University of Reading

Adaptive learning with extreme verification latency in non-stationary environments

Article

Published Version

Creative Commons: Attribution 4.0 (CC-BY)

Open Access

Idrees, M. M. ORCID: https://orcid.org/0000-0002-2572-3251, Stahl, F. ORCID: https://orcid.org/0000-0002-4860-0203 and Badii, A. (2022) Adaptive learning with extreme verification latency in non-stationary environments. IEEE Access, 10. pp. 127345-127364. ISSN 2169-3536 doi: https://doi.org/10.1109/access.2022.3225225 Available at https://centaur.reading.ac.uk/109482/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

To link to this article DOI: http://dx.doi.org/10.1109/access.2022.3225225

Publisher: IEEE

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

CentAUR

University of Reading

Central Archive at the University of Reading

Reading's research outputs online

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 October 2022, accepted 15 November 2022, date of publication 28 November 2022,
date of current version 8 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225225

RESEARCH ARTICLE

Adaptive Learning With Extreme Verification Latency in Non-Stationary Environments

MOBIN M. IDREES[1], FREDERIC STAHL[2], AND ATTA BADII[1]
[1]Department of Computer Science, University of Reading, RG6 6AH Reading, U.K.
[2]German Research Center for Artificial Intelligence GmbH (DFKI), Marine Perception, 26129 Oldenburg, Germany

Corresponding author: Frederic Stahl (frederic_theodor.stahl@dfki.de)

This work was supported by the Ministry for Science and Culture, Lower Saxony, Germany, through Funds from the Niedersachsische Vorab, under Grant ZN3683 and Grant ZN3480.

ABSTRACT Existing Data Stream Mining algorithms assume the availability of labelled and balanced data streams. However, in many real-world applications such as Robotics, Weather Monitoring, Fraud-Detection systems, Cyber Security, and Human Activity Recognition, a vast amount of high-speed data is generated by Internet of Things sensors and real-time data on the Internet are unlabelled. Furthermore, the prediction models need to learn in Non-Stationary Environments due to evolving concepts. Manual labelling of these data streams is not practical due to the need for domain expertise and the time-resource-prohibitive nature of the required effort. To deal with such scenarios, existing approaches are self-Learning or Cluster-Guided Classification (CGC) which predict the pseudo-labels, which further update the prediction models. Previous studies have yet to establish a clear and conclusive view as to when, and why one pseudo-labelling approach should be preferable to another and what causes an approach to fail. In this research, we propose a novel approach, "*Predictor for Streaming Data with Scarce Labels*" (PSDSL), which is capable of intelligently switching between self-learning, CGC and micro-clustering strategies, based on the problem it is applied to, i.e., the different characteristics of the data streams. In PSDSL a novel approach called *Envelope-Clustering* has been introduced to resolve the conflict during the cluster labelling which suggested a confidence measure approach to ensure the quality and correctness of labels assigned to the clusters. The auto parameter tuning mechanism of PSDSL eliminates the human dependency and determines the best value of number of centroids from initial labelled data. The predictive performance of the PSDSL is evaluated on non-stationary datasets, synthetic data-streams, and real-world datasets. The approach has shown promising results on randomised datasets as well as on synthetic data-streams, as compared with state-of-the-art approaches. This is the first large-scale study on an adaptive extreme verification approach that supports automatic parameter tuning and intelligent switching of pseudo-labelling strategy, thus reducing the dependency of machine learning on human input.

INDEX TERMS Concept drift, data stream mining, extreme verification latency, non-stationary environment, semi-supervised learning.

I. INTRODUCTION

Data Stream Mining algorithms assume the availability of labelled data, immediately or after some delay, to update the accuracy of the classifier and update the prediction model. However, with certain applications such as Fraud-Detection Systems [1], Cyber Security [2], [3] and Human Activity Recognition [4], [71] etc. the data stream is unlabelled and manual labelling is impractical due to the cost, time, and the need for domain expertise. In machine learning literature, this scenario is referred to as Verification Latency (VL) [5], [6]. In another scenario, only limited labelled data is followed by completely unlabelled instances; this scenario is referred to as Extreme Verification Latency (EVL) [7], [8], [10], [11].

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

In many real-world applications of online data stream mining, the data originates from different sources such as sensor devices, social media, business/financial transactions, etc. The data evolves over time, and therefore extracting worthwhile knowledge is hard to achieve in such Non-Stationary Environments (NSEs). The underlying probability distributions of the data stream change over time, resulting in concept drifts [12].

This change occurs in a set of input variables 'x' and/or class labels 'y', i.e., $P_t\ (x,\ y) \neq P_{t+1}\ (x,\ y)$ at the time 't'. Two different types of concept drifts exist, i.e. ''Virtual drift'' [19] in which only the distribution of input data 'x' changes, i.e., $P_t(x) \neq P_{t+1}(x)$ and does not affect the class labels, i.e. $P_t\ (y|x) = P_{t+1}(y|x)$ whereas ''Real drift'' refers to any gradual or sudden changes in class labels due to changes in the distribution $P\ (y|x)$.

Initially a Labelled Non-Stationary Environment (ILNSE) addresses both EVL and NSEs issues simultaneously, for example, autonomous robots [65] are initially trained inside a specific environment on labelled data and known classes. Later, they are sent to explore an unknown environment without the supervision of humans. The robots also need to adapt themselves to changing environments and do so under the condition of lack of true class labels from sensor data. Another application is credit card fraud detection [1] in which the true class label of a particular transaction is unknown, and it is impossible to say whether it is fraud or non-fraud until the user receives and reviews the monthly statement. It is a VL scenario, because the true class labels are only available in the future, and it is also an NSE because the customers' patterns of spending change seasonally and/or during holidays due to changes in their geographic locations and these factors can result in concept drifts.

Learning under scarcity of class labels is challenging in data streams because the true class labels for future emerging data instances are not available. In data stream clustering, semantically similar objects are moved closer to each other, and the algorithms try to group similar objects. However, the clusters could easily be misclassified in the absence of true class labels. The choice of pseudo-labels in cluster labelling could be problematic as the pseudo-labels are predicted using the same model and due to NSEs these labels could make the models less reliable. In Cluster Guided Classification (CGC), the labels from the nearest clusters are transferred, however, the algorithm does not implement a confidence measure approach to assure the quality and correctness of labels assigned to the clusters.

In data streams the instances arrive in a sequential order which is directly fed into the online learning models thus storing and referring to the previous data is not practical due to time limitations. The output of an adaptive classifier at every time step depends on instances the classifier has been trained on to-date. Hence, performance depends on the order of instances in the dataset [68], Žliobaitė [68] suggested executing multiple tests with randomised copies of a data stream. Existing benchmarks for non-stationary datasets [8], [26] are designed to evaluate CGC on EVL, by inducing gradual shifting to the clusters. CGC shows promising results due to the high purity of clusters [76]; however, when the order of these datasets is randomised the CGC performance drops considerably. This supports the fact that the existing CGC approaches succeed only under certain conditions.

Real-time streaming data is usually unlabelled and unstructured; therefore, supervised learning algorithms are not very effective due to their dependency on class labels. Most of the algorithms on learning from NSEs - including prior efforts on Heterogeneous Dynamic Weighted Majority (HDWM) [13] - have focused on supervised approaches. The terms homogeneous and heterogeneous refer to the data mining algorithms used in the process, where homogeneous refers to the use of only one data mining algorithm, and heterogeneous refers to the use of different data mining algorithms [44]. HDWM makes use of ''seed'' learners, in which different types of classifiers maintain the diversity of the ensemble. To support EVL, HDWM was extended, and a new approach proposed, PSDSL. Fig.1 shows the positioning of the presented work in the literature. It is placed at the intersection of EVL and NSE, while data stream clustering and stream classification are mutually intersected with both EVL and NSE.

PSDSL automatically decides the use of the best classifier from a pool of heterogeneous classifiers, it can switch on the pseudo-labelling strategy, i.e., cluster guided, self-learning or micro-clustering, and selects whichever approach is beneficial, based on the characteristics of the data stream. We also introduce a new approach called envelope-clustering to resolve the conflict during the cluster labelling and suggested a confidence measure approach to ensure the quality and correctness of labels assigned to the clusters.

PSDSL is empirically evaluated against existing state-of-the-art approaches namely COMPacted Object Sample Extraction (COMPOSE) [21], Learning Extreme VErification Latency with Importance Weighting (LEVEL_{IW}) [11], Stream Classification Algorithm Guided by Clustering (SCARGC) [8] and Micro Cluster for Classification (MClassification) [70] on benchmarks NSE datasets [8] Massive Online Analysis (MOA) [25] data streams and real-world datasets [8]. We also introduced the hyperparameters tuning mechanism in PSDSL which assist the algorithm to automatically suggest the best value for the number of centroids 'k'. The predictive performance of PSDSL and SCARGC were also evaluated after randomising the benchmarks non-stationary datasets in which the training instances were shuffled by changing the training orders. The results showed that PSDSL performed significantly better than existing approaches when the instances of the datasets were randomised or noisy.

This paper is structured as follows. Section II presents related work. Section III then describes the proposed PSDSL approach. The experimental setup and experimental evaluation are described in Section IV. Finally, Section V provides a discussion of the presented research and sets out concluding remarks.

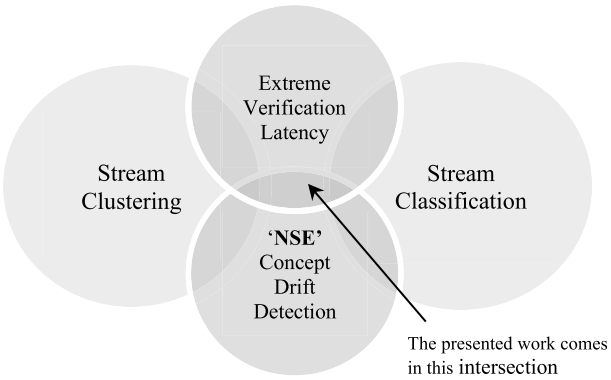

FIGURE 1. Standing of the presented work in the literature.

Several approaches exist that address the problems associated with NSE and EVL in isolation. However, few algorithms address both issues simultaneously. ILNSE is a challenging task because the learning algorithms have no access to the true class labels directly after the drift occurs. From the literature, it is not clear when and under what conditions one approach is better than the other, and what causes one approach to fail. Most of the real-world data streams are continuous and infinite. Unlike data mining, in data streams, there is no prior information about the number of classes and this value may change in the future. In some specific conditions, the CGC algorithms could be more effective than self-learning if the data is favouring clustering, i.e., high purity clusters. These issues make it difficult to choose the right EVL approach to different problems. This parameter has a great influence on clustering results. In offline machine learning, this parameter is iteratively tuned on finite datasets; however, the data streams are infinite and arrive at high speed and the existing EVL approaches are relying on the manual selection of parameter 'k'.

Set out below are the questions addressed in the presented research, which are followed by answers to the research questions and contributions.

- *RQ1: Which ILNSE approach is a better choice to deal with class label scarcity under non-stationary environments and is that approach always successful when applied to different problems? If not, why does it sometimes fail?*
- *RQ2: Do existing ILNSE approaches depend on parameters tuned by the human analyst before building the training models and what other factors are influence the prediction results?*
- *RQ3: What strategy should be adopted if one of the EVL approaches fail?*

The following is a summary of the answers to the research questions as an extract of the findings of the research presented in this paper. To deal with EVL under NSEs, the most successful approaches are self-learning and CGC. SCARGC [8] is a well-known algorithm that is based on CGC to deal with EVL under NSE. SCARGC performs well on certain datasets in which centroids are moving at a constant speed; however, when the order of the training instances is shuffled, its predictive performance was significantly reduced; this confirms the influence of randomisation on the prediction capabilities of the CGC approach (RQ1). To address (RQ3), PSDSL is made capable of intelligently switching between self-learning and CGC, based on the problem it is applied to, i.e., different characteristics of the data streams. In SCARGC [8] and COMPOSE [21], which is another approach that addresses EVL under NSEs, the values of parameter 'k' are chosen manually to achieve the best results in different datasets (RQ2). MClassification [70] does not require the number of clusters to be known prior to execution as SCARGC, however, it is also computationally expensive. LEVEL_{IW} [11] is highly dependent on the value of the Gaussian kernel bandwidth. While PSDSL is capable of hyperparameter tuning for the best values of parameter 'k' and does not depend on the Gaussian kernel.

This paper provides the following novel contributions:

- Empirically evaluated PSDSL against standalone approaches based on micro-clusters, self-learning and CGC. The predictive accuracies of PSDSL, COMPOSE, LEVEL_{IW}, SCARGC and MClassification are evaluated on benchmarks NSE datasets [8].
- Introduced envelope-clustering (centroid-based clustering for micro-instances) approach to resolve conflicts during the cluster labelling and suggested a confidence measure to ensure the quality and correctness of labels assigned to the clusters.
- Introduced the hyperparameters tuning mechanism in PSDSL to automatically suggest the best value for the number of centroids 'k'.
- The predictive performance of PSDSL and SCARGC were also evaluated after randomising the benchmark non-stationary datasets in which the training instances were shuffled by changing the training orders.

In a comparative analysis of existing EVL learning algorithms which was presented by Umer and Polikar [77]. The average prediction accuracy of SCARGC is highest in all the benchmarks non-stationary datasets [26] when compared with COMPOSE, LEVEL_{IW} and MClassification, therefore only SCARGC was implemented in the MOA [25] to compare it with the MOA data streams approach.

II. RELATED WORK

A recent comprehensive survey and comparative analysis of some of the EVL algorithms is available in the literature [77]. Data stream classification is a process to extract effective knowledge and thereby unlock valuable insights arising from large amounts of real-time data. In semi-supervised data stream mining, a clustering step is followed by a classification which also repeatedly applies in a closed loop fashion. The clustering algorithms make use of unlabelled data to predict the pseudo-labels which are fed to the classifiers to update the prediction models. Pseudo-labelling is a process of using the

labelled data model to predict labels for unlabelled data. The clustering task is to group similar objects.

EVL can be handled using self-learning [45], [46], active learning [47], [48] or CGC [8], [49], [50]. The authors of Dyer et al. [21] applied clustering and ensemble learning to deal with Label Scarcity and drift handling. Graph mining, clustering, and ensemble approaches have been used by Zhang et al. [22] for mining data streams with concept drifts. Trees with a clustering approach were used by Xindong et al. [23] to deal with recurrent drifts. The following section discusses some of the more promising online ensemble classifiers that can be used in EVL under NSEs.

A. ONLINE ENSEMBLES FOR NSE

Ensembles are learning models grouped together in an effort to improve the prediction capabilities of single classifiers. Ensemble methods are one of the most promising research directions [51]. A comprehensive survey on ensemble learning for data streams is available in the literature [20]. The following are some of the online ensembles dealing with concept drifts:

Weighted Majority Algorithm (WMA) [17] combines the different types of base classifiers with an initial weight equal to '1'. The weight is updated on each wrong prediction. The final prediction is made based on the weighted majority vote among the base learners. WMA base learners are heterogeneous, potentially helping to produce more diverse ensembles. However, one of the drawbacks of WMA is that it uses fixed numbers of base learners.

Dynamic Weighted Majority algorithm (DWM) [18] is similar to WMA but it uses a dynamic ensemble size. Despite using the WMA weighting mechanism, DWM does not exploit one of the key aspects of WMA, the use of different types of base models.

Heterogeneous *Dynamic Weighted Majority (HDWM)* [13], is suggested by this study for deployment; as this makes use of different types of ''seed'' learners to maintain the diversity of ensemble and to overcome problems of existing dynamic ensembles that may undergo loss of diversity due to the exclusion of base learners.

Accuracy Updated Ensemble (AUE) [15] combines Hoeffding trees [75] using stacking. Adaptive Classifiers-Ensemble [16] consists of one online and offline classifier to detect concept drift.

Additive Expert Ensembles (AddExp) [52] uses weighted majority vote and adds a new base model at every wrong prediction by the ensemble. Online Accuracy Updated Ensemble [53] combines block-based and online ensemble methods.

B. EVL APPROACHES FOR NSE

Several approaches have been proposed to handle EVL and NSE over the last few years. In particular, the following algorithms were developed to address this problem.

SCARGC [8] applies K-Nearest Neighbour to build the classification models. The algorithm stores instances in batches or in a pool. The initial classification is trained using labelled instances and predicts the pseudo-labels for the unlabelled instances and stores them in the pool. When the pool size reaches 'θ' which is a user-provided value, and the clusters are formed, new centroids receive their labels from previous centroids. The new centroids are used for the prediction of new class labels for the pool data. The algorithm follows a closed loop by switching between clustering and classification.

MClassification [70] uses the concept of micro-clusters [38]. The paper that introduced micro-clusters. The algorithm uses tuple $(N, \overrightarrow{LS}, \overrightarrow{SS}, y)$ to store sufficient statistics from a set of examples. The authors of [70] calculated the centre and radius of micro-clusters using eq. (1) and (2).

$$Centroid = \frac{\overrightarrow{LS}}{N} \tag{1}$$

$$Radius = \sqrt{\frac{\overrightarrow{SS}}{N} + \left(\frac{\overrightarrow{LS}}{N}\right)^2} \tag{2}$$

where:

- $\overrightarrow{LS} = \sum_1^N \overrightarrow{x}_i$ is the linear sum in N data points
- $\overrightarrow{SS} = \sum_1^N (\overrightarrow{x}_i)^2$ is the squared sum in N data points
- N = number of data points
- y = class label for a set of data points

New data points are absorbed in the existing micro-clusters and this results in an increase in the radius and centroids. If the radius is increased from the threshold set by the user, it creates a new micro-cluster, and this process repeats in a loop for each newly received unlabelled data point. For example, a new data point $\overrightarrow{x}$ can be absorbed in $MC_A = \left(N_A, \overrightarrow{LS}_A, \overrightarrow{SS}_A\right)$ updating the summary statics in the following way:

$$\begin{aligned} \overrightarrow{LS}_A &\leftarrow \overrightarrow{LS}_A + \vec{x} \\ \overrightarrow{SS}_A &\leftarrow \overrightarrow{SS}_A + (\vec{x})^2 \\ N_A &\leftarrow N_A + 1 \end{aligned}$$

Similarly when merging two disjoint micro-clusters MC_A and MC_B the union of these two clusters is equal to the sum of its parts and the sufficient statistics is calculated as:

$$\begin{aligned} \overrightarrow{LS}_C &\leftarrow \overrightarrow{LS}_A + \overrightarrow{LS}_B \\ \overrightarrow{SS}_C &\leftarrow \overrightarrow{SS}_A + \overrightarrow{SS}_B \\ N_C &\leftarrow N_A + N_B \end{aligned}$$

Micro-clusters are generated for initial labelled examples, and new unlabelled data instances are accorded their respective labels from the nearest clusters based on the Euclidean distance. In this way as new data points are absorbed and this results in an increase in the radius and centroids. If the radius is increased from the threshold set by the user, it creates a new micro-cluster, and this process repeats in a loop for each newly received unlabelled example.

COMPOSE [21] also addresses the EVL problem. Initially, the labelled instances build a base classifier, either Gaussian mixture model or k-nearest neighbour to obtain a hypothesis and predict class labels. It then constructs the

α-shape (density estimation) using Compaction Percentage and assigns the labels that typically lie in the centre of the feature PSDSL for each class. The Core Support Extraction (CSE) extracts those newly labelled data drawn from the central region of the current distribution.

LEVEL_{IW} [11] relies on a least-squares probabilistic wrapper classifier, which predicts the labels for the unlabelled test data and becomes the labelled training data for the current time step. To predict the labels for the unlabelled test data the algorithm takes four parameters. 1) The training data at the current time step, 2) the corresponding label 3) the unlabelled test data at the current time step, and 4) the kernel bandwidth value σ. The algorithm then follows a closed loop.

C. DATA STREAM CLUSTERING

A series of surveys incorporate the latest developments in the field of Semi-Supervised Learning (SSL) methods which are closely related to label scarcity issues [24]. Several surveys and reviews on stream clustering algorithms are available [27], [42], [43], [63]. Examples of data stream clustering algorithms are incremental *k-means* [56], *E-Stream* [54] and *HUE-Stream* [55], *CluStream* [38], *StreamKM++* [57], *StreamLS* [37], *SWClustering* [58]. Density Based algorithms are intended to group arbitrary-shaped clusters. Examples are *DenStream* [59], *LDBSCAN* [60], *D-Stream* [61], and MR-Stream [62]. However, existing surveys focus on offline learning for static data and make two basic assumptions: 1) the availability of large training datasets; and (2) training and test data are stationary. The *CluStream* algorithm divides the clustering process into on-line and offline components. Online micro-clusters compute and store summary statistics of the data stream. The offline macro-clusters apply K-mean on these micro-clusters.

D. DRIFT DETECTION IN EVL

Several approaches for drift adaptation are available in the literature [14], [15], [16], [18], [33]. Some data stream clustering algorithms adapt to concept drift implicitly as part of the learning process. More specifically, in EVL, when new instances arrive, the clusters are updated to reflect new concepts. The number of clustering algorithms explicitly addressing concept drift is very limited.

To address the non-stationary nature of data, most available algorithms apply window models-with the exception of two *ODAC* (Online Divisive-Agglomerative Clustering) [28] and *FEAC-Stream* (Fast Evolutionary Algorithm for Clustering data stream) [29] which use explicit concept drift adaptation. *ODAC* partitions the streams into different time windows. It constructs an incremental tree-like hierarchy of clusters and continuously monitors the diameters of clusters. FEAC-Stream uses the Page–Hinkley Test [30] to detect concept drifts.

CUSUM (cumulative sum approach) [31] was applied in the work of Namitha and Santhosh [32] for identifying virtual drifts in data stream clustering problems. MC-NN (Micro-Cluster Nearest Neighbour) [64], [74] aims to keep a recent and accurate summary of the data stream, and these micro-clusters are used for feature selection and detecting concept drift.

E. HYPERPARAMETER TUNING APPROACHES

Hyperparameters are parameters that need be initialised and before learning begins, these parameters control the learning process. Several data stream clustering algorithms apply *k-means* [34] due to its simplicity, scalability, and empirical success in many real-world applications [35]. However, one of the pitfalls of k-means is its dependency on the number of centroids 'k' that must be specified prior to the learning. To extend k-means-based Algorithms for evolving data streams with a variable number of 'k', de Andrade Silva and Hruschka [36] describe an algorithmic framework that enables the automatic estimation of 'k' based on the data.

The authors applied three state-of-the-art algorithms for clustering data streams - Stream *LSearch* [37], *CluStream* [38], and *Stream KM++* [39] combined with two well-known algorithms for estimating the number of centroids 'k', namely: Ordered Multiple Runs of *k-means* [40] and *Bisecting k-means* [41].

F. LIMITATIONS OF THE APPROACHES DESCRIBED IN THE LITERATURE

This section highlights the gaps of the literature and explains in what way PSDSL fills these gaps. The most influential parameter for LEVEL_{IW} is the value of the kernel width σ as used in the Gaussian kernel. The algorithm relies on Core Support Extraction (CSE), which is computationally very expensive, especially for high-dimensional data. Furthermore, the process is critically dependent on the parameter 'CP' which defines the Compaction Percentage of current labelled instances to use as core supports, and this means that therefore selecting the best value is problematic. Importantly, PSDSL does not rely on CSE and CP parameters.

While analysing the results published in the respective papers of LEVEL_{IW} and COMPOSE, it is difficult to determine which performs better, it seems to be strongly dependent on the application. COMPOSE showed better results than LEVEL_{IW} when there was a significant class overlap. COMPOSE uses the parameter 'k', the number of centroids, and LEVEL_{IW} uses σ which is the value of the kernel bandwidth. In the case of complete class overlap and a condition when no ground truth data is available, it is extremely difficult for the algorithm to recover from this scenario. PSDSL is made capable of switching the learning strategies based on the problem it is applied on, this strategy is explained in Section III-A. PSDSL also introduced envelope-clustering to recover from class overlaps. This is explained in Section III-C.

The predictive performance of SCARGC is highly dependent on clustering, and it also requires some prior knowledge such as the number of centroids 'k' and pool size 'θ' which may significantly affect the predictive performance when such information is not available. To choose the best value of

'k' which is suitable for a particular data stream, the algorithm needs to run several times with different values of 'k' and pick the 'k' that gives the best predictive accuracy. PSDSL applies an auto parameter tuning mechanism which determines the best value of 'k'.

III. THE PSDSL ALGORITHM

PSDSL is implemented in MOA [25] an open-source framework for data stream mining. The PSDSL is designed to work under EVL and NSEs and performs the following tasks on the initial labelled data.

1) Decide on the best classifier from a pool of Heterogeneous classifiers.
2) Decide on the pseudo-labelling strategy, i.e., Cluster guided or self-learning using classifiers.
3) Build offline micro-clusters and apply them online on-demand only in the case of drift detection.
4) Perform hyperparameter tuning to determine the best value of 'k'.

The unlabelled data stream generates and periodically updates the clustering on real-time data streams. To handle the Virtual drifts that occur due to changes in the distribution of input data i.e., $P_t(x) \neq P_{t+1}(x)$, PSDSL establishes a mapping between current and previous clusters ($C_t \longrightarrow C_{t+1}$) by assigning the current centroid the label which is the same label of the 'k' nearest past centroid. An overview of PSDSL is shown in Fig 2. In step 1, a set of heterogeneous classifiers are trained on a small number of labelled data, and ground truth clusters are formed. This information of ground truth clusters is passed to the hyperparameter tuning (step 2) and switching of pseudo-labelling states (step 3) which are explained in Sections III-D and III-A respectively. In step 4 overlapping of the clusters is determined (explained in Section III-C), if confidence levels of cluster labels fall below a user-provided threshold, envelope-clusters are formed to resolve the conflict in labelling. Finally, in step 5, the pseudo-labels are fed back to update the classifiers for predictions.

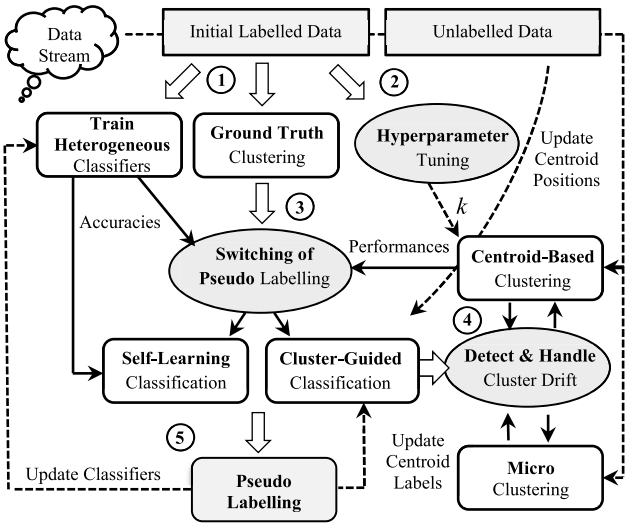

FIGURE 2. Illustrative Pseudo-labelling process of PSDSL algorithm, key steps include training of heterogeneous classifiers and generating clusters by using limited amount of labelled data.

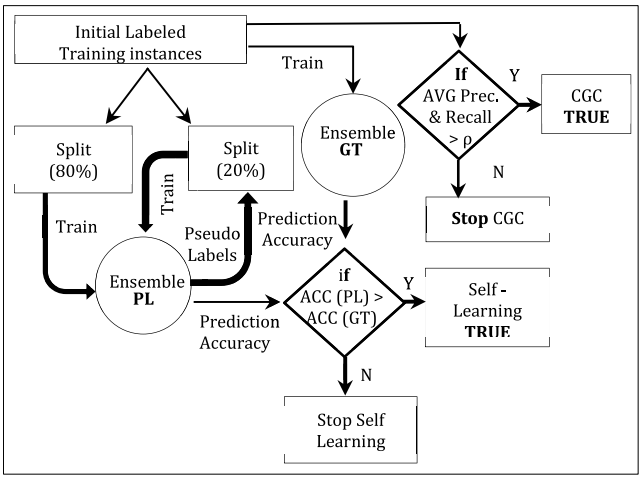

FIGURE 3. Switching of Pseudo-Labelling States between self-learning and CGC based on the prediction accuracies from ground truth and pseudo-labelling ensemble classifiers and comparing it with the average precision and recall of the clusters.

A. SWITCHING OF PSEUDO-LABELING STATES

PSDSL can switch between the three learning states for pseudo-labelling, 1) Cluster guided 2) Self-learning and 3) micro-clustering. The switching mechanism of PSDSL is illustrated in Fig. 3.

In a situation where pseudo-labelling is not improving the predictive performance on initially labelled data, PSDSL switches off the pseudo-labelling state. For this, Ensemble 'GT (Ground Truth)' is trained on the complete set of initial labelled data, while Ensemble 'PL (Pseudo-Labelling)' is trained only on 80% of the training data. Ensemble 'PL' predicts the pseudo-labels for the remaining 20% and trains itself on these pseudo-labels.

If the prediction accuracy of Ensemble 'PL' improves over 'GT', the self-learning state is enabled, otherwise it is suspended. The cluster guided state is enabled when the mean values of *F1-P* and *F1-R* [69] are higher than a user provided threshold 'ρ'. The names *F1-P* and *F1-R* of evaluation metrics are given as the same names mentioned in MOA [25]. The *F1-measure* is the harmonic mean of precision and recall. *F1-P* calculates the total *F1-score* for each found cluster instead of for all ground truth clusters. While the *F1-R* is calculated by maximising *F1* for each ground truth class.

B. CLUSTER GUIDED CLASSIFICATION IN PSDSL

In PSDSL, clusters guide the classification algorithms. Fig. 4 shows Concept $v_1(t)$ and v_2 *(t+1)* as a function of time 't' and 'x' are the input attributes, and 'c' are the classes.

The steps used in this approach are given below.

1) In concept v_1, at time *(t)*, the labelled instances generate $\{C_1 \ldots C_n\}$ clusters representing $\{c_1 \ldots c_n\}$ classes in the initial labelled data.

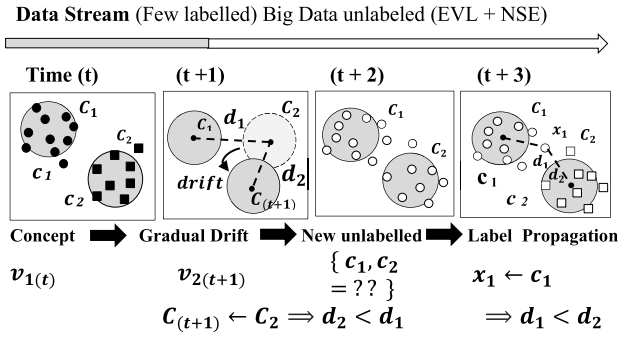

FIGURE 4. Cluster Guided Classification in PSDSL showing representation of different concepts at time 't' due to gradual drifts and the process of label propagation from nearest clusters.

2) When unlabelled data arrives at time *(t+1)* and data distribution changes, the concept v_1 changes to v_2 and the clusters receive labels from the nearest clusters.
3) More new data arrives at time *(t+2)* for which class labels are missing (shown in white circles) and pseudo-labels are required.
4) At time *(t+3),* the unlabelled instances 'x' receive pseudo-labels from the nearest clusters 'C' using the Euclidean distances between 'C' and 'x'.

C. ENVELOPE-CLUSTERING

Micro-clustering state applies on-demand when overlaps between clusters are detected. When clusters overlap, the nearest labelling approach encounters common issues such as losing the correct labels. Envelope-clustering detects and resolves the labels assigned to the clusters. Current micro-clusters receive their labels from the previously labelled clusters and vote for the class labels from 'k' nearest neighbours. This is a scenario where one group of clusters crosses another. As shown in Fig. 5, one group of clusters is stationary i.e., C_2, and C_1 is crossing it. There are two possible outcomes 1) Triangle cluster 'C_1' transfers its label to 'C2' upon intersection with C_2 as the Circle cluster and converts the Circle cluster to Triangle; or, outcome 2) whereby the Triangle cluster 'C_1' gets re-labelled upon intersection with C_2, thus turning the Triangle cluster Circle.

PSDSL generates envelope-clusters by transforming the micro-clusters into micro instances. Envelope-clusters are generated using centroid-based clustering, such as *k-means*. When no cluster overlaps are detected, the concept of envelope-clustering applies online micro-clustering to calculate and store the summary statistics of the data stream; thus, applying it offline to generate macro-clusters when overlaps are detected, increases the processing speed of micro-clustering. Finally, the conflicted clusters receive labels from the corresponding envelope-clusters. Section III-C-1 describes conflict detection and resolution steps in detail.

1) CONFLICT DETECTION

The confidence level for the cluster labelling on the votes is calculated as in (3) below. When the confidence level

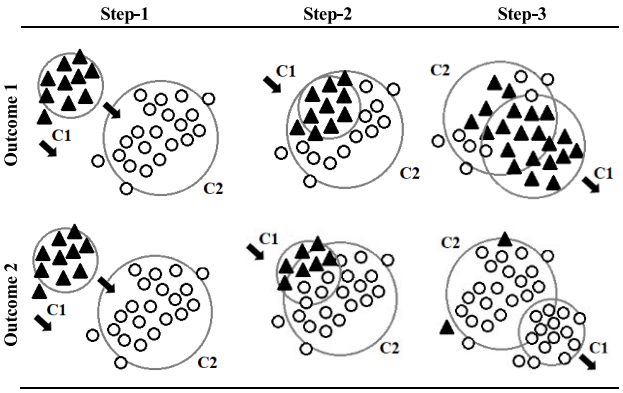

FIGURE 5. Cluster overlapping in 1Csurr dataset [26] showing one class surrounding the other and resulting in two outcomes. 1) C_1 transfers its label to 'C2' or 2) C_1' gets re-labelled upon intersection with C_2.

reaches below a user-provided threshold 'α' it reports the drift; otherwise, it transfers the labels to the corresponding clusters.

$$Confidence\ Level = \frac{\text{Votes(Max}\ (\lambda) - Min\ (\lambda)}{\sum \text{N}} \tag{3}$$

where, λ are the class votes, Min and Max are the minimum and maximum number of votes per class and 'N' are total votes.

2) CONFLICT RESOLUTION

It is necessary to resolve clustering conflicts and label the remaining clusters. The conflict clusters receive the labels from the corresponding nearest envelope-clusters. Fig. 6 shows a plot for the 1Csurr dataset [26] as an example; circle and triangle clusters are successfully labelled from previous clusters (unfilled circle and triangles) with high confidence levels. The figure showing '6' conflicts (diamond) at threshold $\alpha = 0.5$ and 3-nearest neighbours. For $\lambda = [1,2]$ i.e. '1' vote for 'class 0' and '2' votes for class '1', the confidence level is $= (2\text{-}1)/3 = 0.3 < 0.5$ threshold. When there were no conflicts, $\lambda = [3, 0]$ the confidence ratio $= (3\text{-}0)/3 = 1.0 > 0.5$ resulted in a successful label transfer shown in filled circle and triangle clusters.

Fig. 7 shows the resolution of conflict in which the labels to the diamond conflicted clusters are assigned using the nearest envelope-clusters.

D. HYPERPARAMETER TUNING

This step is an essential automated hyper parameter tuning approach used in PSDSL that determines the number of centroids 'k' to be used in clustering using the few initial labelled instances. The cluster evaluation uses extrinsic methods to assign a score to the clustering when the ground truth is available. It applies the mean values of (F1-P), (F1-R) [69], and purity (P) [76] to determine the optimum value of 'k'. The Purity is a measure of the quality of clusters and determines the extent to which clusters contain a single class.

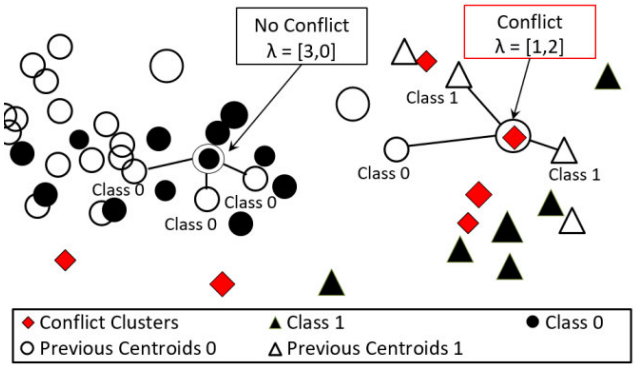

FIGURE 6. Conflict detection in micro-cluster using class votes from 3-nearest neighbours using threshold $\alpha = 0.5$. The diamond shape representing conflicts in cluster labelling due to low confidence value.

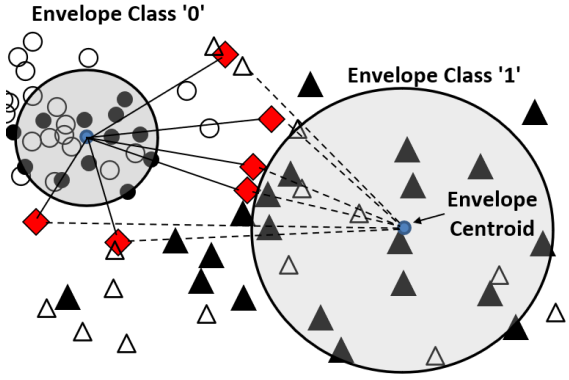

FIGURE 7. Envelope-clustering for conflict resolution. The filled circle and triangle represent recent clusters, and the opaque circle and triangles are previous clusters.

E. PSDSL PSEUDOCODE

The pseudo code for PSDSL is depicted in Algorithm 1. In EVL, initially available labelled examples are of significant use in hyperparameter tuning to determine the optimal values for 'k' (number of centroids). This parameter tuning approach is described in Algorithm 1.1. These labelled examples also play an important role in automatically deciding the best pseudo-labelling approaches, such as self-learning or CGC. The switching mechanism is described in Algorithm 1.2. whereas the concept drift detection and handling using clustering is described in Algorithm 1.3. The symbols and notations used in the algorithms are described in Table 1.

The PSDSL algorithm maintains a set of 'm' base classifiers and clusters. Inputs to the algorithm are 'n' training examples in which τ instances are labelled, followed by complete unlabelled examples. As shown in Algorithm 1, both labelled and unlabelled instances incrementally create the micro-clusters (line 5). When labelled instances arrive (line 6) a clustering algorithm is executed to generate C^t and divide the data into clusters and associates each cluster with one of the classes (line 7) and trains the initial classifier ε.

When unlabelled instances arrive (line 9) it determines the learning state, if the self-learning state is active, it applies a prequential evaluation to predict the pseudo-labels by using the best classifier in the ensemble and re-training the ensemble on these predicted pseudo-labels (line 10-12). When the self-learning state is inactive, it performs CGC (line 14-32). The unlabelled examples are stored in a pool or batch of size θ (line 14) the value of which is set by the user and periodically performs the tasks listed in lines (16 - 30). The pool data is periodically analysed for potential drifts due to cluster overlaps in micro-clusters (line 17). This process returns labelled micro-cluster instances and reports the state of drifts as described in Algorithm 1.3.

If drift is detected, envelope-clusters are formed using micro-cluster instances such that each cluster represents a class in the data (line 19). Envelope-clusters then transfer their labels to the nearest conflicted micro-clusters (lines 20-22). If no drift is detected, the clustering algorithm Φ obtains C_{t+1} on the pool data (line 24) by applying the best values of 'k' obtained in Algorithm 1.1. Each new centroid receives its label from the nearest centroids using the Euclidean distances between C_t and C_{t+1}(line 25). Finally, a set of heterogeneous base classifiers is trained using the pseudo-labelled instances (line 29).

TABLE 1. Symbols and notation used in PSDSL.

Symbol	Description
$\tau = \{x_i ; y_i\}$	Initial set of labelled instances $x \in X$; $y \in Y = \{1, ..., c\}$
τ'	Predicted instances
μ	Micro-clustering algorithm such as C*luStream*
Φ	Clustering algorithm such as K-mean
C^t	Clusters at time 't'
{1, ..., c}	Set of Class Labels
$\mathcal{E}$	Set of heterogeneous base classifiers
$\mathcal{E}_{GT}$	Set of classifiers trained on true class labels
$\mathcal{E}_{PL}$	Classifiers train on pseudo-labels
A	Confidence drift threshold
Λ	Class votes
K	Number of centroids
K_{max}	Max limit of centroids
P	Purity threshold
θ	Pool or batch size
Knn	K-nearest neighbours clusters
Q	Centroids of clusters C {1…k}

1) ALGORITHM: HYPER PARAMETER TUNING

As outlined in Algorithm 1.1, there are three input parameters, a set of labelled instances, a clustering algorithm, and Kmax which is the maximum number of centroids (k) provided by the user. Initially, the ground truth centroids are generated using the labelled instances (line 2) such that {c = k} where 'c' is the number of classes. Lines 3 and 4 generate and evaluate purities μPurit for micro-clusters. Line 5 begins the loop to determine the best value of 'k' by iterating in the range from 'k = 2' to K_{max}. In line 6, new clusters are generated after eliminating the ground truth labels from the labelled data. A user-provided clustering algorithm is applied while passing the incremented values of 'k'. In line 7, the ground truth clustering and current clustering are evaluated,

Algorithm 1: PSDSL(τ, S, ε, Φ, μ, θ, ρ, K_{max})

Input S: $\{x_i; y_i\}$; i = 1, . . . , n: Stream of examples
τ: Initial set of labelled examples
ε: Set of Heterogeneous Base Classifiers {1 . . . m}
μ: Micro-clustering algorithm such as CluStream
Φ: Clustering algorithm like k-mean
θ: Pool Size
ρ: purity threshold, default is 0.85
K_{max}: Maximum number of centroids

1 pool ← ∅
2 K_{best}, Φ_{Purity}, μ_{Purity} ← TuneParameter (τ, Φ, μ, K_{max}) //Alg. (1.1)
3 SwitchingLearningStates (Φ Purity, μPurity) // (Alg. 1.2)
4 **for** i = *1* to n // loop over instances
5 C $_{micro}$ ← Clustering (μ, x_i)
6 **if** (x_i is labelled) **then** // receive initial labelled data
7 C ← GTClustering (Φ, τ) // ground truth cluster
8 ε ← buildClassifier (τ) // build initial classifier
9 **else** // unlabelled instances arrive
10 **if** (self − Learning = true) **then**
11 τ' ← getVoteforinstance (ε_{best}, x_i) // predict labels
12 ε ← TrainClassifier (τ')
13 **else**
14 pool ← pool $\bigcup$ {x_i}
15 **if** (i mod θ = 0) **then** // periodic execution
16 drift ← false;
17 μpoints ← DetectClusterDrift (C, C $_{micro}$) //Alg. (1.3)
18 **if** (drift = TRUE) **then**
19 $C_{Envelope}$ ← Clustering (Φ, C, μpoints, k_{best})
20 LabelCentroids (C, $C_{Envelope}$)
21 $C_{Envelope}$ ← {CEnvelope}$\bigcup$ {C }
22 C $^{t+1}$ ← LabelCentroids ($C_{Envelope}$, C $_{micro}$)
23 **else**
24 C $^{t+1}$ ← Clustering (Φ, C, pool, k_{best})
25 LabelCentroids (C, C $^{t+1}$)
26 C $^{t+1}$ ← {C $^{t+1}$} $\bigcup$ {C} // merge centroids
27 **end if**
28 τ' ← Labeldata (pool, C $^{t+1}$)
29 ε ← TrainClassifier (τ')
30 pool ← ∅
31 **end if**
32 **end if**
33 **end if**
34 **end for**

and the corresponding *F1-P*, *F1-R*, and *P* are stored in sets of '*Fprp*' and 'Purities' lines 8 and 9 respectively.

2) ALGORITHM: SWITCHING LEARNING STATES

Algorithm 1.2 outlines the switching algorithm; it takes μPurity and Φ_{Purity} inputs, and the parameter ρ is the switching threshold set by the user. The ensemble 'ε_{GT}' (Ground Truth) is trained on initial labelled data (line 5), this training set splits in the ratio of 80% and 20% (line 6). Another ensemble ε_{PL} (Pseudo Label) 'trains on 80% of this training examples, then ε_{PL} predicts the pseudo-labels for the remaining 20% and retrains itself on the predicted pseudo-labels (line 7,8). As the ground truth labels of initial training set are known, the predictive performance of both ε_{GT} and ε_{PL} is compared, if the overall prediction accuracy of ε_{PL} becomes higher than the ε_{GT}, the self-learning state becomes active, otherwise the pseudo-labelling is suspended.

3) ALGORITHM: DETECT CLUSTER DRIFT

The algorithm to detect cluster drift is available in Algorithm 1.3, the current micro-clusters C_{t+1} are associated

Algorithm 1.1: PSDSL Auto Tuning Parameter 'k' (τ, Φ, μ, K_{max})

Input: τ: initial set of labelled examples
Φ: Clustering Algorithm
μ: Micro-clustering algorithm
K_{max}: Maximum number of Centroids
Output: K, Φ_{Purity}, μ_{Purity}

1 i ← 0
2 C ← GTClustering (Φ, τ)
3 C μ ← Clustering (μ, τ)
4 μ_{Purity} ← evaluateClustering (C, C μ)
5 **for** k = 2 to K_{max}
6 C $^{t+1}$ ← Clustering (Φ, τ, k)
7 {F1P, F1R, P} ← evaluateClustering (C, C $^{t+1}$)
8 Fprp[i] ← (F1P+F1R + P)/3
9 Purities[i] = P;
10 i++;
11 **end for**
12 k ← argmax (Fprp) + 2
13 Φ_{Purity} ← max (Purities)

with previous clusters C_t by measuring similarity between 'k' nearest centroids q t i; i = {1, . . . , k} using Euclidean distance, i.e., *Dist* (q^t, q^{t+1}) (line 7). The '*k*' nearest clusters vote for the class labels to the current clusters (line 12). To calculate the conflict ratio, min-max values of the votes are applied to the formula (line 15). If the ratio reaches above the user-provided drift threshold, current micro-clusters are assigned the label of the majority class vote.

F. COMPLEXITY OF PSDSL

PSDSL is a single pass algorithm which splits the data stream into batches of predefined size such that each batch contains *n* examples. These batches are sequentially processed, requiring less computational time and space because only the information regarding the centroids and data points of the current batch is stored in the memory. The complexity of PSDSL depends on the choice of learners. PSDSL intelligently switches learning strategies and applies an HDWM classifier for self-learning or *k-means* and *CluStream* for CGC and micro-clustering respectively. Under EVL, when labelled data arrives, PSDSL executes hyperparameter tuning (Algorithm 1.1) and switch learning strategy (Algorithm 1.2) only once.

Hyperparameter tuning (Algorithm 1.1) begins with formation of ground truth clusters which is dominated by the complexity of sorting, which takes $O(nlog\ n)$ time. Next, it iterates on different values of K_{max} to generate clusters, this phase takes ($nkiKmax$) time, where i is the number of iterations. It takes $O(nk)$ space because only the information of distances and centroids are stored in the memory.

State switching (Algorithm 1.2) trains and evaluates HDWM ensemble classifier 'ε', online micro-clusters 'μ' and offline clusters Φ. The overall time complexity for

Algorithm 1.2: SwitchingLearningStates (Φ_{Purity}, μ_{Purity}, ρ)

Input: τ: initial set of labelled examples
$\{\varepsilon_{GT}\}_m^1$: Set of Classifiers train on true class labels
$\{\varepsilon_{PL}\}_m^1$: Classifiers train on pseudo-labels
μ: Micro-clustering algorithm
ρ: purity threshold
Output: none

```
Self-learning ←false
CGC ←false
split_pos ← trainSize ∗ 0.8
for i = 1 to size|τ|
    ε_GT ← train (τ_i)
    if (i < split_pos) then ε_PL ← train (τ_i) else
        τ'_i ← predict (ε_PL, τ_i)
        ε_PL ← train (τ'_i)
    end if
end for
C_micro ← Clustering (μ, τ)
μ ACC% ← evaluate (C_micro, C_GT)
if (μ ACC% > ρ then CGC ←true else self-learning ←true
end if
ifACC (ε_PL) < Acc(ε_GT) then self-learning ← false
end if
```

Algorithm 1.3: DetectClusterDrift (knn, C, C_{micro})

Input: C: past clusters
C_{micro}: Current micro-clusters
α: Drift Threshold
knn: num of nearest neighbours
c: No. of classes
Output: Drift state, labelled micro-clusters

```
Labelledcluster ← ∅
conflict ← false
λ [c]
[q, q^{t+1}] ← getcentroids[C, C_micro]
for (i = 0 to |C|)
    for (j = 0 to |C_micro|)
        distances[ i ][ j ] ← dist (q_i, q_j^{t+1})
    dist ← sort(distances[ i ])
    for ( j = 0 to | dist |)
        if (binarySearch (dist_j, knn) then
            classID ← C_j// 'k' nearest distances
            λ [ classID++] ← 1 // add vote to class
    end for
    maxpair[ ] = minMax(λ);
    ratio ← (maxpair[0] - maxpair[1])/sum(maxpair)
    if (ratio > α) then
        C_micro ← label(C_micro, argmax(λ))
        // Set id of cluster to max class vote.
        C_Labeled ← {C_Labeled} ⋃ {C_micro}
        // Add to labelled clusters
    Else
        conflict ← true
    end if
end for
```

classifiers for 'τ' number of labelled training examples is $O(\tau_\varepsilon)$ and for clustering is $O(\tau_\mu + \tau_\Phi)$. The time complexity of online ensemble classifiers heavily depends on the choice of base classifiers. HDWM applies Naïve Bayes (NB) [78], Hoeffding Tree (HT) [78] and K-Nearest Neighbour (KNN) [80] base classifiers. Based on the worst time complexity of these base classifiers, the total time complexity of HDWM is $O(\tau d)$ $+O(dvc)$. The space complexity for storing the likelihood of each feature with respect to classes is $O(ldvc)$. Where 'd' is dimensionality of the attributes, 'v' values per attribute, 'c' is number of classes and 'l' is the current number of leaves.

When unlabelled examples arrive, additional time and space is required for predicting the pseudo labels for unseen examples. PSDSL selects a best performing base classifier '$\varepsilon_{\text{best}}$' and assigns it for pseudo-labelling in small batches, this phase takes O(n) time for predictions. For clustering the n^{th} batch using i number of iterations it takes $O(nki)$ time and $O(nk)$ space to store the centroid and data points. Micro-clustering takes $O(\ qniN_{Init})$ for the online phase, where q is the number of micro-clusters and N_{Init} is the initial number of examples. For merging two micro-clusters it takes $O(qn)$ and offline phase takes $O(\ qnki)$ time.

The time complexity of drift detection (Algorithm 1.3) depends on the time required to compute the distances from previous centroids to the current centroids. The drifts are detected at a regular interval θ. In the worst case, when all the batches contain drift, the time complexity to compute distances is $O(log\ n)$ using binary search. Once the drifts are detected, transforming the micro-clusters into micro-instances and generation of envelope-clusters requires $O(1)$ and $O(nki)$ time respectively.

Therefore, the total time complexity of PSDSL in the worst scenario is $O(nki)$ $+O(\ qnN_{Init})$ $+$ $O(qn)+$ $O(\ qnki)$ which approximates to O($\ qnki$) $\approx$ $O(N)$ as n is much larger than q, k and i. The value of parameter k is constant which has already been tuned using Algorithm 1.1. PSDSL requires less iterations for convergence because the initial centroids are trained on initial labelled data from the stream and the new centroids obtain their labels from the nearest centroids. COMPOSE on the other hand is of order $O(n^{(d+1)/2})$ i.e., exponential in dimensionality [77]. SCARGC has the worst time complexity, which is $O(nki)$.

IV. EXPERIMENTS AND RESULTS

This section investigates the PSDSL algorithm and compares its performance with SCARGC [8], LEVEL_{IW} [11], COMPOSE [21], and MClassification [70]. To verify statistically significant differences between algorithms, the Friedman test was applied, which is a suitable non-parametric test for multiple algorithms on multiple datasets [66]. The

Friedman test was applied with $\alpha = 0.05$ to test the null hypothesis that there is ''no statistical difference between the algorithms''. The Nemenyi post-hoc test [67] has been applied to identify which pairs of algorithms differ from each other. In EVL few initial ground truth labels are available; therefore, internal evaluations were applied to the clusters i.e., Purity, Precision and Recall [69].

Methods used for evaluating learning models in previous data stream mining studies include prequential, holdout [25] and Kappa statistics [72]. A prequential accuracy estimate is appropriate when all classes are approximately balanced [73]. Kappa statistics is a more sensitive measure for quantifying the predictive performance of streaming classifiers since it cannot be ascertained whether the classes were balanced.

A. EXPERIMENTAL SETUP

The evaluation procedure used is Kappa statistics and prequential testing. Prequential testing is a facility of the MOA [25] in which each instance is used to test the model before it is used for training, and the accuracy is updated incrementally. The PSDSL was compared with existing EVL approaches, static and benchmark setting. To determine how PSDSL performs with and without pseudo-labelling the 'Static' approach was used in which PSDSL does not apply pseudo labelling. Further, to analyse the consequences of unlabelled examples in the data stream and their impact on predictive performance, 95% of the class labels were removed from each dataset and PSDSL was compared in the 'benchmark' setting in which all the training examples are labelled.

All the experiments are evaluated in terms of time consumption and predictive performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM. The experiments performed on non-stationary datasets [26] using MOA-generated streams [25] and real-world datasets. The details of algorithms and parameters used in the experiments for these existing EVL approaches are provided in Table 2.

TABLE 2. Algorithms and parameters used in the experiments.

Algorithm	Description
Static	The PSDSL classifier is not updated after it is trained with the first examples in the data streams. i.e., no pseudo-labelling is applied.
SCARGC	Applied 1-NN base classifier and applying the parameters suggested by Souza et al [8]
Benchmark	PSDSL classifier is applied on 100% labelled examples.
COMPOSE	Gaussian mixture models (GMM) as core supports extraction, applied the parameters suggested by Dyer et al [21]
MClassification	r = 0.1 and |T| = 150 and where |T | represents the size of the initial labelled set and r is the maximum radius of MC
$LEVEL_{IW}$	importance weighted least squares probabilistic base classifier using the default parameters suggested by Umer et al [11]

1) NON-STATIONARY DATASETS

Non-stationary datasets used in the experiments were provided by the authors of SCARGC [8] and are available to the machine learning community [26]. These datasets have been randomised and made available for further research [9]. They provide datasets with incremental changes over time. Here Unimodal Gaussian datasets represent two bi-dimensional Gaussian clusters rotating around a common axis. The distance between the Gaussian components changes over time. Class overlap exists in these datasets. The datasets UG-2C-2D, UG-2C-3D, and MG-2C2D were originally proposed by Dyer et al. [21].

2) MOA DATA STREAMS

The artificial data streams used in the experiments are generated through the MOA workbench [25]; the number of instances is 100,000 and the batch size is 1000 in all the streams. The MOA commands to generate these streams are available in Appendix I.

1) SEA data stream contains three attributes, function $x_i \in R$ and the value of xi is between 1.0 and 10.0. The target concept is determined using the equation $y = [x_0 + x_1 + x_2 \leq \theta]$, such that $\theta \in \{7, 8, 9, 9.5\}$.
2) RandomTree generates a stream based on a randomly generated tree.
3) LED generates a stream defined by a 7-segment LED display and the task is to predict the digit (0-9).
4) Hyperplane is a flat n-dimensional PSDSL useful for simulating gradually drifting concepts. The orientation and position can be modified by slightly changing the relative size of the weights.
5) Random Radial Basis Function (RRBF) consists of a fixed number of randomly positioned centroids with a single standard deviation, class label and weight.
6) Keystroke dataset [8] task is to predict one of four users based on their typing patterns. The dataset contains keystroke records obtained from the users in 8 different sessions who typed a fixed password.

The description of the datasets used in the experiments is provided in Table 3 and Table 4.

The batch size for the MOA Stream is 300. The information about drifts and class overlap is not available for the real-world datasets. Next in Sections IV-B and C, the predictive capabilities of PSDSL were tested on MOA data streams, benchmark non-stationary datasets and real-world datasets.

B. COMPARATIVE ANALYSIS OF PSDSL ON BENCHMARK DATASETS

Predictive accuracies of PSDSL, COMPOSE, $LEVEL_{IW}$, SCARGC and MClassification (MC) were evaluated on benchmark non-stationary datasets [26] that have also been used in the original SCARGC publication. Table 5 shows

TABLE 3. Units for magnetic properties.

Datasets	# of Instances	# of Feature	# of Classes	Drift Interval	Class Overlap
1CDT	16,000	2	2	400	No
1CHT	16,000	2	2	400	No
1CSurr	55,283	2	2	600	yes
2CDT	16,000	2	2	400	yes
2CHT	16,000	2	2	400	yes
4CE1CF	173,000	2	4	750	No
4CR	144,000	2	4	400	No
4CRE-V1	125,000	2	4	1,000	yes
4CRE-V2	183,000	2	4	1,000	yes
5CVT	24,000	2	5	1,000	yes
FG_2C_2D	200,000	2	2	2,000	No
GEARS_2C_2D	200,000	2	2	2,000	No
MG_2C_2D	200,000	2	2	2,000	yes
UG_2C_2D	100,000	2	2	1,000	yes
UG_2C_3D	200,000	3	2	2,000	yes
UG_2C_5D	200,000	5	2	2,000	yes

C=Class, D=Diagonal, V=Vertical, H=Horizontal, T=Transaction, R=Rotating, E=Expanding U= Unimodal, G= Multimodal, G= Gaussian

TABLE 4. Description of MOA streams.

MOA Stream	Instances	Feature	Classes	Drifts
SEA		3	2	2
RandomTree		10	2	2
LED	100K	24	10	1
Wave		21	3	1
Hyperplane		10	2	3
RRBF		2	5	2
Keystroke	1600	8	4	2

the Friedman statistic X_r^2 is 18.93 (df = 5, n = 15). The p-value = .0019 shows significant difference in the algorithms at ($p < .05$). The number in the brackets represents the ranks, the lower the rank and the higher the predictive performance.

Fig. 8 shows a critical difference diagram on ranked accuracies for non-stationary datasets. For 6 algorithms and 16 datasets, the Critical Difference (CD) for the Nemenyi [80] at ($\alpha = 0.05$) is (CD = 1.82). The solid bar shows no significant differences between COMPOSE, $LEVEL_{IW}$, SCARGC, MClassification and PSDSL, however these performed significantly better than 'Static'.

Table 6 presents the Evaluation time in seconds; the results show that PSDSL achieved similar accuracies in less average computation time (8.01 seconds) on non-stationary datasets.

Thus, $LEVEL_{IW}$ is found to be the second lowest performing algorithm in terms of computational complexity after MClassification and performs significantly worse than all other algorithms except SCARGC and PSDSL.

C. ANALYSIS OF MOA DATA STREAMS

Previous set of experiments are performed on offline datasets, a comprehensive analysis was made on MOA data streams. As can be seen from Table 5 the average prediction accuracies

TABLE 5. Average accuracies on benchmark datasets.

Datasets	Static	COMPOSE (GMM)	$LEVEL_{IW}$	SCARGC (1-NN)	MC	PSDSL
1CDT	99.0(6)	99.8(3)	**99.9(1)**	99.7(5)	99.8(2)	99.6(5)
1CHT	94.5(6)	99.3(3)	**99.5(1)**	99.2(4.5)	99.3(2)	99.0(5)
1CSurr	65.6(6)	89.7(4)	91.3(3)	94.3(2)	85.1(5)	**94.5(1)**
2CDT	55.0(6)	**95.9(1)**	58.3(5)	90.9(3)	95.2(2)	90.7(4)
2CHT	54.5(5)	**89.6(1)**	52.1(6)	85.0(4)	87.8(2)	85.8(3)
4CE1CF	94.7(3)	93.9(6)	**97.7(1)**	94.0(5)	94.3(4)	94.7(2)
4CR	25.2(6)	99.9(2.5)	**99.9(1)**	99.9(5)	99.9(2.5)	99.9(4)
4CRE-V2	26.2(5)	**92.3(1)**	24.1(6)	91.9(2.5)	91.5(4)	91.8(3)
5CVT	48.1(4)	45.1(5)	33.1(6)	**90.1(1)**	88.4(2)	84.9(3)
FG_2C_2D	81.3(4)	95.5(2)	**95.7(1)**	95.1(3)	62.4(6)	64.9(5)
GEARS	94.9(5)	95.8(4)	**97.7(1)**	95.9(3)	94.7(6)	95.9(2)
MG_2C_2D	51.6(6)	**93.2(1)**	85.4(3)	92.7(2)	80.5(4)	64.5(5)
UG_2C_2D	45.8(6)	**95.7(1)**	74.3(5)	95.5(2.5)	95.2(4)	95.5(2)
UG_2C_3D	64.1(6)	**95.2(1)**	64.6(5)	94.7(3)	94.7(4)	94.8(2)
UG_2C_5D	69.2(6)	**92.1(1)**	80.1(5)	90.9(4)	91.2(2)	91.0(3)
Keystroke	68.7(6)	87.2(4)	90.5(2)	87.7(3)	**90.6(1)**	85.3(5)
Average	64.93(5.3)	91.2(2.5)	77.8(3.2)	**93.6(3.1)**	90.7(3.3)	89.6(3.3)

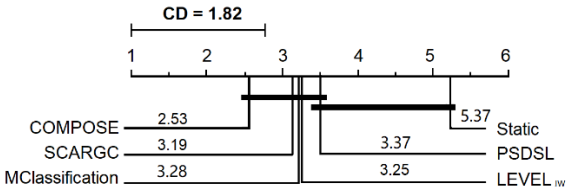

FIGURE 8. Critical Difference diagram for non-stationary dataset accuracies. Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at p = 0.05) are connected.

TABLE 6. Evaluation time in seconds (non-stationary datasets).

Datasets	COMPOSE (GMM)	$LEVEL_{IW}$	SCARGC (1-NN)	MC	PSDSL
1CDT	4.2	15.0	1.5	64.5	**1.2**
1CHT	4.0	15.3	1.2	62.6	**0.4**
1CSurr	7.3	43.8	4.2	220.9	**3.8**
2CDT	2.9	15.7	1.0	62.8	**0.7**
2CHT	3.5	15.8	1.6	60.7	**0.9**
4CE1CF	44.1	137.8	15.1	775.9	**10.7**
4CR	55.9	148.3	12.7	608.0	**7.0**
4CRE-V2	34.8	147.8	15.2	641.6	**10.9**
FG_2C_2D	16.0	185.7	8.2	870.2	**6.2**
GEARS_2C_2D	14.4	186.4	17.5	497.7	**10.0**
MG_2C_2D	15.4	190.8	18.0	740.5	**10.6**
UG_2C_2D	16.9	72.7	9.1	362.8	**6.0**
UG_2C_3D	15.6	176.5	19.4	881.7	**16.0**
UG_2C_5D	15.9	176.8	**21.2**	977.3	27.2
Average	17.9	109.2	10.4	487.5	**8.0**

of SCARGC is highest (93.64%) in all the benchmark datasets therefore we implemented it in the MOA to compare it with our approach. A recent comparative analysis in the literature [77] reports no statistical significance at $\alpha = 0.05$ for classification accuracy among COMPOSE, $LEVEL_{IW}$, MClassification and SCARGC). LEVELIW performs rather poorly on benchmark datasets with significant between-class overlap. MClassification and $LEVEL_{IW}$ are found to be computationally inefficient as shown in Table 6.

TABLE 7. Average accuracies on MOA streams.

MOA Stream	Static	Benchmark	SCARGC	PSDSL
SEA (Sudden Drift)	69.9(3)	77.1(2)	67.9(4)	**78.0(1)**
LED (Sudden Drift)	27.8(3)	**63.6(1)**	22.3(4)	41.7(2)
Wave (Sudden)	75.6(2)	**89.2(1)**	60.4(4)	75.3(3)
RRBF (Gradual Drift)	26.8(4)	**98.3(1)**	97.9(3)	98.0(2)
HP (Incremental)	53.7(3)	**82.8(1)**	50.5(4)	54.5(2)
RandomTree(Recurring)	63.1(3)	63.7(2)	56.0(4)	**71.3(1)**
SEA (No Drift)	77.1(2)	76.9(3)	68.6(4)	**86.5(1)**
LED (No Drift)	45.1(3)	64.1(2)	39.2(4)	**74.0(1)**
Hyperplane (No Drift)	83.1(3)	83.3(2)	61.2(4)	**89.8(1)**
RandomTree (No Drift)	60.3(3)	64.1(2)	54.8(4)	**65.6(1)**
RRBF(No Drift)	13.7(4)	**94.5(1)**	50.5(3)	54.5(2)
Wave (No Drift)	76.4(2)	75.3(3)	54.3(4)	**83.1(1)**
Average (Rank)	56.0(2.9)	77.7(1.7)	57.0(3.8)	**72.7(1.5)**

To analyse SCARGC and PSDSL, Prequential Accuracies, Kappa Statistics and Evaluation time were used and the ranks for each algorithm were calculated. It is noted that SCARGC and PSDSL were compared with the 'Static' and benchmarked approaches which are described in Table 2. The first batch i.e., 300 instances were kept labelled and the class labels of the remaining data stream were removed.

1) PREQUENTIAL ACCURACIES

In EVL these accuracies could not be evaluated due to the scarcity of true class labels; as true labels become available, the accuracy is calculated and presented for comparison. Table 7 presents the average accuracy (in %) achieved by the methods over the 12 MOA streams. The best results were highlighted in a comparison between the proposed method PSDSL and SCARGC, benchmark, and Static.

The overall results show that PSDSL performed better than all other approaches. The Friedman statistic X_F^2 is 24.05 ($df = 3$, $n = 11$), the p-value = .00002 shows a significant difference in the algorithms at ($p < .05$). The number in brackets represents the rank.

To determine which algorithm(s) performed differently, Fig. 9 is the critical difference diagram on ranked accuracies for MOA streams. The connected solid lines represent groups of algorithms that are similar to each other, and any two algorithms are significantly different if the difference between their average ranks is at least CD [66]. For four algorithms and 12 streams, the CD for the Nemenyi [67] at $\alpha = 0.05$ is 1.41. The results show two groups of algorithms, i.e. PSDSL - Benchmark and SCARGC-Statics. Significant differences are found between PSDSL and SCARGC, while the performance of PSDSL is closer to the Benchmark while no significant difference was found between SCARGC and Static.

2) KAPPA STATISTICS

The Kappa evaluation measure is widely used in data stream mining, as it can handle both multi-class and imbalanced class problems. The larger the Kappa value, the more generalised and better the classifier. The kappa statistics show similar results compared with average accuracy, in which PSDSL performs significantly better than other algorithms. Table 8 provides the Kappa statistics for the experiments.

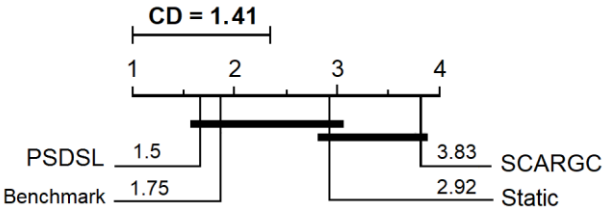

FIGURE 9. Critical Difference diagram for MOA Streams Accuracies, comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at p = 0.05) are connected.

TABLE 8. Average kappa statistics on MOA streams.

MOA Stream	Static	Benchmark	SCARGC	PSDSL
SEA (Sudden Drift)	42.3(3)	52.3(2)	35.6(4)	**55.6(1)**
LED (Sudden Drift)	19.8(3)	**59.5(1)**	13.4(4)	34.6(2)
Wave (Sudden)	33.3(2)	**75.4(1)**	9.3(4)	32.5(3)
RRBF (Gradual Drift)	8.3(4)	**97.9(1)**	97.4(3)	97.5(2)
HP(Incremental)	7.6(3)	**65.6(1)**	0.6(4)	9.0(2)
RandomTree (Recurring)	25.2(3)	25.5(2)	9.8(4)	**41.4(1)**
SEA (No Drift)	52.9(2)	52.8(3)	37.0(4)	**70.0(1)**
LED (No Drift)	39.0(3)	60.1(2)	32.5(4)	**71.1(1)**
Hyperplane (No Drift)	66.3(3)	66.7(2)	22.3(4)	**79.6(1)**
RandomTree (No Drift)	22.7(3)	**60.1(1)**	9.7(4)	29.1(2)
RRBF (No Drift)	1.5(3)	88.9(1)	0.6(4)	9.0(2)
Wave (No Drift)	64.6(2)	63.0(3)	31.6(4)	**74.7(1)**
Average (Rank)	32.0(2.8)	64.0(1.6)	25.0(3.9)	**50.4(1.5)**

3) EVALUATION TIME

Table 9 presents the Evaluation time in Seconds for Static, Benchmark, SCARGC and PSDSL on MOA Streams. The results show that PSDSL achieved better average accuracies (72.7%) in less average computation time (58.38 seconds) than SCARGC Accuracy = 57.0% in 120.11 seconds, but not as far as Benchmark and Static because these do not apply pseudo-labelling.

4) SIGNIFICANT FINDINGS

As the PSDSL does not apply a CGC approach on MOA streams and switches to a self-learning state, this improvement is due to the switching mechanism of heterogeneous base classifiers. Fig. 10 shows the predictive accuracy plots for MOA Streams in which no drift is induced. The results show that PSDSL performed significantly better than SCARGC on all the MOA Streams when there are no concept drifts.

Fig.11 shows the predictive accuracy plots for MOA Streams in which artificial drift is induced. The results show that in EVL, when the CGC fails, restoring from the concept drift is challenging due to unavailability of true class labels. In SEA (Abrupt) and RandomTree (Recurring Drift) streams, all the algorithms restored learning after the sudden drifts. However, the graphs show that, before the first and after the last drifts the PSDSL predictive performance is higher than

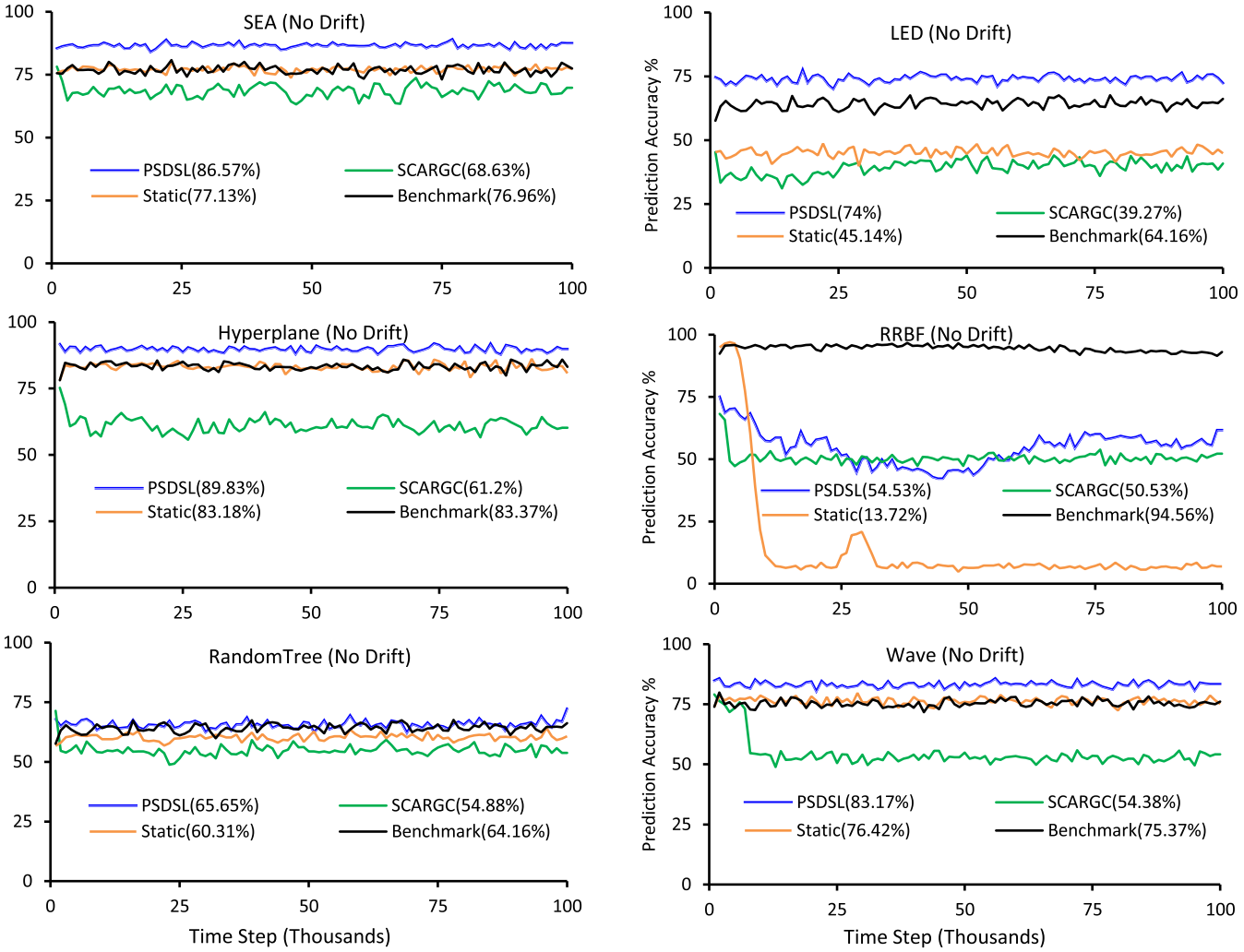

FIGURE 10. Predictive accuracy plots for MOA Streams (No drift), Prediction Accuracy (%) is plotted along the y-axis, instance stream (Time Step) is plotted along the x-axis.

TABLE 9. Evaluation time in seconds (MOA streams).

MOA Stream	Static	Benchmark	SCARGC	PSDSL
SEA (Sudden Drift)	7.5	9.8	120.9	35.89
LED (Sudden Drift)	182.3	55.37	164.9	70.04
Wave (Sudden)	43.81	46.56	109	116.8
RRBF (Gradual Drift)	11.93	10.5	146.2	27.9
HP(Incremental)	29.1	27.18	122.1	50.62
RandomTree (Recurring)	9.56	24.48	136.8	49.45
SEA (No Drift)	6.9	7.3	95.14	36.71
LED (No Drift)	156.75	66.2	164.7	69.20
Hyperplane (No Drift)	20.14	20.54	93.34	49.93
RRBF (No Drift)	4.26	11.87	72.09	3.63
RandomTree (No Drift)	18.89	55.07	100.98	72.17
Wave (No Drift)	37.4	40.76	115.2	118.2
Average	44.05	31.30	120.11	58.38

the competing algorithms. This demonstrates that under EVL conditions PSDSL adapted to the abrupt as well as recurring drifts better than other algorithms. On LED which is a multi-class problem, and Hyperplane which contains incremental drifts, none of the approaches adapted to the drifts in these two streams. Overall, PSDSL performed better than other approaches on drift induced MOA streams.

SCARGC performed best in non-stationary datasets however its predictive performance did not improve when applied to MOA data streams. To further investigate the cause(s) of this failure a Randomisation analysis was made and is presented in Section IV-D

D. ANALYSIS OF RANDOMIZATION

This experiment analyses the sequence of training data and its influence on prediction accuracies for CGC algorithms. In data streams, continuous data arrives at high speed and there is practically no control over the sequence of training data presented to the learning algorithms. Randomisation is thus different to noise, as it is not a random displacement of examples, but a random order in which data instances are presented to the learning algorithm. In this section, RQ1 is addressed- *are existing ILNSE approaches always successful when applied to different problems and why this*

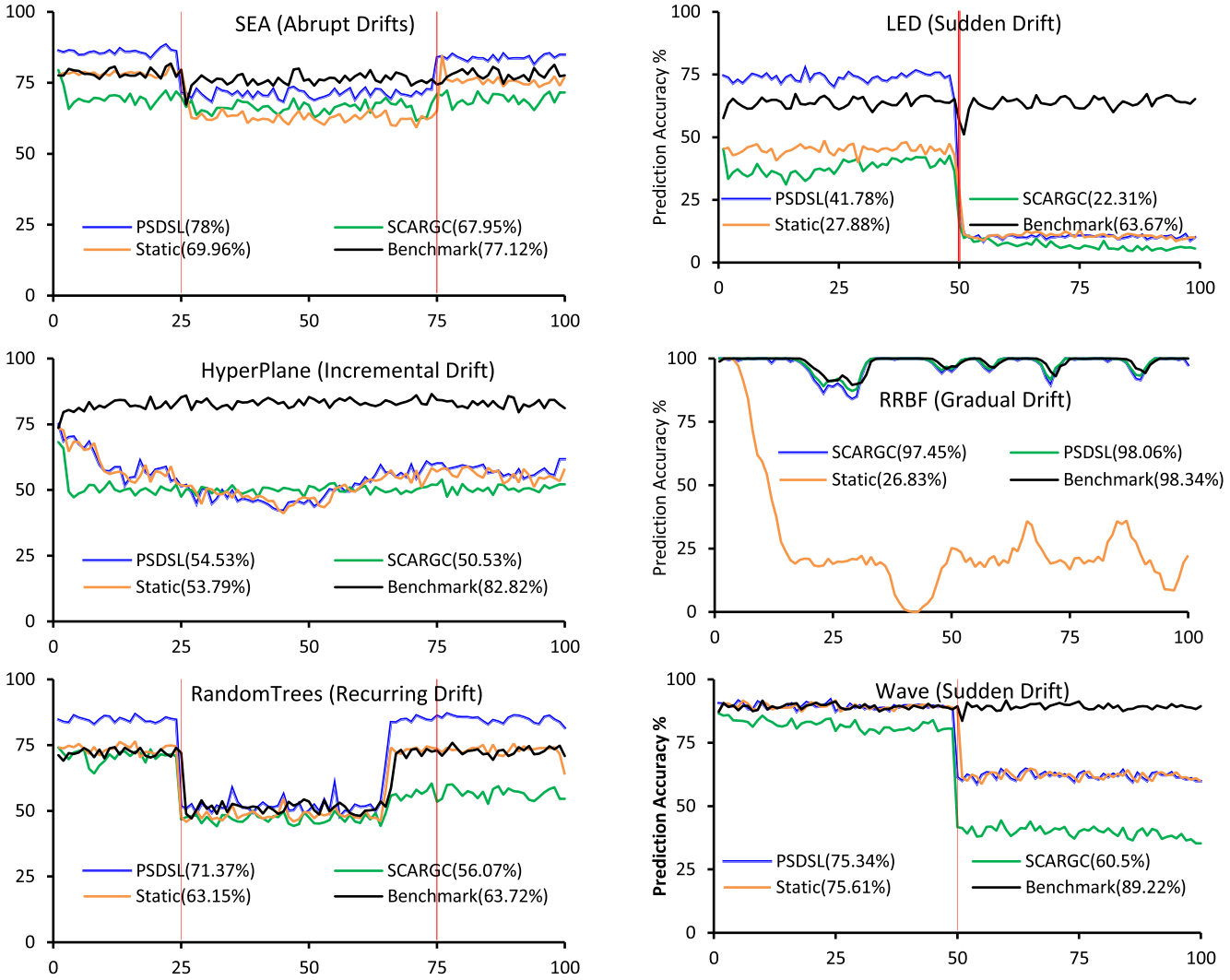

FIGURE 11. Predictive Accuracy Plots for MOA Streams (Artificial drift induced), red vertical lines representing the actual location of abrupt drifts. Prediction Accuracy (%) is plotted along the y-axis, instance stream (Time Step) is plotted along the x-axis.

approach sometimes fails? The benchmark non-stationary datasets [8] [26] are randomised by shuffling the order of examples in the datasets [9]. Fig. 12 shows a plot of a Four Class Rotating (4CR) [26] dataset. The plot '4CR original dataset' on the left shows initial 1000 examples, and on the right '4CR randomised' are initial 1000 instances after shuffling 144k instances in the dataset. The centroids in the dataset are gradually rotating, therefore the examples which are located above the 1000 appeared in the first batch and resulted in a noise effect. The change in the order of examples resulted in the loss of cluster boundaries. This is the scenario in real-time data streams, i.e., no control over the order of examples.

CGC rely on the assumption that the data follows a normal or Gaussian distribution. This supports the clustering process by helping to generate distinct clusters. This assumption also makes CGC a more effective choice in class labels imputation for missing class labels. However, the normalcy (Gaussian) EVL approaches cannot hold for randomised datasets or for real-world data streams, as most such streams are unstructured and contain noise.

The results in Table 10 show the prediction accuracies achieved by the SCARGC and PSDSL algorithm on original and randomised datasets. The results show that SCARGC had a significant drop in average prediction accuracy by 35.3% on randomised datasets, whereas PSDSL only dropped by 20.9%.

E. SWITCHING MECHANISM IN PSDSL

To address (RQ3), *what strategy should be adopted if the CGC or self-learning approaches fail?* PSDSL is made capable of intelligently switching learning states 'CGC with *k-means*, micro-clusters or self-learning. Table 11 shows the switching mode in PSDSL is dependent on {*F1-P, F1-R and purity}* of *k-means* and micro-clusters. Whichever is higher, it adapts the learning mode accordingly. For values lower

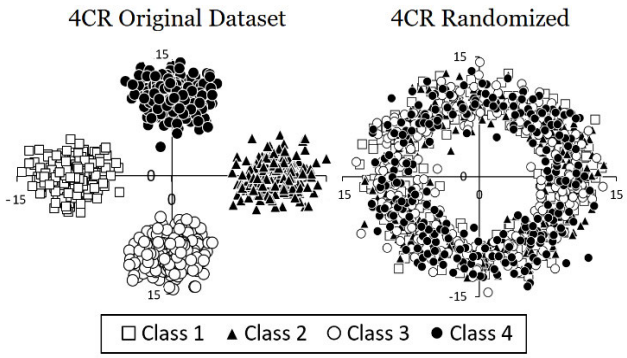

FIGURE 12. Plot for initial 1000 instances of Four Class Rotating (4CR) original dataset (left) [26] versus randomised 4CR dataset (right). The centroids in 4CR dataset are gradually rotating and the instances located after the 1000 appeared before, thus resulted in a noise effect.

TABLE 10. Predictive accuracy (in %) on original and randomised benchmark datasets.

Datasets	SCARGC Original	SCARGC Random	PSDSL Original	PSDSL Random
1CDT	99.8	88.3	99.6	99.2
1CHT	99.3	88.4	99.1	98.2
1CSurr	94.4	59.4	94.5	66.7
2CDT	90.9	59.9	90.7	60.3
2CHT	85	59.7	85.8	60.1
4CE1CF	94.1	92.9	94.7	95.31
4CR	99.9	24.9	99.9	25.1
4CRE-V2	91.9	24.3	91.8	25
5CVT	90.1	38.9	84.9	66.9
FG_2C_2D	95.2	43.2	64.9	74.3
GEARS	95.9	95.4	95.9	95.6
MG_2C_2D	92.7	50.1	64.5	59.6
UG_2C_2D	95.6	53.4	95.6	58.2
UG_2C_3D	94.8	51	94.8	70.5
UG_2C_5D	91	51.1	91	79.8
Average	**94.0**	**58.7**	**89.8**	**68.9**

than threshold 'ρ' such as in randomised datasets or MOA Streams, it switches to self-learning. Further, it monitors the performance of pseudo-labelling. In the case that pseudo-labelling does not improve the predictive performance on initial labelled data, PSDSL suspends the pseudo-labelling.

F. HYPERPARAMETER TUNING

This section presents the analysis carried out to address RQ2: *Does this approach depend on parameters that require manual tuning by the users before inducing the training models?* As shown in Table 12, SCARGC applies k = 4 for (1CSurr) which is a binary class problem; similarly, SCARGC applies k = 4 for (FG_2C_2D) (MG_2C_2D) which contains 5 and 2 classes in the datasets respectively. Furthermore, the real-world dataset 'keystroke' contains 4-classes, but SCARGC applies k = 12 (number of centroids). In SCARGC these values need to be manually chosen by the user to achieve the best results. Contrary to this, PSDSL automatically tuned the best values for the 'k'. As evident in the Table, in most of the datasets, PSDSL predicted values for 'k' which were similar to those in SCARGC. However, the difference is that

TABLE 11. PSDSL purity and switching mode.

Non-stationary Datasets	μ ACC%	Φ Purity	μ Purity	LM
1CDT	100	**0.97** ↑	0.91	C
1CHT	100	0.94	**0.95**↑	M
1CSurr	100	0.97	**0.98**↑	M
2CDT	100	**1.00** ↑	0.98	C
2CHT	100	**0.97** ↑	0.92	C
4CE1CF	100	**0.89** ↑	0.86	C
4CR	100	1.00	1.00	C
4CRE-V2	100	**1.00** ↑	0.97	C
5CVT	100	0.90	**0.96**↑	M
FG_2C_2D	100	**0.92** ↑	0.90	C
GEARS_2C_2D	100	**0.95** ↑	0.92	C
MG_2C_2D	100	1.00	1.00	C
UG_2C_2D	100	1.00	1.00	C
UG_2C_3D	100	1.00	1.00	C
UG_2C_5D	100	1.00	1.00	C
MG_2C_2D	100	1.00	1.00	C
SEA (Sudden Drift)	86.6	0.75	0.76	S
LED (Sudden Drift)	47.1	0.00	0.35	S
Wave (Sudden)	85.3	0.00	0.64	S
RRBF (Gradual Drift)	100	1.00	1.00	C
Hyperplane (Incremental)	76.2	0.67	0.56	S
RandomTree (Recurring)	79.1	**1.00** ↑	0.62	S
SEA (No Drift)	86.3	0.70	0.80	S
LED (No Drift)	47.1	0.00	0.35	S
Hyperplane ((No Drift)	79.2	**1.00** ↑	0.58	S
RandomTree (No Drift)	92.4	**0.99** ↑	0.54	S
Wave (No Drift)	76.7	0.00	0.60	S
KEYSTROKE	99.3	0.34	**0.93**↑	M

μ_{Purity}= CluStream micro-clustering purities, Φ_{Purity}= k-means purities, μACC%= pseudo-labelling Accuracy for μ, LM= Learning modes {C = CGC using k-means, M = micro-clustering, S= Self-learning}

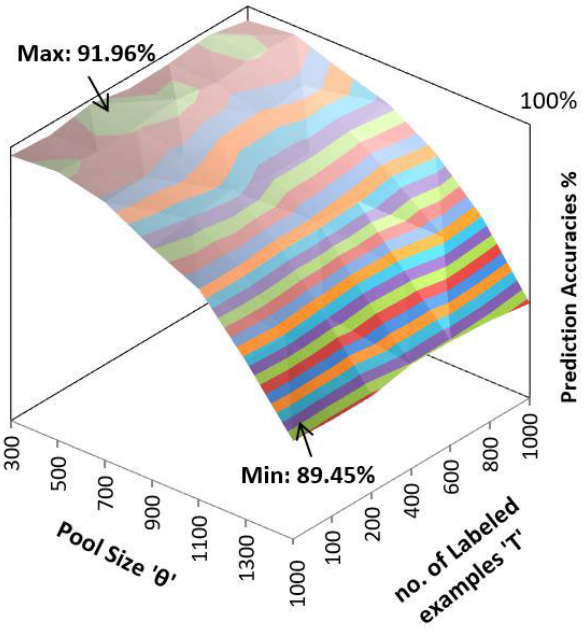

FIGURE 13. Prediction accuracy for 4CRE-V2 dataset changing values of pool size θ and size of initial labelled data 'T'.

the parameter 'k' was sent manually in SCARGC, while PSDSL automatically adapts the parameter 'k' to optimise the classification results over time.

TABLE 12. PSDSL auto-tuned 'k' and learning mode.

Non-stationary Datasets	No. of Classes in datasets	SCARGC Manual 'k'	PSDSL Auto-tune 'k'	SCARGC ACC%	PSDSL ACC%
1CDT	2	= 2	2	**99.75**	99.67
1CHT	2	= 2	2	**99.25**	99.09
1CSurr	2	**≠ 4**	4	94.35	**94.51**
2CDT	2	= 2	2	**90.92**	90.74
2CHT	2	= 2	2	85.02	**85.82**
4CE1CF	4	≠ 5	5	94.08	**94.09**
4CR	4	= 4	4	99.95	**99.97**
4CRE-V2	4	= 4	4	**91.9**	91.88
5CVT	4	≠ 5	5	**90.15**	84.99
FG_2C_2D	5	**≠ 4**	≠ 2	**95.16**	64.94
GEARS_2C_2D	2	= 2	2	95.89	**95.93**
MG_2C_2D	2	**≠ 4**	≠ **2**	**92.71**	64.52
UG_2C_2D	2	= 2	2	95.56	**95.57**
UG_2C_3D	2	= 2	2	94.77	**94.80**
UG_2C_5D	2	= 2	2	90.98	91.05
KEYSTROKE	4	≠ 12	***4**	**87.72**	85.33

G. PARAMETER SENSITIVITY ANALYSIS

The influence of the PSDSL parameters pool size (θ) and number of labelled examples |T| is analysed against the prediction accuracy. Fig. 13 shows the prediction accuracy in % on different values of θ from 300 to 1500 and |T | from 50 to 1000. As it is clear from the plot, increasing the pool size increases the prediction accuracy; however, |T| has no significant effect on the accuracy.

V. DISCUSSION AND CONCLUSIONS

The twin constraints of lack of domain expertise availability and time-resources make the goal of evolving predictive models increasingly more impractical given the relentless volumes of data flowing across the network-centric cyber-physical IoT and semantic media spaces.

This study directly responds to this challenge in proposing a novel algorithm, PSDSL that can responsively switch between Self-Learning, micro-clustering and CGC; whichever approach is beneficial, based on the characteristics of the data stream. Accordingly, a new approach has been introduced, called envelope-clustering, which resolves the conflicts during the cluster labelling process. This approach applies a Confidence Measure to enhance the overall integrity of the labelling by ensuring the quality and correctness of labels assigned to the clusters.

It was concluded that the existing approaches such as SCARGC or COMPOSE perform well for certain datasets in which centroids are moving with a constant velocity. However, when SCARGC was evaluated after shuffling the training instances of the same datasets by changing the training orders, its predictive performance was significantly reduced. The results showed that PSDSL performed significantly better than SCARGC on most real-time data streams including randomised data instances. Thus, the prediction performance of pseudo-labelling has been evaluated by automatically switching between self-labelling and clusters labelling based on the characteristics of the training instances. This study has demonstrated that the PSDSL algorithm performed better than SCARGC for some non-stationary datasets when these were randomised. PSDSL was evaluated on artificially induced MOA streams and real-world data streams and the results showed significantly enhanced performance over SCARGC for most of the MOA streams.

Finally, it was found that, for SCARGC to achieve the best results in different datasets, the values of 'k' needed to be manually chosen, whereas, in contrast, PSDSL achieved similar predictive accuracies without the need for manual selection of the value of the parameter 'k'. Thus, the novel approach proposed in this paper further paves the way for reducing the dependency of machine learning on human input which essentially liberates the process from this hard constraint, as a critical bottleneck, to enable mass-scale deployment of dynamically adaptive labelling of data instances in various emerging data streams.

APPENDIX I

SEA 1 (Sudden Drift): EvaluatePrequential -s (ConceptDriftStream -s (generators.SEAGenerator -f 4) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 2) -p 50000 -w 1) -p 25000 -w 1) -i 100000 -f 1000

SEA 2 (Gradual and Sudden Drift): EvaluatePrequential -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 4) -p 50000 -w 1) -p 25000 -w 10000) -i 100000 -f 1000

HyperPlane (Gradual Drift): EvaluatePrequential -s (generators.HyperplaneGenerator -k 10 -t 0.01) -i 100000 -f 1000

RandomTrees (Recurring Drift): EvaluatePrequential -s (Recurrent ConceptDriftStream -x 10000 -s (generators.RandomTreeGenerator -o 0) -d (generators.RandomTreeGenerator -u 0) -p 25000 -w 1) -i 100000 -f 1000

RandomRBF (Gradual Drift): EvaluatePrequential -s (clustering.RandomRBFGeneratorEvents -n) -i 100000 -f 1000

LED (Sudden Drift): EvaluatePrequential -s (ConceptDriftStream -s generators.LEDGenerator -d (generators.LEDGeneratorDrift -d 7) -p 50000) -i 100000 -f 1000

WaveFormDrift (Sudden Drift): EvaluatePrequential -s (ConceptDriftStream -s generators.WaveformGenerator -d (generators.WaveformGeneratorDrift -d 20) -p 50000 -w 1) -i 100000 -f 1000

REFERENCES

[1] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection and concept-drift adaptation with delayed supervised information," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8, doi: 10.1109/IJCNN.2015.7280527.

[2] L. Watkins, S. Beck, J. Zook, A. Buczak, J. Chavis, W. H. Robinson, J. A. Morales, and S. Mishra, "Using semi-supervised machine learning to address the big data problem in DNS networks," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–6, doi: 10.1109/CCWC.2017.7868376.

[3] A. D. Gabriel, D. T. Gavrilut, B. I. Alexandru, and P. A. Stefan, "Detecting malicious URLs: A semi-supervised machine learning system approach," in *Proc. 18th Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Sep. 2016, pp. 233–239, doi: 10.1109/SYNASC.2016.045.
[4] S. Chernbumroong, A. S. Atkins, and H. Yu, "Activity classification using a single wrist-worn accelerometer," in *Proc. 5th Int. Conf. Softw., Knowl. Inf., Ind. Manage. Appl. (SKIMA)*, Sep. 2011, pp. 1–6, doi: 10.1109/SKIMA.2011.6089975.
[5] G. G. Cabral, L. L. Minku, E. Shihab, and S. Mujahid, "Class imbalance evolution and verification latency in just-in-time software defect prediction," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, May 2019, pp. 666–676, doi: 10.1109/ICSE.2019.00076.
[6] G. R. Marrs, J. Hickey, and M. B. Michaela, "The impact of latency on online classification learning with concept drift," in *Knowledge Science, Engineering and Management* (Lecture Notes in Computer Science), vol. 6291. Berlin, Germany: Springer, Jan. 2010, pp. 459–469, doi: 10.1007/978-3-642-15280-1_42.
[7] R. Razavi-Far, E. Hallaji, M. Saif, and G. Ditzler, "A novelty detector and extreme verification latency model for nonstationary environments," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 561–570, Jan. 2019, doi: 10.1109/TIE.2018.2826477.
[8] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proc. SIAM Int. Conf. Data Mining*, Jun. 2015, pp. 873–881, doi: 10.1137/1.9781611974010.98.
[9] M. M. Idrees. *Non-Stationary Datasets*. Accessed: Sep. 5, 2021. [Online]. Available: https://github.com/mimm1/non-stationary-environments
[10] R. S. Ferreira, G. Zimbrão, and L. G. M. Alvim, "AMANDA: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency," *Inf. Sci.*, vol. 488, pp. 219–237, Jul. 2019, doi: 10.1016/j.ins.2019.03.025.
[11] M. Umer, R. Polikar, and C. Frederickson, "LevelIW: Learning extreme verification latency with importance weighting," in *Proc. IJCNN*, Anchorage, AK, USA, May 2017, pp. 1740–1747, doi: 10.1109/IJCNN.2017.7966061.
[12] J. Gama and G. Castillo, "Learning with local drift detection," in *Advanced Data Mining and Applications*. Berlin, Germany: Springer, Aug. 2006, pp. 42–55, doi: 10.1007/11811305_4.
[13] M. M. Idrees, L. L. Minku, F. Stahl, and A. Badii, "A heterogeneous online learning ensemble for non-stationary environments," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 104983, doi: 10.1016/j.knosys.2019.104983.
[14] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012, doi: 10.1109/TKDE.2011.58.
[15] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer, "Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking," in *Proc. 2nd Asian Conf*, Tokyo, Japan, vol. 13, Nov. 2010, pp. 225–240. Accessed: Jun. 15, 2022.[Online]. Available: https://proceedings.mlr.press/v13/bifet10a.html
[16] K. Nishida and K. Yamauchi, "Adaptive classifiers-ensemble system for tracking concept drift," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2007, pp. 3607–3612, doi: 10.1109/ICMLC.2007.4370772.
[17] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Comput. Eng. Inf. Sci.*, vol. 108, no. 2, pp. 212–261, Feb. 1994, doi: 10.1006/inco.1994.1009.
[18] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Jun. 2003, pp. 123–130, doi: 10.1109/ICDM.2003.1250911.
[19] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Apr. 2014, doi: 10.1145/2523813.
[20] B. Krawczyk, L. L. Minkub, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017, doi: 10.1016/j.inffus.2017.02.004.
[21] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 12–26, Jan. 2014, doi: 10.1109/TNNLS.2013.2277712.
[22] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and cluster ensembles for mining concept drifting data streams," in *Proc. IEEE Int. Conf. Data Mining*, Sydney, NSW, Australia, Dec. 2010, pp. 1175–1180, doi: 10.1109/ICDM.2010.125.
[23] X. Wu, P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing*, vol. 92, pp. 145–155, Sep. 2012, doi: 10.1016/j.neucom.2011.08.041.
[24] Z. Xiaojin. (Sep. 2005). *Semi-Supervised Learning Literature Survey*. University of Wisconsin-Madison Dept. of Comp. Sciences. Accessed: Jun. 13, 2022. [Online]. Available: https://minds.wisconsin.edu/handle/1793/60444
[25] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: Massive online analysis, a framework for stream classification and clustering," *Proc. 1st Workshop Appl. Pattern Anal.*, 2010, pp. 44–50. Accessed: Jun. 13, 2022. [Online]. Available: http://proceedings.mlr.press/v11/bifet10a/bifet10a.pdf
[26] V. M. Souza. *Non-Stationary Datasets Archive*. Accessed: Jun. 13, 2022. [Online]. Available: https://sites.google.com/site/nonstationaryarchive/datasets
[27] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. D. Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–31, Jul. 2013, doi: 10.1145/2522968.2522981.
[28] P. P. Rodrigues, J. Gama, and J. P. Pedroso, "Hierarchical clustering of time-series data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 615–627, May 2008, doi: 10.1109/TKDE.2007.190727.
[29] J. de A. Silva, E. R. Hruschka, and J. Gama, "An evolutionary algorithm for clustering data streams with a variable number of clusters," *Expert Syst. Appl.*, vol. 67, pp. 228–238, Jan. 2017, doi: 10.1016/j.eswa.2016.09.020.
[30] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of page-Hinckley, an approach for fault detection in an agro-alimentary production system," in *Proc. 5th Asian Cont. Conf.*, vol. 2, Jul. 2004, pp. 815–818.
[31] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, 1954, doi: 10.2307/2333009.
[32] N. K. and G. S. Kumar, "CUSUM based concept drift detector for data stream clustering," in *Proc. 4th Int. Conf. Big Data Internet Things*, New York, NY, USA, Aug. 2020, pp. 90–95, doi: 10.1145/3421537.3421548.
[33] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discovery Data Streams, ECML PKDD*, Berlin, Germany, vol. 6, Sep. 2006, pp. 77–86. Accessed: Sep. 5, 2021. [Online]. Available: https://www.cs.upc.edu/ abifet/EDDM.pdf
[34] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist, Prob.*, vol. 1, Jun. 1967, pp. 281–297.
[35] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Dec. 2007, doi: 10.1007/s10115-007-0114-2.
[36] J. de Andrade Silva and E. R. Hruschka, "Extending k-means-based algorithms for evolving data streams with variable number of clusters," in *Proc. 10th Int. Conf. Mach. Learn. Appl. Workshops*, Dec. 2011, pp. 14–19, doi: 10.1109/ICMLA.2011.67.
[37] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. 18th Int. Conf. Data Eng.*, Feb. 2002, pp. 685–694, doi: 10.1109/ICDE.2002.994785.
[38] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proc. VLDB Conf.*, 2003, pp. 81–92, doi: 10.1016/B978-012722442-8/50016-1.
[39] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," *ACM J. Experim. Algorithmics*, vol. 17, pp. 1–30, Jul. 2012, doi: 10.1145/2133803.2184450.
[40] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho, "Efficiency issues of evolutionary k-means," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1938–1952, 2011, doi: 10.1016/j.asoc.2010.06.010.
[41] Y. Yin, C. Wei, G. Zhang, and C. Li, "Implementation of space optimized bisecting K-means (BKM) based on Hadoop," in *Proc. 9th Web Inf. Syst. Appl. Conf.*, Nov. 2012, pp. 170–175, doi: 10.1109/WISA.2012.47.
[42] A. Zubaroğlu and V. Atalay, "Data stream clustering: A review," *Artif. Intell. Rev.*, vol. 54, no. 2, pp. 1201–1236, Feb. 2021, doi: 10.1007/s10462-020-09874-x.
[43] H. L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 535–569, 2015, doi: 10.1007/s10115-014-0808-1.
[44] F. Stahl, M. M. Gaber, P. Aldridge, D. May, H. Liu, M. Bramer, and P. S. Yu, "Homogeneous and heterogeneous distributed classification for pocket data mining," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems V*. Berlin, Germany: Springer, 2012, pp. 183–205, doi: 10.1007/978-3-642-28148-8_8.

[45] Z. Hu, Y. V. Bodyanskiy, O. K. Tyshchenko, and O. O. Boiko, "A neuro-fuzzy Kohonen network for data stream possibilistic clustering and its online self-learning procedure," *Appl. Soft Comput.*, vol. 68, pp. 710–718, Jul. 2018, doi: 10.1016/j.asoc.2017.09.042.
[46] L. Korycki and B. Krawczyk, "Combining active learning and self-labeling for data stream mining," *Proc. 10th Int. Conf. CORES*, vol. 578. Cham, Switzerland: Springer, May 2017, pp. 481–490, doi: 10.1007/978-3-319-59162-9_50.
[47] W. Fan, Y.-A. Huang, and P. S. Yu, "Decision tree evolution using limited number of labeled data items from drifting data streams," in *Proc. 4th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2004, pp. 379–382, doi: 10.1109/ICDM.2004.10026.
[48] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1607–1621, Apr. 2010, doi: 10.1109/TSMCB.2010.2042445.
[49] E. J. Spinosa, A. P. de Leon, F. de Carvalho, and J. Gama, "OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams," in *Proc. ACM SAC*, Seoul, South Korea, Mar. 2007, pp. 448–452, doi: 10.1145/1244002.1244107.
[50] S. U. Din and J. Shao, "Exploiting evolving micro-clusters for data stream classification with emerging class detection," *Inf. Sci.*, vol. 507, pp. 404–420, Jan. 2020, doi: 10.1016/j.ins.2019.08.050.
[51] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Inf. Fusion*, vol. 16, pp. 3–17, May 2014, doi: 10.1016/j.inffus.2013.04.006.
[52] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 449–456, doi: 10.1145/1102351.1102408.
[53] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, May 2014, doi: 10.1016/j.ins.2013.12.011.
[54] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai, "E-stream: Evolution-based technique for stream clustering," in *Advanced Data Mining and Applications*. Berlin, Germany: Springer, 2007, pp. 605–615, doi: 10.1007/978-3-540-73871-8_58.
[55] W. Meesuksabai, T. Kangkachit, and K. Waiyamai, "Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty," in *Advanced Data Mining and Applications*. Berlin, Germany: Springer, 2011, pp. 27–40, doi: 10.1007/978-3-642-25856-5_3.
[56] C. Ordonez, "Clustering binary data streams with K-means," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery (DMKD)*, San Diego, CA, USA, 2003, pp. 12–19, doi: 10.1145/882082.882087.
[57] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," *ACM J. Experim. Algorithmics*, vol. 17, pp. 1–2, Jul. 2012, doi: 10.1145/2133803.2184450.
[58] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding Windows," *Knowl. Inf. Syst.*, vol. 15, no. 2, pp. 181–214, May 2008, doi: 10.1007/s10115-007-0070-x.
[59] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Mining*, Bethesda, MD, USA, Apr. 2006, pp. 328–339, doi: 10.1137/1.9781611972764.29.
[60] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A local-density based spatial clustering algorithm with noise," *Inf. Syst.*, vol. 32, no. 7, pp. 978–986, Nov. 2007, doi: 10.1016/j.is.2006.10.006.
[61] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2007, pp. 133–142, doi: 10.1145/1281192.1281210.
[62] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 3, pp. 1–28, Jul. 2009, doi: 10.1145/1552303.1552307.
[63] Q. Tu, J. F. Lu, B. Yuan, J. B. Tang, and J. Y. Yang, "Density-based hierarchical clustering for streaming data," *Pattern Recognit. Lett.*, vol. 33, no. 5, pp. 641–645, Apr. 2012, doi: 10.1016/j.patrec.2011.11.022.
[64] M. S. Hammoodi, F. Stahl, and A. Badii, "Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining," *Knowl.-Based Syst.*, vol. 161, pp. 205–239, Dec. 2018, doi: 10.1016/j.knosys.2018.08.007.
[65] E. Suzuki et al., "Towards facilitating the development of monitoring systems with low-cost autonomous mobile robots," in *Proc. Int. Workshop ISIP*, vol. 421, Sep. 2013, pp. 57–70, doi: 10.1007/978-3-319-08732-0_5.
[66] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006, doi: 10.5555/1248547.1248548.
[67] P. Nemenyi, "Distribution-free multiple comparisons," M.S. thesis, Princeton Univ., Princeton, NJ, USA, 1963.
[68] I. Žliobaitė, "Controlled permutations for testing adaptive classifiers," in *Proc. Int. Conf. Discovery Sci.* Berlin, Germany: Springer, Oct. 2011, pp. 365–379, doi: 10.1007/978-3-642-24477-3_29.
[69] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, Jul. 2008.
[70] V. M. A. Souza, D. F. Silva, G. E. A. P. A. Batista, and J. Gama, "Classification of evolving data streams with infinitely delayed labels," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 214–219, doi: 10.1109/ICMLA.2015.174.
[71] H. Hachiya, M. Sugiyama, and N. Ueda, "Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition," *Neurocomputing*, vol. 80, pp. 93–101, Mar. 2012, doi: 10.1016/j.neucom.2011.09.016.
[72] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.
[73] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective*.Cambridge, U.K.: Cambridge Univ. Press, Aug. 2011, doi: 10.1017/CBO9780511921803.
[74] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Gener. Comput. Syst.*, vol. 75, pp. 187–199, Oct. 2017, doi: 10.1016/j.future.2017.03.026.
[75] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD*, Aug. 2000, pp. 71–80, doi: 10.1145/347090.347107.
[76] H. V. Reddy, S. V. Raju, and P. Agrawal, "Data labeling method based on cluster purity using relative rough entropy for categorical data clustering," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2013, doi: 10.1109/ICACCI.2013.6637222.
[77] M. Umer and R. Polikar, "Comparative analysis of extreme verification latency learning algorithms," 2020, *arXiv:2011.14917*.
[78] C. Fleizach and S. Fukushima, "A naive Bayes classifier on 1998 KDD cup," Dept. Comput. Sci. Eng., University of California, Los Angeles, CA, USA, Tech. Rep., 1998.
[79] E. Garcia-Martin, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, "Hoeffding trees with nmin adaptation," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Turin, Italy, Oct. 2018, pp. 70–79, doi: 10.1109/dsaa.2018.00017.
[80] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, Jun. 2016, doi: 10.1016/j.neucom.2015.08.112.

MOBIN M. IDREES received the P.G. Diploma degree in computer and information sciences from the University of Karachi, Pakistan, and the M.Sc. degree (Hons.) in advanced software engineering from the University of Leicester. He is currently pursuing the Ph.D. degree with the University of Reading under the supervision of Dr. Frederic Stahl.

He has worked as an Information Technology Consultant at new University City campus for Imam Abdulrahman Bin Faisal University (IABF), Dammam, where he actively involved in research and participated in government funded research. He has published three peer-reviewed journals and contributed for five international conferences while working at IABF. His research interests include, among many others, the development of data mining algorithms, especially the online semi supervised algorithms for big data streams.

Mr. Idrees has been a member of International Association for Engineers (IAENG), since 2015, IAENG is a non-profit international association for engineers and computer scientists. He was awarded with the Best Student of the Year Prize in the year 2017. He has developed several open-source AI programs which are publicly available at GitHub and Source Forge. He has developed a blood donors' program for developing countries which uses online maps. He has reviewed manuscript for Neurocomputing which is a peer-reviewed journal published by Elsevier.

FREDERIC STAHL received the Dipl.-Ing. (FH) degree in bioinformatics from the University of Applied Science, Weihenstephan, Germany, in 2006, and the Ph.D. degree in computer science from the University of Portsmouth, U.K., in 2010.

From 2010 to 2012, he was a Senior Research Associate at the Department of Computer Science, University of Portsmouth. In 2012, he worked as a Lecturer at the Department of Design Engineering and Computing, Bournemouth University, U.K. From 2012 to 2019, he was a Lecturer and an Associate Professor at the University of Reading, U.K. Since 2019, he has been the Deputy Head, a Team Leader, and a Senior Researcher for Subject Area Marine Perception at the German Research Centre for Artificial Intelligence (DFKI GmbH). He has published over 60 articles in peer-reviewed conferences, journals, and book chapters. He has been working in the field of data mining for more than ten years focusing on the research domain of big data analytics. His particular research interests are lie in (i) developing scalable algorithms for building adaptive models for real-time streaming data; (ii) developing scalable parallel data mining algorithms and workflows; and (iii) applications in big data analytics.

Dr. Stahl is a member of the British Computer Society (BCS) and has been elected three times as a Committee Member of the BCS's Specialist Group on Artificial Intelligence (SGAI), servicing on the committee, since 2013.

ATTA BADII is currently a Research Professor of computer science with a focus in the fields of AI and data science. He has established a track record of key contributions to over 50 projects, including over 30 large-scale collaborative research programs many of which he has initiated and led as the technical coordinator. He has pioneered several paradigms in user-centered assistive-ambient technologies particularly personalized context-aware security-privacy protective design; and hardware-accelerated ML for real-time AI-assisted application domains such as an Anomaly Detection in online time series data streams as applied to safety alerting, cyber-attack detection and financial transactions fraud detection.

He has served on various editorial and research steering boards as a Coordinator/Technical Leader/Invited Expert, for example, as the Chair of the Security Architectures and Virtualisation Taskforce of the European Road Map Project SECURIST, the Chair of the VideoSense: European Video-Analytics Network of Excellence and Expert Advisor on the large-scale EPSRC-funded Research Program on Big Data and Human Rights at the University of Essex.