Lazy Learning*. Kluwer Academic Publishers, 1997, pp. 133–155.

[24] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Comput. Sci. Dept., Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[25] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," in *Proc. Int. Conf. Innov. Techn. Appl. Artif. Intell.*, 2004, pp. 3–16.

[26] G. Widmer and M. Kubat, "Effective learning in dynamic environments by explicit context tracking," in *Proc. Eur. Conf. Mach. Learn.*, 1993, pp. 227–243.

[27] M. M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, Mar. 2004.

[28] L. I. Kuncheva, "Classifier ensembles for changing environments," in *Proc. Int. Workshop Multiple Classifier Syst.*, 2004, pp. 1–15.

[29] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[30] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symp. Artif. Intell. (SBIA)*, 2004, pp. 286–295.

[31] M. Baena-Garca, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discovery Data Streams*, 2006, pp. 77–86.

[32] R. S. M. de Barros, "Advances in data stream mining with concept drift," Universidade Federal de Pernambuco (UFPE), Centro de Informática, Recife, Brazil, Tech. Rep., 2017.

[33] R. S. M. Barros, D. R. L. Cabral, P. M. Gonçalves, and S. G. T. C. Santos, "RDDM: Reactive drift detection method," *Expert Syst. Appl.*, vol. 90, pp. 344–355, Dec. 2017.

[34] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Diversity measure as a new drift detection method in data streaming," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105227.

[35] O. A. Mahdi, E. Pardede, and N. Ali, "KAPPA as drift detector in data stream mining," *Proc. Comput. Sci.*, vol. 184, pp. 314–321, Jan. 2021.

[36] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Fast reaction to sudden concept drift in the absence of class labels," *Appl. Sci.*, vol. 10, no. 2, p. 606, Jan. 2020.

[37] O. A. Mahdi, E. Pardede, and N. Ali, "A hybrid block-based ensemble framework for the multi-class problem to react to different types of drifts," *Cluster Comput.*, vol. 24, no. 3, pp. 2327–2340, Sep. 2021.

[38] O. A. Mahdi, E. Pardede, and J. Cao, "Combination of information entropy and ensemble classification for detecting concept drift in data stream," in *Proc. Australas. Comput. Sci. Week Multiconf.*, Jan. 2018, pp. 1–5.

[39] D. R. D. L. Cabral and R. S. M. D. Barros, "Concept drift detection based on Fisher's exact test," *Inf. Sci.*, vols. 442–443, pp. 220–234, May 2018.

[40] M. M. W. Yan, "Accurate detecting concept drift in evolving data streams," *ICT Exp.*, vol. 6, no. 4, pp. 332–338, Dec. 2020.

[41] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 443–448.

[42] A. Bifet, "Adaptive learning and mining for data streams and frequent patterns," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 55–56, Nov. 2009.

[43] A. Pesaranghader and H. L. Viktor, "Fast Hoeffding drift detection method for evolving data streams," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2016, pp. 96–111.

[44] D. T. J. Huang, Y. S. Koh, G. Dobbie, and R. Pears, "Detecting volatility shift in data streams," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 863–868.

[45] I. Frías-Blanco, J. de Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

[46] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, Jan. 2012.

[47] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Proc. Int. Conf. Discovery Sci.*, 2007, pp. 264–269.

[48] R. S. M. D. Barros, J. I. G. Hidalgo, and D. R. D. L. Cabral, "Wilcoxon rank sum test drift detector," *Neurocomputing*, vol. 275, pp. 1954–1963, Jan. 2018.

[49] Ö. Gözüaçık and F. Can, "Concept learning using one-class classifiers for implicit drift detection in evolving data streams," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3725–3747, Jun. 2021.

[50] I. Baidari and N. Honnikoll, "Bhattacharyya distance based concept drift detection method for evolving data stream," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115303.

[51] A. Pesaranghader, H. L. Viktor, and E. Paquet, "McDiarmid drift detection methods for evolving data streams," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–9.

[52] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[53] Z. Ouyang, M. Zhou, T. Wang, and Q. Wu, "Mining concept-drifting and noisy data streams using ensemble classifiers," in *Proc. Int. Conf. Artif. Intell. Comput. Intell.*, vol. 4, Nov. 2009, pp. 360–364.

[54] B. Krawczyk and M. Wozniak, "Reacting to different types of concept drift with adaptive and incremental one-class classifiers," in *Proc. IEEE 2nd Int. Conf. Cybern. (CYBCONF)*, Jun. 2015, pp. 30–35.

[55] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Proc. 3rd IEEE Int. Conf. Data Mining*, pp. 123–130.

[56] K. Nishida, K. Yamauchi, and T. Omori, "ACE: Adaptive classifiers-ensemble system for concept-drifting environments," in *Proc. Int. Workshop Multiple Classifier Syst.*, Seaside, CA, USA, Jun. 2005, pp. 176–185.

[57] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[58] J. G. Carbonell, T. M. Mitchell, and R. S. Michalski, *Machine Learning: A Guide to Current Research*. London, U.K.: Kluwer Academic Publishers, 1986.

[59] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2001, pp. 377–382.

[60] J.-W. Liao and B.-R. Dai, "An ensemble learning approach for concept drift," in *Proc. Int. Conf. Inf. Sci. Appl. (ICISA)*, May 2014, pp. 1–4.

[61] M. M. Idrees, L. L. Minku, F. Stahl, and A. Badii, "A heterogeneous online learning ensemble for non-stationary environments," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 104983.

[62] P. Sidhu and M. P. S. Bhatia, "A two ensemble system to handle concept drifting data streams: Recurring dynamic weighted majority," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 3, pp. 563–578, Mar. 2019.

[63] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012.

[64] I. Frías-Blanco, A. Verdecia-Cabrera, A. Ortiz-Díaz, and A. Carvalho, "Fast adaptive stacking of ensembles," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, Apr. 2016, pp. 929–934.

[65] R. S. M. de Barros, S. Garrido T. de Carvalho Santos, and P. M. G. Júnior, "A boosting-like online learning ensemble," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1871–1878.

[66] L. Du, Q. Song, L. Zhu, and X. Zhu, "A selective detector ensemble for concept drift detection," *Comput. J.*, vol. 58, no. 3, pp. 457–471, Mar. 2015.

[67] B. I. F. Maciel, S. G. T. C. Santos, and R. S. M. Barros, "A lightweight concept drift detection ensemble," in *Proc. IEEE 27th Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2015, pp. 1061–1068.

[68] *Monitor Feature Skew and Drift | Vertex AI | Google Cloud*. Accessed: Nov. 4, 2023. [Online]. Available: https://cloud.google.com/vertex-ai/docs/model-monitoring/using-model-monitoring

[69] *Machine Learning Governance—ML Governance With Amazon SageMaker—Amazon Web Services*. Accessed: Nov. 4, 2023. [Online]. Available: https://aws.amazon.com/sagemaker/ml-governance/

[70] *Monitor Model Drift With Watson OpenScale—IBM Developer*. Accessed: Nov. 4, 2023. [Online]. Available: https://developer.ibm.com/tutorials/monitoring-model-drift-with-watson-openscale/

[71] *Configuring Drift Evaluations in Watson OpenScale—Docs | IBM Cloud Pak for Data As a Service*. Accessed: Nov. 4, 2023. [Online]. Available: https://dataplatform.cloud.ibm.com/docs/content/wsj/model/wos-monitor-drift.html?locale=en&context=cpdaas

[72] *Detect Data Drift on Datasets (Preview)—Azure Machine Learning | Microsoft Learn*. Accessed: Nov. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets?view=azureml-api-1&tabs=python

[73] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: Massive online analysis, a framework for stream classification and clustering," in *Proc. 1st Workshop Appl. Pattern Anal.*, 2010, pp. 44–50.

[74] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, and A. Zouitine, "River: Machine learning for streaming data in Python," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 4945–4952, Dec. 2020.

[75] V. M. A. Souza, D. M. Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining Knowl. Discovery*, vol. 34, pp. 1805–1858, Jul. 2020.

[76] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data Mining Knowl. Discovery*, vol. 23, no. 1, pp. 128–168, Jul. 2011.

[77] I. Zliobaite, "How good is the electricity benchmark for evaluating concept drift adaptation," 2013, *arXiv:1301.3524*.

[78] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Comput. Electron. Agricult.*, vol. 24, no. 3, pp. 131–151, Dec. 1999.

[79] *Poker Hand—UCI Machine Learning Repository*. Accessed: Nov. 4, 2023. [Online]. Available: https://archive.ics.uci.edu/dataset/158/poker

[80] *GitHub—Alipsgh/data-streams: You Will Find (about) Synthetic and Real-world Data Streams in This Repository*. Accessed: Nov. 5, 2023. [Online]. Available: https://github.com/alipsgh/data-streams

[81] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Mach. Learn.*, vol. 1, no. 3, pp. 317–354, Sep. 1986.

[82] O. A. Wahab, "Intrusion detection in the IoT under data and concept drifts: Online deep learning approach," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19706–19716, Oct. 2022.

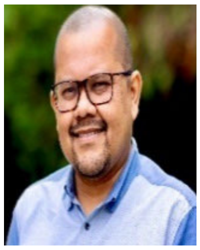
[83] A. Alazab, A. Khraisat, S. Singh, S. Bevinakoppa, and O. A. Mahdi, "Routing attacks detection in 6LoWPAN-based Internet of Things," *Electronics*, vol. 12, no. 6, p. 1320, Mar. 2023.

[84] O. A. Mahdi, A. Alazab, S. Bevinakoppa, N. Ali, and A. Khraisat, "Enhancing IoT intrusion detection system performance with the diversity measure as a novel drift detection method," in *Proc. 9th Int. Conf. Inf. Technol. Trends (ITT)*, May 2023, pp. 50–54.

[85] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. dos Santos, "An overview of unsupervised drift detection methods," *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 6, p. e1381, Nov. 2020.

OSAMA A. MAHDI received the B.Sc. degree in computer science from Babylon University, Iraq, in 2009, the M.Sc. degree in database systems from Universiti Putra Malaysia, in 2014, and the Ph.D. degree in data stream analytics from La Trobe University, in 2021. He is currently a full-time Lecturer with the Master of Data Analytics Program, Melbourne Institute of Technology (MIT). His research interests include data stream mining (with a focus on concept drift), data analytics, and machine\deep learning.

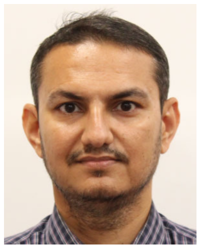
NAWFAL ALI received the B.Sc. and M.Sc. degrees from Al-Nahrain University, Iraq, in 2000 and 2003, respectively, and the Ph.D. degree from the University of Putra Malaysia, in 2012. He is currently a Lecturer with the Faculty of Computing Science, Monash University, VIC, Australia. He has published more than 15 papers in peer-reviewed journals and conferences. His research interests include cloud computing, job scheduling, federated learning, and data mining.



ERIC PARDEDE (Senior Member, IEEE) is currently an Associate Professor with the School of Computing, Engineering and Mathematical Sciences, La Trobe University, Melbourne, Australia. He has published more than 140 peer-reviewed papers in books, journals, and conference proceedings. He has supervised 15 Ph.D. students to completion. His current research interests include data processing and analytics and IT/science higher education pedagogies and practices.

AMMAR ALAZAB is currently a Senior Cybersecurity Lecturer with a wealth of industry experience. His expertise extends beyond academia, as he has a strong background in the cybersecurity industry. This practical experience enriches his teaching and research, providing valuable real-world insights to his students. With numerous published articles and over 50 research papers in top-tier journals and conferences, he is a prominent figure in the cybersecurity field. His remarkable contributions have led him to secure research grants amounting to over one million from reputed industry players, further validating the practical significance of his work. His current research interests include cyber security, digital forensics of computer systems, and cybercrime detection and prevention.

TAHSIEN AL-QURAISHI is currently a Senior Data Science Lecturer and contributes to the academic community by drawing on his extensive professional experience, particularly in the area of health informatics. His research spans a wide range, with contributions to prestigious journals and conferences in the areas of data science, data analysis, bioinformatics, and health informatics. He has a well-deserved reputation as a respected expert in the field of health informatics, specifically cancer research, and is praised for his priceless real-world knowledge.



BHAGWAN DAS (Senior Member, IEEE) is currently a Faculty Member of the School of Information Technology and Engineering, Melbourne Institute of Technology. Recognized for his contributions, he holds a patent and eight copyrights. He has published over 40 articles in prominent journals and conferences. His research interests include the Internet of Things, AI, and data sciences. He received the 2016 Best Man Inventor Award from IFIA, Geneva, and other accolades from institutions in Malaysia. He was the Chair of the IEEE Computer Society, from 2020 to 2023. He served as the Secretary for IEEE IES, from 2021 to 2023.