SLEADE: Disagreement-Based Semi-Supervised Learning for Sparsely Labeled Evolving Data Streams

Heitor Murilo Gomes, Jesse Read, Maciej Grzenda, Bernhard Pfahringer, Albert Bifet

***Abstract*—Semi-supervised learning (SSL) problems are challenging, appear in many domains, and are particularly relevant to streaming applications, where data are abundant but labels are not. The problem tackled here is classification over an evolving data stream where labels are rare and distributed randomly. We propose SLEADE (Stream LEArning by Disagreement Ensemble), a novel method that exploits disagreement-based learning and unsupervised drift detection to leverage unlabeled data during training. SLEADE uses pseudo-labeled instances to augment the training set of each member of an ensemble using a *majority trains minority* scheme. The pseudo-labeled data impact is controlled by a weighting function that considers the confidence in the prediction attributed by the ensemble members. SLEADE exploits unsupervised drift detection, which allows the ensemble to respond to changes. We present several experiments using real and synthetic data to illustrate the benefits and limitations of SLEADE compared to existing algorithms.**

***Index Terms*—Data stream, semi-supervised learning, concept drift, disagreement-based learning.**

I. Introduction

THE use of machine learning (ML) is pervasive across the whole spectrum of applications [1]. Alongside all the challenges and concerns that this pervasiveness entails, one crucial problem is how to learn from massive and ephemeral data streams (e.g., sensor data, social network news feeds, data from wearable devices, and others). This context requires ML methods that are accurate and robust to changes. In an ML streaming setting, copious amounts of data are generated in short periods [2]. This imposes computational challenges on the learning algorithm, as it is expected to be able to predict and train at any time [3]. Furthermore, since a data stream is theoretically infinite, the data cannot be stored for future processing. Nor is it desirable to skip predictions or discard labeled data that could be used for training. Besides computational constraints, which have been thoroughly studied, there are two particularly challenging problems that arise when dealing with data streams: *evolving data* [4] and *partially labeled data* [5], [6].

Evolving data refers to the phenomenon that, in most domains, data is expected to change over time (i.e. concept drift [7] can occur). Therefore, learning systems should be capable of self-diagnosis when performance degrades by identifying the causes of degradation and self-adapting to recover to a stable status [8]. Furthermore, a self-adaptive learner can be achieved by detecting and reacting to changes [9], or by continuously updating the model.

Partially labeled data refers to the fact that unlabeled data is abundant in a streaming setting, but labeled data is not, which hinders supervised learning approaches [6]. Furthermore, labeling can be costly and the time needed to annotate an instance can impact its relevance [10]. Moreover, in some cases *post factum* labeling and active learning may not be feasible. As an example, even an expert may not be able to unequivocally decide in the case of travel mode choice modeling [11], which travel mode was selected under given circumstances in the past by a traveler for a trip in urban area. Moreover, even if labeling is possible, if it takes too long for an instance to be labeled, it might not be representative of the current data distribution [12].

A general abstraction to cope with partially labeled data is semi-supervised learning (SSL) [13]. Some SSL approaches such as self-training [14] and cluster-based models [14]–[16] rely on strong assumptions about the underlying data distribution, which, when not met, causes a catastrophic impact on the predictive performance [5]. Furthermore, given the ephemeral characteristics of a data stream (attributed to concept drifts), situations where the data no longer abide by the method bias are expected. Thus, on one hand, training a learning model using only a few labeled data is challenging even when all the data items are available at once (i.e. batch setting). On the other hand, there is no *de facto* solution to coping with concept drifts, even when the streaming data is fully labeled. At the intersection of these two research problems lies the difficult, and pervasive problem, of learning with *partially labeled evolving data streams*.

Problems in which only few instances are labeled have been previously investigated for data streams [14], [17], [18], often assuming an active learning problem setting [19]–[22]. However, an important challenge related to many of these methods is that the assumption of active learning that true labels are available on request is not satisfied for many problems. Furthermore, models learned with semi-supervised methods exploiting unlabeled data may be negatively affected by such data and yield lower performance than those developed with labeled data only.

Hence, in this work, to address the need for semi-supervised

Heitor Murilo Gomes is with Victoria University of Wellington, Jesse Read is with LIX, École Polytechnique, Institut Polytechnique de Paris, France, Maciej Grzenda is with Warsaw University of Technology, Faculty of Mathematics and Information Science, Poland, Bernhard Pfahringer is with Computer Science Department, University of Waikato, New Zealand, Albert Bifet is with AI Institute, University of Waikato, New Zealand & TELECOM Paris, LTCI, France

learning with partially labeled data streams, we propose a disagreement-based learning (DBL) method [23]. DBL presents a departure from the conventional paradigms such as self-training and clustering-based techniques, as it imposes fewer assumptions on the data [24], [25]. Its core assumption revolves around the idea that ensemble base members can effectively enhance and rectify each other's perspectives, provided they exhibit both diversity and accuracy. Our investigation centers on the validity of this fundamental assumption, along with an exploration of the necessary conditions for its successful application in practice. As the method does not rely on the *post factum* labeling, it can be applied when active learning assumptions are not satisfied, e.g. when an expert cannot unequivocally label instances.

On top of that, we combine the algorithm with an unsupervised drift detection strategy. In general terms, we train a supervised ensemble learner on the labeled data, such that the ensemble is expected to enforce multiple views [26] of a (potentially) single view data stream. These multiple views are used to fit diverse models, which are used to exploit unlabeled data by training base learners using other base learners predictions, i.e. a *majority trains minority* scheme. In our context, we refer to instances which were labeled by an uncertain process (e.g. another classifier or an unsupervised method) as instances with pseudo-labels.

We introduce SLEADE (Streaming LEArning by Disagreement Ensemble). SLEADE relies on reinforcing the *disagreement* [23] among base learners based on the ensemble algorithm, thus inducing multiple views from the input data. We hypothesise that through using only high confidence predictions provided by majority of an ensemble, the performance of the entire ensemble could be improved. Hence, disagreement between learners could be exploited to provide instances with pseudo-labels and increase the volume of labeled instances used to train the learners. We assume only a few restrictions for the base ensemble method. First, it must be able to enforce diversity among the base models. Second, it must be an incremental learner. Ensemble methods such as the Streaming Random Patches (SRP) ensemble [27] or the Adaptive Random Forest (ARF) [9] are good candidates since they were shown to produce diverse incremental base models [28].

Paper contributions and roadmap. Our main contributions can be summarized as follows:

1) SLEADE algorithm[1] that leverages abundant unlabeled data by training base learners on instances with pseudo-labels predicted by other base learners trained on different views of the data. The algorithm relies on unsupervised drift detection to trigger substantial changes to the ensemble model.
2) Experiments comparing SLEADE against supervised and semi-supervised algorithms with varying levels of scarce labeling (from 1% to 5%) using data streams with different characteristics (e.g. multi-class, concept drifts, real-world and synthetic data).
3) A thorough examination of the SLEADE algorithm, including a breakdown of its components (ablation study), evaluation of the effectivity of its pseudo-labelling approach compared to other SSL methods, and investigation of the impact of the unsupervised drift detection method on SLEADE's performance.

[1]Algorithm available as part of CapyMOA [29] at https://github.com/adaptive-machine-learning/CapyMOA/blob/main/src/capymoa/ssl/_sleade.py.

The rest of this work is organized as follows. First, we explain the problem of learning from evolving partially labeled data streams. The following section discusses related work and highlights the differences between their approaches and ours. The third section presents SLEADE. In the fourth section, the experiments are reported and evaluated. Finally, the last section presents some final remarks and presents directions for future works.

II. Problem setting

The **supervised immediate labeling scenario** involves a data stream S that presents new unlabeled instance $\mathbf{x}^t$ to the classifier at regular intervals. The features of instance $\mathbf{x}^t$ are available at time t and its corresponding true class label y^t is immediately available before the next instance $\mathbf{x}^{t+1}$ is presented to the learner. As a result, the class label y^t can be used for training immediately after the prediction for $\mathbf{x}^t$ is made.

In the **semi-supervised labeling scenario**, the assumption that y^t is always available when $\mathbf{x}^{t+1}$ arrives is relaxed. This means that for some instances $\mathbf{x}^t$, the corresponding class labels may never be available. In this study, we focus on the **semi-supervised immediate labeling scenario**, where labels are immediately available, but only for a limited number of instances.

III. Related work

Our research resides at the intersection of Disagreement-Based Learning (DBL) and data stream learning. We commence by discussing DBL strategies and then focus on existing semi-supervised approaches for streaming data.

A. Disagreement-based learning

A possible approach to SSL, which has yet to be thoroughly investigated for streaming data [5], relies on co-training [24] and, more generally, disagreement-based learning [23]. Learning by disagreement exploits the predictions on unlabeled instances yielded by a learner to train another learner, such that these learners were trained on different views of the data. This can be achieved when the data is naturally split on multiple views using the co-training algorithm [24]. However, a more general approach is to create multiple views from single-view data by leveraging the diversity of the ensemble members. This strategy was explored in the canonical co-forest algorithm [30]. Wang and Zhou [31] provided a thorough theoretical analysis of co-training and learning by disagreement in a traditional batch learning setting, i.e. assuming that learners can perform multiple passes over the training data, and that this data originates from the same concept.

Examples of learning by disagreement for batch learning include tri-training [32] and co-forest [30]. Tri-training was

designed to enhance the classification model's accuracy by promoting synergy between three classifiers. Tri-training operates iteratively, similarly to co-training, expanding the training sets of these classifiers. On the other hand, in its classic form, co-training induces two learners on separate feature sets, i.e. on views, which are both sufficient for learning the underlying concept while being conditionally independent to one another given the class label. Since independent views are challenging to obtain in practice, more relaxed assumptions were proposed, such as the 'expansion' assumption. Authors in [33] investigated the expansion assumption and proved that it is sufficient for iterative co-training to succeed given strong probably approximately correct (PAC) learners on each subset of features. The expansion assumption posits that the distribution of data allows for a certain degree of 'expansion' between the subsets of features. Specifically, it means that for any sets of confident predictions in each view, a significant portion of these sets must contain instances that are not confident predictions in the other view. This ensures that co-training can iteratively improve both classifiers by providing useful training data from the confident predictions of one classifier to the other. One challenge when dealing with data stream learning is that the original 'expansion' assumption cannot be used as it is not viable to iterate through the input data multiple times. Furthermore, the more restrictive assumptions from the original co-training theory [24], which were applicable to one-shot co-training, might not be appropriate since the different induced views are unlikely to be conditionally independent.

In our current work, we rely on the assumption of semi-supervised learning by disagreement, where base members of an ensemble can be used to boost and correct each other as long as they are diverse and accurate themselves, thus not depending on the prior multi-view assumption.

B. Concept drift detection and active approach to concept drift

Learning with data streams in the presence of concept drift necessitates model adaptation mechanisms. These rely on either a passive or active approach [34]. In the case of the passive approach, a model is constantly updated irrespective of whether concept drift occurred in the data. Under the active approach, model updates are triggered by drift detections. This necessitates the use of drift detection methods. A benefit of this approach is that only after drift is detected, a model is updated. Multiple drift detection methods were proposed. The most popular group of methods are error rate-based drift detection algorithms, i.e. methods which track changes in online error rates of classifiers [35]. Error rate-based drift detection algorithms include inter alia ADaptive WINdowing (ADWIN) [9], [36], and Page Hinkley Test [9], [37]. The ADWIN method adaptively sets the width of two subwindows, which are used to test if the averages of values present in these windows have diverged. This enables the detection of changes in the mean error of a learner over time, when ADWIN is supplied with the errors made by the learner. In a recent extensive aggregated comparison of drift detectors for all considered drift scenarios [37], ADWIN was found to be the detector in terms of F1 score. However, other performance indicators used to evaluate drift detectors, such as detection delay, may also suggest the use of other detectors.

Importantly, drift detectors may track changes in other values than errors of a single learner. In the student-teacher approach [38], the primary model, referred to as the teacher, is established using a labeled training data set. The predictions made by the teacher serve as class labels for a surrogate model, the student, which aims to imitate the teacher. A drift detection algorithm is then employed to monitor the variations in the student's mimicking error. The idea is that if the mimicking error increases, it suggests that a concept drift has occurred.

The active approach to drift detection relies on drift detectors and has been used successfully combined with ensemble methods. This is because drift detection can be followed by the replacement of both an entire model and one (or many) of the base learners of an ensemble. One of the popular methods implementing this approach is the Adaptive Random Forest [9] method. It combines an online bootstrap aggregating process with training an ensemble of base learners. Hoeffding trees [39] are used in this role. Errors made by a tree are monitored by e.g. an ADWIN detector. Every time drift at a warning level is detected, a background tree is initiated. Once the drift is confirmed, which is based on the drift threshold controlling the sensitivity of drift detection, the background tree replaces the tree for which the drift was detected. The successor of the ARF method, i.e. the Streaming Random Patches method which was shown to offer superior performance to the ARF method [27], relies on the same error-rate-based drift detection to replace ensemble member(s). Moreover, its base learners are trained on random subspaces of features in addition to different subsets of instances, which promotes an even higher diversity of base learners. Importantly, ARF and SRP have been shown to provide models of particularly high performance [9], [27]. However, these methods exploit fully labeled data only.

C. Stream Learning for Partially Labeled Data

Methods that cope with partially labeled data streams often rely on active learning [16], [19], [20], [40]. Given our problem setting, such algorithms are not applicable as we assume there is no control over which instances are labeled. Therefore, we turn our attention to streaming SSL strategies, which commonly relied on cluster-based models [15], [41] and self-training [14]. Cluster-based models assume closer instances in the input space are likely to have the same label [18]. Self-training is a wrapper algorithm that takes any arbitrary classifier and uses its predictions to train itself.

An early clustering-based approach was proposed in [42], where an individual model created K micro-clusters from a window of data. The prediction was given after determining the closest k nearest clusters from it. The predicted class was that which had the highest frequency of labeled data across all of the closest k clusters. The Cluster-Then-Label algorithm proposed in [14] utilized CluStream [43] to cluster instances, such that each cluster has an associated class label frequency counter. Pseudo-labels are assigned to arriving unlabeled data according to the most frequent label associated with its closest cluster.

In [14], the authors proposed a self-training learner (i.e. SelfTrain) designed to receive as input either a single instance or a batch of instances at a time. A confidence threshold that determines whether instances are used for self-training could be fixed or adaptive concerning the average confidence scores observed in a window. The authors observed that the variant using a windowed input, distance-based scoring, and fixed confidence threshold achieved the best predictive performance. In CPSSDS [44], authors explore inductive conformal prediction to select unlabeled instances for self-training. To detect concept drifts, the outputs of the conformal prediction for two data windows were compared using the Kolmogorov-Smirnov test. The recovery from drift strategy relies on the classic strategy of resetting the base learner.

The Improved Online Ensembles (IOE) algorithm [45] introduced an adaptive ensemble approach for semi-supervised data stream classification, capable of leveraging various base learners, including Hoeffding Trees. IOE dynamically adjusts the ensemble size based on per-class recall estimates to address class imbalance. It also incorporates self-training with a pseudo-label confidence threshold and adaptive weighting to mitigate bias in evolving class distributions.

SDSL [46], tackles semi-supervised stream learning with concept drift using a short lookback window. It combines supervised learning with pseudo-labeling from unsupervised clustering, while using adversarial anti-forgetting mechanisms to balance adaptation and stability. Unlike other methods, which primarily focus on self-training (e.g. IOE) or neural network adaptation (e.g. SSALD), SDSL explicitly models drift by maintaining a short memory of past labeled and pseudo-labeled instances. Importantly, the method assumes the existence of initial labeled data prior to unlabeled data stream.

More recently, SSALD [22] extends radial basis function neural networks (RBFNNs) to the streaming SSL setting. It maintains a dynamic set of radial basis function centers that adapt to the evolving data distribution. SSALD updates its pseudo-labeling strategy using a similarity-based weighting mechanism and leverages a concept drift adaptation mechanism to ensure robustness in non-stationary environments. Notably, SSALD relies on active learning to label selected instances within a sliding window of recent instances.

While these methods present different approaches to semi-supervised stream learning, they also come with notable limitations. Algorithms that incorporate active learning strategies, such as SSALD, may struggle in scenarios where labeled instances are assigned independently of the model's control. This aligns with observations in [5], where the authors highlighted the challenges of active learning in drifting streams, particularly due to the delay in obtaining ground-truth labels and the risk of shifts rendering past queries obsolete. SDSL, on the other hand, depends on an initial labeled set, which can be reasonable in many practical settings but introduces constraints regarding how far pseudo-labeled instances can diverge from the initial labeled data. If concept drift is too drastic, these constraints may become difficult to justify. Similarly, methods such as Cluster-Then-Label (ClusterTL) and SelfTrain rely on assumptions about data structure that may not always hold in streaming environments. ClusterTL assumes that class distributions remain stable within clusters and that pseudo-labels can be reliably inferred from cluster frequency statistics. However, in dynamically evolving data streams, clusters may shift or overlap, leading to cascading errors in pseudo-labeling. SelfTrain, while more flexible, relies on confidence-based pseudo-labeling, which can reinforce incorrect predictions when the model's confidence is misplaced—especially in early stages or after abrupt concept drifts. Finally, a common gap in these studies is the lack of rigorous benchmarking against competitive supervised baselines trained on sparsely labeled data. As pointed out in [47], semi-supervised methods should be evaluated against robust supervised models to ensure their practical utility. If an SSL approach fails to outperform a well-optimized supervised learner trained on the available labeled data, its adoption becomes difficult to justify.

IV. SLEADE

As other DBL methods [30], SLEADE is built from the assumption that a model l can be improved by learning from pseudo-labeled instances where the pseudo label has been predicted by other models with greater confidence than model l itself would apply.

In SSL settings, such as we consider, labels y^t are frequently absent. SLEADE aims to use the pseudo-label $\widehat{y}_l^t$ instead, where l is a learned model that provides predictions according to

$$\widehat{y}_l^t = l(\boldsymbol{x}^t) \tag{1}$$

An issue here is the potential for l to commit an error. The expectation of such an error is:

$$P(\widehat{y}_l^t \neq y^t) \tag{2}$$

where the pseudo-label $\widehat{y}_l^t$ is as reliable as the true label y^t when $P(\widehat{y}_l^t = y^t) = 1$.

Unfortunately, we cannot measure this, because we do not have true label y^t neither this P, so as a substitute, in SLEADE we consider confidence,

$$c_l(\widehat{y}_l^t) \approx P(\widehat{y}_l^t = y^t) \tag{3}$$

which can be obtained in some form from practically any learner, e.g., the number of positive vs negative votes in an ensemble. The confidence of an ensemble is likely to be more reliable than an individual learner. This is where the concept of disagreement comes in. SLEADE considers the confidence $c_{\hat{L}}$ of an ensemble of learners $\hat{L} = L \setminus l$ (i.e., an ensemble excluding l), when considering to use $\widehat{y}_{\hat{L}}^t$ as a pseudo-label to update model l. More specifically, only when $c_{\hat{L}} > c_l$; the majority teaches the minority; along the lines of [32] and [30].

Hence, SLEADE can be expected to reduce error on average across all examples for which a training label is missing and for which there is sufficient confidence to incorporate it, thus increasing the quality of the training data seen by each learner l, reducing its error, and so on.

However, the streaming setting presents unique challenges, because instances are observed one at a time and confidences are likely to be highly variable (uncalibrated) at the beginning of the stream and also – likewise – at the beginning of a new

concept. This problem is magnified by the extreme sparsity of labels; we may have to wait for hundreds of examples to get a single training label which refers to one class and one example only.

Therefore, taking raw confidences might provoke a 'run-away' effect where confidence inspires confidence, and so confidence becomes increasingly baseless, driving the model off course long before it has seen enough true labels to become accurate, and at a rate greater than what can be corrected by true labels. To diminish the chance of this, SLEADE incorporates a novel weighting scheme for pseudo-labeled instances, proportional to the supervised experience of the models, defined in line 18 of Alg. 1. In other words, the confidence of relatively more experienced models should have more weight on the training mechanism. Specifically, to train a model with a pseudo-labeled instance $(\boldsymbol{x}, \hat{y})$, we propose to use a weight $\omega = c(\hat{y}) \times w_{\mathrm{s}}$, where $c(\hat{y})$ is the confidence on $\hat{y}$ prediction and w_{s} is the weight shrinkage factor determined proportional to the true labeling density, since we assume that more true labels should correspond to stronger predictions and higher confidence. Similarly, sparse labeling suggests that the impact of a single pseudo-labeled instance on base model updates should be reduced. This is to maintain a relatively high influence of instances having true labels on the training of base models. Furthermore, this weighting scheme relies on the base models' ability to process fractional updates, as the impact of a pseudo-labeled instance is scaled rather than treated as a full observation. Models such as Hoeffding Trees support this by allowing weighted updates to their leaf statistics.

Labeled instances are always weighted $\omega = 1$, as shown in line 10 of Alg. 1; thus, they have a full impact on the training process.

A further risk is that learners may converge toward homogeneous predictions when repeatedly trained on each other's outputs, removing the effect of an ensemble [31]. To maintain disagreement among learners is therefore essential. To achieve this, we use an ensembling technique where base learners are trained on random patches (random subsets of features and instances) akin to the Streaming Random Patches (SRP) [27] algorithm to promote highly diverse base models by training them on different random patches.

The SLEADE algorithm is depicted in Algorithm 1, with lines 2 to 5 creating an ensemble L composed of new untrained base models, an empty window W to be populated with labeled instances, a new untrained student model l_S taking the form of Hoeffding tree, and new instance of a concept drift detector d, respectively. Next, lines 8 to 23 showcase the central learning through disagreement approach and lines 24 to 38 demonstrate the handling of unsupervised drift detection, which will be discussed further in a subsequent section. The focus of the code is on SLEADE training, as predictions are handled by the underlying ensemble method.

Algorithm 1 SLEADE.

INPUT: S: Data stream; n: size of sliding window labeled instances; Φ: minimum confidence threshold.
DATA: L: ensemble; l: base model from ensemble L; $c_l(\hat{y}) \in [0, 1]$: Confidence of learner l on prediction $\hat{y}$; w_{s}: Weight shrinkage factor determined proportional to the labeling density; $|X_L|$: Number of labeled instances observed so far; $|X|$: Total number of instances observed so far.

1: **function** TRAIN(S)
2: $L \leftarrow$ initialise_ensemble()
3: $W \leftarrow$ empty_queue()
4: $l_S \leftarrow$ new_student_model()
5: $d \leftarrow$ new_detector()
6: $w_{\mathrm{s}} \leftarrow 1.0$
7: **for all** $(\mathbf{x}, y) \in S$ **do**
8: **for all** $l \in L$ **do**
9: **if** y is present **then**
10: train($l, (\mathbf{x}, y), 1$)
11: **else**
12: $\hat{y}_{\mathrm{l}} \leftarrow$ predict($l, \mathbf{x}$)
13: $\hat{L} \leftarrow L \setminus l$
14: $\hat{y}_{\hat{\mathrm{L}}} \leftarrow$ predict($\hat{L}, \mathbf{x}$)
15: **if** $\hat{y}_{\mathrm{l}} \neq \hat{y}_{\mathrm{L}}$ **then**
16: **if** $c_{\hat{L}} > \Phi$ **and** $c_l < c_{\hat{L}}$ **then**
17: $w_{\mathrm{s}} \leftarrow |X_L|/|X|$
18: $\omega \leftarrow c(\hat{y}_{\hat{\mathrm{L}}}) \times w_{\mathrm{s}}$
19: train($l, (\mathbf{x}, \hat{y}_{\hat{\mathrm{L}}}), \omega$)
20: **end if**
21: **end if**
22: **end if**
23: **end for**
24: **if** y is present **then**
25: **if** size(W)$\geq n$ **then**
26: pop(W)
27: **end if**
28: push(W, $(\mathbf{x}, y), 1$)
29: **end if**
30: $\hat{y}_{\mathrm{L}} \leftarrow$ predict($L, \mathbf{x}$)
31: $\hat{y}_{l_S} \leftarrow$ predict($l_S, \mathbf{x}$)
32: train(l_S, ($\mathbf{x}$, $\hat{y}_{\mathrm{L}}$))
33: update_detector(d, $\hat{y}_{l_S}$, $\hat{y}_{\mathrm{L}}$)
34: **if** drift_detected(d) **then**
35: $\ddot{L} \leftarrow \{\, l \in L \mid acc(l) < avg\_acc(L) \,\}$
36: reset($\ddot{L}$)
37: train($\ddot{L}$, W)
38: **end if**
39: **end for**
40: **end function**

A. Unsupervised drift detection

In this study, we implement a concept-drift detection method based on the student-teacher approach inspired by Cerqueira et al in [38]. Our approach differs from the original student-teacher method in that the teacher model, which is an ensemble model trained on sparsely labeled data and pseudo-labeled data (according to Alg. 1), and the student model, which is an incremental learner in the form of a single Hoeffding Tree, are both continually updated.

The student-teacher unsupervised drift detection model detects changes in the input data distribution $P(X)$ by monitoring discrepancies between the predictions of the student and teacher models. The teacher is continuously trained

on incoming data, while the student is trained to mimic the teacher's predictions. If the input distribution shifts, the teacher's predictions may also shift, but the student—having learned from prior teacher predictions—may struggle to adapt immediately, leading to an increase in prediction mismatches. This divergence between the student and teacher signals a potential concept drift. In addition to detecting shifts in $P(X)$, this mechanism can also capture changes in $P(Y|X)$. Since the student learns to approximate the teacher's decision boundaries, variations in the relationship between features and labels will cause discrepancies between the models' outputs. Likewise, changes in the joint distribution $P(X, Y)$ can be inferred through increasing divergence in student-teacher predictions. To track these discrepancies, SLEADE monitors whether the student correctly predicts the teacher's outputs as a binary signal. These agreement values are then fed into the ADWIN drift detector [36] (line 33), which detects statistically significant changes over time. If the student's classification error increases beyond a statistically significant threshold, the drift detector triggers a concept drift alarm. However, this approach is not without limitations. If the student struggles to generalize the teacher's predictions due to biases or limited capacity, it may introduce false positives, erroneously signaling drift even when no real change has occurred. This is a common challenge in unsupervised drift detection methods, as they rely solely on indirect indicators of distribution shifts.

To respond to concept drifts in a way that benefits the learning algorithm and mitigates potential negative impact of false drift detections on SLEADE performance, it is thus crucial to implement an effective recovery strategy. Our approach uses a sliding window technique, keeping the n most recent labeled instances (lines 24–29). When a concept drift is detected (line 34), any ensemble member with test-then-train accuracy lower than the average is reset. Such models $\ddot{L}$ are then trained on the n latest labeled instances (line 37). However, choosing an appropriate value for n is crucial. A larger n may contain instances from previous concepts, while a smaller n may result in poorly initialized models.

Similarly to other active approaches for learning a classifier in nonstationary environments [34], i.e. approaches relying on drift detectors to detect changes in data distribution and then possibly trigger updates of learners, including the mechanisms present in ARF [9] and SRP [27], in SLEADE possible false drift detections can occur, but cause the replacement of only some base learners. The replaced models are the reset models $\ddot{L}$. Moreover, new ensemble members $\ddot{L}$ are trained with the data present in a window of recent labeled instances W. This solution promotes diversity of ensemble members and avoids replacing former learners $\ddot{L}$ with newly initialised, but not trained yet base learners.

V. Experiments

The experiments focus on the classification performance of SLEADE in comparison to other semi-supervised and supervised algorithms (trained only on the available labeled instances). Following the discussion in [47], we aim at clearly showing when SLEADE outperforms the supervised algorithms, specially its underlying supervised ensemble method. We complement the experiments with an analysis of other characteristics of SLEADE, such as the impact of increasing the number of base learners or disabling the drift detection and recovery strategy.

Before presenting the results, we discuss the **algorithms**, **hyperparameters**, **evaluation framework**, and **datasets**. The experiments were executed using the Massive Online Analysis (MOA) [3] (Java) and CapyMOA [29] (Python) frameworks on a machine with 128 cores[2] and 200 GB of RAM.

The **algorithms** used for comparisons include two supervised and six semi-supervised algorithms, namely: ARF [9], SRP [27], SelfTrain [14], ClusterTL [14], CPSSDS [44], IOE [45] (adaptive ensemble), SSALD [22] (neural network adaptation), and SDSL [46] (memory-driven adversarial learning). SRP and ARF are ensemble methods that outperform other classifiers in the supervised setting [28], serving as strong baselines for comparison. SelfTrain is a semi-supervised meta-algorithm that can be applied to any base classifier and in this study it utilizes SRP. ClusterTL is a cluster-then-label approach that assigns pseudo-labels based on clustering structure. Among the more recent SSL approaches, IOE dynamically adjusts its ensemble size based on class recall estimates, incorporating adaptive weighting and self-training with pseudo-labeling confidence thresholds. SSALD extends radial basis function neural networks (RBFNNs) to the streaming setting, leveraging adaptive pseudo-labeling and drift adaptation mechanisms. Additionally, SSALD incorporates strategies to handle class imbalance, using similarity-weighted confidence scores to adjust the impact of pseudo-labeled instances and mitigate bias toward the majority class. SDSL, in turn, combines supervised learning with pseudo-labeling from unsupervised clustering, integrating adversarial anti-forgetting mechanisms to balance adaptation and stability. SRP, SLEADE, and IOE leverage an ensemble of Hoeffding Trees, ensuring consistency in comparing ensemble-based methods. Given that SRP and ARF achieve high accuracy on fully labeled data streams, they serve as strong baselines against which the performance of semi-supervised methods can be evaluated [47].

Regarding the **hyperparameters**, the number of base models was set to 10 or 30, depending on the experiment for ensemble-based methods. SLEADE has two hyperparameters, the sliding window (n) and the minimum confidence threshold (Φ), which were set as $\mathrm{n} = 100$ and $\Phi = 0.9$. SelfTrain and ClusterTL were parametrized according to the best-performing values for their hyperparameters as described in [14]. ClusterTL does not employ an underlying supervised model, instead it relies solely on clustering. Therefore, in experiments where we increase the capacity of the underlying models by using more learners, we attempt to do the same in ClusterTL by using more clusters (i.e. from 100 to 300 clusters). Even though this configuration led to slight improvements in [14], in most cases, we could not observe significant accuracy gains by using 300 clusters. In theory, the CPSSDS approach can be utilized with any type of base learning model, including ensemble learning models. However, it requires an appropriate

[2]AMD EPYC 7702 64-Core

non-conformity measure to be implemented. In the original publication of CPSSDS, no such measures were proposed for ensemble methods. Therefore, we adopt the configuration that produced the best results in [44], which employs a Hoeffding Tree as the base learner. For SDSL, we follow a configuration that closely aligns with the original publication while adjusting certain aspects based on preliminary experiments. The replay buffer size was set to 200, which ensures greater stability when incorporating pseudo-labeled data. The pretraining phase was performed with $1,000$ instances, providing SDSL with an initial labeled dataset that, in theory, offers an advantage over the other methods. However, as seen in the results, this additional labeled data did not significantly improve its performance. The pseudo-label confidence threshold was set to 0.8, with other key parameters (learning rate $= 0.01$, adversarial step size $= 0.01$), and centroid refinement iterations $= 5$) following the original implementation. For SSALD, we maintain the window size at $= 100$ and the label budget $= 0$, ensuring that the active learning mechanism is disabled. The number of radial basis function (RBF) centers was set to 10, reflecting the configuration used in the original paper. The model also employs a loss threshold $= 0.0001$ to determine convergence during training, preventing excessive updates when pseudo-labeling stabilizes. The number of training rounds was limited to 100 iterations, preventing excessive computational overhead while ensuring the model adapts effectively to changing distributions. For IOE, we set the forgetting factor to 0.9, controlling how quickly past observations lose influence, and the threshold to 0.05, which dictates when ensemble adjustments occur. The pseudo-label confidence threshold was set to 0.8, aligning with similar self-training methods that employ confidence-based filtering.

The **evaluation framework** is based on the test-then-train approach. Every experiment was repeated five times, and the metrics presented are the averages and standard deviations of such experiments. To simulate the partially labeled stream, first we take a fully labeled stream. Next, we add a random chance p of a given instance never receiving its label. Either way, every instance is used for testing. Each one of the five executions varies the random seed of the random labeling process, therefore we anticipate high standard deviations for the results as which instances are labeled varies drastically from one execution to the other. The intent behind this evaluation framework is that at the core of our problem setting is the lack of control over which instances are labeled, therefore we need to investigate what is the impact on the overall performance as we use different labeled instances for the same data. Similar evaluation frameworks for semi-supervised experiments can be found in the literature [14]. However, some frameworks assume that labeled data is only available during the beginning of the execution [48], while others also assume that labeled instances can be requested on demand (active learning) [16], [22], [40], which would modify our problem setting assumptions.

The **datasets** include seven synthetic data streams, three real data streams (snapshots), and reshuffled versions of the real streams. Our goal was to represent different problems with a different number of features, class labels, different kinds of concept drifts, synthetic generators with different biases and reshuffled datasets. In Table I, we provide the characteristics of each dataset.

TABLE I
DATASETS (DRIFTS: (A) ABRUPT, (G) GRADUAL, (F) FEATURE, (I_m) INCREMENTAL (MODERATE) AND (I_f) INCREMENTAL (FAST)).

Dataset	# Instances	# Features	Type	Drifts	# Classes
LED_a	100,000	24	Synthetic	A	10
LED_g	100,000	24	Synthetic	G	10
AGR_a	100,000	9	Synthetic	A	2
AGR_g	100,000	9	Synthetic	G	2
RBF_m	100,000	10	Synthetic	I_m	5
RBF_f	100,000	10	Synthetic	I_f	5
CovtFD	581,012	104	Synthetic	A,F	7
AIRL	539,383	7	Real	-	2
ELEC	45,312	8	Real	-	2
COVT	581,012	54	Real	-	7
$AIRL_r$	539,383	7	Reshuffled	-	2
$ELEC_r$	45,312	8	Reshuffled	-	2
$COVT_r$	581,012	54	Reshuffled	-	7

Six of the synthetic datasets were generated to simulate noisy (LED_a and LED_g) and drifting data (abrupt (LED_a, AGR_a), gradual (LED_g, AGR_g) and incremental (RBF_m: moderate changes, RBF_f: fast changes)). All synthetic streams were generated using the MOA library [3]. LED_a and LED_g both have 10% label noise. AGR_a and AGR_g do not include label noise but were configured with perturbation fraction $= 0.05$[3]. Neither RBF_m nor RBF_f present label noise. The last synthetic dataset, CovtFD [49], is a version of Covertype [50] that simulates feature drift: after processing $1/3$ and $2/3$ of the data, the location of each valid numeric feature is randomly switched with one of fifty random noise features. Also, CovtFD can be challenging if the learning algorithm is susceptible to noisy features as it contains 50 random numeric variables. The real datasets are well-known in the community AIRLINES (AIRL), Covertype (COVT), and Electricity (ELEC) data sets. It is impossible to determine if concept drifts occur on these datasets. However, they have some challenging characteristics, such as a known temporal dependence between instances in ELEC, complex nominal features in AIRL, and an uneven distribution of class labels over time in COVT. We included "reshuffled" versions of each real dataset to investigate how SLEADE behaves when the temporal aspects (e.g. concept drifts) are removed.

A. SLEADE in comparison to others

In this section, the results of all experiments show the average and standard deviation of the test-then-train accuracy calculated over 5 runs. Tables II and III depict the results for 1% and 5% labeled data using 10 learners, and Tables IV and V show results for 1% and 5% labeled data using 30 learners. Tables IV and V include results only for ensemble methods (and ClusterTL), since results for other methods are not affected by the number of learners. We highlight that SSALD can only deal with binary problems, therefore it was not included in the ranking calculations as all the multi-class results are missing. For the binary classification

[3]The perturbation fraction (perturbFraction) is the amount of perturbation (noise) introduced to numeric values.

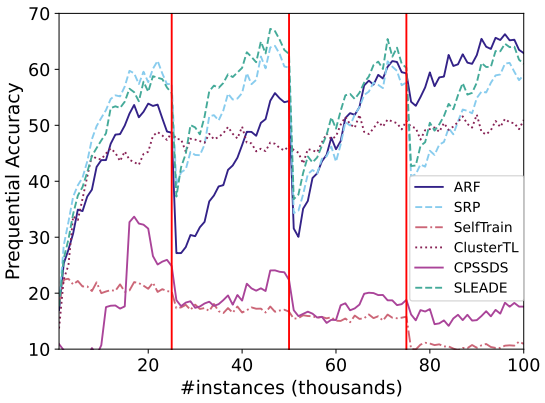
tasks, in all cases SSALD provided lower accuracy results than those of the best supervised method, i.e. SRP, and SLEADE. SLEADE outperforms the fully supervised models ARF and SRP consistently in all experiments. Also, by using 30 base learners, SLEADE consistently outperforms its version using only 10 learners and increases the difference to the other supervised and semi-supervised learners. At the same time, results provided in Tables II – V confirm how difficult learning with sparsely labeled evolving data streams is. In both cases, 1% and 5% labeling, the average rankings of SelfTrain, ClusterTL, CPSSDS, SSADL, IOE and SDSL methods are worse than those of the fully-supervised methods.

We observe that ClusterTL yields undisputed results for RBF variants, which is reasonable because although RBF_m and RBF_f are challenging incremental drift problems, they perfectly match ClusterTL clustering assumption, i.e. instances close together in the space belong to the same class. The predictive performance of CPSSDS was comparable to SelfTrain, but it was not as effective as SLEADE or ClusterTL. We see from the results presented in Tables II and III that when 5% labeled data was used, CPSSDS showed substantial improvement compared to both when it was used with only 1% labeled data and to SelfTrain in both labeling settings. This is particularly noticeable in LED and AGR variants.

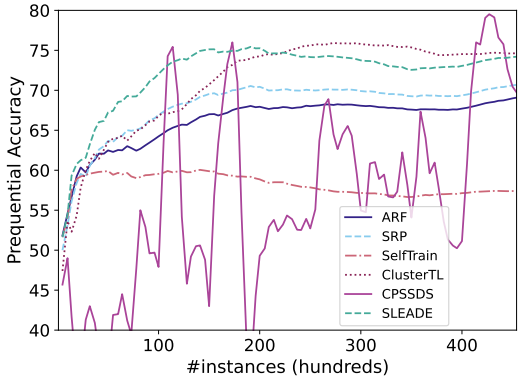
Importantly, ClusterTL, i.e. the second best semi-supervised method compared to SLEADE, in the case of some of the streams suffers from major accuracy reduction compared to SRP. As documented in Table II, the accuracy for e.g. the LED_a data (SRP: 51.34%, ClusterTL: 46.64%) and the COVT data (SRP: 74.59%, ClusterTL: 63.11%) is much higher when SRP rather than the semi-supervised ClusterTL method is used. This performance gap between the SRP method ClusterTL is even larger for some of the sets, when SRP ensembles with 30 members are used. Importantly, while ClusterTL also yields accuracy benefits for some of the other streams, SLEADE avoids such significant performance reductions compared to the fully supervised SRP, which is crucial in practical settings. Out of the remaining SSL methods, only IOE yields better accuracy than SLEADE for two datasets, i.e. RBF_f and ELEC_r data, in the 1% labeled setting, but the average rank of IOE is only 4.31 in the best case. Furthermore, IOE does not benefit from having more labeled instances as the accuracy on 5% labeled data is reduced in comparison to the 1% setting. Despite its pretraining advantage (1000 labeled instances before streaming), SDSL's accuracy remained lower than SLEADE in all datasets. This suggests that its short memory mechanism and adversarial regularization are insufficient for handling evolving data streams.

In general, SLEADE outperforms other algorithms, even though the gap narrows between SLEADE and SRP when 5% of the labels are available and 30 learners are used (Table V). This can be explained by the fact that as more labels are available, a robust supervised method is expected to gradually reach its maximum performance. Hence, performance gains arising from adding more labeled data in the form of pseudo-labeled instances are inevitably limited. In other words, already available instances with ground truth labels may be sufficient to (almost) reach maximum performance attainable by a fully supervised method (such as SRP). However, real-world applications typically operate in highly label-scarce environments. In those cases, SLEADE's ability to leverage disagreement and pseudo-labeling remains a crucial advantage.

In Figure 1a, we observe the accuracy impact of abrupt concept drifts in the LED_a data. We note that after each concept drift SLEADE consistently outperformed SRP, even though it was not as effective as ARF after the last drift. ClusterTL and SelfTrain produce stable results, even though both depict consistently low accuracy with SelfTrain reaching base accuracy in the last concept (around 10% given 10 equally distributed class labels). In Figure 1b, we present the accuracy achieved by each learner in ELEC. As shown in Table II, overall, ClusterTL outperforms others, but SLEADE closely follows ClusterTL and yields actually an even higher accuracy during the first approximately $20,000$ instances. Also, SLEADE outperforms SRP consistently in this experiment. We notice that the CPSSDS method was quite inconsistent in this dataset, likely due to multiple false detections and restarts, and this may explain the low accuracy seen in the scenario with 1% labeled data, as shown in Table II. IOE, SSALD, and SDSL were omitted from the plots because their accuracy was consistently lower than that of the top-performing methods. Including them would not provide additional insight and would clutter the visualization.



(a) LED_a - 10 learners, 1% labeled. Red vertical line denotes drift.



(b) ELEC - 10 learners, 1% labeled

Fig. 1. Prequential accuracy for LED_a and ELEC.

TABLE II
ACCURACY FOR 1% LABELED DATA. ENSEMBLES WITH 10 LEARNERS AND 100 CLUSTERS FOR CLUSTERTL.

	ARF	SRP	SelfTrain	ClusterTL	CPSSDS	IOE	SDSL	SSALD	SLEADE
LED_a	49.22 ± 1.1	51.34 ± 1.81	16.07 ± 3.82	46.64 ± 1.84	18.11 ± 2.18	37.13 ± 2.06	26.12 ± 1.24	n/a	**53.45 ± 3.22**
LED_g	47.02 ± 2.56	50.88 ± 1.60	16.15 ± 3.87	46.38 ± 2.21	18.29 ± 1.16	38.16 ± 1.71	26.01 ± 0.87	n/a	**53.54 ± 1.94**
AGR_a	53.73 ± 0.43	60.17 ± 3.08	55.91 ± 4.09	49.30 ± 0	57.58 ± 2.30	51.49 ± 1.37	49.77 ± 0.27	49.26 ± 0.08	**64.64 ± 2.62**
AGR_g	53.73 ± 0.75	60.97 ± 4.26	56.69 ± 2.27	49.30 ± 0	59.55 ± 0.88	51.86 ± 3.14	49.48 ± 0.14	49.88 ± 0.83	**62.76 ± 2.39**
RBF_m	33.93 ± 0.97	35.94 ± 2.17	29.24 ± 1.82	**92.12 ± 1.05**	26.58 ± 2.51	28.65 ± 1.34	28.30 ± 0.59	n/a	39.90 ± 0.49
RBF_f	29.42 ± 0.39	29.71 ± 0.48	29.15 ± 1.80	**92.75 ± 0.73**	23.51 ± 2.98	31.10 ± 1.90	29.55 ± 0.52	n/a	29.91 ± 0.28
CovtFD	68.89 ± 0.56	68.29 ± 0.63	13.99 ± 9.13	37.35 ± 20.16	25.66 ± 9.83	53.12 ± 1.11	26.00 ± 4.45	n/a	**70.37 ± 0.15**
AIRL	60.09 ± 0.14	59.80 ± 0.33	55.22 ± 1.07	55.46 ± 0	52.42 ± 2.79	57.70 ± 0.31	53.22 ± 1.16	51.87 ± 4.05	**60.45 ± 0.68**
ELEC	69.08 ± 3.69	70.68 ± 3.35	57.40 ± 11.17	**74.63 ± 2.55**	56.57 ± 2.07	69.14 ± 1.82	53.47 ± 3.06	52.61 ± 0.75	74.20 ± 1.94
COVT	72.42 ± 0.64	74.59 ± 0.79	40.26 ± 10.49	63.11 ± 7.98	48.30 ± 6.16	56.30 ± 0.64	34.74 ± 13.13	n/a	**75.08 ± 0.51**
$AIRL_r$	**58.47 ± 0.29**	57.55 ± 0.77	55.81 ± 0.93	55.46 ± 0	49.85 ± 0.84	55.18 ± 0.35	51.32 ± 1.91	54.10 ± 1.13	57.62 ± 0.24
$ELEC_r$	67.42 ± 4.18	70.04 ± 2.83	54.06 ± 4.52	**77.02 ± 3.26**	65.38 ± 3.02	73.69 ± 1.08	56.20 ± 2.94	50.95 ± 0.43	71.71 ± 0.78
$COVT_r$	64.96 ± 1.62	68.92 ± 0.86	47.88 ± 5.51	**81.66 ± 1.82**	25.47 ± 2.67	62.87 ± 0.54	46.68 ± 0.51	n/a	68.64 ± 0.33
Avg Rank.	3.69	2.69	6.23	3.69	6.62	4.62	6.77	n/a	**1.69**

TABLE III
ACCURACY FOR 5% LABELED DATA. ENSEMBLES WITH 10 LEARNERS AND 100 CLUSTERS FOR CLUSTERTL.

	ARF	SRP	SelfTrain	ClusterTL	CPSSDS	IOE	SDSL	SSALD	SLEADE
LED_a	63.34 ± 1.96	64.49 ± 1.10	23.95 ± 4.09	48.46 ± 2.71	53.33 ± 3.33	50.32 ± 0.82	26.48 ± 1.41	n/a	**64.73 ± 2.09**
LED_g	63.25 ± 2.28	63.96 ± 1.87	20.84 ± 2.04	46.31 ± 4.17	48.22 ± 5.12	50.85 ± 1.06	26.30 ± 1.07	n/a	**64.27 ± 1.18**
AGR_a	57.10 ± 0.66	67.32 ± 2.60	68.64 ± 1.76	49.30 ± 0	61.19 ± 2.26	55.87 ± 1.52	49.18 ± 0.48	49.91 ± 1.51	**73.83 ± 3.95**
AGR_g	57.29 ± 0.93	66.90 ± 4.30	66.08 ± 4.38	49.30 ± 0	60.56 ± 1.65	55.53 ± 0.66	49.33 ± 0.34	49.81 ± 0.83	**71.83 ± 2.20**
RBF_m	56.76 ± 0.99	58.53 ± 0.51	34.92 ± 1.71	**96.99 ± 0.36**	43.33 ± 1.04	30.19 ± 0.15	28.59 ± 0.47	n/a	56.27 ± 0.96
RBF_f	33.48 ± 0.21	35.05 ± 0.50	29.05 ± 1.48	**97.47 ± 0.41**	22.12 ± 0.91	34.04 ± 1.45	29.67 ± 0.36	n/a	35.08 ± 0.20
CovtFD	76.39 ± 0.21	76.89 ± 0.51	59.30 ± 3.54	39.92 ± 0.63	10.89 ± 1.78	63.97 ± 1.07	20.78 ± 2.13	n/a	**77.64 ± 0.27**
AIRL	61.71 ± 0.14	62.55 ± 0.52	62.24 ± 0.44	55.46 ± 0	57.55 ± 0.40	58.43 ± 0.23	53.13 ± 1.19	54.52 ± 0.69	**63.22 ± 0.36**
ELEC	76.73 ± 0.69	77.85 ± 0.65	73.24 ± 1.15	**86.73 ± 7.53**	59.03 ± 0.84	74.52 ± 0.37	50.28 ± 3.77	56.72 ± 0.41	78.75 ± 0.57
COVT	81.76 ± 0.18	**83.21 ± 0.27**	71.05 ± 0.45	69.41 ± 2.43	49.81 ± 4.55	68.11 ± 0.84	31.06 ± 14.02	n/a	82.03 ± 0.27
$AIRL_r$	59.76 ± 0.19	59.43 ± 0.40	**59.99 ± 0.23**	55.46 ± 0	55.18 ± 0.70	55.33 ± 0.09	51.22 ± 2.08	54.26 ± 1.50	59.77 ± 0.87
$ELEC_r$	74.96 ± 1.20	75.41 ± 1.36	60.62 ± 9.45	**87.48 ± 4.73**	66.02 ± 1.20	73.33 ± 0.21	54.33 ± 3.20	51.71 ± 0.29	75.60 ± 0.52
$COVT_r$	72.16 ± 0.22	72.32 ± 0.47	50.49 ± 2.60	**88.76 ± 1.08**	14.42 ± 3.20	63.79 ± 0.42	46.66 ± 0.42	n/a	71.92 ± 0.54
Avg Rank.	3.69	2.38	5.08	4.23	6.08	5.23	7.46	n/a	**1.85**

TABLE IV
ACCURACY FOR 1% LABELED DATA. ENSEMBLES WITH 30 LEARNERS AND 300 CLUSTERS FOR CLUSTERTL.

	ARF	SRP	SelfTrain	ClusterTL	IOE	SLEADE
LED_a	50.38 ± 1.12	55.12 ± 1.57	17.68 ± 4.38	48.48 ± 6.05	37.10 ± 1.88	**56.08 ± 1.30**
LED_g	48.50 ± 1.73	55.34 ± 2.30	17.51 ± 4.12	47.74 ± 6.17	35.48 ± 2.65	**55.62 ± 0.83**
AGR_a	53.75 ± 0.62	59.75 ± 3.97	56.88 ± 4.61	49.30 ± 0.00	51.19 ± 1.04	**66.40 ± 2.36**
AGR_g	53.67 ± 0.77	59.92 ± 2.96	55.17 ± 1.10	49.30 ± 0.00	50.81 ± 2.47	**65.34 ± 1.52**
RBF_m	34.47 ± 1.05	36.57 ± 1.56	29.27 ± 1.82	**96.98 ± 1.00**	29.70 ± 0.31	41.30 ± 0.98
RBF_f	29.49 ± 0.52	29.89 ± 0.53	29.24 ± 1.84	**98.04 ± 0.57**	30.94 ± 0.41	30.12 ± 0.20
CovtFD	69.14 ± 0.23	69.31 ± 0.49	21.27 ± 7.47	24.67 ± 18.43	52.97 ± 0.81	**70.80 ± 0.26**
AIRL	60.33 ± 0.18	60.42 ± 0.21	55.77 ± 2.19	55.46 ± 0.00	57.42 ± 0.16	**61.30 ± 0.57**
ELEC	69.04 ± 3.74	73.26 ± 2.42	57.54 ± 7.44	61.66 ± 1.28	68.99 ± 2.04	**74.58 ± 1.64**
COVT	73.03 ± 0.69	75.46 ± 0.32	48.79 ± 5.13	70.14 ± 4.23	55.65 ± 0.30	**75.56 ± 0.33**
$AIRL_r$	**58.66 ± 0.24**	58.21 ± 0.52	56.82 ± 0.91	55.46 ± 0.00	55.09 ± 0.27	57.90 ± 0.59
$ELEC_r$	67.56 ± 3.79	71.20 ± 2.20	54.22 ± 7.74	63.34 ± 4.96	**73.49 ± 0.60**	72.47 ± 0.98
$COVT_r$	65.92 ± 1.90	**69.75 ± 0.51**	48.75 ± 7.31	68.86 ± 1.28	61.93 ± 1.15	69.09 ± 0.32
Avg Rank.	3.38	2.23	5.31	4.23	4.31	**1.54**

TABLE V
ACCURACY FOR 5% LABELED DATA. ENSEMBLES WITH 30 LEARNERS AND 300 CLUSTERS FOR CLUSTERTL.

	ARF	SRP	SelfTrain	ClusterTL	IOE	SLEADE
LED_a	64.77 ± 1.26	67.45 ± 0.48	24.42 ± 3.44	60.07 ± 7.02	50.10 ± 0.75	**68.15 ± 0.87**
LED_g	64.27 ± 1.83	66.39 ± 0.93	23.13 ± 3.12	59.46 ± 5.94	50.55 ± 1.44	**66.87 ± 0.89**
AGR_a	58.22 ± 1.06	68.04 ± 2.20	68.77 ± 1.19	49.30 ± 0.00	56.43 ± 2.07	**73.12 ± 1.87**
AGR_g	57.61 ± 1.29	68.68 ± 3.20	66.14 ± 1.90	49.30 ± 0.00	55.28 ± 1.33	**71.89 ± 3.59**
RBF_m	59.78 ± 0.60	62.63 ± 1.69	34.71 ± 0.88	**98.68 ± 0.33**	30.11 ± 0.06	57.75 ± 1.31
RBF_f	34.60 ± 0.71	36.55 ± 0.72	29.61 ± 1.91	**98.91 ± 0.23**	33.69 ± 1.73	35.76 ± 0.55
CovtFD	77.36 ± 0.28	78.14 ± 0.31	58.46 ± 3.05	39.84 ± 0.73	64.88 ± 0.78	**78.22 ± 0.15**
AIRL	62.13 ± 0.07	63.86 ± 0.28	63.79 ± 0.29	55.46 ± 0.00	57.99 ± 0.24	**64.06 ± 0.26**
ELEC	77.42 ± 0.64	78.84 ± 0.29	74.16 ± 1.43	**87.60 ± 6.13**	75.08 ± 0.12	79.33 ± 0.29
COVTYPE	82.66 ± 0.19	**84.21 ± 0.19**	69.90 ± 0.77	77.46 ± 5.25	66.92 ± 0.53	82.88 ± 0.21
$AIRL_r$	60.17 ± 0.15	**61.14 ± 0.20**	60.72 ± 0.27	55.46 ± 0.00	55.20 ± 0.08	60.99 ± 0.76
$ELEC_r$	75.59 ± 0.82	76.15 ± 0.65	59.17 ± 5.58	**84.49 ± 6.85**	73.72 ± 0.72	76.20 ± 0.35
$COVT_r$	72.57 ± 0.30	**73.21 ± 0.37**	49.99 ± 1.89	71.91 ± 4.04	63.35 ± 0.54	72.74 ± 0.30
Avg Rank.	3.54	2.00	4.77	3.77	5.15	**1.77**

B. SLEADE analysis

In this section, we analyze SLEADE's characteristics to understand its advantages and limitations. The previous experiments show that SLEADE outperforms other supervised and semi-supervised learners using either 5% or 1% labeled data. Furthermore, it also avoids the risk of significant accuracy loss compared to a robust supervised method like SRP, an important characteristic that grants more stability to the method. In practical terms, it is desirable that when SLEADE cannot improve upon the supervised method, it at least does not catastrophically jeopardize it.

At the core of SLEADE resides the Disagreement-Based Learning strategy and the unsupervised drift coping mechanism. On top of that, SLEADE execution is influenced by its hyperparameters n and Φ. An ablation study is warranted to shed light on the impact of each of these aspects in

TABLE VI
ACCURACY FOR 1% LABELED DATA VARYING THE SLIDING WINDOW SIZE n, AND MINIMUM CONFIDENCE THRESHOLD Φ. ENSEMBLE LEARNER SIZE = 10, EXCEPT FOR THE LAST TWO COLUMNS (30 LEARNERS). HIGHLIGHTED IN BOLD THE BEST RESULTS COMPARING THE LAST TWO COLUMNS ONLY.

	SLEADE	***No DD***	**Sliding Window Size n, $\Phi = 0.9$**						**Min. Confidence Φ, $n = 100$**							**30 Learners (n, Φ)**	
	$(100, 0.9)$	$(-, 0.9)$	0	50	150	250	500	1000	0.0	0.3	0.5	0.6	0.7	0.8	1.0	$(100, 0.9)$	$(100, 1.0)$
LED_a	53.40	48.95	52.10	53.35	54.17	53.70	53.04	52.13	52.61	53.60	52.43	53.24	53.99	53.60	53.41	**56.08**	55.46
LED_g	53.54	48.92	50.35	52.89	53.50	53.51	52.20	51.29	52.94	53.26	52.12	53.23	53.48	52.48	51.58	**55.62**	55.48
AGR_a	64.64	59.22	61.44	63.54	63.50	63.27	64.68	65.21	64.33	64.33	64.33	64.49	64.65	64.21	65.49	**66.40**	65.98
AGR_g	62.76	59.37	61.73	63.51	63.09	63.41	63.39	63.50	63.99	63.99	63.99	64.16	63.75	63.83	64.69	**65.34**	64.95
RBF_m	39.90	34.83	35.52	40.09	38.90	37.41	36.14	36.34	39.84	39.44	39.74	40.15	39.24	40.21	41.34	**41.30**	40.76
RBF_f	29.91	29.72	29.30	29.92	29.98	29.99	30.02	29.99	29.97	29.97	29.95	30.22	29.82	30.00	30.07	**30.12**	30.10
CovtFD	70.37	65.78	64.85	69.79	70.15	69.72	68.54	67.46	70.15	70.15	70.25	70.41	70.41	70.27	70.40	70.80	**70.84**
AIRL	60.45	59.10	60.60	59.91	60.41	60.34	61.27	61.21	60.39	60.39	60.39	60.26	60.24	60.25	59.73	**61.30**	61.12
ELEC	74.20	72.31	71.50	74.33	74.37	74.33	74.47	74.47	74.41	74.41	74.41	74.16	73.81	74.41	74.44	74.58	**74.76**
COVT	75.08	72.41	70.33	74.98	75.00	74.60	74.08	73.65	75.07	75.07	75.02	74.65	74.94	74.99	74.95	75.56	**75.65**
AIRL_r	57.62	57.09	57.33	57.36	57.42	58.05	58.39	58.95	57.68	57.68	57.68	57.45	57.33	57.20	57.19	**57.90**	57.60
ELEC_r	71.71	70.18	70.77	71.26	71.64	71.77	71.69	71.69	72.05	72.05	72.05	71.68	71.70	71.84	71.56	**72.47**	71.80
COVT_r	68.64	68.42	67.42	68.34	68.53	68.53	68.79	68.72	67.86	67.86	68.04	68.32	68.38	68.21	68.79	69.09	**69.27**

SLEADE. We focus on answering the most pressing questions related to SLEADE using Table VI [4] , such as the impact of disabling the concept drift detection (*No DD*) and disabling the pseudo-labeling strategy while maintaining the drift detection ($\Phi = 1.0$, 3rd column from right to left in Table VI). On top of that, we also investigate the impact of varying hyperparameters n and Φ.

1) Ablation study and hyperparameter settings: In Table VI, as expected, we observed that a version of SLEADE without drift detection (*No DD*) yields significantly worse results, especially for the synthetic datasets, and no significant changes in the predictive performance for the reshuffled datasets. We also verified the impact of the size of the sliding window of labeled instances (hyperparameter n) and noticed that $n = 0$ yields the worst results in comparison to others as it only detects and resets models but does not train the new models on labeled data. The default value used, $n = 100$, was the most stable throughout all the datasets, and as expected, as we increased n, such as $n = 1000$, there was a decrease in accuracy for datasets in which drift happens, and some marginal improvements in the reshuffled versions. The improvements were marginal because there are only spurious detections in the reshuffled datasets (as expected). This leads to fewer resets, thus less impact of whatever n value is used. To sum up, results provided in Table VI show that while we set $n = 100$ in the other experiments summarised in Tables II-V, SLEADE is relatively robust to other $n > 0$ settings i.e. $n \in \{50, 150, 250, 500, 1000\}$.

The pseudo-labeled instance selection process (i.e. 'majority trains the minority') is a fundamental characteristic of SLEADE. Several aspects, such as the weight assigned to the pseudo-labeled instances, influence this process. Some external control over the process is possible through the minimum confidence threshold Φ. Therefore, in Table VI we show the impact of Φ from $\Phi = 0.0$ to $\Phi = 1.0$. We noticed that smaller values are not advisable as they permit low confident pseudo-labels and negatively impact the overall performance. The best overall results were obtained using $\Phi = 0.9$, which is the value used in the default configuration of the algorithm. However, $\Phi = 1.0$ also yields reasonably good results in the 1% labeled setting when using 10 learners. This result led us to investigate if that would also happen if 30 learners were used, so we repeated the experiment comparing $\Phi = 0.9$ and $\Phi = 1.0$ using 30 learners in the last two columns of Table VI. The difference in accuracy widened in favor of $\Phi = 0.9$. This confirms the positive contribution of the 'majority trains the minority' process to the performance of the best SLEADE models.

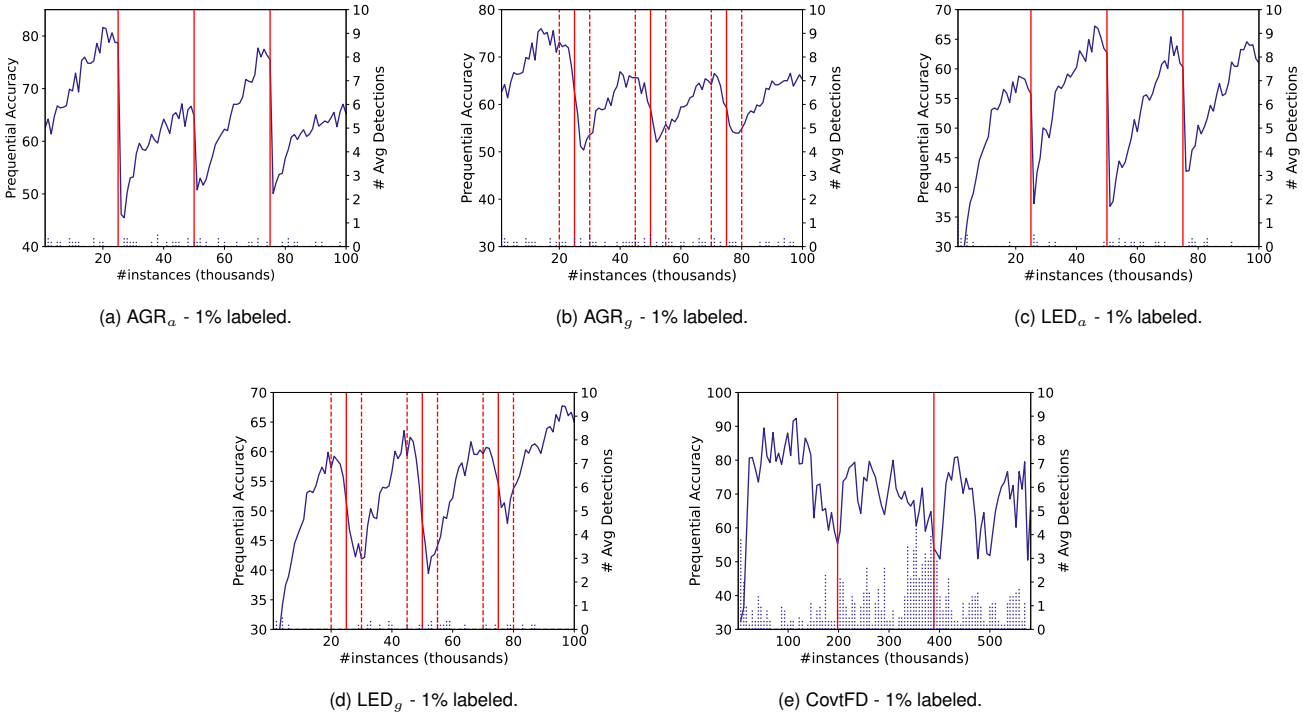
[4]Standard deviation omitted due to space limitations

TABLE VII
AVERAGE NUMBER OF DRIFT DETECTIONS (5 RUNS) IN SLEADE VARYING THE NUMBER OF LABELS FROM 1% TO 5%. 10 LEARNERS FOR ENSEMBLE METHODS.

Labeled	1%	2%	3%	4%	5%
LED_a	8.2	4.8	7.2	5.4	6.0
LED_g	8.6	6.4	6.0	5.2	5.8
AGR_a	11.4	8.8	10.8	8.6	8.2
AGR_g	13.8	10.2	9.2	8.8	7.2
RBF_m	14.2	12.0	13.6	11.2	10.0
RBF_f	19.0	17.8	16.6	17.6	18.6
CovtFD	118.8	114.6	110.0	112.8	112.6
AIRL	33.2	30.6	31.8	30.0	28.4
ELEC	10.6	10.4	7.4	7.0	8.6
COVT	143.2	159.8	151.8	166.4	175
AIRL_r	26.2	22.8	23.0	21.6	19.6
ELEC_r	5.6	3.0	0.8	1.2	0.8
COVT_r	9.8	6.0	5.6	4.0	3.2

2) Drift detection analysis: The results in Table VI show that the unsupervised drift detection and recovery approach has a positive impact on the overall performance, especially when concept drifts take place (e.g. LED and AGR variants). To further examine the impact of the drift coping mechanism, Table VII displays the average number of detections for the 1% labeled data setting with 10 learners. These results show that in all streams drift has been detected. Every time this happened, some ensemble members were replaced, which directly follows from Alg. 1. This confirms the contribution of drift detection to the results obtained by SLEADE for all streams. To visualize where drift detections occurred, we plot the predictive performance of SLEADE over time, and the detected drifts in Figures 2a to 2e. The results in these plots are also the average of 5 executions, thus we grouped detections that took place closer together (in the block of 1/100 of the data) and averaged the results.

In Table VII, we expect the number of detections to approach zero for reshuffled datasets. However, in complex datasets like AIRL_r a high number of detections is anticipated due to the instability of ensemble (teacher) predictions, leading to spurious detections. Conversely, the low number of detections in drifting streams like LED and AGR is promising. Even though the Drift locations for LED and AGR variants



(a) AGR$_a$ - 1% labeled. (b) AGR$_g$ - 1% labeled. (c) LED$_a$ - 1% labeled.

(d) LED$_g$ - 1% labeled. (e) CovtFD - 1% labeled.

Fig. 2. The accuracy of SLEADE (10 learners), true drift locations, prequential accuracy and the average amount of detections every 1/100 of the stream. Red solid vertical lines represent true drift locations, and red dashed lines the period of gradual drift, while blue dashed lines at the bottom represent SLEADE drift detections.

depicted in Figs. 2a to 2d seem to exhibit constant detections the average number of drifts per detection is typically below 1, often around 0.2 or 0.4. As these results are averaged over five runs, detections likely occurred at different stream points in each run. This is supported by the increasing trend in average prequential accuracy after each drift, indicating that continuous resets were not occurring.

3) Pseudo-label accuracy: Finally, the performance of SSL algorithms in terms of pseudo-label accuracy is depicted in Figure 3. Each method has a unique approach for deciding which instances to pseudo-label and how to incorporate these pseudo-labels into their model. SLEADE tries to pseudo-label each incoming instance for each ensemble member, leading to a potential increase in the number of pseudo-labeled instances, up to $(|L| - 1) \times |X_u|$, where $|X_u|$ is the total number of unlabeled instances. However, not all of the unlabeled instances are actually used for pseudo-labeling in SLEADE.

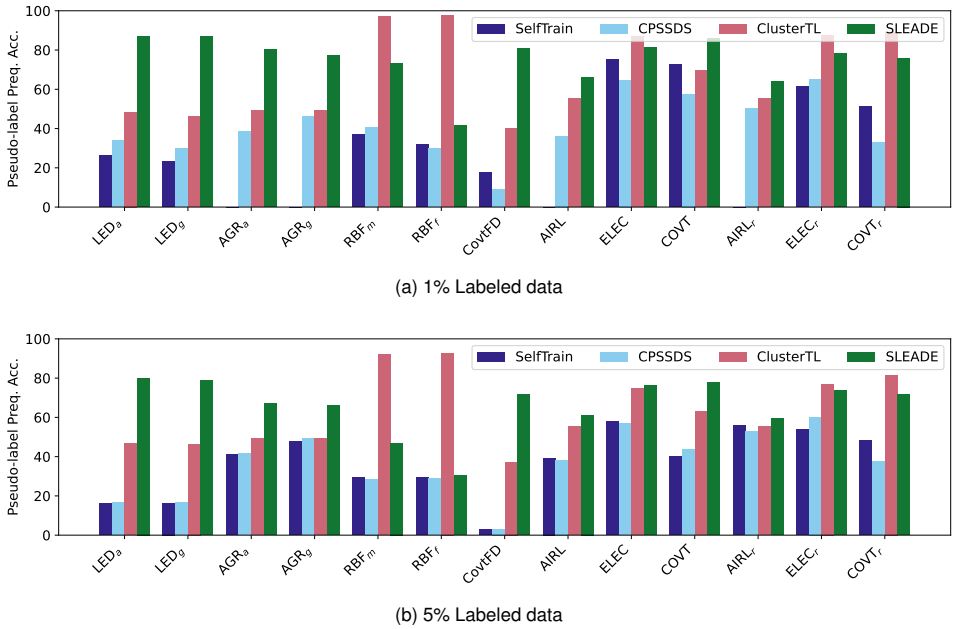
Upon inspection of Figure 3, we can observe that SLEADE pseudo-labeling strategy yield accurate results in comparison to other methods. Furthermore, SLEADE presented a significant improvement in terms of pseudo-labeling accuracy comparing 1% to 5% labeling results. In all levels of labeling, ClusterTL had the highest pseudo-label accuracy for RBF$_m$ and RBF$_f$. This is because the data generator aligns with ClusterTL's clustering assumption. As the number of labeled instances increases, SLEADE's pseudo-label accuracy improves, which is reflected in the increase of overall accuracy in Tables II to V. The results of SelfTrain and CPSSDS show similar behavior and align with the predictive performance of each method from previous experiments.

VI. Conclusions

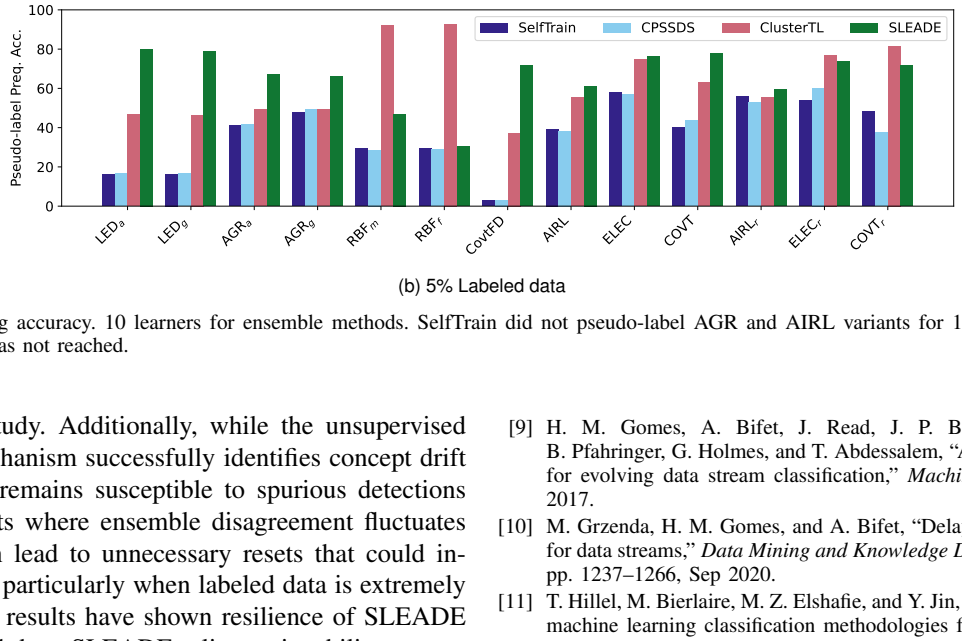
We present SLEADE, a semi-supervised learning algorithm that relies on learning by disagreement and unsupervised drift detection and recovery. We show how combining these techniques can leverage the predictive performance of an already accurate supervised ensemble model (SRP) given a significantly small number of labeled data (1% up to 5%). Furthermore, SLEADE's weight shrinkage and 'majority train the minority' mechanisms reduce the risks of a 'runaway' effect, where base models yield baseless confident predictions, which in turn impact the underlying supervised model, driving it off course long before it has seen enough true labels to become accurate. Importantly, SLEADE provides on average the highest ranked models and for all tested synthetic and real data streams avoids the risk of major performance losses compared to the fully supervised methods. This is unlike other SSL methods, which for some data streams are negatively influenced by the abundance of unlabeled data and yield models of substantially lower accuracy than those provided by the best fully supervised methods.

SLEADE requires two intuitive hyperparameters n and Φ. We analysed them and presented default values that work well for various datasets. In future work, we plan to expand the drift detection and recovery strategy further, including experiments with different algorithms for the student and detection algorithms.

While SLEADE demonstrates strong performance in various scenarios, it also has limitations. The effectiveness of its disagreement-based learning mechanism may depend on the diversity of the base learners, though this was not explicitly



(a) 1% Labeled data



(b) 5% Labeled data

Fig. 3. Pseudo-labeling accuracy. 10 learners for ensemble methods. SelfTrain did not pseudo-label AGR and AIRL variants for 1% labeled as minimum confidence threshold was not reached.

analyzed in our study. Additionally, while the unsupervised drift detection mechanism successfully identifies concept drift in many cases, it remains susceptible to spurious detections in complex datasets where ensemble disagreement fluctuates naturally. This can lead to unnecessary resets that could introduce instability, particularly when labeled data is extremely sparse, though our results have shown resilience of SLEADE to sparsely labelled data. SLEADE relies on its ability to generate meaningful pseudo-labels, and in highly imbalanced or evolving data streams, incorrect pseudo-labels can accumulate, affecting long-term performance. Future work could explore more robust drift detection strategies and alternative pseudo-labeling mechanisms to further mitigate these challenges.

Acknowledgment

Heitor Murilo Gomes acknowledges the financial support of the Marsden Fund under award number VUW2213.

References

[1] Z.-H. Zhou, *Machine learning*. Springer Singapore, 2021.

[2] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*. MIT Press, 2018.

[3] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *JMLR*, vol. 11, pp. 1601–1604, 2010.

[4] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[5] H. M. Gomes, M. Grzenda, R. Mello, J. Read, M. H. Le Nguyen, and A. Bifet, "A survey on semi-supervised learning for delayed partially labelled data streams," *ACM Computing Surveys (CSUR)*, 2022.

[6] C. Fahy, S. Yang, and M. Gongora, "Scarcity of labels in non-stationary data streams: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–39, 2022.

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[8] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633, 2011.

[9] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfahringer, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, pp. 1–27, 2017.

[10] M. Grzenda, H. M. Gomes, and A. Bifet, "Delayed labelling evaluation for data streams," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1237–1266, Sep 2020.

[11] T. Hillel, M. Bierlaire, M. Z. Elshafie, and Y. Jin, "A systematic review of machine learning classification methodologies for modelling passenger mode choice," *Journal of Choice Modelling*, vol. 38, p. 100221, 2021.

[12] I. Žliobaite, "Change with delayed labeling: When is it detectable?" in *IEEE ICDMW*, 2010, pp. 843–850.

[13] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-supervised learning*. MIT Press, 2006, vol. 2.

[14] M. H. Le Nguyen, H. M. Gomes, and A. Bifet, "Semi-supervised learning over streaming data using moa," in *IEEE International Conference on Big Data*, 2019, pp. 553–562.

[15] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *IJCNN*. IEEE, 2011, pp. 2741–2748.

[16] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *IEEE ICDE*, 2016, pp. 481–492.

[17] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 12–26, 2014.

[18] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowledge and Information Systems*, vol. 46, no. 3, pp. 567–597, 2016.

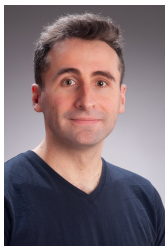
[19] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1607–1621, 2010.

[20] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *ACM SIGKDD*. ACM, 2011, pp. 195–203.

[21] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE TNNLS*, vol. 25, no. 1, pp. 27–39, 2013.

[22] B. Jiao, H. M. Gomes, B. Xue, Y. Guo, and M. Zhang, "A semi-supervised active learning neural network for data streams with concept drift," *IEEE Computational Intelligence Magazine*, vol. 20, no. 1, pp. 18–33, 2025.

[23] Z.-H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 415–439, 2010.
[24] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT*. ACM, 1998, pp. 92–100.
[25] H. He, D. Han, and J. Dezert, "Disagreement based semi-supervised learning approaches with belief functions," *Knowledge-Based Systems*, vol. 193, p. 105426, 2020.
[26] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.
[27] H. M. Gomes, J. Read, and A. Bifet, "Streaming random patches for evolving data stream classification," in *IEEE ICDM*, 2019, pp. 240–249.
[28] H. M. Gomes, J. Read, A. Bifet, and R. J. Durrant, "Learning from evolving data streams through ensembles of random patches," *Knowledge and Information Systems*, vol. 63, no. 7, pp. 1597–1625, 2021.
[29] H. M. Gomes, A. Lee, N. Gunasekara, Y. Sun, G. W. Cassales, J. Liu, M. Heyden, V. Cerqueira, M. Bahri, Y. S. Koh *et al.*, "Capymoa: Efficient machine learning for data streams in python," *arXiv preprint arXiv:2502.07432*, 2025.
[30] M. Li and Z.-H. Zhou, "Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 1088–1098, 2007.
[31] W. Wang and Z.-H. Zhou, "Theoretical foundation of co-training and disagreement-based algorithms," *arXiv preprint arXiv:1708.04403*, 2017.
[32] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE TKDE*, vol. 17, no. 11, pp. 1529–1541, 2005.
[33] M.-F. Balcan, A. Blum, and K. Yang, "Co-training and expansion: Towards bridging theory and practice," in *Advances in neural information processing systems*, 2005, pp. 89–96.
[34] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.
[35] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE TKDE*, vol. 31, no. 12, 2019.
[36] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *SIAM*, 2007.
[37] G. J. Aguiar and A. Cano, "A comprehensive analysis of concept drift locality in data streams," *Knowledge-Based Systems*, vol. 289, 2024.
[38] V. Cerqueira, H. M. Gomes, and A. Bifet, "Unsupervised concept drift detection using a student–teacher approach," in *International Conference on Discovery Science*. Springer, 2020, pp. 190–204.
[39] P. Domingos and G. Hulten, "Mining high-speed data streams," in *ACM SIGKDD*, Sep. 2000, pp. 71–80.
[40] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream." in *AAAI*, 2016, pp. 1652–1658.
[41] B. S. Parker and L. Khan, "Detecting and tracking concept class drift and emergence in non-stationary fast data streams." in *AAAI*, 2015.
[42] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "A practical approach to classify evolving data streams: Training with limited amount of labeled data," in *ICDM*. IEEE, 2008, pp. 929–934.
[43] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *International Conference on Very Large Data Bases (VLDB)*, 2003, pp. 81–92.
[44] J. Tanha, N. Samadi, Y. Abdi, and N. Razzaghi-Asl, "Cpssds: Conformal prediction for semi-supervised classification on data streams," *Information Sciences*, vol. 584, pp. 212–234, 2022.
[45] P. Vafaie, H. Viktor, and W. Michalowski, "Multi-class imbalanced semi-supervised learning from streams through online ensembles," in *IEEE ICDMW*, 2020, pp. 867–874.
[46] W. Ren, P. Wang, X. Li, C. E. Hughes, and Y. Fu, "Semi-supervised drifted stream learning with short lookback," in *ACM SIGKDD*, 2022.
[47] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *arXiv preprint arXiv:1804.09170*, 2018.
[48] K. B. Dyer and R. Polikar, "Semi-supervised learning in initially labeled non-stationary environments with gradual drift," in *IJCNN*. IEEE, 2012.
[49] H. M. Gomes, R. F. de Mello, B. Pfahringer, and A. Bifet, "Feature scoring using tree-based ensembles for evolving data streams," in *IEEE International Conference on Big Data*, 2019, pp. 761–769.
[50] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.



Heitor Murilo Gomes is a senior lecturer at the Victoria University of Wellington (VuW) in New Zealand. Before joining VuW, Heitor was a senior research fellow and co-director of the AI Institute at the University of Waikato. His main research area is the application of machine learning for data streams in a variety of tasks, such as ensemble learning, drift detection and adaptation, and the intersection of online continual learning and streaming data. He has contributed to several open-source projects like MOA (Massive Online Analysis) and recently lead the development of CapyMOA (www.capymoa.org), a python library for efficient data stream learning.

Jesse Read is a Professor in the Computer Science Laboratory (LIX) of Ecole Polytechnique in France. He obtained his PhD from the University of Waikato (New Zealand) in 2010, followed by a number of years of postdoctoral research at the Carlos III University of Madrid, Aalto University in Helsinki, and Télécom Paris (France). His research interests involve machine learning, and particularly multi-label classification, multi-target and structured prediction, and models for sequential data and evolving data streams, including reinforcement learning.

Maciej Grzenda (Senior Member, IEEE) received his Ph.D. from the Warsaw University of Technology (WUT). He is currently an Associate Professor at the Faculty of Mathematics and Information Science of WUT. His primary research interests include data preprocessing for ML, and the use of ML methods for industrial applications. Maciej Grzenda has been supervising the development of BSc and MSc programs in Data Science at WUT since 2016.

Bernhard Pfahringer received his PhD degree from the University of Technology in Vienna, Austria, in 1995. He is a Professor with the Department of Computer Science, and a co-director for the AI Institute, at the University of Waikato in New Zealand. His interests span a range of data mining and machine learning sub-fields, with a focus on streaming, randomization, and complex data. Bernhard is the co-author of the book "Machine Learning from Data Streams" published at MIT Press.

Albert Bifet (Member, IEEE) is the Director of the Te Ipu o te Mahara AI Institute at the University of Waikato and Co-chair of the Artificial Intelligence Researchers Association (AIRA). His research focuses on Artificial Intelligence, Big Data Science, and Machine Learning for Data Streams. He leads the TAIAO Environmental Data Science project and co-leading the open-source projects MOA Massive On-line Analysis, StreamDM for Spark Streaming and SAMOA Scalable Advanced Massive Online Analysis. He is the co-author of a book on Machine Learning from Data Streams published at MIT Press.