OEBench: Investigating Open Environment Challenges in Real-World Relational Data Streams

Yiqun Diao
National University of Singapore
yiqun@comp.nus.edu.sg

Yutong Yang
National University of Singapore
e0425605@u.nus.edu

Qinbin Li*
University of California, Berkeley
qinbin@berkeley.edu

Bingsheng He
National University of Singapore
hebs@comp.nus.edu.sg

Mian Lu
4Paradigm Inc.
lumian@4paradigm.com

ABSTRACT

How to get insights from relational data streams in a timely manner is a hot research topic. Data streams can present unique challenges, such as distribution drifts, outliers, emerging classes, and changing features, which have recently been described as *open environment* challenges for machine learning. While existing studies have been done on incremental learning for data streams, their evaluations are mostly conducted with synthetic datasets. Thus, a natural question is how those open environment challenges look like and how existing incremental learning algorithms perform on real-world relational data streams. To fill this gap, we develop an Open Environment Benchmark named *OEBench* to evaluate open environment challenges in real-world relational data streams. Specifically, we investigate 55 real-world relational data streams and establish that open environment scenarios are indeed widespread, which presents significant challenges for stream learning algorithms. Through benchmarks with existing incremental learning algorithms, we find that increased data quantity may not consistently enhance the model accuracy when applied in open environment scenarios, where machine learning models can be significantly compromised by missing values, distribution drifts, or anomalies in real-world data streams. The current techniques are insufficient in effectively mitigating these challenges brought by open environments. More researches are needed to address real-world open environment challenges. All datasets and code are open-sourced in https://github.com/Xtra-Computing/OEBench.

PVLDB Reference Format:
Yiqun Diao, Yutong Yang, Qinbin Li, Bingsheng He, and Mian Lu. OEBench: Investigating Open Environment Challenges in Real-World Relational Data Streams. PVLDB, 17(6): 1283 - 1296, 2024.
doi:10.14778/3648160.3648170

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Xtra-Computing/OEBench.

*Qinbin Li is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 6 ISSN 2150-8097.
doi:10.14778/3648160.3648170

1 INTRODUCTION

Recently, the concept of *open environment* learning, where data or tasks can change over time, has been introduced by Zhou [84]. Four primary open environment challenges are identified. Firstly, the data distribution may shift owing to subtle environmental changes. Secondly, outliers or emerging new classes can occur, like new viruses or diseases. Thirdly, the feature set can evolve, with new attributes being added or existing ones being dropped due to factors such as the installation or breakdown of sensors. Lastly, the machine learning task itself may transform, e.g., from a focus on accuracy to computational efficiency. In this work, we focus on the change of data in real-world relational data streams, covering the first three challenges mentioned. The change of learning task is much more complicated and we regard it as future works.

Relational data streams have a lot of applications such as stock market and sensor data. Thus, various stream learning algorithms have been developed to obtain the insights from the data streams timely. Previous stream learning studies generally focus on drift detection, efficiency, and real-time processing [24, 31, 39, 68]. Open environment learning acknowledges that the environment can change dynamically and unpredictably over time. It recognizes a series of open environment challenges, including (1) shifting data distributions over time, (2) the emergence of outliers or new classes, and (3) incremental/decremental dimensions in feature space. Open environment challenges focus more on the model's adaptability to these changes and robustness to unforeseen situations. We list some real-world examples of open environment challenges as follow.

Air quality surveillance. Consider an air quality surveillance system. (1) Factors such as industrial emissions, vehicular traffic, and meteorological variations can change unpredictably, which can cause the challenge of distribution drifts. (2) Unexpected events, like industrial spills or large-scale fires, can introduce new types of pollutants or unprecedented pollution levels not present in the training data, leading to the challenge of outliers or new classes. (3) Technological advancements lead to newer, more accurate air quality sensors replacing older ones, causing potential changes in the metrics or pollutants being monitored. This poses the challenge of incremental/decremental feature space.

Energy prediction. In the realm of energy usage prediction, dynamic changes are common. (1) Societal behavior shifts or new industry practices can modify energy consumption or production patterns, leading to the challenge of distribution drifts. (2) Rapid technological adoption, like a surge in electric vehicle usage, or the launch of a new energy-intensive industry can introduce energy

patterns not seen during model training, thereby presenting the problem of outliers or novel classes. (3) The evolution of technology might introduce new energy sources or retire older ones. This can bring about changes in the kind of data collected, representing the challenge of incremental/decremental feature space.

However, there lacks empirical investigation of these scenarios in real-world relational data streams. Although various incremental learning algorithms have been developed [14, 52, 81], their evaluations are mostly conducted with manually partitioned datasets. *How do the open environment challenges such as distribution drifts, outliers, emerging classes, and changing features look like in real-world relational data streams? How do these open environment challenges affect the effectiveness and efficiency of stream learning algorithms?* It calls for a systematic methodology to identify the proposed three challenges and a comprehensive evaluation of existing incremental learning algorithms on real-world relational data streams.

In response, this work makes the first attempt to systematically study and benchmark the open environment challenges in real-world relational data streams, which narrows down the gap between the vision paper [84] and real-world scenarios. Specifically, we investigate 55 real-world relational data streams collected from public repositories to show that open environment challenges are widespread. We develop a scalable, open-source benchmark to quantitatively study the open environment challenges in real-world relational data streams and evaluate their magnitude. As it is rather complex to analyze all data streams, and also for the simplicity of benchmark design, we select a small number of representative datasets, according to three different open environment aspects. Then we evaluate 10 existing stream learning and incremental learning algorithms in real-world data streams. Our collected datasets cover a much wider range in the three open environment challenges than previous datasets and benchmarks. Our methodology of processing and selecting real-world relational data streams can be easily extended to future new datasets. Our evaluation framework can be applied to new algorithms.

We verify that the open environment challenges widely exist in our collected 55 datasets: 90% datasets have over 2% detected outliers; 80% datasets have over 10% windows where distribution drifts are detected with its adjacent window. 40% datasets have over 5% data items with missing values. We also observe that the model accuracy sometimes degrades significantly at the occurrence of open environment challenges. Despite the ubiquity of open environment challenges in real-world relational data streams, it is still under explored how to address them.

We further categorize our findings in two categories: similar findings to previous studies and contrary findings to previous studies. Similar to previous studies, we verify that (1) more updates (smaller batch size or more epochs) can generally improve the model accuracy [36]; (2) large models are prone to overfitting and lightweight models are recommended in relatively simple relational data streams [13, 66]; (3) distribution drifts and outliers can severely harm the accuracy of stream learning models [32]. Note that, previous studies are mostly based on synthetic data streams, or have limited coverage on real-world datasets but overlook the complex open environment challenges in different real-world data streams.

Contrary to findings of prior studies: (1) Previous studies [53, 69] show that more training data usually improves model accuracy. In our study, we find that more training data does not necessarily improve model accuracy under some open environment settings. As we observe in case studies, at the point of obvious distribution drifts or outliers, the test loss surges since old data or unreliable data harm the model accuracy. (2) Previous studies [41, 56] conclude that removing outliers helps improve model accuracy. In our study, it does not always hold in some open environment scenarios, since it is unknown whether the detected outliers are really outliers in real-world data streams. (3) Previous studies [70, 83] claim that larger exemplar storage buffer size or larger ensemble size can help improve model accuracy. However, we find such conclusion is not guaranteed in some real-world data streams with open environment challenges. Under distribution drifts, sometimes old exemplars or old models may not well adapt to new environments and lead to biased supervision. Thus, open environment problems bring great challenges in learning real-world relational data streams.

2 OPEN ENVIRONMENT CHALLENGES

Consider a data stream T. We view it as a sequence of windows $\mathbf{T} = \{T_1, T_2, ..., T_n, ...\}$. The window size is a tunable parameter. In each window T_k, we train a model f_k using only the previous model f_{k-1}, data of the current window, and a limited number of samples from previous windows (if available). In an open environment learning context, the learning objective is that the trained model can generalize well on the upcoming window T_{k+1}.

Consider the following example. Assume we want to predict air quality from observed statistics. Suppose the window size is one day. A model works for one day to perform inference on the observations. Then we update the model with observations in the past day. The goal is to maintain a good predictor that can work well on the near future data, under scenarios where the environments may change, e.g. climate change, extreme weather, sensor damage, and etc. This requires the model to well adapt to the following possible open environment challenges, including distribution drifts, outliers and incremental/decremental features.

2.1 Distribution Drifts

In discussing distribution drifts, we adhere to the definitions in Souza et al. [68]. Denote the feature and label of window k as X_k and Y_k. Distribution drifts can be categorized into three main types:

- Prior probability drift occurs when $P(Y_i|X_i) = P(Y_j|X_j)$ and $P(Y_i) \neq P(Y_j)$. It happens exclusively in $\mathcal{Y} \rightarrow \mathcal{X}$ problems (the features are dependent on the labels).
- Covariate drift occurs when $P(Y_i|X_i) = P(Y_j|X_j)$ and $P(X_i) \neq P(X_j)$. It only happens in $\mathcal{X} \rightarrow \mathcal{Y}$ problems (the labels are dependent on the features).
- Concept drift occurs when $P(Y_i|X_i) \neq P(Y_j|X_j)$.

Distribution drifts are challenging since they can cause some historical data to be misleading for current windows. For example, an environment predictor trained on statistics during summer may not generalize well in winter. A possible solution is to apply drift detectors and re-train the model after drift alerts.

2.2 Outliers or New Classes

Another challenge encountered in open environment learning is the emergence of outliers or new classes. In this context, an outlier

refers to an observation that deviates significantly from the other observations, often due to measurement error or an exception in the data. Meanwhile, a new class refers to a novel category or label that is not present in the previous windows.

The appearance of outliers or new classes can substantially harm the model accuracy. For example, an abnormal value of the target can lead to very high loss of the specific element, which could greatly affect the model parameters and lead to poor accuracy. To effectively manage this challenge, the learning model needs to be capable of identifying these outliers or new classes and adapt accordingly, in an online fashion under possible open environment challenges [85].

2.3 Incremental/Decremental Features

Traditional machine learning methodologies are based on the assumption that all samples reside in the same feature space. Open environment learning challenges this assumption, considering instead a dynamic feature space that may incrementally expand or decrementally shrink. In other words, new data may come with additional features (incremental features) or lack some of the previously observed features (decremental features).

Many existing studies have addressed the issue of missing values in machine learning, yet they often operate under the assumption that the entire feature space is known. This is not the case in open environment learning, where incremental or decremental features in the data stream are unexpected. This difference creates a challenge, as the model trained on previous data would have been based on a different feature space.

Decremental features can be addressed by filling missing values. However, incremental features are difficult to address since the model does not account for the incoming feature. One simple approach to dealing with incremental features is to discard them, although this strategy leads to an under-utilization of potentially valuable information. An alternative solution is to retrain the model to incorporate the new features, at the risk of causing the model to forget previously acquired knowledge.

3 RELATED WORKS

3.1 Incremental Learning Algorithms

Most previous studies on incremental learning [14, 52, 81] can be formulated as learning on a data stream T divided into a sequence of windows $\mathbf{T} = \{T_1, T_2, ..., T_n, ...\}$. All windows are non-overlapping with different classes. In each window T_k, the task is to train a model f_k using only the previous model f_{k-1}, data of the current window, and a limited number of exemplars (if available). The goal of incremental learning is to learn a good model f_k working well on seen classes in $T_1, T_2, ..., T_k$.

However, incremental learning methods may not be suited to the open environment context, as the goal of open environment learning is to work well for near future window T_{k+1} where unpredictable changes can happen. Such changes could render old data unsuitable for a new environment.

From the perspective of regularization, EWC [37] penalizes the change in crucial parameters based on the Fisher Information Matrix. LwF [42] integrates the prediction of the previous model, which reduces the over-confidence towards seen classes of the current window. From the perspective of storing exemplars, iCaRL [61] selects exemplars that are close to the mean representation of each class. From the perspective of parameter isolation, PackNet [50] prunes the neural network for new windows while keeping important parameters frozen. More detailed discussions are in Appendix A.2 of our full version [17].

A summary of incremental learning works is in Table 1. As we can see, none of these incremental learning algorithms are designed to tackle changing feature spaces or outliers. Some algorithms are even inapplicable to real-world relational data streams due to the specific design for image, requiring auxiliary datasets or no scalability towards infinite streams. Therefore, prior incremental learning algorithms can hardly address open environment challenges in real-world relational data streams.

Table 1: Summary of incremental learning works and difficulties in adapting to real-world relational data streams. ✓ and ✗ indicate whether the methods consider each open environment challenge in real-world relational data streams.

Learn from	Difficulties	Incremental/ decremental features	Drifts	Outliers
Critical parameters [2, 14, 37, 81]	N/A	✗	✓	✗
Prior model outputs [35, 42, 72]	N/A	✗	✓	✗
Prior model outputs [16, 67, 77]	Specific for image data	✗	✗	✗
Prior model outputs [82]	No auxiliary dataset	✗	✗	✗
Stored exemplars [14, 33, 61, 78, 79]	Not designed for regression tasks	✗	✓	✗
Fixed model part [3, 50]	Not scalable for infinite streams	✗	✗	✗

3.2 Incremental Learning Benchmarks

The evaluation of incremental learning techniques has led to various benchmarks. For example, Masana et al. [52] compare 13 incremental learning algorithms. They divide the dataset into several windows by class, and evaluate the average accuracy across all previously seen classes in each window. Similar settings are widely adopted by incremental learning works [14, 52, 81].

Besides splitting the datasets, other evaluation methods include permuting or rotating image datasets to construct drifts or introduce new classes [20, 47, 49]. Real-world video datasets are also proposed for incremental learning [46, 63, 65]. However, these datasets merely simulate emerging classes and drifts in open environment challenges, and do not consider relational data streams.

For real-world relational data streams, there are few prior benchmarks. Bifet et al. [10] proposes a stream learning system MOA, but it only has five real-world stream datasets. Souza et al. [68] proposes to conduct experiments on insects and alter the temperature to simulate drifts. Both balanced and imbalanced insect classes are designed, and raw data are recorded to form new data streams. The metric is prequential accuracy, which means first testing and then training each window of data.

We summarize the data streams used in previous incremental learning works in Table 2. As we can see, none of prior works consider all the three aspects of open environment challenges in real-world relational data streams. The first two rows are incremental learning benchmarks on computer vision datasets, which consider neither relational datasets nor incremental/decremental feature space challenges. USP DS Repository [68] proposes real-world relational data streams, however it only explores the distribution drift challenge. Our OEBench considers three categories of open environment challenges in real-world relational data streams and covers a total of 55 datasets, which will be introduced in detail in the following section.

Table 2: Summary of the datasets explored in various incremental learning works.

Paper	#Datasets	Real-world relational data	Outliers	Drifts	Changing feature space
[20, 47, 49, 52, 81]	16	✗	✗	✓	✗
[46, 63, 65]	3	✗	✓	✓	✗
[68]	27	✓	✗	✓	✗
Ours	**55**	✓	✓	✓	✓

4 DESIGN OF OEBENCH

4.1 Design Goals

Our OEBench is crafted according to the four benchmark design principles posited by Gray [27].

Relevance. Our benchmark covers a broad range of open environment challenges including distribution drifts, outliers, and feature space evolution. We have selected 55 real-world relational data streams from diverse sources, including UCI Datasets, Kaggle Datasets, and the USP DS Repository [68]. These datasets cover a variety of contexts on real-time stream learning, including environment surveillance, sales prediction, fraud detection, and etc.

Simplicity. With an aim to identify typical open environment patterns and eliminate redundant testing, we choose five representative datasets for our benchmark. We first extract statistics on the three perspectives of the open environment challenges. Then we apply clustering and select the datasets nearest each cluster center to conduct further experiments.

Portability. Our benchmark is applicable to new relational data streams. The pipeline for extracting open environment statistics and evaluating stream learning algorithms can be easily invoked or adapted in new systems.

Scalability. The benchmark incorporates real-world relational data streams of diverse sizes (Table 3) and varying open environment contexts (Figure 3). This allows for the simulation of multiple incoming data sizes and evolving patterns. Through this variation, we can test the performance of different algorithms under varying open environment conditions.

4.2 Overview

Based on the design goals, we build our benchmark as depicted in Figure 1. It composes of six parts: dataset collection, preprocessing, extracting open environment statistics, visualization, representative dataset selection and incremental learning algorithm evaluation.

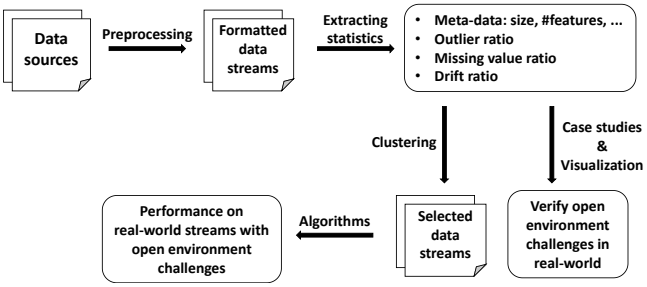


Figure 1: Open environment benchmark flowchart.

Dataset collection. We first search the candidate datasets in public relational dataset resources including the UCI Machine Learning Repository, Kaggle, and the USP DS Repository [68]. Then we keep the datasets with enough rows, columns and meaningful real-world scenarios. We finally keep a total of 55 datasets.

Preprocessing. We record dataset metadata, discard textual features, sort the datasets by time, normalize each feature, and set the target and window size to assign a machine learning task for each dataset. This aims to transfer raw datasets into structured relational data streams for further processing.

Extracting open environment statistics. We design criteria to evaluate the extent of three open environment challenges in each dataset. Specifically, statistics related to distribution drifts, outliers, and missing values are calculated. Since there are no ground truth label for drifts and outliers in the real-world datasets, we apply ensemble of drift detectors or outlier detectors to estimate their ratios.

Visualization. To help data scientists to easily understand the open environment challenges in real-world relational data streams, we conduct visualization based on our extracted open environment statistics. Visualization also serves as a validation of detected drifts and outliers due to the lack of ground truth. Data distributions are plotted in scatter plots to illustrate distribution drifts and outliers. High-dimensional features are reduced to 2-D space by t-SNE [73] for visualization.

Representative dataset selection. As 55 real-world datasets are too many for further experiments, we propose a clustering approach to select a limited number of representative datasets. Specifically, based on the extracted open environment statistics and dataset metadata statistics, we conduct K-means clustering on these information. Inside each cluster, the dataset closest to each center is selected as the representative dataset for the cluster.

Incremental learning algorithm evaluation. Based on the selected representative datasets, we evaluate the effectiveness and efficiency of 10 incremental learning algorithms under different open environment scenarios. We also evaluate factors such as window size, batch size, epochs, model size on incremental learning under open environment challenges.

4.3 Dataset Collection

In order to thoroughly investigate open environment learning scenarios on real-world relational data streams, we select the datasets with the following criteria.

Table 3: Histogram information of the collected datasets.

Size	5,000-20,000	20,001-50,000	50,001-200,000	>200,000
#Datasets (USP DS)	1	2	4	9
#Datasets (ours)	**13**	**17**	**13**	**12**
#Features	5-10	11-20	21-50	>50
#Datasets (USP DS)	3	0	12	1
#Datasets (ours)	**15**	**23**	**14**	**3**

- The sample size must exceed 5,000.
- The features should be relational data streams (excluding text data such as emails) with at least 5 features. The feature dimension should not exceed 1,000 after one-hot encoding. This prevents huge computational burdens.
- The stream learning scenario should be meaningful. For instance, we exclude the PokerHand dataset as the randomness of each hand undermines meaningful stream analysis.

Upon applying these selection criteria to potential datasets from sources including the UCI Machine Learning Repository, Kaggle, and the USP DS Repository [68], we get a collection of 55 publicly available real-world relational data streams that fulfill all the requirements. As shown in Table 3, our collection covers a vast range in terms of data size and feature size. Notably, we have over three times the number of real-world relational data streams compared to the qualified datasets in USP DS Repository, which holds the record for the most comprehensive benchmark with its collection of real-world relational data streams. Detailed dataset descriptions are listed in Appendix C of our full version [17].

4.4 Extracting Open Environment Statistics

Real-world relational data streams display a variety of formats, feature spaces, and scales. To extract open environment statistical features from them, we implement the following preprocessing procedures:

(1) Document dataset metadata, such as task, feature type, target, window size, and null indicators.
(2) Order instances by time, then remove time-related attributes to maintain the temporal context without interfering with the dataset's statistical characteristics.
(3) Employ one-hot encoding to convert categorical features into numerical format.
(4) Utilize KNNImputer to fill in missing values, due to its generally reasonable and bounded predictions. The default value of k is set to 2.
(5) Normalize each feature within the dataset.
(6) Partition the dataset into non-overlapping windows to facilitate temporal analysis. The size of these windows is determined based on the time span and specific characteristics of each dataset.

For datasets consisting of multiple tables, we focus solely on the main table as integrating multiple tables significantly complicates the analysis. For instance, in the fraud detection dataset [34], many transaction records lack associated client identity information. Similarly, in loan risk prediction [54], some clients may have no prior loan applications, while others may have multiple application histories. These cases can lead to data items possessing highly varied feature sizes. We leave these complex cases as future works.

Upon completion of preprocessing, we compute the statistics related to distribution drifts, outliers, and missing values for each dataset, in accordance with the open environment challenges proposed in Zhou [84]. Both global and per-window statistics are documented. For per-window statistics, we record both the maximum and average values across all windows as open environment statistics of the dataset.

Distribution Drifts. We detect data drifts using HDDDM [18], KdqTree [15], CDBD [44], PCACD [60] and the Kolmogorov-Smirnov (KS) test. We apply these methods since they are implemented in an open-source library Menelaus and easy to use. KS statistic is selected because it has statistical guarantee about the drift. While HDDDM and KdqTree are applicable to multi-dimensional datasets, the remaining three methods must be applied to each dimension separately. We document the results of all methods as open environment statistics. By default, we utilize the first two principal components in PCACD and set $p = 0.05$ for the KS test. Elaborations on drift detectors are in Appendix A.3 of our full version [17].

For each algorithm, we document the drift and warning percentages. The average and maximum percentages are stored as a global feature for each dataset. For one-dimensional drift detectors, we note the average and maximum percentages across all columns.

Regarding concept drifts, we employ DDM [22], EDDM [6], ADWIN accuracy [8], and PERM [30]. Similarly, the first three algorithms can be called from Menelaus library. PERM [30] is additionally selected since it is applicable to regression tasks. Following the examples in Menelaus, we use Gaussian Naive Bayes for classification tasks and linear regression models for regression tasks. In each window, the predictions are compared with the ground truth to detect concept drifts. Upon the detection of a drift, the model is retrained with the most recent data slices. Similarly, we store the drift and warning percentages as open environment statistics.

Outliers. There are a lot of outlier detection algorithms in ADBench [29]. We follow the recommendation in ADBench [29] to choose ECOD [43] and IForest [45] to detect outliers. ECOD [43] estimates the underlying distribution of input data to detect outliers in tail probabilities in each dimension. IForest [45] randomly selects an attribute and makes a binary split. The binary tree partition process is conducted recursively to identify easily-isolated data points as outliers. Within each window, we detect outliers by setting the threshold at three standard deviations above the mean score. We then calculate the average and maximum anomaly ratios within each window as open environment statistics of the dataset.

Missing Values. We compute the following statistics about missing values.

(1) **Ratio of data items with missing values**: This measures the completeness of the data items for each row.
(2) **Ratio of missing columns**: This measures the completeness of the feature dimensions for each column.
(3) **Ratio of empty cells**: This measures the completeness of cells within the dataset for the entire table.

Validation. The statistics for missing values are straightforward. To validate the statistics for drifts and outliers, we use generated datasets with different levels of drifts and outliers. We generate a data stream of 50,000 samples and partition it into 100 windows. By default, each sample contains three dimensions f_1, f_2, f_3 sampled uniformly in (0,10).

For concept drifts, we use the SEA concept generator [9]. We divide the dataset into 4, 8 or 16 blocks with different concepts $f_1 + f_2 \leq \theta$, with binary classification threshold $\theta \in \{9, 8, 7, 9.5\}$. For 8 blocks, the threshold values are $\{9, 8, 7, 9.5, 9, 8, 7, 9.5\}$, and similarly for 16 blocks, we repeat the threshold values for four times. Our average concept drift statistics for 4, 8, 16 different concepts are 0.000114, 0.000229, 0.000263 respectively, which complies with the frequency of concept drifts.

For data drifts, similarly, we divide the dataset into 4, 8 or 16 blocks, each with features generated in $(0, \phi)$ where $\phi \in \{10, 9, 11, 7\}$. Our average data drift statistics for 4, 8, 16 different blocks are 0.289, 0.322, 0.356 respectively, which is in accordance with the frequency of data drift.

For outliers, we randomly change 400, 800 or 1600 data samples to outliers in the stream, by randomly adding or deducting 20 in their feature values. Our average anomaly statistics for 400, 800, 1600 outliers are 0.0088, 0.0161, 0.0320 respectively, which conforms to the expected order. The above experiments verify that our extracted statistics are reasonable.

4.5 Selection of Representative Datasets

Since it is expensive to evaluate existing algorithms on all collected 55 datasets, we pick up representative datasets for further exploration. To select representative datasets, we first normalize all open environment statistics and dataset metadata statistics to the same scale. Then for each category of statistics (dataset basic information, missing values, data drifts, concept drifts, outliers), we transform them into 3-D space using Principal Component Analysis (PCA). This transformation ensures that each category is represented with the same number of statistics. We then apply K-means clustering to partition the datasets into five distinct clusters to group the datasets with similar characteristics. The datasets nearest to each cluster center are selected as representative datasets.

Figure 2 visualizes the clustering results on three open environment dimensions. Missing value ratio can be directly calculated by counting the ratio of missing items. Drift ratio is calculated by the ratio of both data drifts and concept drifts detected between consecutive two windows. Anomaly ratio is calculated by the ratio of detected anomalies among each window. Both drift and anomaly do not have ground truth and are calculated through ensemble of detectors.

As we can see, red, purple and cyan represent high missing value ratio, high drift ratio and high anomaly ratio respectively. Blue and green seems mixed, but blue represents datasets of commercial field while green represents datasets of nature science field, since we also take the task, field, dataset size into consideration when clustering. Table 4 provides details of the five selected datasets, each showcasing unique characteristics.

We use the results of K-means clustering due to its popularity and simplicity. Besides K-means, we also try other clustering algorithms

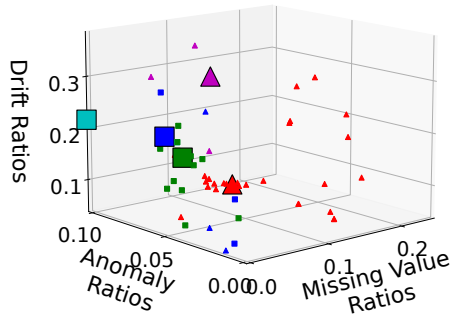


Figure 2: Visualization of clustering results on three open environment dimensions. Large points are selected datasets. Square represents classification task and triangle represents regression task.

including spectral clustering [74], agglomerative clustering [55], and Gaussian mixture model (GMM) [62]. The number of clusters remains five. We find out that Beijing Air Quality (Shunyi), Room Occupancy and Electricity Prices appear in the selected datasets by at least three out of four clustering algorithms. Besides, the results of all four algorithms contain one of the INSECTS dataset. The consensus of different clustering algorithms indicate that those datasets are representative.

Figure 3 presents a box plot of open environment statistics from four data sources: six popular synthetic datasets (SEA, STAGGER, Rotating Hyperplane, RBF, LED, and Waveform) in prior stream learning works [9], the USP DS Repository [68], our collected datasets, and our selected representative datasets. As shown in Figure 3, our 55 collected datasets OE-All cover a much broader spectrum of open environment statistics than prior synthetic datasets and the real-world benchmark USP DS Repository. It is particularly evident in the aspect of missing values, where synthetic datasets and USP DS Repository do not explore. Further, OEBench representative datasets emulate the distribution of the 55 OE-All datasets, which also span a diverse range of open environment scenarios.

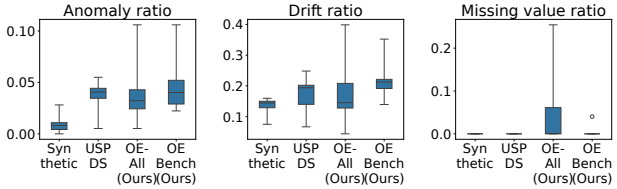


Figure 3: Statistical distribution of synthetic datasets, USP DS Repository and our datasets. We denote all 55 datasets as *OE-All* and the selected five datasets as *OEBench*. In missing value ratio, 1 out of the 5 datasets in OEBench contains missing values, which is the individual circle in the box plot.

Table 4: Five selected representative datasets. The levels of three open environment challenges are determined by the extracted statistics compared with the average statistics of all 55 datasets.

Dataset	Default window size	Instances	Features	Type	Task	Missing value ratio	Drift ratio	Anomaly ratio
Room Occupancy Estimation	2 hours	10,129	6	Others	Classification	Low	Medium high	High
Electricity Prices	2 weeks	45,312	7	Commerce	Classification	Low	Medium high	Medium high
INSECTS-incremental-reoccurring (balanced)	100 items	79,986	33	S&T	Classification	Low	Medium low	Medium high
Beijing Multi-Site Air-Quality Shunyi	30 days	35,064	11	Ecology	Regression	High	Low	Medium low
Power Consumption of Tetouan City	15 days	52,417	7	Power	Regression	Low	High	Medium low

4.6 Selected Incremental Learning Algorithms

We select representative incremental learning algorithms from the perspectives of regularization and experience replay. Besides, we also explore tree-based and ensemble-based algorithms for real-world relational data stream learning.

Regularization. We implement Elastic Weight Consolidation (EWC) [37] and Learning without Forgetting (LwF) [42], considering their popularity and ease of implementation.

Experience replay. We implement iCaRL [61] since it is popular and can be extended to infinite data streams.

Tree model. We apply the Adaptive Random Forest (ARF) [25]. In this approach, each tree is trained with a subset of features and is subjected to drift detection. After drift is identified, a background tree is trained to replace the current tree.

Ensemble. We choose the Streaming Ensemble Algorithm (SEA) [70]. SEA maintains an ensemble and replaces older models with current models of better quality.

5 CASE STUDIES OF OPEN ENVIRONMENT SCENARIOS

In this section, we visualize and analyze some real-world relational data streams. Through these cases, we study how the open environment challenges look like in real-world relational data streams, and provide a deeper understanding of their implications.

5.1 Distribution Drifts

Distribution drifts can be classified into two categories: data drifts and concept drifts. Data drifts refer to changes in the distribution of the feature set. Given the dynamic nature of most real-world data, such drifts are common. Figure 4 provides a clear illustration of cyclical data drift of air quality surveillance, occurring approximately on an annual basis. This cyclical pattern aligns with the inherent seasonality of the dataset. Similar patterns are observed across other air quality datasets as well.

On the other hand, concept drifts imply changes in the environment. For example, the decision boundaries of different months clearly differ in Figure 4, probably due to different weathers in different seasons.

Impact of distribution drifts. In Figure 4, we observe that distribution drifts happen around window 7, 11, 19, 23, 32, 35. To investigate the impact of drifts on incremental learning on real-world data streams, we train a decision tree on (1) the first 11 windows, and (2) window 7 to 11 of the air quality dataset in Tiantan. Both models are tested on the next window 12. The test loss of training on all first 11 windows is 0.347, while only 0.299 for training on window 7 to 11. Considering the distribution drifts at around window 7, we can conclude that including historical data with different distributions harms the model accuracy towards new data distributions, since memorizing old data cannot well adapt to the new environment.

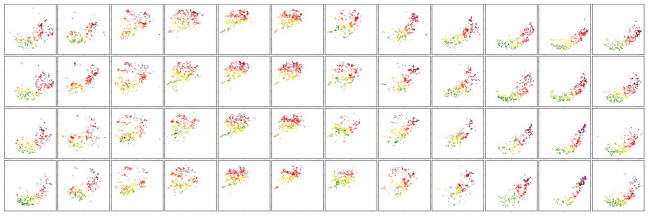


Figure 4: T-SNE visualization of air quality dataset in Tiantan, Beijing. Each line spans a year and each sub-figure represents a month. Data items are labelled according to 6 categories of AQI based on the severity of health concern.

We also train a decision tree and a neural network on the air quality dataset in Tiantan. The test loss is shown in Figure 5. We mark the windows around drift occurrences by vertical lines, where we can clearly see the sudden increase of test loss. Under distribution drifts, old knowledge from the past data can have a negative impact for models to better adapt to the new environment. Such scenarios contradict with the goal of prior incremental learning works to perform well on all seen data. This implies that prior incremental learning algorithms may not work well on real-world open environment learning scenarios.

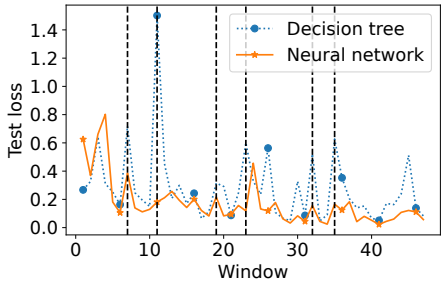


Figure 5: Test loss of training a decision tree and a neural network on Beijing air quality dataset, Tiantan. Vertical lines denote windows around the happening of distribution drifts.

5.2 Outliers

We conduct case study on the Beijing PM2.5 dataset (from five Chinese cities PM2.5 dataset). Our analysis of outliers focuses on

two extreme weather events in Beijing: the flood of July 21, 2012, and the period of intense haze from 2014 to 2015.

Both iForest and ECOD are applied to these datasets, and they yield similar outcomes, especially in identifying the two extreme weather events. We present the visualization of the two events in our full version [17].

Outliers can bring great challenges to data stream learning. In Figure 6, we locate the mentioned extreme weather situations with vertical lines. As we can see, after the abnormal situations, the model test loss increases. Especially in Beijing 2012 flood incident where the outliers happen in an abrupt way, we can witness a spike in the test loss (after the left vertical line). As machine learning models are data-driven, outliers may significantly harm the model since it can greatly affect the loss.

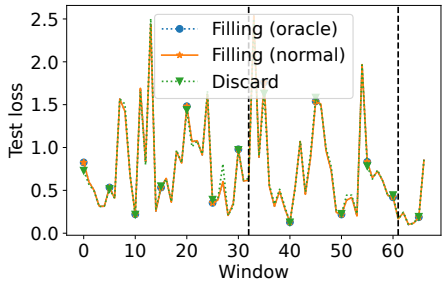


Figure 6: Test loss of training a neural network on the Beijing PM2.5 Data of Five Chinese Cities dataset. Vertical lines are extreme weather events. "Filling (oracle)" means filling up missing values with the knowledge of the whole dataset. "Filling (normal)" means filling only with the knowledge of data in its current window. "Discard" means to discard the three frequently missing features.

The most extreme case in this dataset is the No. 51,278 data, where the precipitation becomes 999,990. The normal range of precipitation is 0-100. We are unable to plot this problem in Figure 6 since the test loss of the neural network suddenly becomes very large or even infinite in the following windows. It illustrates the vulnerability of neural networks when encountering outliers. On the decision tree, we also observe a 3x spike of the test loss after the No. 51,278 data, but it does not crash. Even a single unreliable data item can corrupt the model. Thus, in open environment scenarios, it is an important and challenging problem to detect outliers to reduce their harm to stream learning models.

5.3 Incremental/Decremental Features

Incremental or decremental features is widespread in real-world relational data streams, especially in ecology datasets where sensors are deployed to collect data. For instance, all the air quality datasets in our study have incremental or decremental features, due to the malfunction, repair, removal, or installation of sensors.

Figure 7 shows the ratio of missing values of two features in the Beijing PM2.5 dataset (from five Chinese cities PM2.5 dataset). The appearance of filled missing values is associated with an incremental feature space, while missing values in an attribute correspond to a decremental feature space.

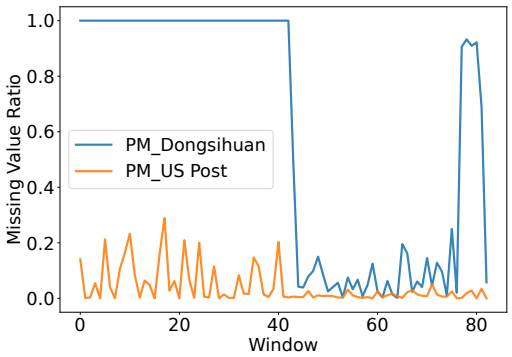


Figure 7: Ratio of missing values per window in the air quality data in Beijing, PM2.5 Data of Five Chinese Cities dataset.

Prior missing value filling studies [19, 59] mainly focus on cases like the orange line, where the model knows the existence of the feature and the ratio of missing values is small. However, open environment challenges include scenarios like the blue line. In the first 40 windows, the model is unaware of the existence of such feature. The sudden appearance of the feature leads to incremental feature space, which is rarely considered in prior works. Such scenarios can result from installing new sensors. This is more challenging due to the unpredictable changes.

To further explore the impact of incremental and decremental features on data stream learning, we train a neural network on the dataset with different methods of processing the incremental and decremental features. The setups are elaborated in Section 6.1. As shown in Figure 6, we observe an interesting phenomenon that discarding these always-missing features have similar test loss compared with filling them up, which shows that more data does not necessarily lead to better model accuracy in some real-world data streams.

In reality, if a sensor always fails to record values, its recorded values are more likely to be unreliable. Unreliable data can harm the model accuracy, which is also discussed in Section 5.2. Due to these complex scenarios in real-world data streams, open environment machine learning tasks are quite challenging.

6 EXPERIMENTS

6.1 Setups

Models. We adopt a multi-layer perceptron (MLP) as our default neural network (NN) architecture, which includes three hidden layers consisting of 32, 16, and 8 neurons respectively with ReLu activation. For tree-based models, we adopt decision tree or GBDT with an ensemble of five trees by default. For advanced models on relational datasets, we test TabNet [5] and ARM-Net [12].

Hyper-parameters. For each window, we train NN for 10 local epochs with batch size 64 and learning rate 0.01. The buffer size is set to 100. For EWC, we observe that regularization factors below 10^3 yield results akin to naive NN, while factors exceeding 10^5 can lead to loss explosions. Therefore, we tune the factor in $\{10^3, 10^4, 10^5\}$.

For LwF, since the regularization loss is in the similar order of magnitude with the naive NN loss, we tune the regularization factor in $\{0.01, 0.1, 1\}$. Missing values are handled by employing the KNNImputer with $k = 2$. We use an ensemble of five models by default for ARF and SEA.

Algorithm implementations. Given that real-world data streams may be infinite, it is unfeasible to keep models from all windows in EWC and LwF. As an alternative, we employ the model from the most recent window to perform regularization. Additionally, LwF and iCaRL are originally designed for classification tasks. To adapt them to regression tasks, we substitute the LwF regularizer with mean square error (MSE) loss and consider all data items as a single class for iCaRL. To normalize each feature, we use the mean and variance of the first window to rescale each dataset dimension. The purpose is to simulate the real-world scenarios where only the statistics of early samples are available to get started.

Metrics. Our evaluation methodology employs a test-then-train paradigm within each window. Aside from the initial warm-up window, we test the model on the data of each window, followed by training on these data to update the model. We calculate the error rate for classification tasks and MSE loss for regression tasks. The final error rate or MSE loss is determined by averaging the results across all windows.

Hardware setups. All experiments run on a machine with 64 Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz, 376 GB memory, and we use 2.7 TB hard disk as storage. The OS is Ubuntu 22.04.3 LTS.

6.2 Evaluating Existing Algorithms on Real-World Relational Data Streams

Finding (1): There is no silver bullet that performs well in all cases of open environment challenges. Therefore, open environment challenges are difficult to deal with and previous algorithms are not specifically designed to address them.

We compare the model accuracy of existing algorithms on real-world relational data streams as shown in Table 5. Besides the five representative datasets, we also experiment on the other 50 datasets of our benchmark to enhance our assessment of the strengths of these algorithms. The full results are shown in Appendix B.1 of our full version [17]. From the results, it is evident that no single algorithm consistently outperforms the others across all cases of open environment challenges. Based on the results of all 55 datasets, we synthesize our recommendations for different contexts into a decision tree, which can be viewed in Figure 8.

In terms of effectiveness, tree models are generally recommended in classification tasks with low anomalies and regression tasks with high missing values. NN models are recommended in regression tasks with low missing values. Another work [66] also verifies that tree models work better than NN models in classification tasks on tabular datasets. NN models are generally over-parameterized than tree models, which results in NN models prone to overfitting in simple tasks like tabular dataset classification. Regression tasks generally require the model to learn more complex functions compared with learning decision boundaries in classification tasks, which is probably the reason why over-parameterized NN models have better accuracy in regression tasks with low missing values. However, NN models are sensitive to the variations of input data [26, 40], therefore tree models are recommended if there are many missing values. When time or memory constraints are tight, tree models are better suited due to their efficiency, as discussed in the following section.

Among NN-based methods, naive NN and iCaRL typically outperform other NN-based techniques in classification tasks with high drifts. ICaRL is designed for classification tasks to store exemplars of each class, and it has the advantage of alleviating forgetting when the drift is relatively high. It also works well on regression tasks with high missing values. For data streams with relatively low drifts, we recommend naive NN. For regression tasks with low missing values, naive NN and SEA-NN tend to outperform other methods.

Among tree-based methods, GBDT and SEA-GBDT yield the best accuracy under high drifts. When the drifts are relatively low, SEA-DT is recommended. SEA-DT also works well in classification tasks with high drifts.

6.3 Time and Memory Consumption

On efficiency, we compare the throughput in Table 6 and memory consumption in Table 7. Decision trees have much higher throughput and lower memory costs than NN-based methods. ARF is very bad since it detects drifts in the background, which takes too much computation and memory. Therefore, we recommend to use DT or GBDT when time or memory constraints are tight.

When considering both effectiveness and efficiency, EWC, LwF and ARF can be excluded as suitable choices for these explored real-world data streams. EWC and LwF have marginal improvement on training a naive NN while doubling the computational costs, which illustrates that simply applying incremental learning algorithms does not necessarily work well in open environment learning tasks. ARF incurs significantly longer computation time, ranging from 30 to 1,000 times longer than other tree-based algorithms, without delivering a significant boost in accuracy. Therefore, these methods are not well-suited to real-world relational data streams.

6.4 Challenges of Distribution Drifts

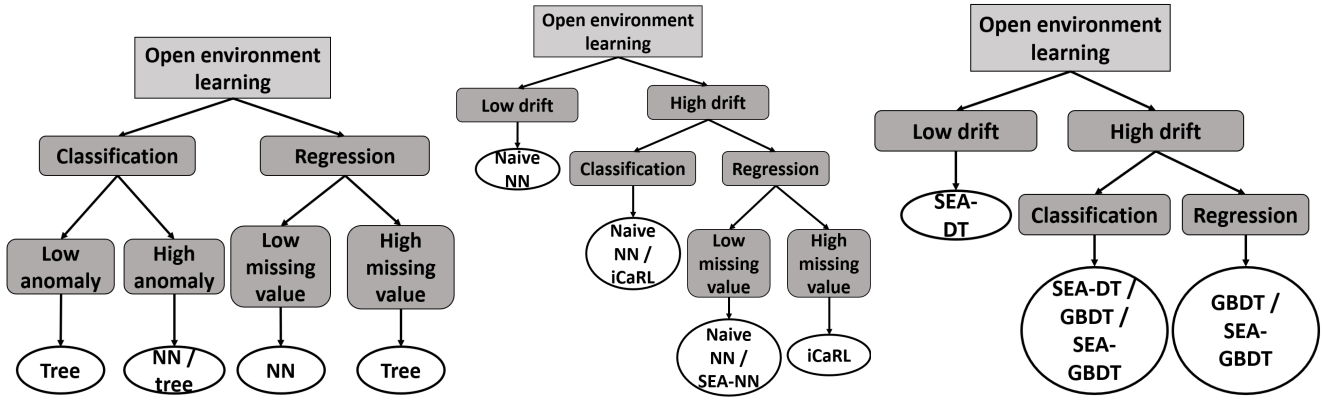
Finding (2): Distribution drifts significantly degrade the effectiveness of stream learning algorithms. NN-based algorithms can better adapt to drifts than tree-based algorithms.

In this section, we explore the challenges of distribution drifts by comparing the accuracy of stream learning algorithms on drifted datasets and non-drifted datasets. The non-drifted datasets are constructed by randomly shuffling the original datasets. We test the recommended algorithms on the ROOM and AIR dataset, and results are shown in Figure 9. Drifted datasets lead to spikes in the test loss, while we can witness steady loss decrease in non-drifted datasets. This illustrates the great challenges of the distribution drifts in real-world data streams.

A difficulty to address distribution drifts in real-world data streams is that there is no ground truth of the drift occurrences. Therefore, it is challenging to compare among drift detectors on real-world data streams. In our study, we directly compare the test loss of all windows. A stream learning algorithm handling drifts well can quickly adapt to new environment to achieve a lower loss. In Figure 9, we find out that the selected NN-based algorithms can

Table 5: Test loss / test error of stream learning algorithms on different characters of real-world datasets. Lower value indicates better result. We repeat all experiments for three times with different random seeds.

Dataset	Naive-NN	EWC	LwF	iCaRL	SEA-NN	Naive-DT	Naive-GBDT	SEA-DT	SEA-GBDT	ARF
ROOM	0.214±0.004	0.207±0.003	0.207±0.014	**0.136±0.021**	0.207±0.037	0.198±0.006	0.181±0.004	0.191±0.004	**0.151±0.002**	0.250±0.004
ELECTRICITY	0.311±0.012	0.311±0.012	0.311±0.012	**0.286±0.013**	0.332±0.022	0.272±0.001	0.256±0.000	0.263±0.009	0.264±0.001	**0.250±0.002**
INSECTS	**0.269±0.006**	**0.269±0.006**	**0.269±0.006**	0.306±0.005	0.321±0.007	0.329±0.001	0.306±0.000	**0.291±0.004**	**0.291±0.002**	0.294±0.001
AIR	**0.166±0.002**	**0.166±0.002**	**0.166±0.002**	0.182±0.008	0.213±0.023	0.263±0.013	0.498±0.002	**0.199±0.010**	0.519±0.003	N/A
POWER	0.793±0.005	0.794±0.004	**0.779±0.004**	0.818±0.014	0.783±0.015	1.278±0.003	**0.800±0.000**	0.845±0.007	0.835±0.002	N/A



(a) NN vs tree (b) NN models (c) Tree models

Figure 8: Recommendations of the (almost) best algorithm in different cases of open environment learning.

Table 6: Throughput of explored stream learning algorithms on selected real-world datasets. All experiments run on 4 CPU threads. For NN-based methods, we set the default number of epochs as 10.

Dataset	Naive-NN	EWC	LwF	iCaRL	SEA-NN	Naive-DT	Naive-GBDT	ARF	SEA-DT	SEA-GBDT
ROOM	8,168	3,376	4,646	4,185	7,083	**202,580**	44,039	234	27,375	27,375
ELECTRICITY	8,375	3,437	4,800	4,572	7,501	**266,541**	71,923	320	50,912	28,814
INSECTS	6,124	2,630	3,692	3,372	5,052	**55,545**	3,146	44	7,180	1,713
AIR	7,861	3,770	5,118	4,466	7,381	**48,032**	32,466	N/A	20,871	15,311
POWER	9,260	4,973	6,315	5,747	9,006	**169,087**	134,402	N/A	119,129	91,959

Table 7: Memory consumption (KB) of explored stream learning algorithms on selected real-world datasets.

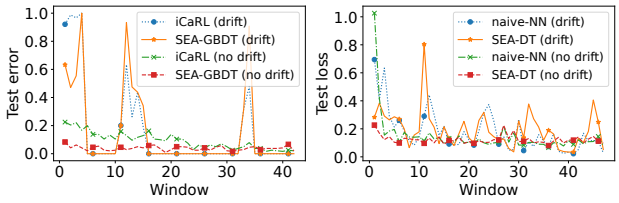
Dataset	Naive-NN	EWC	LwF	iCaRL	SEA-NN	Naive-DT	Naive-GBDT	ARF	SEA-DT	SEA-GBDT
ROOM	22.7	50.8	45.3	23.4	106.6	1.9	6.6	228.1	4.2	20.1
ELECTRICITY	22.7	50.9	45.4	23.1	106.6	1.9	6.4	961.4	4.2	19.7
INSECTS	22.7	50.9	45.4	23.9	106.6	2.0	6.8	2,223.2	4.3	21.2
AIR	22.7	50.9	45.4	22.8	106.6	1.7	6.1	N/A	3.3	18.6
POWER	22.7	50.9	45.4	22.8	106.6	1.7	6.0	N/A	3.3	18.6

better adapt to drifts than tree-based algorithms, generally having lower loss at the drift occurrence points. A possible explanation is that neural networks are over-parameterized and can quickly converge to a good solution in the new environment.

6.5 Challenges of Outliers

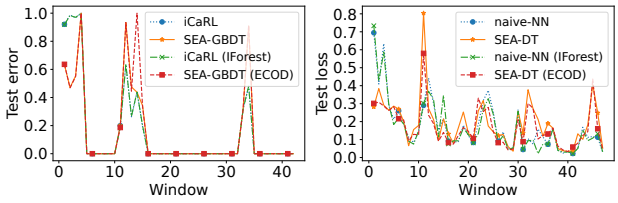
Finding (3): Among streaming outlier detectors, RShash and HSTree are recommended in data streams with more anomalies, while xStream is recommended in data streams with fewer anomalies. Removing detected outliers does not necessarily improve accuracy in open environment scenarios.

As discussed in Section 5.2, even a single outlier can destroy the stream learning model. Therefore, it is crucial to detect and remove outliers in data streams. However, in real-world data streams, the



(a) ROOM (b) AIR

Figure 9: Test loss / error curves on the ROOM and AIR dataset. Drift means the original dataset, while no drift means randomly shuffling the dataset to eliminate drifts.



(a) ROOM (b) AIR

Figure 10: Test loss / error on the ROOM and AIR dataset. ECOD and IForest mean removing outliers with the detector.

lack of ground truth makes it challenging to compare streaming outlier detectors. To address this challenge, we mark the outliers by the ensemble of anomaly detectors on the whole dataset, and test streaming outlier detectors with the marked ground truth.

To mark the outliers in the whole dataset, we apply the ensemble of ECOD [43] and IForest [45], as they are recommended by a popular benchmark [29]. We evaluate the five multivariate streaming outlier detectors in StreamAD library: xStream [51], RShash [64], HSTree [71], LODA [58], and RRCF [28]. Results are shown in Table 8. We find that RShash and HSTree work better in data streams with relatively more anomalies, while xStream is recommended when the data streams have relatively fewer anomalies. When designing multivariate streaming outlier detectors, it may be good to consider applying dimensionality reduction or ensemble. Dimensionality reduction helps filter out noisy features and detect the abnormal features. Ensemble can boost accuracy from models of different perspectives. A contemporary work [85] proposes METER, a state-of-the-art method to conduct online outlier detection under distribution drifts, which will be studied and integrated into our benchmark in the future.

Table 8: AUROC of streaming outlier detectors.

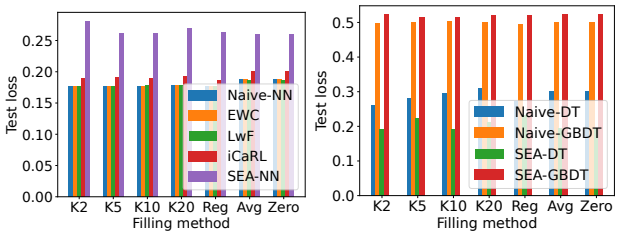
Anomaly statistic	Dataset	xStream	RShash	HSTree	LODA	RRCF
High	ROOM	0.941	**0.967**	0.948	0.214	0.592
Medium high	ELECTRICITY	0.675	**0.862**	0.851	0.632	0.667
Medium high	INSECTS	0.842	0.927	**0.965**	0.807	0.678
Medium low	AIR	**0.884**	0.707	0.649	0.562	0.599
Medium low	POWER	**0.879**	0.784	0.829	0.499	0.570

We also compare the test loss on whether to remove detected outliers or not before training. We experiment on the ROOM and AIR dataset using recommended algorithms. Results are shown in Figure 10. As we can see, removing outliers improves accuracy in the AIR dataset, but it is ineffective in the ROOM dataset. This illustrates that addressing outliers in real-world data streams is a challenging topic and requires further researches.

6.6 Challenges of Missing Values

Finding (4): KNN imputer is generally more effective than regression imputer, filling with average and filling with zero. This indicates that missing values can usually be effectively estimated from nearby samples in real-world data streams.

In this section, we explore various missing value filling methods on the AIR dataset. We experiment with KNN imputer (k = 2, 5, 10, 20), regression imputer, filling with average and filling with zero. Figure 11 reveals that KNN imputer generally outperforms other methods in terms of accuracy. Moreover, different values of k for the KNN imputer do not significantly influence the accuracy. Considering that a smaller k can save computation, we recommend using the KNN imputer with $k = 2$ as a default setting.



(a) NN models (b) Tree models

Figure 11: Test loss with respect to different missing value filling methods on the AIR dataset. K means KNN. Reg means regression imputer. Avg means filling with average.

6.7 Experiments on Advanced Models

Finding (5): Advanced and complex models may easily overfit in open-environment regression tasks, creating a demand for models capable of learning generalizable knowledge for streaming data.

To explore the effectiveness and efficiency of advanced models for relational dataset, we test TabNet [5] and ARM-Net [12]. TabNet [5] combines neural network with decision tree to achieve end-to-end learning with interpretability. ARM-Net [12] learns adaptive feature interaction in the exponential space to capture cross feature information, which can help filter noisy or irrelevant features.

We use their public codes with default settings. To be consistent, the number of epochs is set as 10. Results are shown in Table 9. As we can see, ARM-Net has clear accuracy improvement on classification tasks (the first three rows). However, both TabNet and ARM-Net cannot work well in open environment regression tasks. As shown in Figure 12, we observe that the test loss is much higher

than train loss in TabNet and ARM-Net, due to the overfitting of complex models. Considering that the running time of TabNet and ARM-Net is over 20x than three-layer naive NN, both models may not be suitable to time-sensitive open environment learning in relational data streams. More results are shown in Appendix B.3 of our full version [17].

Table 9: Test loss of advanced models on relational datasets.

Dataset	Naive-NN	TabNet	ARM-Net
ROOM	0.214±0.004	0.213±0.011	**0.204±0.010**
ELECTRICITY	0.311±0.012	0.340±0.006	**0.293±0.009**
INSECTS	0.269±0.006	0.507±0.007	**0.249±0.000**
AIR	**0.166±0.002**	1.942±0.074	0.408±0.014
POWER	**0.793±0.005**	1.578±0.102	0.958±0.007

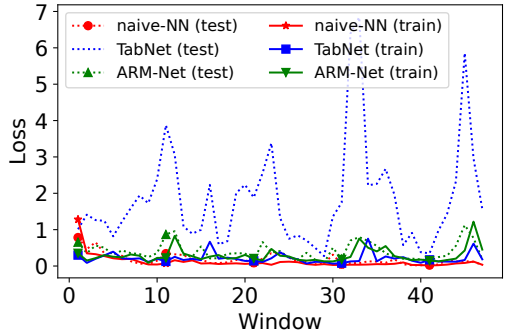


Figure 12: Train and test loss on the AIR dataset.

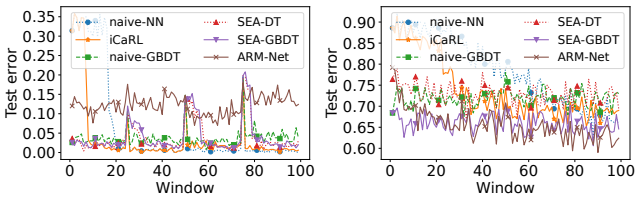
6.8 Investigating Synthetic Datasets

Finding (6): Synthetic datasets have limited coverage on open environment challenges. Conclusions drawn from experiments on synthetic datasets are much biased from experiments on our OEBench.

We investigate six popular synthetic data streams summarized in a highly cited work [9]: SEA (used in [1, 11, 21, 48, 75, 76]), STAGGER (used in [4, 7, 21, 80]), Rotating Hyperplane (used in [1, 7, 11, 21, 38, 75, 80]), RBF (used in [1, 4, 38]), LED (used in [1, 4, 11, 21, 23, 57, 76]), and Waveform (used in [11, 21, 23]). Among the six datasets, SEA and STAGGER create abrupt concept drifts by suddenly changing the decision rules at certain points. Rotating Hyperplane and RBF create incremental drifts by continuously altering the parameters of the decision boundaries. LED generates drifts by swapping features, and it also includes many irrelevant features. Waveform focuses on simulating noisy and irrelevant attributes.

These synthetic datasets do not cover regression task and have no missing values. Besides, as shown in Figure 3, the anomaly ratio of synthetic datasets is significantly lower than that of our collected real-world datasets. Among the three open environment challenges, these synthetic datasets only simulate the drifts, while ignoring the other two challenges of outliers and missing values. Moreover, synthetic datasets have very narrow range of drift statistics, and their drift ratios are also relatively low compared to our collected real-world data streams.

We also evaluate stream learning algorithms on these synthetic datasets. According to previous recommendations, we test naive NN, iCaRL, naive GBDT, SEA-DT, SEA-GBDT and ARM-Net. Results on SEA and LED are shown in Figure 13. More results are shown in Appendix B.1 of our full version [17]. As we can see, GBDT or SEA-GBDT can work well on these synthetic datsets. ARM-Net can help tackle irrelevant features and achieve better results in the LED dataset. This is in contrast with our findings in Section 6.2. Thus, as these synthetic datasets have limited coverage on open environment challenges, the conclusions drawn from experiments on synthetic datasets are much biased from experiments on our OEBench. Algorithms that can handle synthetic datasets well may not work well on real-world data streams.



(a) SEA (b) LED

Figure 13: Test error on synthetic datsets.

7 CONCLUSION

In this paper, we explore real-world relational data streams considering the recently proposed open environment challenges [84]. We propose a benchmark *OEBench* consisting of six stages: dataset collection, preprocessing, extracting open environment statistics, visualization, representative dataset selection and incremental learning algorithm evaluation. Specifically, we study 55 diverse real-world relational data streams, select five representative datasets based on the open environment statistics, and evaluate the performance of 10 existing incremental learning algorithms. We also conduct case studies with visualizations to explore how open environment challenges look like and their impacts on real-world relational data stream learning. The results highlight the inadequacy of current methods in effectively addressing these challenges in real-world relational data streams. Further researches are encouraged towards more effective and efficient approaches to data stream learning under open environment challenges.

ACKNOWLEDGMENTS

This research / project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

REFERENCES

[1] Ahmad Abbasi, Abdul Rehman Javed, Chinmay Chakraborty, Jamel Nebhen, Wisha Zehra, and Zunera Jalil. 2021. ElStream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning. *IEEE Access* 9 (2021), 66408–66419.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 139–154.

[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3366–3375.

[4] Robert Anderson, Yun Sing Koh, Gillian Dobbie, and Albert Bifet. 2019. Recurring concept meta-learning for evolving data streams. *Expert Systems with Applications* 138 (2019), 112832.

[5] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 6679–6687.

[6] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and Rafael Morales-Bueno. 2006. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, Vol. 6. 77–86.

[7] Jürgen Beringer and Eyke Hüllermeier. 2007. Efficient instance-based learning on data streams. *Intelligent Data Analysis* 11, 6 (2007), 627–650.

[8] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 443–448.

[9] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. 2009. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 139–148.

[10] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. 2010. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*. PMLR, 44–50.

[11] Dariusz Brzeziński and Jerzy Stefanowski. 2011. Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems*. Springer, 155–163.

[12] Shaofeng Cai, Kaiping Zheng, Gang Chen, HV Jagadish, Beng Chin Ooi, and Meihui Zhang. 2021. Arm-net: Adaptive relation modeling network for structured data. In *Proceedings of the 2021 International Conference on Management of Data*. 207–220.

[13] Rich Caruana, Steve Lawrence, and C Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems* 13 (2000).

[14] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 532–547.

[15] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. 2006. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proc. Symposium on the Interface of Statistics, Computing Science, and Applications (Interface)*.

[16] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. 2019. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5138–5146.

[17] Yiqun Diao, Yutong Yang, Qinbin Li, Bingsheng He, and Mian Lu. 2023. OEBench: Investigating Open Environment Challenges in Real-World Relational Data Streams. arXiv:2308.15059

[18] Gregory Ditzler and Robi Polikar. 2011. Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*. IEEE, 41–48.

[19] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. 2021. A survey on missing data in machine learning. *Journal of Big Data* 8, 1 (2021), 1–37.

[20] Sebastian Farquhar and Yarin Gal. 2018. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733* (2018).

[21] Joao Gama and Petr Kosina. 2011. Learning decision rules from data streams. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[22] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *Brazilian symposium on artificial intelligence*. Springer, 286–295.

[23] Joao Gama, Ricardo Rocha, and Pedro Medas. 2003. Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 523–528.

[24] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. 2009. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 329–338.

[25] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106 (2017), 1469–1495.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[27] Jim Gray. 1993. The benchmark handbook for database and transasction systems. *Mergan Kaufmann, San Mateo* (1993).

[28] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*. PMLR, 2712–2721.

[29] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. 2022. ADBench: Anomaly Detection Benchmark. In *Neural Information Processing Systems (NeurIPS)*.

[30] Maayan Harel, Shie Mannor, Ran El-Yaniv, and Koby Crammer. 2014. Concept drift detection through resampling. In *International conference on machine learning*. PMLR, 1009–1017.

[31] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. 2019. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 9769–9776.

[32] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. 2012. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence* 1 (2012), 89–101.

[33] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 831–839.

[34] Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, and Hussein Abbass. 2019. IEEE-CIS Fraud Detection. https://kaggle.com/competitions/ieee-fraud-detection

[35] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. 2021. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3957–3966.

[36] Ibrahem Kandel and Mauro Castelli. 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express* 6, 4 (2020), 312–315.

[37] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[38] Bartosz Krawczyk, Bernhard Pfahringer, and Michał Woźniak. 2018. Combining active learning with concept drift detection for data stream mining. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2239–2244.

[39] Pawel Ksieniewicz and Pawel Zyblewski. 2022. Stream-learn—Open-source Python library for difficult data stream batch analysis. *Neurocomputing* 478 (2022), 11–21.

[40] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[41] Weizhi Li, Weirong Mo, Xu Zhang, John J Squiers, Yang Lu, Eric W Sellke, Wensheng Fan, J Michael DiMaio, and Jeffrey E Thatcher. 2015. Outlier detection and removal improves accuracy of machine learning approach to multispectral burn diagnostic imaging. *Journal of biomedical optics* 20, 12 (2015), 121305–121305.

[42] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[43] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. 2022. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[44] Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. 2013. Drift detection using uncertainty distribution divergence. *Evolving Systems* 4 (2013), 13–25.

[45] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[46] Vincenzo Lomonaco and Davide Maltoni. 2017. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*. PMLR, 17–26.

[47] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M Van de Ven, et al. 2021. Avalanche: an end-to-end library for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3600–3610.

[48] Osama A Mahdi, Eric Pardede, Nawfal Ali, and Jinli Cao. 2020. Diversity measure as a new drift detection method in data streaming. *Knowledge-Based Systems* 191 (2020), 105227.

[49] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. 2022. Online continual learning in image classification: An empirical survey. *Neurocomputing* 469 (2022), 28–51.

[50] Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7765–7773.

[51] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. 2018. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1963–1972.

[52] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[53] Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 117–122.

[54] Anna Montoya, inversion, KirillOdintsov, and Martin Kotek. 2018. Home Credit Default Risk. https://kaggle.com/competitions/home-credit-default-risk

[55] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011).

[56] Husein Perez and Joseph HM Tah. 2020. Improving the accuracy of convolutional neural networks by identifying and removing outlier images in datasets using t-SNE. *Mathematics* 8, 5 (2020), 662.

[57] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. 2018. McDiarmid drift detection methods for evolving data streams. In *2018 International joint conference on neural networks (IJCNN)*. IEEE, 1–9.

[58] Tomáš Pevnỳ. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102 (2016), 275–304.

[59] Utomo Pujianto, Aji Prasetya Wibawa, Muhammad Iqbal Akbar, et al. 2019. K-nearest neighbor (k-NN) based missing data imputation. In *2019 5th International Conference on Science in Information Technology (ICSITech)*. IEEE, 83–88.

[60] Abdulhakim A Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. 2015. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 935–944.

[61] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[62] Douglas A Reynolds et al. 2009. Gaussian mixture models. *Encyclopedia of biometrics* 741, 659-663 (2009).

[63] Ryne Roady, Tyler L Hayes, Hitesh Vaidya, and Christopher Kanan. 2020. Stream-51: Streaming classification and novelty detection from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 228–229.

[64] Saket Sathe and Charu C Aggarwal. 2016. Subspace outlier detection in linear time with randomized hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 459–468.

[65] Qi She, Fan Feng, Xinyue Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin Zhang, et al. 2020. OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 4767–4773.

[66] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90.

[67] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2021. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9374–9384.

[68] Vinicius Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. 2020. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1805–1858.

[69] David RB Stockwell and A Townsend Peterson. 2002. Effects of sample size on accuracy of species distribution models. *Ecological modelling* 148, 1 (2002), 1–13.

[70] W Nick Street and YongSeog Kim. 2001. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 377–382.

[71] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. 2011. Fast anomaly detection for streaming data. In *Twenty-second international joint conference on artificial intelligence*. Citeseer.

[72] Marco Toldo and Mete Ozay. 2022. Bring Evanescent Representations to Life in Lifelong Class Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16732–16741.

[73] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[74] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[75] Heng Wang and Zubin Abraham. 2015. Concept drift detection for streaming data. In *2015 international joint conference on neural networks (IJCNN)*. IEEE, 1–9.

[76] Pingfan Wang, Nanlin Jin, Wai Lok Woo, John R Woodward, and Duncan Davies. 2022. Noise tolerant drift detection method for data stream mining. *Information Sciences* 609 (2022), 1318–1333.

[77] Guile Wu, Shaogang Gong, and Pan Li. 2021. Striking a balance between stability and plasticity for class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1124–1133.

[78] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 374–382.

[79] Shipeng Yan, Jiangwei Xie, and Xuming He. 2021. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3014–3023.

[80] Ying Yang, Xindong Wu, and Xingquan Zhu. 2006. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery* 13 (2006), 261–289.

[81] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*. PMLR, 3987–3995.

[82] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. 2020. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1131–1140.

[83] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. 2022. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *arXiv preprint arXiv:2205.13218* (2022).

[84] Zhi-Hua Zhou. 2022. Open-environment machine learning. *National Science Review* 9, 8 (2022).

[85] Jiaqi Zhu, Shaofeng Cai, Fang Deng, Beng Chin Ooi, and Wenqiao Zhang. 2023. METER: A Dynamic Concept Adaptation Framework for Online Anomaly Detection. *arXiv preprint arXiv:2312.16831* (2023).