Drift-Aware Multi-Memory Model for Imbalanced Data Streams

Amir Abolfazli
L3S Research Center
Leibniz University of Hanover
Germany
abolfazli@L3S.de

Eirini Ntoutsi
L3S Research Center
Leibniz University of Hanover
Germany
ntoutsi@L3S.de

***Abstract*—Online class imbalance learning deals with data streams that are affected by both concept drift and class imbalance. Online learning tries to find a trade-off between exploiting previously learned information and incorporating new information into the model. This requires both the incremental update of the model and the ability to unlearn outdated information. The improper use of unlearning, however, can lead to the *retroactive interference problem*, a phenomenon that occurs when newly learned information interferes with the old information and impedes the recall of previously learned information. The problem becomes more severe when the classes are not equally represented, resulting in the removal of minority information from the model. In this work, we propose the Drift-Aware Multi-Memory Model (DAM3), which addresses the class imbalance problem in online learning for *memory-based models*, namely KNNs. DAM3 mitigates class imbalance by incorporating an imbalance-sensitive drift detector, preserving a balanced representation of classes in the model, and resolving retroactive interference using a working memory that prevents the forgetting of old information. We show through experiments on real-world and synthetic datasets that the proposed method mitigates class imbalance and outperforms the state-of-the-art methods.**

***Index Terms*—online learning, class imbalance, concept drift, retroactive interference, multi-memory model.**

I. Introduction

The challenge of learning from imbalanced data streams with concept drift has attracted a lot of attention from both academia and industry in recent years. The term *concept drift* refers to changes in the underlying data distribution over time. *Class imbalance* occurs when the classes are not equally represented. Online class imbalance learning deals with data streams that are affected by both concept drift and class imbalance and exists in many real-world applications such as anomaly detection, risk management, and social media.

Online learning algorithms, dealing with imbalanced streams, not only try to better represent the minority class for the learning model (e.g., by oversampling the minority class), but they also try to find a trade-off between retaining previously learned information and adapting to new information from the stream, known as the *stability-plasticity dilemma* [1]. The basic idea is that a learning model requires *plasticity* for the integration of new information, but also *stability* in order to prevent the forgetting of old information [2]. A too adaptive model forgets previously learned information and a too stable model cannot learn new information, hence, finding a trade-off between plasticity and stability is required.

In recent years, many methods have been proposed to deal with online class imbalance learning (e.g., [3], [4]). Some methods employed *unlearning* to deal with concept drift (e.g., [5]–[7]) by removing (historical) information that is inconsistent, i.e., have contradicting class labels, with the incoming data from the stream. SAM-kNN [6] is one of such models based on kNNs that makes use of unlearning in the locality of the instances. SAM-kNN is a dual-memory model [6] that partitions the knowledge between short-term memory (STM) and long-term memory (LTM), containing the information of the current and former concepts, respectively. Preserving the consistency, in SAM-kNN, is based on a cleaning operation that unlearns the information of former concepts in the LTM that contradicts the information of the most recent concept, stored in the STM. Although *unlearning* is a desired property for the model adaptation, if not applied carefully, it could lead to the *retroactive interference* problem [8] that occurs when new information interferes with previously learned information, causing the (unintentional) forgetting of old information.

Figure 1 illustrates the problem of retroactive interference in a dual-memory model, where old information in the LTM is removed because the model adapts to new data in the STM. The problem becomes more severe when the classes are not equally represented in the stream as it could lead to the removal of minority instances, which are of higher interest than majority instances in many real-world applications. In this work, we propose a multi-memory model which deals with the class imbalance by 1) incorporating an imbalance-sensitive drift detector, 2) preserving a balanced representation of classes in the model, and 3) resolving the retroactive interference by means of a *working memory* (WM) [9], manipulating information in the LTM for every incoming instance to the STM. We also contribute new synthetic benchmarks with different drift types and class imbalance ratios.

II. Preliminaries and basic concepts

A data stream D is a potentially infinite sequence of instances arriving at distinct time points $1, \cdots, t, \cdots$, where t is the current timepoint. Each instance $\mathbf{x} \in D$ is described in a d-dimensional feature space, i.e., $x \in \mathbb{R}^d$. Without loss

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

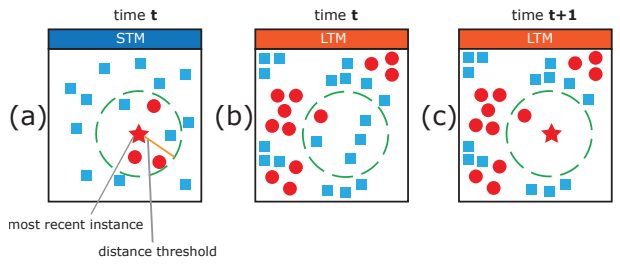

Figure 1. Illustration of retroactive interference problem in a dual-memory model. **(a)** the red star instance is the most recent instance in the STM. Its locality (indicated by the green dashed circle) is defined by the maximum distance of consistent instances (i.e., instances with the same class: red) in its neighborhood ($k = 5$). **(b)** state of the LTM at t, the affected area (indicated by the green dashed circle) centered at the new point's location. **(c)** state of the LTM at $t+1$ where (the majority of) the previously stored instances have been removed due to inconsistency with new information of the STM.

of generality, we assume a binary classification problem, i.e., $Y = \{+, -\}$. We follow the first-test-then-train or prequential evaluation setup [10]. Assuming a probability distribution $P(\mathbf{x}, y)$ generating the instances of D, the characteristics of P might change with time, i.e., for two time-points i, j, it might hold that $P_i(\mathbf{x}, y) \neq P_j(\mathbf{x}, y)$, a phenomenon called *concept drift* [10]. The drift type could also be characterized based on the rate at which drift occurs. *Sudden drift* results in a severe change in the distribution of data. *Incremental drift* occurs when the concept incrementally changes. *Gradual drift* occurs when the instances belonging to two different concepts are interleaved for a certain period of time. *Recurring drift* describes a case in which the concept which has already been observed, reoccurs. Apart from the occurrence of concept drifts, we also assume that the stream is *imbalanced* with the majority class (assumed to be $-$) occurring more often than the minority class (assumed to be $+$). To express the degree of imbalance, we use the *imbalance ratio (IR)* [11] defined as the number of minority instances over majority instances. IR is commonly denoted by 1:r (r is a value corresponding to the majority class) which specifies the ratio between the minority and majority class [12]. In a streaming data environment, imbalance can be either *static* assuming a fixed class ratio or *dynamic* assuming varying class ratio over the stream. Learning under imbalance is harder in a stream environment as there is no prior knowledge about the IR and often the role of minority and majority exchanges over the stream [13].

III. Related Work

Online class imbalance learning methods deal with data streams, affected by both concept drift and class imbalance. [14] proposed OBA which is an online ensemble method that improves the Online Bagging algorithm [15] by adding the ADWIN change detector. When a change is detected, the worst-performing base learner of the ensemble is replaced with a new one. [16] proposed Leverage Bagging (LB), an online ensemble method that leverages the performance of bagging by increasing the weights of the resampling using a larger value λ to compute the value of the Poisson distribution in order to increase diversity of the ensemble. LB uses the ADWIN change detector to deal with concept drifts. When a concept drift is detected, the worst base learner is reset. Both OBA and LB indirectly deal with class imbalance as [17] showed that diversity-increasing techniques such as bagging improve the performance of ensemble methods for imbalanced problems. [18] assume the likelihood of different features including class follows different trends and propose an ensemble method that predicts closely the best trend detector. [3] proposed online AdaC2, an online boosting algorithm that considers the different misclassification costs when calculating the weights of base learners and updates the weights of instances accordingly. More precisely, AdaC2 increases weight more on the misclassified positive instances than the misclassified negative instances. The same authors in [3] proposed the online RUSBoost, an online boosting algorithm that removes instances from the majority class by randomly undersampling the majority-class instances in each boosting round. The original version of both AdaC2 and RUSBoost do not deal with concept drift. However, the improved version of these methods deal with concept drifts using an ADWIN change detector.

The most relevant work to our work is the Self Adjusting Memory model for the k Nearest Neighbor algorithm (SAM-kNN) [6]. SAM-kNN builds an ensemble of classifiers induced on different memories: the short-term memory (STM) for the current concept and the long-term memory (LTM) for the former concepts, and the combined memory (CM) which is the union of STM and LTM. The authors propose a cleaning operation during the transfer that deletes instances of the LTM that are inconsistent with transferred instances of the STM.

The original SAM-kNN model does not consider class imbalance. In case of imbalance, the memories and in particular the LTM memory is increasingly dominated by the majority-class instances (see Figure 5 (b)). As a result, the performance of the model on minority instances is dropping. As we show in our experiments, this is not only because of the reduced representation of the minority class in the input stream but also because of the cleaning operation which deletes more instances of the minority class (see Figure 6 (e)).

Despite the lack of a compact model, KNNs comprise a popular choice for streams mainly because of the seamless addition and deletion of instances. [19] propose to learn specialized KNN models (for each entity) and a global KNN (for the whole stream) to ensure adequate representation of the entities in the learning models, independent of their volume, and moreover, they leverage the global model to deal with the cold-start problem. Recently, the problem of fairness-aware learning in the online setting and under class-imbalance has been introduced [20]; the proposed solution adapts the training distribution to take into account the evolving imbalance-and discriminatory-behavior of the model, both evaluated upon the historical stream.

IV. Drift-Aware Multi-Memory Model

In the research field of human memory, multi-memory models [21] have been proposed to overcome the limitations of

dual-memory models and better represent the human memory. Such models consist of the sensory register (SR), short-term memory (STM), and long-term memory (LTM). The basic idea is that first, the sensory information enters the SR, keeping the information for a very short time. The sensory information moves into the STM for temporary storage, and is encoded visually, acoustically or, semantically. The information is then transferred to the LTM after getting enough attention by processes such as active rehearsal [21]. The information that enters the STM is joined by context-relevant information in the LTM, which requires the retrieval of information from the LTM. Sometimes, the information of the LTM cannot be retrieved due to the *retroactive interference (RI)* [8]. RI is the interference occurring when newly learned information impedes the recall of previously learned information. A theoretical concept proposed in the field of cognitive psychology is the *working memory* [22], which introduces a memory that temporarily stores information relevant to the current task.

In a dual-memory model (see Figure 1), the problem of retroactive interference occurs when inconsistent information of the LTM is replaced with new information of the STM. Such a replacement, intended for dealing with concept drifts in SAM-KNN, can result in loss of information that happens 1) when information is transferred from the STM to the LTM, and 2) when the LTM is cleaned with respect to the STM for every incoming instance from the stream. As we will see in the experiments (Figure 5 (b)), such a replacement greatly affects the minority class, and therefore, the RI problem becomes more severe for the minority class.

To deal with this problem, we propose to incorporate a *working memory* (WM) that prevents the model from removing inconsistent instances. Due to the fact that stream contains more instances from the majority class and also the fact that the LTM is cleaned with respect to every incoming instance from the stream, more minority instances become inconsistent and thus removed from the LTM. Therefore, the minority class benefits more from the use of the WM. The proposed model, DAM3, is a multi-memory model for data streams with class imbalance and concept drifts that: i) introduces a working memory to deal with retroactive interference (Section IV-A), ii) incorporates an imbalance-sensitive drift detector (Section IV-C) to take into account the inherent imbalance, iii) preserves a balanced representation of classes using oversampling, (Section IV-D), and iv) removes noisy instances in the working memory that are generated after exchanging information (Section IV-F). The architecture of DAM3 is shown in Figure 2.

A. Model memories

DAM3 consists of four memories: STM, WM, LTM, and CM, each represented by a set of labeled instances.

Short-term memory (STM) is dedicated to the current concept and is a dynamic sliding window containing the most recent m instances from the stream (t is the current timepoint):

$$STM = \left\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{+1, -1\} | i = t - m + 1, \ldots, t\right\}.$$

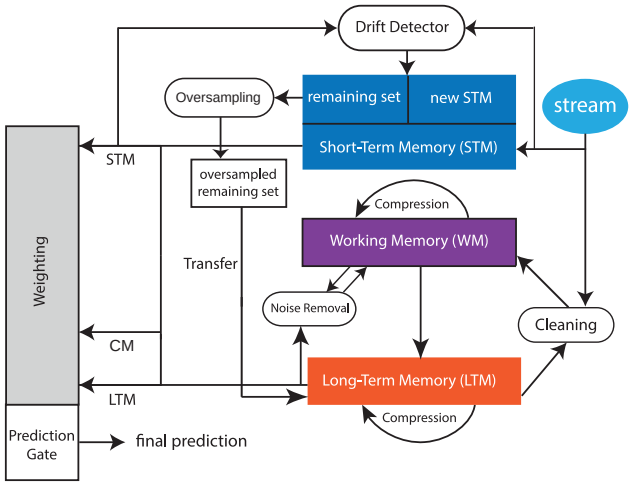

Figure 2. Architecture of DAM3.

Long-term memory (LTM) maintains information (p points) of former concepts that is *consistent* with the current concept:

$$LTM = \left\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{+1, -1\} | i = 1, \ldots, p\right\}.$$

Combined memory CM (STM ∪ LTM) represents the combination of short-term and long-term memories. It comprises just the union of STM and LTM and has the size $m + p$.

Working memory (WM) lends itself to resolving the retroactive interference problem. It preserves inconsistent information of the LTM and also transfers back (to the LTM) information that becomes consistent with the most recently stored information in the STM. In this way, the WM makes its consistent information available to the LTM for current predictions, made by the classifiers LTM and CM, and also retains valuable information for later predictions. The WM is a set of q points:

$$WM = \left\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{+1, -1\} | i = 1, \ldots, q\right\}.$$

B. DAM3 training, weighting, and prediction

Each memory induces a classifier; therefore, DAM3 could be considered as an ensemble method.

1) DAM3 training: In the SAM-KNN [6] model, weighted kNN classifiers were employed for all memories. In this work, we use the weighted kNN for the *LTM* and *CM* as it allows the seamless implementation of cleaning operation. kNN assigns a label to an instance $\mathbf{x}$ based on the memory instances:

$$\mathrm{kNN}_M(\mathrm{x}) = \arg\max_{\hat{y}} \sum_{x_i \in N_k(\mathbf{x}, M) | y_i = \hat{y}} \frac{1}{d(\mathbf{x}_i, \mathbf{x})} \quad (1)$$

where $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance between two points, $N_k(\mathbf{x}, M)$ is the set of k nearest neighbors of $\mathbf{x}$ in M, and $M \in \{LTM, CM\}$.

We use the *full Bayes* classifier [23] for the *STM* instead of the weighted kNN as the instance-based learning classifiers are quite sensitive to noisy data [24]. In our case, the use of the kNN as the STM classifier might result in incorrect predictions passed to the drift detector (c.f., Section IV-C).

The full Bayes classifier assumes that the distribution of data can be modeled with a multivariate Gaussian distribution [23]. A new instance $\mathbf{x}$ is classified as follows:

$$\mathrm{FB}_{\mathrm{STM}}(\mathbf{x}) = \arg\max_{y} p(y) f(\mathbf{x}|y) \tag{2}$$

where $p(y)$ is the class prior and $f(\mathbf{x}|y)$ is the multivariate Gaussian density function [23].

2) DAM3 weighting and prediction: Each of the base classifiers STM, LTM, and CM is weighted based on its balanced accuracy on the most recent ms instances of the stream, where ms is equal to the minimum size of the STM. The best performing model is chosen to predict the class label of the current instance.

C. Imbalance-sensitive drift detection

Information is transferred from the STM to the LTM when concept changes. A change in concept is signaled by significant degradation of the STM classifier, corresponding to the model learned on the current concept. Due to inherent class imbalance of the stream, performance of the model on both classes should be taken into account. Hence, we propose a drift detector that relies on balanced accuracy. The detector takes as inputs the incoming instances and their corresponding predictions made by the STM classifier.

Our proposed drift detector divides all the balanced accuracy values into two windows (reference window and test window) and performs the non-parametric Kolmogorov–Smirnov test. If the balanced accuracy values of the reference window are significantly different from those of the test window, the drift is detected and the STM size is reduced from m to ws, where the m is the current size of the STM and ws is the window size of the drift detector (each of the reference and test windows has the size ws), respectively. In this way, the test window becomes the new STM (STM_{t+1}, with the size ws) and the remaining set (denoted by Δ), with the size $m - ws$, is first oversampled and then transferred to the LTM.

D. Balanced representation of remaining set

Before transferring the information from the STM to the LTM, we perform oversampling on the remaining set Δ to ensure a balanced representation of both classes. We denote the oversampled remaining set by O_{Δ}. We use *BorderlineSMOTE*[1] [25], which selects instances of the minority class that are misclassified by a kNN classifier and oversamples only those difficult instances, being more important for classification. The oversampled set is then transferred to the LTM.

E. Transfer and cleaning:

DAM3 keeps the LTM consistent with the STM, similar to the SAM-kNN [6]. The LTM is cleaned with respect to every incoming instance from the stream. However, DAM3 does not perform any cleaning on the remaining set (transferred from the STM to the LTM) due to its designed drift detector. In contrast, SAM-kNN performs the cleaning as all the instances of the remaining set might not correspond to the previous concept. In both cases (transfer of information and cleaning of the LTM), deletion of inconsistent instances might lead to the retroactive interference problem (c.f., Figure 1). To allow for the cleaning/deletion of outdated information but also to prevent deletion of older information due to the interference with newer information, we use the working memory. This memory resolves the retroactive interference by exchanging its consistent information with the inconsistent information of the LTM for every incoming instance from the stream. Figure 3 illustrates how this problem is resolved.

We use the basic cleaning operation, proposed by the SAM-kNN [6], that deletes inconsistent instances of different classes in the locality of an instance. The main idea is that most recent instances of the stream convey the correct class-label, and instances of different labels in its locality should be considered as outdated and thus deleted.

Transfer of information from STM to LTM: The instances of O_{Δ} (oversampled remaining set) are transferred from the STM to the LTM and the LTM is updated as follows:

$$LTM_{t_d+1} = LTM_{t_d} \cup O_{\Delta_{t_d}},$$

where t_d is the time at which drift occurs.

Transfer of inconsistent information from LTM to WM: The k nearest neighbors of $\mathbf{x}_i$ in $STM \setminus (\mathbf{x}_i, y_i)$, at time t, are determined and the ones with label y_i are selected. The *distance threshold* θ at time t is then defined as:

$$\theta_t = \max \{ d(\mathbf{x}_i, \mathbf{x})) \mid \mathbf{x} \in N_k(\mathbf{x}_i, STM_t \setminus (\mathbf{x}_i, y_i)), \\ y(\mathbf{x}) = y_i \}.$$

On the basis of the found distance threshold of the STM_t, we define the *inconsistent set* of the LTM at time t (IS_{LTM_t}) with respect to the instance $(\mathbf{x}_i, y_i)$ in the STM_t:

$$IS_{LTM_t} = LTM_t \cap \{ (\mathbf{x}_j, y(\mathbf{x}_j)) \mid \mathbf{x}_j \in N_k(\mathbf{x}_i, LTM_t), \\ d(\mathbf{x}_j, \mathbf{x}_i) \leq \theta,\ y(\mathbf{x}_j) \neq y_i \}.$$

Similarly, we define the *consistent set* of the WM at time t (CS_{WM_t}) with respect to the set LTM_t and the instance $(\mathbf{x}_i, y_i)$ in the STM:

$$CS_{WM_t} = WM_t \cap \{ (\mathbf{x}_j, y(\mathbf{x}_j)) \mid \mathbf{x}_j \in N_k(\mathbf{x}_i, WM_t), \\ d(\mathbf{x}_j, \mathbf{x}_i) \leq \theta,\ y(\mathbf{x}_j) = y_i \}.$$

Based on the IS_{LTM_t} and CS_{WM_t}, inconsistent instances of the LTM (IS_{LTM_t}) are transferred to the WM and information of the WM is updated as follows:

$$WM_{t+1} = (WM_t \setminus CS_{WM_t}) \cup IS_{LTM_t}.$$

Transfer of consistent information from WM to LTM: Consistent instances of the WM are transferred back to the LTM and information of the LTM is updated as follows:

$$LTM_{t+1} = (LTM_t \setminus IS_{LTM_t}) \cup CS_{WM_t}.$$

The WM preserves inconsistent information of the LTM, as it might be useful for later predictions, and makes its consistent information available to the LTM for current predictions.

[1]We used the parameter values $k = 5$, $m = 5$ for the BorderlineSMOTE.

Compression of the LTM and WM. The LTM content is not discarded when its size exceeds the maximum threshold. Instead, as in SAM-KNN [6], we compress the data through class-wise *K-Means++* clustering, such that the number of instances per class is reduced by half. Similarly for WM.

F. Noise removal

The exchange of inconsistent instances of the LTM with the consistent instances of the WM might result in the generation of noisy instances. This means that some instances might become consistent with the LTM. We remove those instances which could be correctly classified based on the information of only the LTM, at each time point.

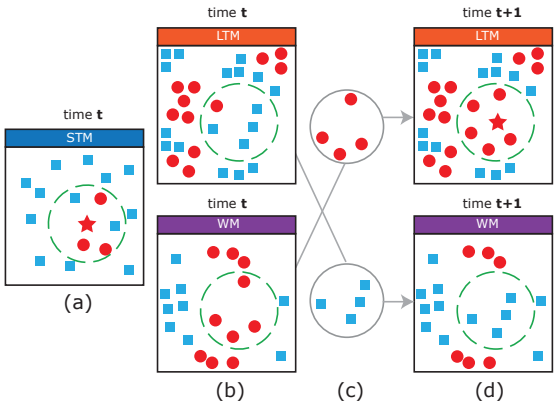

Figure 3. Resolving the retroactive interference using a working memory incorporated into a multi-memory model. **(a)** state of the STM at time point t; the red star point denotes the most recent instance in the STM. **(b)** state of the LTM and WM at time point t; there are four *inconsistent* instances in the LTM and four *consistent* instances in the WM. **(c)** consistent instances (in red) and inconsistent instances (in blue) with respect to the most recent instance. **(d)** state of the LTM and WM at the time point $t+1$; the inconsistent instances in the LTM are exchanged with the consistent instances in the WM.

V. Experiments

First, we compare DAM3 with state-of-the-art competitors on a variety of datasets (Section V-A) using appropriate measures and evaluation setup (Section V-B). The results are discussed in Section V-C. Then, we focus on the behavior of our proposed model, namely the class-imbalance ratio of each memory (Section V-D1), the interaction between the different memories, i.e., cleaning and transfer operations, (Section V-D2), and the size of memories with respect to the minority and majority classes (Section V-D3).

All the experiments were implemented and evaluated in Python using the scikit-multiflow framework [26]. The code and datasets are available on GitHub[2].

A. Datasets

We experimented with a variety of synthetic and real datasets summarized in terms of their cardinality, dimensionality, class ratio, and drift type in Table I, as described below.

Synthetic datasets: Synthetic datasets have the advantage that any desired drift behavior can be explicitly simulated. We used the MOA framework [27] to generate 4 synthetic streams with different types of concept drift. The SEA generator [28] was used to generate two streams: one stream with three sudden drifts and a constant 1:10 IR (*SEA_S*); and one stream with three gradual drifts where each concept has a different IR 1:4/1:5/1:2/1:10 (*SEA_G*). Similarly, the Hyperplane generator [29] was used to simulate two streams with incremental drifts: one with a constant IR 1:10 (*HyperFast*); and one with a dynamic IR 1:1 $\rightarrow$ 1:100 (*HyperSlow*). For the streams *SEA_S* and *HyperFast* (constant IR=1:10), we included 10% noise, and for the streams *SEA_G* and *HyperSlow* (having different IRs and dynamic IR, respectively), we included 5% noise.

Real-world datasets: Real-world datasets are used to show how well the stream classifiers perform in practice. A few real-world drift benchmarks are available for binary classification, of which we considered weather, electricity, and PIMA.

The *weather* dataset [30] contains 18,159 instances and 8 features corresponding to the measurements such as temperature and wind speed. The goal is to predict whether it will be a rainy day (minority class) or not.

The *electricity* dataset [31] contains 45,312 instances and 8 features such as date, demand, and price, and the goal is to predict whether the price will increase (minority class) or not, according to the moving average of last 24 hours.

The *PIMA* Indian dataset [32] contains 768 instances and 8 features, such as blood pressure, insulin, and age. The goal is to diagnostically predict whether a patient will have diabetes mellitus (minority class) or not, in 1-5 years.

Table I
THE CHARACTERISTICS OF THE DATASETS USED IN THE EXPERIMENTS.

Type	Datasets	#Instances	#Features	Class Ratio (+:-)	Noise	#Drifts	Drift Type
Synthetic	SEA_S	100K	3	1:10	10%	3	sudden; real
	SEA_G	100K	3	1:4/1:5/1:2/1:10	5%	3	gradual; real
	HyperFast	50K	5	1:10	10%	1	incremental; real
	HyperSlow	50K	5	1:1 $\rightarrow$ 1:100	5%	1	incremental; real
Real-world	Weather	18159	8	1:2.17	N/A	N/A	N/A
	Electricity	45312	8	1:1.35	N/A	N/A	N/A
	PIMA	768	8	1:1.85	N/A	N/A	N/A

B. Evaluation setup

1) Performance metrics: An appropriate performance metric takes into account the performance on all classes, rather than the overall performance which is heavily affected by the majority class. The *balanced accuracy* is an appropriate performance metric for imbalanced data. For binary classification, it is defined as the arithmetic mean of the sensitivity and specificity [12]. Another performance metric, often used for imbalanced data, is the *geometric mean (G-Mean)*. For binary classification, G-Mean is defined as the squared root of the product of the sensitivity and specificity. G-Mean punishes those models for which there is a big disparity between the sensitivity and specificity. It is different from the balanced accuracy which treats both classes equally [33].

2) Evaluation method: In data stream classification, the most commonly used evaluation method is the *prequential evaluation* [34]. The prequential evaluation is specifically designed for streaming settings, where instances arrive in

[2]https://github.com/amir-abolfazli/DAM3

sequential order. The idea is to first test the model on the instance, and then that instance is used to update the model. In this way, the model is always tested on the instances, not seen yet. The prequential evaluation is preferred over the traditional holdout evaluation as it makes the maximum use of the available data (i.e., no test set is needed) [35].

For the experiments, We evaluate the performance of the classifiers using the prequential evaluation and report on sensitivity, specificity, G-Mean, and balanced accuracy.

C. Predictive performance

In Table II, the predictive performance of DAM3 and compared classifiers on the different datasets is shown. The proposed method DAM3 outperforms all the compared methods in terms of G-Mean and balanced accuracy. This indicates that DAM3 finds a trade-off between the sensitivity and specificity better than other methods. To further investigate the differences in the average G-Mean and balanced accuracy (i.e., average ranks) of the compared methods on the considered datasets, we used the post-hoc Bonferroni-Dunn test [36] to compute the critical difference (CD). The results are shown in Figure 4 and as we can see, the performance of DAM3 is significantly better than AdaC2, LB, and SAM-kNN in terms of G-Mean, and significantly better than AdaC2, RusBoost, and LB, in terms of balanced accuracy.

Table II
PREDICTIVE PERFORMANCE OF THE CLASSIFIERS. CLASSIFIERS WITH THE BEST AND SECOND-BEST PERFORMANCE ARE MARKED IN BOLD AND UNDERLINED, RESPECTIVELY.

Dataset	Classifier	Sensitivity	Specificity	G-Mean	Balanced Accuracy
SEA_S	DAM3	0.4614	0.9190	**0.6512**	**0.6902**
	SAM-kNN	0.3778	**0.9884**	0.6111	0.6831
	AdaC2	0.1449	0.9883	0.3783	0.5666
	RUSBoost	**0.4777**	0.7961	0.6167	0.6369
	LB	0.3923	0.8826	0.5884	0.6374
	OBA	0.3845	0.9795	0.6137	0.6820
SEA_G	DAM3	**0.6504**	0.9393	**0.7816**	**0.7946**
	SAM-kNN	0.5467	**0.9919**	0.7364	0.7693
	AdaC2	0.2843	0.9913	0.5308	0.6378
	RUSBoost	0.5417	0.8523	0.6794	0.6970
	LB	0.5307	0.9345	0.7043	0.7326
	OBA	0.5569	0.9901	0.7426	0.7735
HyperFast	DAM3	0.4026	0.9021	**0.6026**	**0.6523**
	SAM-kNN	0.2205	**0.9930**	0.4680	0.6067
	AdaC2	0.0912	0.9900	0.3005	0.5406
	RUSBoost	**0.4144**	0.7883	0.5716	0.6014
	LB	0.2976	0.8846	0.5131	0.5911
	OBA	0.2596	0.9809	0.5046	0.6202
HyperSlow	DAM3	**0.3990**	0.9230	**0.6069**	**0.6610**
	SAM-kNN	0.2852	**0.9960**	0.5330	0.6406
	AdaC2	0.1180	0.9942	0.3426	0.5561
	RUSBoost	0.3768	0.8728	0.5734	0.6248
	LB	0.2795	0.9428	0.5133	0.6112
	OBA	0.2668	0.9949	0.5152	0.6308
Weather	DAM3	0.7479	0.7663	**0.7570**	**0.7571**
	SAM-kNN	0.5080	**0.9068**	0.6788	0.7074
	AdaC2	**0.8982**	0.4211	0.6150	0.6596
	RUSBoost	0.5549	0.7869	0.6608	0.6709
	LB	0.5501	0.8052	0.6655	0.6777
	OBA	0.5627	0.8785	0.7031	0.7206
Electricity	DAM3	**0.8335**	0.8780	**0.8555**	**0.8558**
	SAM-kNN	0.7879	0.8566	0.8215	0.8222
	AdaC2	0.3824	**0.9681**	0.6085	0.6753
	RUSBoost	0.8017	0.8214	0.8115	0.8115
	LB	0.8233	0.8709	0.8468	0.8471
	OBA	0.7274	0.8285	0.7763	0.7780
PIMA	DAM3	0.7143	0.7079	**0.7111**	**0.7111**
	SAM-kNN	0.3308	0.8727	0.5373	0.6017
	AdaC2	0.2247	**0.9699**	0.4669	0.5346
	RUSBoost	0.4962	0.7228	0.5989	0.6095
	LB	**0.7491**	0.5113	0.6189	0.6302
	OBA	0.5489	0.8090	0.6664	0.6789

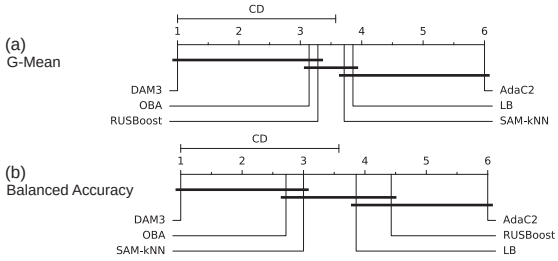

Figure 4. Critical difference diagram for the post-hoc Bonferroni-Dunn test.

Ablation study. In DAM3, all the components (drift detector, oversampling, working memory, and weighting of classifiers based on balanced accuracy) work together to mitigate the class imbalance. Therefore, showing the impact of each component alone does not make sense. Apart from these components, one of the main differences between DAM3 and SAM-kNN is the use of full Bayes as the STM classifier (instead of the weighted kNN).

In Table III, we compare the performance of DAM3 in terms of G-Mean and balanced accuracy with the performance of SAM-kNN with both kNN and full Bayes as the STM classifier, and show the difference in the performance. The results indicate that the use of the full Bayes as the STM classifier in SAM-kNN slightly improves the performance on all datasets (except SEA_S and Weather). The results also show that for all the datasets (except Electricity), the superior performance of DAM3 is mainly due to the considered components and not the use of the full Bayes classifier.

Table III
IMPACT OF THE FULL BAYES CLASSIFIER USED AS STM CLASSIFIER FOR SAM-KNN, COMPARED WITH THE ORIGINAL SAM-KNN AND DAM3.

Dataset	Classifier	G-Mean	G-Mean Diff	Balanced Accuracy	Balanced Accuracy Diff
SEA_S	DAM3	0.6512	↑ 0.0457	0.6902	↑ 0.0103
	SAMkNN – STM_FB	0.6055	↓ 0.0056	0.6799	↓ 0.0032
	SAMkNN – STM_kNN	0.6111		0.6831	
SEA_G	DAM3	0.7816	↑ 0.0364	0.7946	↑ 0.0185
	SAMkNN – STM_FB	0.7452	↑ 0.0088	0.7761	↑ 0.0068
	SAMkNN – STM_kNN	0.7364		0.7693	
HyperFast	DAM3	0.6026	↑ 0.0957	0.6523	↑ 0.0290
	SAMkNN – STM_FB	0.5069	↑ 0.0389	0.6233	↑ 0.0166
	SAMkNN – STM_kNN	0.4680		0.6067	
HyperSlow	DAM3	0.6069	↑ 0.0494	0.6610	↑ 0.0076
	SAMkNN – STM_FB	0.5575	↑ 0.0245	0.6534	↑ 0.0128
	SAMkNN – STM_kNN	0.5330		0.6406	
Weather	DAM3	0.7570	↑ 0.0828	0.7571	↑ 0.0528
	SAMkNN – STM_FB	0.6742	↓ 0.0046	0.7043	↓ 0.0031
	SAMkNN – STM_kNN	0.6788		0.7074	
Electricity	DAM3	0.8555	↑ 0.0137	0.8558	↑ 0.0133
	SAMkNN – STM_FB	0.8418	↑ 0.0203	0.8425	↑ 0.0203
	SAMkNN – STM_kNN	0.8215		0.8222	
PIMA	DAM3	0.7111	↑ 0.0986	0.7111	↑ 0.0698
	SAMkNN – STM_FB	0.6125	↑ 0.0752	0.6413	↑ 0.0396
	SAMkNN – STM_kNN	0.5373		0.6017	

D. Model behavior

The goal of this section is to shed light on the internal mechanisms of DAM3 and their contribution towards tackling both class imbalance and concept drifts.

1) Imbalance perception by the model: In this section, we examine the IR of the memories of our model, DAM3, compared with the cumulative IR of the stream. We also

examine the IR of the memories of the SAM-kNN [6] which is the most similar model to our model in terms of architecture.

Figure 5 (a) shows the IR of all memories for our model DAM3, compared with SAM-kNN, on the Weather dataset. Since our model incorporate a drift detector sensitive to the balanced accuracy, it shortens the STM faster than SAM-kNN. The drift detector helps the model to not make the STM full with majority instances. As a result, the IR of the STM in DAM3 is lower than that of SAM-kNN. In (b), the green and red lines correspond to the IRs of the LTM for DAM3 and SAM-kNN, respectively. The lines reveal that what is reflected by SAM-kNN is significantly higher than the actual IR shown in (d). Moreover, for SAM-KNN, the IR of the LTM gradually increases over time, implying that the majority class becomes increasingly dominant in the LTM. Unlike SAM-kNN, DAM3 reflects a balanced representation of classes (IR ≈ 1). In (c), the purple line shows the IR of the WM that corresponds to the instances which are inconsistent with the LTM.

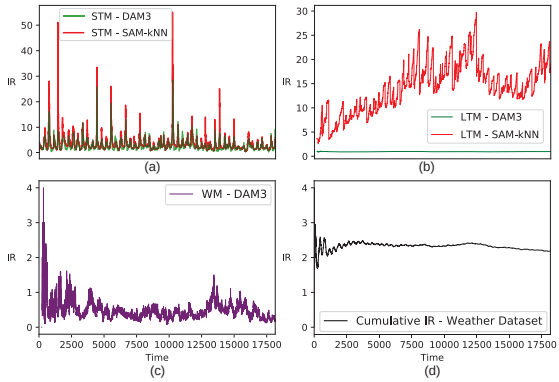

Figure 5. The IR of the memories of DAM3 compared with that of SAM-kNN on the Weather dataset.

2) Removal and transfer of instances in LTM and WM: In this section, we demonstrate the "removal" and "transfer" of information within the long-term and working memories.

In Figure 6, (a) and (b) show the number of *inconsistent* minority and majority instances, removed and transferred from the LTM to the WM in DAM3. (c) and (d) show the number of *consistent* minority and majority instances, removed and transferred from the WM to the LTM in DAM3. (e) and (f) show the number of *inconsistent* minority and majority instances which are removed from the LTM in SAM-kNN. Since we used the kNN with k=5, at most, there could be 5 inconsistent instances to be removed and transferred at each time point. Comparing the subfigures (a) and (b) with (e) and (f) shows that DAM3 removes fewer minority instances compared with SAM-kNN. This statement is supported by the subfigure (b) in Figure 5, showing the IR of the LTM (in red), where the IR increases gradually. This implies that SAM-kNN removes more minority instances over time. The subfigures (c) and (d) show the number of removed and transferred *consistent* instances of the WM for both minority and majority classes. Both (c) and (d) show a similar behavior, revealing that there are some consistent instances which could be transferred back almost all the time. This means that the DAM3 resolves the problem of retroactive interference, impeding the model's ability to retrieve the old minority instances.

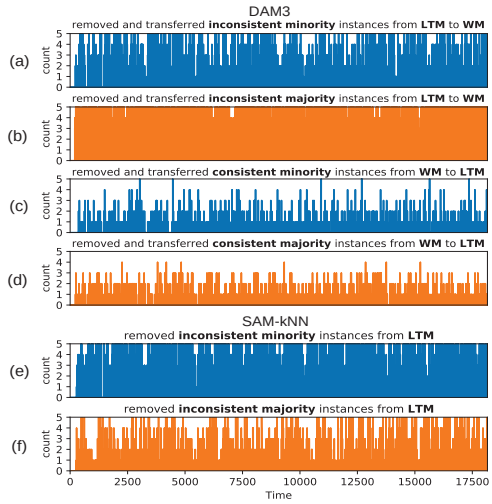

Figure 6. The number of removed and transferred instances from the LTM to WM and vice versa for DAM3, compared with the number of removed inconsistent instances from LTM for SAM-kNN, on the Weather dataset.

3) Size of memories: In Figure 7, (a) and (b) correspond to the size of STM with respect to the minority and majority for the models DAM3 and SAM-kNN, respectively. Both subfigures show a similar behavior, however, the size of STM in DAM3 is, on average, 32% smaller than that of the SAM-kNN, all the time, due to the use of the drift detector. In (c), the number of majority instances in the LTM, for the SAM-kNN (blue line), gradually increases while the number of minority instances (red line) remains almost the same. The behavior is completely different for DAM3, where the number of minority instances is almost equal to that of majority instances. However, preserving a balanced representation of classes in the LTM of DAM3 causes the information of the LTM to be compressed more. The sudden drops in the size of the LTM correspond to the times at which compression occurs. In (d), green and red lines correspond to the number of minority and majority instances, respectively, in the WM.

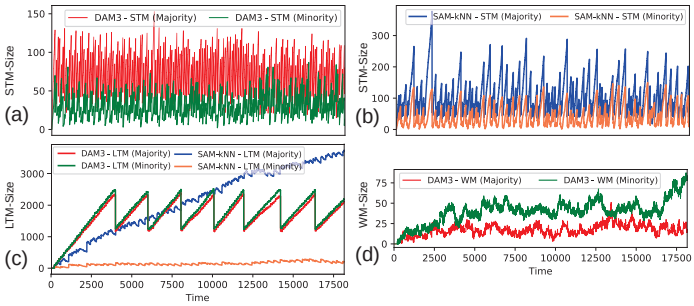

Figure 7. The size of all memories with respect to the minority and majority classes for both models DAM3 and SAM-kNN, on the Weather dataset.

VI. Conclusion

In this paper, we proposed the Drift-Aware Multi-Memory Model (DAM3), designed to mitigate the class imbalance in dual-memory models, dedicating short-term and long-term memories to the current and former concepts, respectively. DAM3 mitigates the class imbalance by incorporating an imbalance-sensitive drift detector, preserving a balanced representation of classes in the long-term memory, resolving the retroactive interference using a working memory preventing the removal of old information, and weighting the classifiers induced on different memories based on their balanced accuracy. Our experimental results showed that the proposed method outperforms the state-of-the-art methods in terms of G-Mean and balanced accuracy. For future work, we intend to design a multi-memory model that deals with recurring drifts as well as with limited feedback [37].

Acknowledgment

The work of the first author was supported by the German Research Foundation (DFG) within the project OSCAR (Opinion Stream Classification with Ensembles and Active leaRners) and HEPHAESTUS (Machine learning methods for adaptive process planning of 5-axis milling) for both of which the second author is a principal investigator.

References

[1] G. A. Carpenter and S. Grossberg, "Art 2: Self-organization of stable category recognition codes for analog input patterns," *Applied optics*, vol. 26, no. 23, pp. 4919–4930, 1987.

[2] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in psychology*, vol. 4, p. 504, 2013.

[3] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3353–3366, 2016.

[4] H. Zhang, W. Liu, S. Wang, J. Shan, and Q. Liu, "Resample-based ensemble framework for drifting imbalanced data streams," *IEEE Access*, vol. 7, pp. 65 103–65 115, 2019.

[5] B. Krawczyk and M. Wozniak, "Weighted naive bayes classifier with forgetting for drifting data streams," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015, pp. 2147–2152.

[6] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 291–300.

[7] H. Yu and G. I. Webb, "Adaptive online extreme learning machine by regulating forgetting factor by concept drift map," *Neurocomputing*, vol. 343, pp. 141–153, 2019.

[8] Z. Sosic-Vasic, K. Hille, J. Kröner, M. Spitzer, and J. Kornmeier, "When learning disturbs memory–temporal profile of retroactive interference of learning on memory formation," *Frontiers in psychology*, vol. 9, p. 82, 2018.

[9] A. Diamond, "Executive functions," *Annual review of psychology*, vol. 64, pp. 135–168, 2013.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[11] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced data sets," *Soft Computing*, vol. 13, no. 3, p. 213, 2009.

[12] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[13] E. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel, "Density-based projected clustering over high dimensional data streams," in *SIAM SDM*. SIAM, 2012, pp. 987–998.

[14] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD*, 2009, pp. 139–148.

[15] N. C. Oza, "Online bagging and boosting," in *2005 IEEE international conference on systems, man and cybernetics*, vol. 3. Ieee, 2005, pp. 2340–2345.

[16] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2010, pp. 135–150.

[17] J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio, and L. I. Kuncheva, "Diversity techniques improve the performance of the best imbalance learning ensembles," *Information Sciences*, vol. 325, pp. 98–117, 2015.

[18] D. P. Melidis, M. Spiliopoulou, and E. Ntoutsi, "Learning under feature drifts in textual streams," in *ACM CIKM*, 2018, pp. 527–536.

[19] V. Unnikrishnan, C. Beyer, P. Matuszyk, U. Niemann, R. Pryss, W. Schlee, E. Ntoutsi, and M. Spiliopoulou, "Entity-level stream classification: exploiting entity similarity to label the future observations referring to an entity," *International Journal of Data Science and Analytics*, vol. 9, no. 1, pp. 1–15, 2020.

[20] V. Iosifidis and E. Ntoutsi, "FABBOO- online fairness-aware learning under class imbalance," in *Discovery Science*. Cham: Springer International Publishing, 2020, pp. 159–174.

[21] R. C. Atkinson and R. M. Shiffrin, "The control of short-term memory," *Scientific American*, vol. 225, no. 2, pp. 82–91, 1971.

[22] A. D. Baddeley and G. Hitch, "Working memory," in *Psychology of learning and motivation*. Elsevier, 1974, vol. 8, pp. 47–89.

[23] M. R. Berthold, C. Borgelt, F. Höppner, and F. Klawonn, *Guide to intelligent data analysis: how to intelligently make sense of real data*. Springer Science & Business Media, 2010.

[24] Z. Qin, A. T. Wang, C. Zhang, and S. Zhang, "Cost-sensitive classification with k-nearest neighbors," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2013, pp. 112–131.

[25] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[26] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2914, 2018.

[27] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[28] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *ACM SIGKDD*, 2001, pp. 377–382.

[29] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD*, 2003, pp. 226–235.

[30] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[31] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[32] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association, 1988, p. 261.

[33] V. García, R. A. Mollineda, and J. S. Sánchez, "Index of balanced accuracy: A performance measure for skewed class distributions," in *Iberian conference on pattern recognition and image analysis*. Springer, 2009, pp. 441–448.

[34] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine learning*, vol. 90, no. 3, pp. 317–346, 2013.

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

[36] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[37] C. Blake and E. Ntoutsi, "Reinforcement learning based decision tree induction over data streams with concept drifts," in *IEEE ICBK*. IEEE, 2018, pp. 328–335.