SLAYER: A Semi-supervised Learning Approach for Drifting Data Streams under Extreme Verification Latency

Maria Arostegi[1], Jesus L. Lobo[1], and Javier Del Ser[1,2]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
[2] University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
[maria.arostegi,jesus.lopez,javier.delser]@tecnalia.com

Abstract. Classification models learned from data streams often assume the availability of true labels after predicting new examples, either instantly or with some delay with respect to inference time. However, in many real-world scenarios comprising sensors, actuators and robotic swarms, this assumption may not realistically hold, since the supervision of newly classified samples can be unfeasible to achieve in practice. The extreme case where such a supervision is never available is referred to as extreme verification latency. Furthermore, streaming data is also known to undergo the effects of exogenous non-stationary phenomena, by which patterns to be learned from the streams can evolve over time, thereby requiring the adaptation of the classifier for its knowledge to match to the prevailing concept. When these two circumstances (extreme verification latency and concept drift) concur in a given scenario, adapting the model to the evolving dynamics of stream data becomes a challenging task, as the lack of supervision requires rethinking this functionality from a semi-supervised perspective. In this context we present SLAYER, a semi-supervised learning approach capable of tracking the evolution of concepts in the feature space, and analyzing their characteristics towards alleviating the effects of concept drift in the classification accuracy. Besides its continuous adaptation to evolving concepts, another advantage of SLAYER is its resilience against the appearance and disappearance of concepts over time, adapting its knowledge seamlessly when it occurs. We assess the performance of SLAYER over several datasets and compare it to that of state-of-the-art approaches proposed to deal with this stream learning setup. The discussion on the obtained results is conclusive: SLAYER offers a competitive behavior, performing best over several of the datasets considered in the benchmark.

Keywords: Stream learning · extreme verification latency · semi-supervised learning · concept drift

1 Introduction

Nowadays increasing volumes of data are generated at unprecedented speeds, pushing the derivation of new learning models suitable for data analysis under

stringent computational constraints. In order to gain value from these data flows, efficient analytical models are needed, which are at the core of research efforts around the Big Data paradigm [13]. Among the technological Big Data landscape, real-time Big Data analytics aim to extract useful information from large datasets, often produced in the form of data streams, namely, sequences of data items that arrive fast and continuously over time. Models that learn from such streaming data while complying with the restrictions imposed by their flowing nature have given rise to a profitable research area widely referred to as data stream learning. Examples of real-world stream learning applications abound in a manifold of domains, including recommendation systems, energy demand modeling, climate data analysis, malware/spam detection, industrial prognosis or traffic data analysis, among many others. As a matter of fact, the upsurge of scenarios resorting to Internet of Things (IoT) devices and functionalities has significantly propelled the necessity of new advances in stream learning, since applications where IoT sensors are deployed often produce huge amounts of data continuously over time [25].

In this context, devices that capture and process such data streams are usually limited in terms of memory and computing power (e.g. sensors, tiny devices), which causes that in most practical cases, the processing time is the main limiting issue to tackle when designing models for stream learning. In response to these restrictions, preprocessing and learning methods have been proposed in the literature, as for instance, selective sampling strategies, divide and conquer strategies and distributed computing [22]. Even if the relative maturity of those developed techniques can satisfy the computational constraints to a certain extent, stream learning models must also be endowed with the capability to adapt their captured knowledge to eventual changes in their received stream data. This phenomenon, known as *concept drift*, is usually due to non-stationary environmental conditions that exogenously affect the production of such data flows, which yields that the characteristics of the data streams evolve over time and impact on the prevalence of the knowledge embedded in the model [11, 12].

The community has hitherto been particularly active in the derivation of new approaches suited to deal with different types of drift as per their speed (abrupt/gradual), severity level or casuistry (feature/label drift). Actually, significant efforts in this vein have been devoted towards the study of concept drift together with known traditional problems in machine learning, such as class imbalance or multi-label classification. However, a scenario that has been less addressed to date is that where concept drift collides with a indefinite lack of supervision of the incoming data. In many practical scenarios annotated stream data can be costly to obtain, or even unfeasible by any means [20]. Therefore, the immediate availability of the supervision associated to stream data (e.g. true labels in classification tasks) cannot be assumed any longer or, at best, supposed to be available after some application-dependent time delay. This circumstance, known in the field as *verification latency*, may hold also in drifting data streams, hence imposing not only efficiency of the learning algorithm, but also a continuous adaptation of the model to varying concepts without any supervision of

the input data whatsoever. This need for adaptation is of special relevance in the so-called *extreme* verification latency, in which the supervision of the stream data disappears at a time and is never available again for modeling purposes [18]. The real-world examples provided in the first part of this section also serve as an example where extreme verification latency holds. For instance, IoT sensors that capture temperature or humidity data are often subject to decalibration in the physical and chemical properties of their transducers. In many practical setups companies cannot afford to recalibrate such sensors every time decalibration occurs, nor can they cope with the investment needed to newly annotate data in the new operating regime of the decalibrated sensor. Therefore, they assume that extreme verification latency is an inherent circumstance to be faced by solutions designed for the considered task. Many contributions reported to date around data stream classification over drifting data focus on scenarios characterized by a lagged label supervision with respect to prediction time [17].

This work addresses data stream learning under the above two premises (concept drift and extreme verification latency), focusing on slowly evolving drifts that can be traced and characterized in a non-supervised fashion. To deal with concept drift adaptation in these challenging conditions, we present SLAYER (*Semi-supervised stream LeArning with densitY-basEd dRift adaptation*), a learning algorithm for classification tasks formulated over data streams that can incrementally characterize the evolution over time of the class-dependent modes of streaming data. This characterization relies on an continuous non-supervised analysis of incoming data in order to anticipate changes in their structural characteristics. In other words, the cornerstone of SLAYER is that the analysis is driven by the number of clusters found in data at every time, instead of the number of classes of the formulated classification problem. At this point, we emphasize that, unlike online clustering, which aims to have a good characterization of clusters over time but the correspondence between labels and clusters is not taken into account, in our setup it is crucial to trace the correspondence between labels and concepts over time. Therefore, the goal is to predict examples to classes rather than to infer how data organize in clusters over time [21].

We assess the performance of the proposed approach over a set of public synthetic datasets featuring evolving drifts of very diverse nature, assuming extreme verification latency in all of them. Results of our proposed scheme are compared to the ones obtained by other methods reported in the scarce literature that has so far undertaken this same problem. Results elucidate that the unsupervised drift tracking mechanisms embodied in SLAYER can lead to superior accuracies than its counterparts, as they allow for a fine-grained modeling of the information continuously flowing and drifting over time.

The rest of the paper is structured as follows: first a brief review of related contributions is made in Section 2. Next, Section 3 describes the overall algorithmic steps of SLAYER, jointly with a description and design rationale of methods underneath. The experimental setup is detailed in Section 4, whereas results are presented and discussed in Section 5. Finally, Section 6 concludes this work and outlines future research directions stimulated by our findings.

2 Background

Among the extensive bibliography corpus related to stream learning, we first pause at the conclusions drawn in [15] and [16]. The importance of dealing with verification latency when learning from streams was already pointed out in the former, whereas the latter reviewed more than 130 works about concept drift, analyzing assumptions, methodologies and techniques, and concluding with prospects and guidelines for future research in the field. Among them, extreme verification latency was identified as an area deserving further research. The delayed or null supervision of incoming data samples implies the use of different strategies that blindly monitor the distribution of arriving data samples and adapt the model to changes detected therein [17]. Within such strategies to handle verification latency, we highlight 1) semi-supervised learning [12], in which a few labeled samples are available for initially training a model, which allows subsequently extracting further knowledge from the large amount of unsupervised streaming data; and 2) active learning [19], by which the learning method itself chooses the instances to be learned. Next, we review the main algorithms contributed so far for data stream classification in non-stationary environments subject to extreme verification latency.

To begin with, the Arbitrary Sub-Populations Tracker (APT) approach proposed in [14] is characterized by a two-stage learning strategy. Expectation-maximization is used to determine the optimal one-to-one assignment between the unlabeled and labeled data (by using kernel density estimation techniques), and next the classifier is updated to reflect the population parameters of newly received data, as well as the drift characteristics. Shortly thereafter, the renowned Compacted Object Sample Extraction (COMPOSE) semi-supervised approach was reported in [5], which is essentially a geometry-based framework capable of learning from non-stationary streaming data by following three steps: 1) the extraction of so-called α-shapes to represent the current class conditional distribution; 2) the shrinkage of such α-shapes to properly model the geometric center of each class distribution; and 3) from the compacted α-shapes new instances are extracted to serve as labeled data for future time steps.

Ever since COMPOSE has originated multiple variants, such as FAST COMPOSE, MASS, or LEVELIW, which improve the original version of the algorithm in different aspects (e.g. speed). Before commenting on them, we follow our review by mentioning the Classification Algorithm Guided by Clustering (SCARGC) in [20], which consists of clustering followed by classification applied repeatedly in a closed-loop fashion. The algorithm exploits the current and past cluster positions extracted from unlabeled data to track drifts over time. Later in time, the first improvement of COMPOSE, denoted as FAST COMPOSE, was published in [23], which reduces the computational fingerprint of COMPOSE by alleviating the complexity of their shape extraction step. At the time, another modified version of COMPOSE – Modular Adaptive Sensor System (MASS) [8] – was proposed as a workaround to extreme verification latency in stream data, yet was still found to be computationally unaffordable for resource constrained applications (e.g. IoT sensor networks). Likewise, LEVELIW [23, 24] was pro-

posed as a framework for learning in extreme verification latency scenarios by using importance weighting in gradual concept drift scenarios. LEVELIW leverages weighting to match distributions between two consecutive time steps, and estimates the posterior distribution of the unlabeled data using a weighted least-squares probabilistic classifier. The work in [1] proposed TRACE, a technique that predicts the trajectory of concepts in the feature space by means of Kalman filtering, which was shown to adapt to drifting environments without any external supervision. Finally, [7], a semi-supervised density-based adaptive model for non-stationary data (AMANDA) has been recently proposed in [7]. AMANDA weights and filters samples that best represent the concepts in the distribution. To this end, it identifies which instances lie in the core region of the existing class distributions, so that these selected instances are chosen as training data for the next iteration. The weighting method receives a set of instances as input and returns the same set of instances associated to weights.

In conclusion, the recent activity in the field reviewed above suggests that this is a topic eager for new algorithmic approaches. This is the main motivation for the development of SLAYER, which incorporates a novel perspective to predict the evolution of drifts over stream data in an unassisted manner.

3 SLAYER: Description and Design Rationale

Learning in non-stationary streaming environments can be approached from two different perspectives: active or passive [6]. The difference among them resides on the adaptation mechanism: the active procedure depends on the use of a drift detector that processes arriving data and triggers an alarm when a change is detected. By contrast, a passive method continuously modifies the model over time without explicitly detecting the drift in order to prepare the model for any concept drift eventually present in the stream data. Our scheme builds upon a passive approach: even if a drift is not detected anyhow, SLAYER constantly updates its knowledge in order to yield a prediction conforming to the prevalent status of the stream. For this to occur, a fundamental premise of SLAYER is that drifts over the streams are gradual, so that changes can be monitored and tracked over non-supervised data. This is important to be noted, as conventional drift detectors and passive adaptation strategies operate by assuming immediate access to the annotation of the received data instances, so that changes can be inferred by quantifying the performance degradation of the model over time.

Figure 1 and Algorithm 1 summarize the main steps of SLAYER. As can be read in the algorithm, we departing from the arrival of the initial set of labeled instances $\{\mathbf{x}_t\}_{t=1}^{T_0}$, from which SLAYER infers the clusters in which these initial samples can be grouped (line 2). Without loss of generality, this initial step can be done by very assorted means. For the sake of simplicity and given the limited computational effort imposed by streaming setups, this is done by using K-means together with the well-known elbow method to compute the optimal number of clusters K_0: the total within-cluster variation is calculated for increasing values of K_0 as the sum of squared distances Euclidean distances between items and

the corresponding centroid, so that the optimal value of K_0 is declared when the addition of a new cluster does not imply an improvement (decrease) of the variation measure.

Once clusters $\{\mathcal{X}_0^k\}_{k=1}^{K_0}$ have been computed over the initial set of samples $\{\mathbf{x}_t\}_{t=0}^{T_0}$, two structural characteristics are quantified for every cluster $\mathcal{X}_0^k$: the amount Q_0^k of examples assigned to the cluster, i.e. $Q_0^k = |\{\mathbf{x}_t \; \forall t \leq T_0 : \texttt{cluster}(\mathbf{x}_t) = k\}|$, and its *radius* R_0^k (namely, the maximum distance from every example to the centroid of its assigned cluster). From these characteristics and assuming a globular shape of the cluster, a rough estimation of the cluster density is given by $D_0^k = Q_0^k/(R_0^k)^D$, where D is the dimension of the feature space (line 2). After computing this measure over the cluster space of the initial batch of labeled examples, a 1-nearest neighbor model can be used over such samples to predict the samples of the next batch that follows immediately thereafter (Lines 3 and 4).

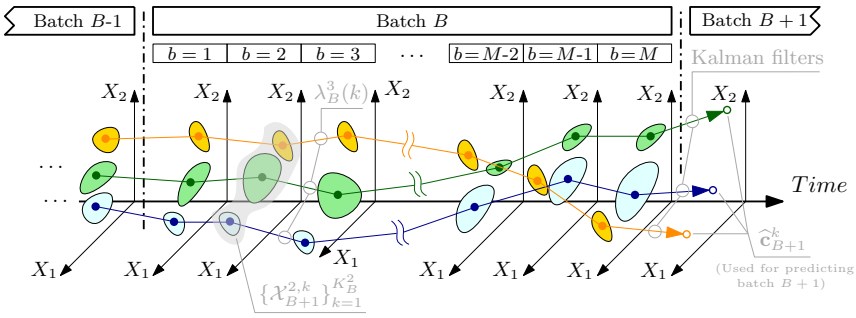


Fig. 1: Schematic diagram illustrating the internal processing of unsupervised batches featured by SLAYER, including the successive clustering and mapping procedure performed at every mini-batch $b \in \{1, \ldots, M\}$.

When the next batch arrives and once samples therein have been predicted (line 6), another clustering phase is performed over such samples (line 7) so that the new cluster space $\{\mathcal{X}_1^k\}_{k=1}^{K_1}$ can reflect the emergence of new clusters or the disappearance of others. A matching between the clusters from the previous batch and those arising for the current one is done (line 8), yielding a mapping function $\Omega_1 : \{1, \ldots, K_1\} \mapsto \{1, \ldots, K_0\}$ given by:

$$\Omega_1(k') = \underset{k \in \{1,\ldots,K\}}{\arg\min} \; \|\mathbf{c}_1^{k'} - \mathbf{c}_0^k\|_F, \tag{1}$$

where $\|\cdot\|_F$ stands for Frobenius (Euclidean) norm, and $\mathbf{c}_1^k$ denotes the centroid of the k-th cluster. Despite its simplicity, this simple mapping rule permits to trace how the cluster space evolves between subsequently received batches. Departing from this matching, densities $\{D_1^k\}_{k=1}^{K_1}$ of the K_1 clusters are computed and compared to those of the previous clusters to which they are mapped. SLAYER implements this comparison as $|D_1^{k'} - D_0^{\Omega(k')}|$, which, together with an increase or decrease of the number of clusters (namely, when $K_1 \neq K_0$), is an

Algorithm 1: SLAYER

Input : Initially annotated data instances $\{\mathbf{x}_t, y_t\}_{t=1}^{T_0}$, threshold ϵ for declaring a density-driven change, number of mini-batches M, unsupervised stream instances $\{\mathbf{x}_t\}_{t>T_0}$.
Output: Predictions $\{\widehat{y}_t\}_{t=1}^{\infty}$.

1 Let L denote the number of classes
2 Compute clusters $\{\mathcal{X}_0^k\}_{k=1}^{K_0}$ and densities $\{D_0^k\}_{k=1}^{K_0}$ for $\{\mathbf{x}_t\}_{t=1}^{T_0}$
3 Set $\widehat{\mathbf{c}}_1^k$ equal to the average of all $\mathbf{x}_t$ such that $\mathbf{x}_t \in \mathcal{X}_0^k$
4 Set $\ell(k)$ to the majority class of annotated instances assigned to cluster k
5 **foreach** *incoming batch* $B \in [1, \ldots, \infty]$ **do**
6 Predict samples $y_t \in B$ as the label $\ell(k)$ of the closest $\{\widehat{\mathbf{c}}_B^k\}_{k=1}^{K_{B-1}}$
7 Compute clusters $\{\mathcal{X}_B^k\}_{k=1}^{K_B}$ and densities $\{D_B^k\}_{k=1}^{K_B}$ for $\{\mathbf{x}_t\}_{t\in B}$
8 Compute mapping $\Omega_B()$ between $\{\mathcal{X}_B^k\}_{k=1}^{K_B}$ and $\{\mathcal{X}_{B-1}^k\}_{k=1}^{K_{B-1}}$
9 Divide batch B in M mini-batches
10 **foreach** *minibatch* b **do**
11 Compute clusters $\{\mathcal{X}_B^{b,k}\}_{k=1}^{K_B^b}$ and centroids $\{\mathbf{c}_B^{b,k}\}_{k=1}^{K_B^b}$
12 Infer mapping $\lambda_B^b : \{1, \ldots, K_B^b\} \mapsto \{1, \ldots, K_B^{b-1}\}$
13 Correct each $\mathbf{c}_B^{b,k}$ based on $\lambda_B^b(\cdot)$ as per Expr. (2)
14 **if** $\exists k$ *such that* $|D_B^k - D_B^{\Omega_B(k)}| > \epsilon$ **then**
15 Set $\alpha_\ell = 0\ \forall \ell \in \{1, \ldots, L\}$
16 **else**
17 Adjust α_ℓ for each $\ell \in \{1, \ldots, L\}$ as in Expression (3)
18 **end**
19 Predict $\widehat{\mathbf{c}}_{B+1}^k$ by means of a Kalman filter, using as prior estimation $\widehat{\mathbf{c}}_B^{k^*}$, where $k^* = \lambda_B^1(\lambda_B^2(\ldots(\lambda_B^{M-1}(\lambda_B^M(k)))\ldots)$
20 **end**
21 **end**

indicator of a potential change of the cluster space over time. For this purpose, a change is declared when there exists at least a cluster $k' \in \{1, \ldots, K_1\}$ for which $|D_1^{k'} - D_0^{\Omega_1(k')}| > \epsilon$, where ϵ is an hyper-parameter of SLAYER that tunes the sensitivity of the model to the speed and intensity at which clusters vary.

When a change is identified, a forgetting mechanism must be triggered to discard previous knowledge about the stream. In the unsupervised regime, SLAYER predicts arriving samples by assigning them the label associated to the prevailing cluster space whose centroid is closest to each sample. To this end, SLAYER attains a finer level of granularity by dividing each batch into M *mini-batches* of equal size (line 9). Samples falling inside each mini-batch are clustered by means of a memory-based K-means algorithm wherein, once K_1^b clusters have been extracted from mini-batch b (line 11), a distance-based matching $\lambda_1^b : \{1, \ldots, K_1^b\} \mapsto \{1, \ldots, K_1^{b-1}\}$ is computed as in (1) (line 12), so that centroids can be corrected as (line 13):

$$\mathbf{c}_1^{b,k} = \frac{\alpha_{\ell(k)} \cdot \mathbf{c}_1^{b-1,\lambda_1^b(k)} \cdot N_1^{b-1,\lambda_1^b(k)} + (1 - \alpha_{\ell(k)}) \cdot \mathbf{c}_1^{b,k} \cdot N_1^{b,k}}{\alpha_{\ell(k)} \cdot N_1^{b-1,\lambda_1^b(k)} + (1 - \alpha_{\ell(k)}) \cdot N_1^{b,k}}, \tag{2}$$

where $N_B^{b,k}$ denotes the number of samples assigned to cluster k in mini-batch b of batch B, $\mathbf{c}_B^{b,k}$ its centroid, and $\alpha_\ell(k)$ is a forgetting factor that depends on $\ell(k)$, i.e. the label of cluster k that is *tied* through consecutive batches by virtue of repeated clustering and mapping over mini-batches. In the above expression we note that $\alpha_{\ell(k)} = 0$ implies no correction of the cluster center, whereas $\alpha_{\ell(k)} = 1$ denotes full persistence of the cluster space over the batch. Given this role of $\alpha_{\ell(k)}$ in the update dynamics of the cluster space, SLAYER modifies its value whenever a new batch arrives: if a change is declared, $\alpha_\ell = 0$ for all classes (line 15). Otherwise (line 17), α_ℓ is set inversely proportional to the average distance between centroids of tied clusters that are assigned label ℓ, i.e.:

$$\alpha_\ell \propto \left(\sum_{k=1}^{K_1^b} \| \mathbf{c}_1^{b,k} - \mathbf{c}_1^{b-1,\lambda_1^b(k)} \|_F \cdot \mathtt{I}(\ell(k) = \ell) \right)^{-1}, \tag{3}$$

where $\mathtt{I}(\cdot)$ is an auxiliary binary function taking value 1 if its argument is true (0 otherwise). By updating the forgetting factor as in the above expression, α_ℓ can be thought to be a rough estimation of the average *speed* at which label concepts move over the feature space during the batch. The above adaptation of α is done in a per label basis and not in a finer granularity (per every cluster), as this could increase significantly the overall computational complexity of SLAYER, specially in those cases with scattered cluster spaces.

Finally, SLAYER attains a higher level of adaptability against drifts over the stream by *predicting* where clusters will reside during the incoming batch (line 19). Since there is a correspondence (thoroughly tied through mini-batches) between the clusters discovered in consecutive batches, a lightweight Kalman filter is used to estimate the future coordinates $\widehat{\mathbf{c}}_{B+1}^k$ of every centroid at the end of the current batch B. A Kalman filter is a simple recursive system used to calculate the state of a linear dynamic system and the variance or uncertainty of the estimate. In SLAYER, a Kalman filter keeps track of the estimated position of cluster centers in the feature space, yielding a vector of estimated centroids $\{\widehat{\mathbf{c}}_{B+1}^k\}$ that is used for predicting the next batch of samples. This is possible thanks to their assigned labels propagated through the mini-batch-wise cluster-and-mapping process explained above.

4 Experimental Setup

In order to assess the performance of SLAYER, we make use of a public repository of non-stationary data streams widely adopted by the stream mining community working with gradual drifts [20]. Specifically, the repository contains 15 synthetic datasets featuring different drift changes over time, including translations, rotation, warps and other transformations of the feature space. Table 1 summarizes their main characteristics. The column labeled as D and L refer to the number of features and classes, respectively. In these datasets, the initial 5% of the samples are assumed to be supervised, whereas the remaining stream instances arrive in 100 batches in chronological order.

After reviewing the latest contributions on data stream classification under extreme verification latency in non-stationary environments (Section 2), we have built a comparison benchmark that includes several proposals published so far in this research area:

- A static classifier learned from the first labeled samples (STATIC).
- A sliding window classifier that learns initially from labeled samples, and updates its knowledge with predicted upcoming samples, discarding samples that do not fall inside the sliding window for predicting new instances (SLIDING).
- An incremental window classifier, which works similarly to the sliding window classifier, but that does not forget any past instance for training (INCR).
- COMPOSE [5], which creates a boundary from the current data and defines a shape that represents the distribution of each class. After several iterations, COMPOSE draws instances from the core region(s) to support training as labeled data. Finally, upon the reception of new unlabeled data, new instances are combined with that of the core regions to retrain the model and adapt to eventual non-stationary behaviors over the stream.
- LEVELIW [23, 24], an iterative weighting approach that relies on the assumption that there is an overlap among class-conditional distributions between consecutive time steps, a premise that holds in slow drifting data streams.
- AMANDA [7], in which the distributions of each class are first estimated over the received labeled samples. Then, a semisupervised learning classifier is learned and used to predict upcoming batches with unlabeled samples. A density-based algorithm measures the importance of the classified instances by weighting them, and only retaining the most representative samples. This method has two variations: AMANDA-FCP (A-FCP), which selects a fixed number of samples; and AMANDA-DCP (A-DCP), which dynamically selects samples from data.

Dataset	L	D	Instances	Class distribution	Description
1CDT	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), one moving in diagonal
1CHT	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), one moving horizontally
1CSurr	2	2	55283	37%/63%	Two clusters (one per class), one surrounding the other
2CDT	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), moving in the same diagonal
2CHT	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), moving together horizontally
FG2C2D	2	2	$2 \cdot 10^5$	75%/25%	Three clusters for one class, one moving cluster for the other class
GEARS	2	2	$2 \cdot 10^5$	50%/50%	Two rotating gears, one per class
MG2C2D	2	2	$2 \cdot 10^5$	50%/50%	Two clusters per class, moving and overlapping with each other
UG2C2D	2	2	$1 \cdot 10^5$	50%/50%	One cluster per class, moving without overlapping with each other
UG2C5D	2	5	$2 \cdot 10^5$	50%/50%	Two 5-dimensional clusters, one per class, moving and overlapping
4CE1CF	5	2	173250	20% per label	Four classes expanding and one class fixed, one cluster each
4CR	4	2	144400	25% per label	Four clusters, one per class, rotating with no overlap
4CRE-V1	4	2	$125 \cdot 10^3$	25% per label	Four clusters, one per class, rotating with expansion (version 1)
4CRE-V2	4	2	$183 \cdot 10^3$	25% per label	Four clusters, one per class, rotating with expansion (version 2)
5CVT	5	2	$4 \cdot 10^4$	33%×1, 16%×4	Five clusters, one per class, moving together vertically

Table 1: Slow drifting stream datasets from [20] utilized in this work.

In what refers to performance metrics, we use the so-called *prequential error*, which has been thoroughly used in the literature [9, 10]. For the sake of compli-

ance with the methodological practices in the above prior work, the prequential error is computed based on an accumulated sum of a loss function between the prediction and observed values [3], i.e.:

$$preqError(t) = \frac{1}{t}\sum_{t'=1}^{t}\mathcal{L}(y_{t'}, \widehat{y}_{t'}), \tag{4}$$

where the prequential error is computed at time t, $\widehat{y}_{t'}$ represents the prediction, and $y_{t'}$ represents the real value at time $t' \leq t$. Among the possible loss functions for classification, we specifically use the 0-1 loss, i.e. $\mathcal{L}(y_{t'}, \widehat{y}_{t'}) = 0$ if $y_{t'} = \widehat{y}_{t'}$ and 1 otherwise. The prequential error allows monitoring the evolution of the models' performance over time. However, it is convenient to gauge other performance metrics that are sensitive to mild class imbalance, as there is no certainty that classes are equally represented in batches received over time. Therefore, we also report on the macro F1 score, which grants the same importance to each label/class. Source code and datasets will be made available at a public repository available in https://git.code.tecnalia.com/maria.arostegi/slayer/.

5 Results and Discussion

The obtained results are summarized in Tables 2 (average prequential error) and 3 (average macro F1 score) for the methods and datasets considered in our study. In these tables, the best results for each datasets are highlighted in bold.

Dataset	STATIC	SLIDING	INCR	COMPOSE	LEVELIW	A-FCP	A-DCP	SLAYER
1CDT	0.76	0.06	0.3	0.08	0.04	0.02	0.05	**0.006**
1CHT	3.93	0.43	3.2	0.48	0.4	**0.33**	0.39	0.38
1CSURR	35.86	9.05	36.06	9.43	9.2	**4.39**	7.93	5.91
2CDT	46.3	6.13	46.14	6.73	49.74	5.46	5.83	**3.8**
2CHT	45.97	48.45	46.01	47.41	47.41	14.39	19.93	**10.27**
FG2C2D	17.79	4.43	18.29	12.15	4.31	5.12	16.39	4.32
GEARS	5.43	0.81	5.33	4.03	6.18	**0.81**	3.74	3.84
MG2C2D	51.63	22.86	50.66	49.17	9.31	8.7	14.88	**8.59**
UG2C2D	55.81	4.97	54.42	5.32	26.34	4.3	12.64	**4.26**
UG2C5D	30.97	20.11	30.62	20.82	20.82	8.21	8.53	**8.08**
4CE1CF	1.98	1.9	1.82	2.09	2.21	**1.73**	1.92	2.029
4CR	78.83	0.02	78.75	0.04	0.02	0.02	0.03	**0.009**
4CRE-V1	78.15	81.29	79.44	79.55	79	73.5	73.28	**2.21**
4CRE-V2	79.61	82.88	79.67	77.38	80.77	69.97	71.81	**7.69**
5CVT	54.51	60.97	52.04	65.5	59.18	24.11	52.38	**12.4**

Table 2: Prequential error of the compared methods for the considered datasets.

We begin our discussion by inspecting the prequential error scores in Table 2. It is first relevant to notice that the use of naïve methods (STATIC, SLIDING, INCR) yields in general comparatively bad results due to the fact that they are not designed to cope with non-stationary environments under extreme verification latency. If we take a closer look at these scores, except for GEARS, 4CR and 4CEF1CF, the prequential error of those methods are significantly higher that

the rest of algorithms in the benchmark. By contrast, if we consider approaches tailored to deal with unsupervised drifting streams such as COMPOSE or LEVELIW, results improve across datasets, yet still lagging behind those obtained by A-FCP, A-DCP and SLAYER.

Among the weak points of our proposed approach we pause at the importance of the density of the clusters, and the speed at which they move over the feature space. For example, in the `4CE1CF` dataset, in the beginning of the stream the 5 classes are very close to each other in the feature space, represented by single clusters with very similar densities. This poses a challenge for the change detection mechanism of SLAYER, as reflected by the 6th position in the ranking between models for this dataset. Therefore, SLAYER fails to properly characterize the evolution of concepts from unsupervised data streams when these two circumstances collide together.

We now focus on the comparison between the two AMANDA-based approaches (A-FCP and A-DCP) and SLAYER. First we observe that SLAYER obtains a slight advantage over A-FCP, as SLAYER scores best in 10 out of the 15 datasets under consideration. A-FCP attains the best performance in 4 datasets, falling down once to the 6^{th} position in the ranking. By contrast, A-DCP is never below the fourth position among the methods, but it does not score best in any of the datasets. Similar conclusions can be drawn when analyzing the macro F1 results shown in Table 3. SLAYER yields the best results for 9 datasets, followed by A-FCP that scores first in 5 datasets, and A-DCP in just one dataset.

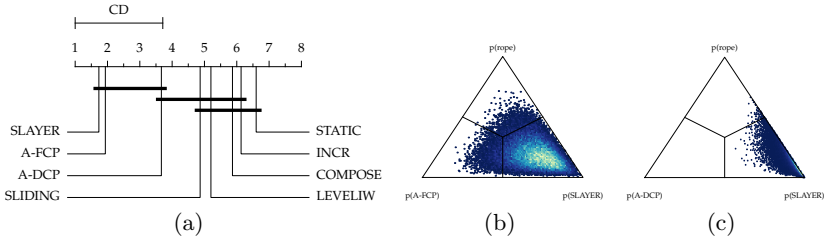
Dataset	STATIC	SLIDING	INCR	COMPOSE	LEVELIW	A-FCP	A-DCP	SLAYER
1CDT	0.9935	0.9994	0.9971	0.9995	0.9996	0.9997	0.9994	**0.9999**
1CHT	0.96	0.995	0.9681	0.9949	0.996	**0.9963**	0.9955	0.996
1CSURR	0.6403	0.9137	0.6384	0.9094	0.6368	**0.9607**	0.9267	0.9385
2CDT	0.3871	0.9418	0.3884	0.9362	0.4836	0.948	0.9416	**0.961**
2CHT	0.3954	0.356	0.3942	0.4758	0.4758	0.8526	0.788	**0.897**
FG2C2D	0.7322	0.9391	0.7298	0.8596	**0.9469**	0.9319	0.819	0.9419
GEARS	0.9474	0.9957	0.9485	0.9637	0.9382	**0.9957**	0.963	0.9579
MG2C2D	0.4795	0.7543	0.4936	0.505	0.5923	**0.9143**	0.8499	0.9133
UG2C2D	0.4425	0.9514	0.4546	0.9491	0.7366	**0.9581**	0.8706	0.9538
UG2C5D	0.668	0.7549	0.6782	0.7918	0.7918	0.9151	0.9129	**0.9153**
4CE1CF	0.9807	0.9795	**0.9821**	0.9781	0.9779	0.9808	0.9803	0.9798
4CR	0.2099	0.9998	0.2154	**0.9999**	0.9998	0.9998	**0.9999**	**0.9999**
4CRE-V1	0.2073	0.1804	0.1997	0.2035	0.2486	0.267	0.2651	**0.9778**
4CRE-V2	0.2043	0.1259	0.2039	0.1971	0.2464	0.3035	0.181	**0.9229**
5CVT	0.3537	0.1812	0.3707	0.2385	0.1767	0.7297	0.3802	**0.8849**

Table 3: Macro F1 results of the compared methods for the considered datasets.

We end our discussion by inspecting the statistical significance of the differences observed in the above tables. To shed light on this matter, Demsar's critical distance diagrams are often used [4]. These diagrams show the average ranks of a number of models under comparison across multiple datasets, wherein significant differences are declared when the difference between the average ranks of two models is larger than a critical distance (CD). The CD value is given by

a post-hoc Nemenyi test at a certain confidence level (usually set to 0.95). The critical distance diagram corresponding to the results in Table 2 is shown in Figure 2.a: unfortunately, the average rank achieved by SLAYER cannot be declared to be statistically significant, mainly due to the relatively high value of CD (2.71) that results from the large number of models being compared.

Although critical distance diagrams are widely used by the community, they have been reported to be misleading and scarcely interpretable [2]. Therefore, we complement this analysis with a pairwise Bayesian analysis of differences between the best performing methods in the benchmark, namely, A-FCP, A-DCP and SLAYER. Specifically, we model the probability that one model outperforms another based on the results obtained by each of them over all datasets. Once fitted, the probability distribution is sampled and displayed in barycentric coordinates, wherein three regions can be observed: the first algorithm outperforms the second (and vice versa), and a region of practical equivalence (the two methods perform similarly). A *rope* parameter establishes the minimum difference that scores of both methods must have for them to be declared different to each other, thereby ensuring that statistical significance relies on an interpretable parameter. Figure 2.b depicts the Bayesian posterior plot between A-FCP and SLAYER performed over the prequential error scores with a rope equal to 0.1. As shown in this plot, the sampled distribution appears to be clearly skewed towards SLAYER, revealing that it is likely that our proposal outperforms A-FCP, with prequential error differences larger than the selected *rope* value. When comparing SLAYER to A-DCP (Figure 2.c), the significance of the better prequential error performance observed for our proposal is even more significant.



(a) (b) (c)

Fig. 2: (a) Critical distance diagram of prequential error results in Table 2; (b) Bayesian posterior plot of A-FCP versus SLAYER corresponding to the same table; (c) Bayesian posterior plot of A-DCP versus SLAYER.

6 Conclusions and Future Work

This work has gravitated on a research area that has been paid considerably lower attention in the stream learning field than other widely studied paradigms: learning to classify streams subject to the effects of concept drift and extreme

verification latency. In this setup, classification models for streaming data become obsolete at a point due to the drift experienced by concepts to be classified over time, whereas the absence of supervision about the stream samples hinders severely the change detection, tracking and adaptation processes. This confluence of conditions has been the main motivation for SLAYER, the novel approach presented in this paper, which continuously analyzes the evolution of clusters and *ties* them together across batches, so that a correspondence between labels and evolving clusters can be constantly maintained over time. By virtue of this mechanism, SLAYER seamlessly accommodates the appearance/disappearance of new clusters, and is particularly suitable for environments where the speed of change in the feature space is not abrupt, so that posterior distributions overlap to a certain extent between consecutive time ticks (*slow* feature drift). Simulation results over 15 datasets with different drift dynamics have elucidated that in most of them, SLAYER outperforms state-of-the-art approaches contributed to address this kind of scenarios. Furthermore, differences have been found to be significant as concluded from a Bayesian posterior analysis.

Future research work will be devoted towards enhancing different constituent parts of SLAYER. We foresee to resort to methods capable of learning topological relationships between multi-dimensional samples (e.g. Growing Neural Gas) for a better characterization of the patterns associated to each of the classes are not corpuscular (as in the `GEARS` dataset). Also, we will assess the extent to which we can extrapolate core SLAYER concepts (especially the prediction of the evolution of the concepts using Kalman) to other data stream learning paradigms, including online clustering. Likewise, improvements are planned in the way SLAYER connect the different clusters over mini-batches, for which an alternative measure of similarity between clusters more reliable than the distance between their centroids will be sought. Finally, we will explore whether mini-batches of variable size can be considered in the design of SLAYER, so that mini-batches are enlarged whenever the drift dynamics between the previous consecutive mini-batches are slow. This would lessen the computational effort required to run SLAYER, and could give rise to 1) a more accurate representation of the prevailing cluster distribution; and 2) a better traceability of the label-cluster assignment over time.

Acknowledgments

This work has received funding support from the Basque Government through the ELKARTEK program (3KIA project, ref. KK-2020/00049). J. Del Ser also acknowledges support from the Department of Education of the same institution (Consolidated Research Group MATHMODE, IT1294-19).

References

1. Arostegi, M., Torre-Bastida, A.I., Lobo, J.L., Bilbao, M.N., Del Ser, J.: Concept tracking and adaptation for drifting data streams under extreme verification la-

tency. In: International Symposium on Intelligent and Distributed Computing. pp. 11–25. Springer (2018)
2. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. The Journal of Machine Learning Research **18**(1), 2653–2688 (2017)
3. Dawid, A.P.: Present position and potential developments: Some personal views statistical theory the prequential approach. Journal of the Royal Statistical Society: Series A (General) **147**(2), 278–290 (1984)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
5. Dyer, K.B., Capo, R., Polikar, R.: Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. IEEE transactions on neural networks and learning systems **25**(1), 12–26 (2013)
6. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011)
7. Ferreira, R.S., Zimbrão, G., Alvim, L.G.: Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency. Information Sciences **488**, 219–237 (2019)
8. Frederickson, C., Gracie, T., Portley, S., Moore, M., Cahall, D., Polikar, R.: Adding adaptive intelligence to sensor systems with mass. In: 2017 IEEE Sensors Applications Symposium (SAS). pp. 1–6. IEEE (2017)
9. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 329–338 (2009)
10. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine learning **90**(3), 317–346 (2013)
11. Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 44 (2014)
12. Gomes, H.M., Read, J., Bifet, A., Barddal, J.P., Gama, J.: Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explorations Newsletter **21**(2), 6–22 (2019)
13. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. Journal of Parallel and Distributed Computing **74**(7), 2561–2573 (2014)
14. Krempl, G.: The algorithm apt to classify in concurrence of latency and drift. In: International Symposium on Intelligent Data Analysis. pp. 222–233. Springer (2011)
15. Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al.: Open challenges for data stream mining research. ACM SIGKDD explorations newsletter **16**(1), 1–10 (2014)
16. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2018)
17. Marrs, G.R., Hickey, R.J., Black, M.M.: The impact of latency on online classification learning with concept drift. In: International Conference on Knowledge Science, Engineering and Management. pp. 459–469. Springer (2010)
18. Razavi-Far, R., Hallaji, E., Saif, M., Ditzler, G.: A novelty detector and extreme verification latency model for nonstationary environments. IEEE Transactions on Industrial Electronics **66**(1), 561–570 (2019)
19. Settles, B.: Active learning. Synthesis lectures on artificial intelligence and machine learning **6**(1), 1–114 (2012)

20. Souza, V.M.A., Silva, D.F., Gama, J., Batista, G.E.A.P.A.: Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SIAM International Conference on Data Mining (SDM). pp. 873–881. SIAM (2015)
21. Spiliopoulou, M., Ntoutsi, E., Theodoridis, Y., Schult, R.: Monic and followups on modeling and monitoring cluster transitions. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 622–626. Springer (2013)
22. Tsai, C.W., Lai, C.F., Chao, H.C., Vasilakos, A.V.: Big data analytics: a survey. Journal of Big Data **2**(1), 21 (2015)
23. Umer, M., Frederickson, C., Polikar, R.: Learning under extreme verification latency quickly: Fast compose. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8. IEEE (2016)
24. Umer, M., Polikar, R., Frederickson, C.: Level iw: Learning extreme verification latency with importance weighting. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 1740–1747. IEEE (2017)
25. Yang, S.: Iot stream processing and analytics in the fog. IEEE Communications Magazine **55**(8), 21–27 (2017)