# A Diversity Framework for Dealing with Multiple Types of Concept Drift Based on Clustering in the Model Space

Chun Wai Chiu, Leandro L. Minku, *Member, IEEE*

***Abstract*—Data stream applications usually suffer from multiple types of concept drift. However, most existing approaches are only able to handle a subset of types of drift well, hindering predictive performance. We propose to use diversity as a framework to handle multiple types of drift. The motivation is that a diverse ensemble can not only contain models representing different concepts, which may be useful to handle recurring concepts, but also accelerate the adaptation to different types of concept drift. Our framework innovatively uses clustering in the model space to build a diverse ensemble and identify recurring concepts. The resulting diversity also accelerates adaptation to different types of drift where the new concept shares similarities with past concepts. Experiments with 20 synthetic and 3 real-world data streams containing different types of drift show that our diversity framework usually achieves similar or better prequential accuracy than existing approaches, especially when there are recurring concepts or when new concepts share similarities with past concepts.**

***Index Terms*—Online Ensemble Learning, Concept Drift, Recurring Concepts, Clustering in the Model Space, Diversity**

## I. Introduction

Due to the fast growth and high incoming speed of data, it is inefficient or even impossible to wait for all the data to arrive before analysing them. *Data stream learning* [1], which is able to analyse data sequentially as they arrive, have thus become increasingly important. It has been used in several real-world applications, such as software defect prediction [2], software effort estimation [3], credit card fraud detection [4], spam filtering [5], and soft sensors in process industry [6].

Typically, data streams suffer changes in the underlying distribution of the data. Such changes are referred to as *concept drifts* [7], whereas the underlying joint probability distribution of the data is commonly referred to as a *concept* [7]. To enable a swift reaction to concept drifts, data stream learning algorithms desirably operate in an online way, where the predictive model is updated upon the arrival of each separate incoming training example [1]. Thus, this paper focuses on online learning.

Concept drift can be categorised into three main types: *abrupt, gradual, and recurrent* [8]. Abrupt drifts replace the current concept $C_{BeforeDrift}$ by another concept $C_{AfterDrift}$ very suddenly – in a single time step [8]. Gradual drifts replace $C_{BeforeDrift}$ by $C_{AfterDrift}$ over a transitional period where the probability of an example being drawn from $C_{BeforeDrift}$ ($C_{AfterDrift}$) gradually decreases (increases) [1], [8], [9]. Depending on how large the differences between $C_{BeforeDrift}$ and $C_{AfterDrift}$ are, drifts can be further categorised as severe or mild severity drifts. Active approaches, which employ explicit drift detection methods, are typically the most effective in dealing with abrupt drifts [8], even though their effectiveness typically relies on hyper-parameters that control the trade-off between drift detection delay and false-positive drift detection. Passive approaches that continuously update predictive models without relying on explicit drift detection tend to be more effective for gradual drifts [1]. Recurrent drifts (or recurring concepts) occur when a concept previously seen in the data stream reappears [8], [10]. Approaches that maintain a memory of models representing different past concepts can exploit past knowledge, being appropriate for this type of drift [10].

Many data stream learning algorithms have been proposed to deal with concept drift, but they tend to be better at handling a small subset of types of drift. We posit that diversity could be used as an overall framework to handle multiple types of drift effectively and efficiently. Diversity can be perceived as the level of disagreement among a set of models. It has been shown that it can improve the recovery of predictive performance from gradual and mild severity drifts while achieving acceptable predictive performance on severe and abrupt drifts [11]. However, the benefits of diversity could potentially go beyond these types of drift. For example, diversity has the potential to be a memory management strategy, which can effectively maintain a memory of diverse past models to better handle recurrent drifts [10]. Yet, no existing work has exploited the full potential of using diversity to deal with multiple types of drifts at the same time.

This paper thus aims to investigate whether diversity can provide a robust framework to deal with multiple types of drift. In particular, it answers the following research questions: ***RQ1)*** *How can diversity be used as a memory strategy to deal with multiple types of concept drift, including recurrent and non-recurrent drifts?* ***RQ2)*** *How effective is this framework in dealing with multiple types of concept drift compared to existing approaches?* ***RQ3)*** *Why is this framework successful or unsuccessful in dealing with multiple types of concept drift compared to existing approaches?*

The novel framework exploits the full potential of using diversity to deal with multiple types of concept drift. It manages a memory of models by using a diversity metric to decide which models to discard when the memory has

C. W. Chiu and L. L. Minku are with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 7TT, UK. E-mail: {cxc1015, L.L.Minku}@cs.bham.ac.uk

reached its maximum size. A clustering algorithm is employed in the model space to decide which of these models to recover from the memory for learning and making predictions, aiming at achieving robustness to multiple types of concept drift (RQ1). The proposed approach is compared against six existing approaches through experiments on twenty synthetic data streams with multiple types of concept drift and three real-world data streams. The results show that the proposed approach can deal with multiple types of concept drift well (RQ2), especially recurrent drifts and drifts leading to concepts that share similarities with past concepts. This is because it managed to identify and exploit similar past knowledge to accelerate recovery from drifts (RQ3).

## II. Related Work

### A. *Approaches for Dealing with Concept Drift*

Many different approaches have been proposed to deal with concept drift [7], [12]–[14]. As explained in Section I, this paper focuses on online learning approaches. One of the most classic online learning approaches is the Drift Detection Method (DDM) [15]. DDM monitors the error rate of a predictive model. If the error increases significantly, it triggers a concept drift detection and resets the model. The monitored model can be a single model [15] or an ensemble [11]. Diversity for Dealing with Drift (DDD) [11] is an active approach that uses a highly diverse ensemble with a drift detection method to recover the predictive performance from different types of concept drift. However, DDM and DDD do not maintain a memory of past models. So, they cannot exploit potentially useful past knowledge to handle recurrent drifts.

Recurring Concept Drift (RCD) [16] maintains a memory of past models. Each of these models is associated with a buffer of data representing the respective underlying concept. Upon concept drift detection, if there is no significant difference between the most recent batch of data and one of the past buffers, RCD deems that a recurring concept is detected and the corresponding past model is retrieved from the memory to react to the drift. However, RCD adopted a First In First Out (FIFO) queue as its memory management strategy. Thus, if a concept reoccurs after a long period, it loses the advantage of exploiting past knowledge to recover from the drift. Diversity Pool (DP) [10] uses a diversity metric to maintain a diverse memory of models, increasing the chances to exploit relevant past knowledge when past concepts reoccur. However, DP focuses on handling recurring concepts and does not aim at achieving robustness to multiple types of drift. Also, the time complexity of its diversity-based memory management strategy is high and it adopts a conservative method that may fail to identify recurrent concepts.

Passive ensemble approaches could also potentially handle recurring concepts, as they could maintain models representing past concepts in the ensemble, depending on their memory management strategy. Dynamic Weighted Majority (DWM) [17] and Online Accuracy Update Ensemble (OAUE) [18] maintain a weighted majority vote ensemble, and use memory management strategies that delete ensemble members based on their predictive performance on the current concept. These strategies may not always be ideal because concepts that are very different from the current concept can potentially reappear at any time. Deleting them could thus be detrimental. Nevertheless, these two approaches performed quite well in the previous work [10], with OAUE being more competitive against DP than DWM.

### B. *Approaches for Learning in the Model Space*

Some existing work on learning in the model space focuses on offline learning [19]–[21]. Chen et al. [22], [23] proposed an online learning approach for fault diagnosis based on time series classification. The approach creates models to represent different portions of a time series that describes the behaviour of a monitored system. One-class SVMs are trained on such models to recognise previously seen states of the system, where new states are assumed to represent new types of fault. The different states of the monitored system can be seen as different concepts in time series, suggesting that learning in the model space could potentially be useful to support strategies to handle drifts in data stream classification problems. However, no existing work has investigated this potential so far.

## III. Proposed Approach

To answer the research questions proposed in Section I, we propose a new approach called Concept Drift Handling Based on Clustering in the Model Space (CDCMS). The main novelty of CDCMS is its diversity memory strategy, which includes a clustering in the model space strategy and a memory management strategy aimed at dealing with multiple types of concept drift. CDCMS is presented in Algorithm 1 and its general working mechanism is described below.

CDCMS uses a *memory* $R$ of size $e \times r$ to store past models representing different concepts, where $e$ is the predefined maximum size of the ensembles and $r$ is the memory size multiplier. A representative ensemble $E_{nl}$ consisting of a new model is initialised to learn the current concept (line 1). During learning, each training example (line 3) is stored into a sliding window $B$ (line 4), which will be used later by the clustering in the model space strategy to deal with drifts. A drift detection method is employed (line 5) to determine whether strategies to react to concept drift should be triggered. Any drift detection method can be used, e.g., [15], [24]–[28].

At every $b$ time steps (line 7), CDCMS adds a new model $c_n$ to the representative ensemble $E_{nl}$ (line 17), as a strategy to deal with gradual drifts. Before adding this model, CDCMS first checks whether $E_{nl}$ has reached its maximum size $e$ (line 12). If so, the least performing model $c_{worst}$ in $E_{nl}$ is removed from $E_{nl}$ and saved into the memory $R$ using the diversity-based memory management strategy (line 15) explained in Section III-C. In other time steps, $c_n$ learns alongside with $E_{nl}$ (line 19). Together with the creation of a new model at every $b$ time steps, this helps to fill up the memory quickly so that the approach holds a good number of past models for the clustering in the model space strategy explained in Section III-A. Alternatively, if the current time step $t$ is $b$ time steps after a drift detection (line 8), the representative ensemble $E_{nl}$ is replaced by $e-1$ existing models from the memory that have

**Algorithm 1** Concept Drift Handling Based on Clustering in the Model Space - CDCMS

**Hyper-Parameters**: Ensemble Size ($e$), Memory Size ($e \times r$), Window Size ($b$), Similarity Threshold ($\theta$), Data Stream ($S$)
**Variables**: New Ensemble ($E_{nl}$), Old Ensemble ($E_{ol}$), Highly Diverse Ensemble ($E_{nh}$) New Model ($c_n$), Sliding Window ($B$), Memory ($R$), Clustering Result (C), Cluster (cluster)

```
1: $E_{nl} \leftarrow E_{nl}.add(createModel())$
2: $E_{ol} \leftarrow null; E_{nh} \leftarrow null; c_n \leftarrow createModel()$
3: for $s_t \in S$ do
4:     $B.saveFIFO(s_t)$
5:     $drift\_level \leftarrow DriftDetection(E_{nl}, s_t)$
6:     if $drift\_level == NORMAL$ then
7:         if $t$ mod $b == 0$ then
8:             if $t$ is $b$ time steps after a drift then
9:                 $C \leftarrow (R \cup c_n).clusteringModels(B)$
10:                $cluster \leftarrow C.getCluster(c_n)$
11:                $E_{nl}.add(cluster.getMostTrained(e-1))$
12:            else if $|E_{nl}| \geq e$ then
13:                $c_{worst} \leftarrow E_{nl}.getWorstModel()$
14:                $c_{worst}.resetWeight()$
15:                $R.saveByDiversity(c_{worst})$
16:            end if
17:            $c_n \leftarrow createModel()$; $E_{nl}.add(c_n)$
18:        end if
19:        $c_n.updateWeight(s_t); c_n.trainOnInstane(s_t)$
20:        $drift\_level_{previous} \leftarrow NORMAL$
21:    else if $drift\_level == DRIFT$ then
22:        $E_{ol} \leftarrow E_{nl}$
23:        $E_{nl}.resetWeight(); R.saveByDiversity(E_{nl})$
24:        $E_{nl}.clear(); E_{nh} \leftarrow createEnsemble(e)$
25:        if $drift\_level_{previous} == NORMAL$ then
26:            $c_n \leftarrow createModel()$
27:            $C \leftarrow R.clusteringModels(B)$
28:            $E_{nh} \leftarrow C.getRepresentativeModels()$
29:        end if
30:        $E_{nl}.add(c_n); c_n \leftarrow createModel()$
31:        $E_{nh}.resetWeight(); E_{nl}.resetWeight()$
32:        $E_{ol}.resetWeight(); B.clear()$
33:        $drift\_level_{previous} \leftarrow DRIFT$
34:    end if
35:    if $E_{ol} \neq null$ then $E_{ol}.updateWeight(s_t)$
36:    end if
37:    if $E_{nh} \neq null$ then $E_{nh}.updateWeight(s_t)$
38:    end if
39:    $E_{nl}.updateWeight(s_t); E_{nl}.trainOnInstane(s_t)$
40: end for
```

similar underlying concepts to the current concept, based on the clustering in the model space strategy. This strategy aims to deal with recurrent drifts.

Upon drift detection (line 21), $E_{nl}$ is denominated as the old representative ensemble $E_{ol}$ (line 22). Its base models are stored into the memory, using the diversity-based memory management strategy (line 23) explained in Section III-C. A highly diverse ensemble $E_{nh}$ is built by taking a representative model from each cluster of models from the memory (line 28), using the clustering in the model space strategy explained in Section III-A. Before building $E_{nh}$, CDCMS checks whether the previous state of the system is "NORMAL" (line 25). This is to prevent running the clustering method to rebuild $E_{nh}$ in consecutive time step caused by false-positive drift detection, which may detrimentally affect the prequential accuracy and the running speed. This ensemble is created to help to deal with drifts by making use of previously learnt knowledge. It can potentially help not only with recurrent concepts but also with drifts leading to new concepts that share similarities with old concepts. At the same time, a new representative ensemble $E_{nl}$ is also created, which consists of a new model $c_n$ created upon drift detection (line 30, 26).

To make predictions, CDCMS uses the representative ensemble $E_{nl}$ with weighted majority vote over the base models most of the time. The weights of the base models are based on their prequential accuracy. During the uncertain period after the drift detection, all three ensembles are used to make predictions by weighted majority vote until the system deems to have confidence in using $E_{nl}$ only. Similarly, the weights of the ensembles are also based on their prequential accuracy. Section III-B discusses this in more detail.

The key aspect of CDCMS is the use of clustering in the model space (lines 9, 27). The idea behind is that, ideally, models learnt from the same concept should make the same set of mistakes when predicting the same chunk of data. This means that clustering models by the predictions they made on the sliding window $B$, should cause models holding the same concept to belong to the same cluster. This technique plays an important role in CDCMS because it enables a convenient way to identify recurring concepts and to construct ensembles with different levels of diversity for handling concept drifts.

Another key aspect of CDCMS is its diversity-based memory management strategy, which decides which models to discard when the memory reaches its maximum size (line 15, 23). This strategy aims to maintain a memory of models holding concepts as different as possible. It is more time-efficient than the preliminary strategy in [10], as will be discussed in Section III-D3. CDCMS is further detailed in Sections III-A to III-D.

## A. Clustering Models in the Diverse Memory

Clustering models in the diverse memory is crucial to CDCMS. It is used to tackle multiple types of concept drift as part of the unified framework. For instance, it is used to form a highly diverse ensemble by picking a representative model from each cluster, which is then used to deal with abrupt drifts and drifts leading to concepts that share similarities with past concepts. It is also used to enable identifying recurring concepts and to accelerate adaptation to new concepts that share similarities with past concepts by determining the cluster to which a newly created model belongs. This section concentrates on our method for clustering in the model space, whereas Section III-B explains how the results of the clustering are used to deal with concept drift.

The clustering in the model space strategy is based on the predictions given by the models to a batch of examples. Therefore, a sliding window of examples is needed for this purpose. This sliding window is emptied when a drift is detected, to prevent it from holding examples that belong to past concepts (line 32). That said, we believe that CDCMS has certain robustness to the presence of old examples in the sliding window, because similar models will make similar mistakes on old examples.

Whenever CDCMS needs to cluster the models in the memory, it first requests these models to make predictions to the examples in the sliding window. A data set is then created where each example is a list of correctnesses of a model's predictions to the examples in the sliding window. A value of 1 represents correct while a value of 0 represents incorrect. Therefore, if the window has size $b$, each training example fed to the clustering algorithm is a list of $b$ binary attributes, and if the memory has $e \times r$ models, there are $e \times r$ examples. We refer to these examples as "clustering examples", to avoid confusion with the training examples from the data stream that are stored in the sliding window. Any clustering algorithm can be used to cluster the clustering examples, however, CDCMS only needs a "snapshot" of the model space to build ensembles with different diversity levels or to identify recurring concepts *when a drift is detected*. Thus, an offline clustering algorithm is sufficient. In this study, we used Expectation Maximisation (EM) clustering [29] with 10-fold cross-validation to decide the number of potential clusters.

## B. Drift Handling and Recurring Concepts Identification

DDD has shown that past knowledge can help to speed up learning of a new similar concept and to deal with gradual concept drifts [11]. As this could result in swifter recovery from drifts than resetting the learning system upon drift detection [11], we also create a highly diverse ensemble upon drift detection. However, even though DDD can deal with new concepts well, it cannot handle recurring concepts well, as it does not have a memory of past models. To achieve an approach that can deal with both new and recurring concepts, we hereby propose a new method to create ensembles with different diversity levels by making use of the results of the clustering algorithm explained in Section III-A.

Models clustered together are likely to represent the same or similar concepts, whereas models in different clusters are likely to represent different concepts. CDCMS thus creates highly diverse ensembles by taking the model trained with the largest number of examples from each cluster (line 28). In the event of having only one cluster, models at every $r$ index number interval are taken to build the highly diverse ensemble $E_{nh}$. The fact that our proposed method creates a highly diverse ensemble consisting of different past concepts means that, in the case of encountering a recurring concept, one of these models may hold the same or a similar concept, helping to achieve faster recovery from the drift. This is a key potential advantage over DDD, which creates highly diverse ensembles by enforcing high diversity on an ensemble trained from a single concept. DDD's strategy may be inadequate to handle recurring concepts, as the resulting highly diverse ensemble does not incorporate models representing concepts older than the most recent one.

To identify and deal with recurring concepts, once $b$ new training examples are received after the drift detection (line 8), the models in the memory and the new model created upon the last drift detection are clustered (line 9). If the new model belongs to a cluster with other past models, the current concept is regarded as a recurring concept (line 10). The algorithm then recovers the $e-1$ models trained with the largest number of training examples from that cluster to fill up the new representative ensemble (line 11), where $e$ is the size of the new representative ensemble. If the new model does not belong to a cluster containing other past models, the current underlying concept is regarded as a new concept. The new representative ensemble will then carry on being built from scratch, starting as an ensemble containing the new model as its sole member. Besides, if the drift leads to a concept that shares similarities with any past concept, this method of using clustering to recover relevant past models can also help the system to adapt the drift.

Since drift detection, all three ensembles (the highly diverse ensemble $E_{nh}$, the old and new representative ensembles $E_{ol}, E_{nl}$) are used to make combined predictions until $E_{nl}$ performs better than either of $E_{ol}$ and $E_{nh}$. After that, the system uses $E_{nl}$ only. $E_{ol}$ and $E_{nl}$ give predictions based on weighted majority among their base models, while $E_{nh}$ gives predictions by a simple majority among its base models. This is because $E_{nh}$ consists of models from different concepts and we do not have enough examples from the new concept to assign the base models with reliable weights right after drift detection. The final combined prediction of CDCMS is by weighted majority among the predictions given by $E_{ol}$, $E_{nl}$ and $E_{nh}$. The weights used for the majority votes are computed as follows:

$$w_t^i = \begin{cases} 1, & \text{if } c_i \in E_{nh} \\ \frac{acc_t(c_i)}{\sum_{l=1}^{M} acc_t(c_l)}, & \text{otherwise} \end{cases} \tag{1}$$

where $c_i$ is one of the $L$ base models and $M$=$L$ when computing the predictions by $E_{ol}$, $E_{nl}$ and $E_{nh}$, or $c_i \in \{E_{ol}, E_{nl}, E_{nl}\}$ and $M$=3 when computing the overall prediction by CDCMS. The prequential accuracy $acc_t(c_i)$ is:

$$acc_t(c_i) = \begin{cases} acc_t^{ex}(c_i), & \text{if } t = f \\ \frac{acc_t^{ex}(c_i)+\alpha \times acc_{t-1}(c_i) \times (t-f)}{t-f+1}, & \text{otherwise} \end{cases} \tag{2}$$

where $t$ is the current time step, $acc_{ex}$ is 0 if the prediction to the current training example $ex$ before learning is wrong and 1 if it is correct, $f$ is the first time step used in the calculation, and $\alpha$ is a fading factor.

The calculation of the overall weighted majority vote among $M$ models is shown below:

$$W.M.V._t = \underset{y \in Y}{\arg\max} \sum_{i=1}^{M} w_t^i I(c_i(x_t) = y_t) \tag{3}$$

where $I$ is the indicator function and $x_t$ are the input attributes of the example being predicted.

## C. diversity-based Memory Management Strategy

As described in the general mechanism of CDCMS, the algorithm regularly (at every $b$ time steps) creates a new model

$c_n$ to learn the current concept. If the maximum size $e$ of the representative ensemble $E_{nl}$ is reached, the least performing model $c_{worst} \in E_{nl}$ is removed and placed into the memory $R$ before adding $c_n$ to $E_{nl}$. If $R$ had already reached its maximum size ($e \times r$), the algorithm searches for the model $c_{sim} \in R$ that is the most similar to $c_{worst}$. Only either $c_{worst}$ or $c_{sim}$ will be kept in $R$. This will maintain the memory with a maximum size while ensuring that diversity is preserved, increasing the chances that various past knowledge is kept.

Yule's Q-statistics (Q) [30] is employed to find the most similar model. It is used to calculate the similarity between two models $c_i$ and $c_k$ as follows:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{4}$$

where $N^{a,b}$ is the number of examples where the classification by $c_i$ is $a$ and the classification by $c_j$ is $b$, 1 represents a correct classification and 0 represents a misclassification. $Q$ varies between 1 and -1. Models that tend to classify the same/different examples correctly will have positive/negative values of $Q$.

A predefined threshold ($\theta$) is used to ensure that $c_{worst}$ and $c_{sim}$ are similar enough. If $Q_{worst,sim} \leq \theta$, the least performing between the two models is discarded. Otherwise, the model trained with more examples is kept in memory. This memory management strategy is an improvement over DP [10], as will be discussed in Section III-D.

### D. Time Complexity

The time complexity of CDCMS consists of its time complexities for making predictions, training models, clustering models and managing the diverse memory.

*1) Prediction and Training Time Complexity:* Considering each ensemble has $e$ base models with prediction time complexity in $O(M_p)$, the time complexity of CDCMS to make a prediction at any time step during concept stationary period is $O(eM_p)$, which is just the combined vote of the base models in the ensemble. During the period between the drift detection and until the new representative ensemble is deemed to be ready to make predictions by itself, the time complexity of CDCMS to make a prediction is $O(3eM_p) = O(eM_p)$ because all three ensemble ($E_{nl}$, $E_{nh}$ $E_{ol}$) participate in making combined prediction. The training time complexity of CDCMS can be determined similarly because it always trains the new representative ensemble only. Thus, the training time complexity is $O(eM_t)$, where $O(M_t)$ is the training time complexity of the base model. Note that the time complexities $O(M_p)$ and $O(M_t)$ depend on the base learner used.

*2) Clustering in the Model Space Strategy and Concept Drift Handling:* Consider that we have $H$ models with prediction time complexity in $O(M_p)$ and we need to cluster them based on $b$ examples. As described in Section III-A, the models are first required to make predictions to the sliding window $B$ of examples, such that a data set can be built to hold the results of the predictions for the clustering. The time complexity of this procedure is $O(HM_pb)$. A clustering method with time complexity $O(G)$ is then used to cluster the models based on this data set, where $O(G)$ depends on the clustering method used. Thus, the overall time complexity to obtain the clustering results of those $H$ models based on $b$ examples is in $O(HM_pb + G)$. This time complexity applies to both building the highly diverse ensemble and identifying recurring concepts because the clustering in the model space strategy is used in both circumstances despite the results of the clustering being used differently.

*3) diversity-based Memory Management Strategy:* Consider the memory has $L = e \times r$ models with prediction time complexity $O(M_p)$. The time complexity of the memory management strategy proposed in Section III-C is in $O(LM_p)$.

In contrast, DP [10] uses an exhaustive search method. It simulates all possible memory states by eliminating each model from the memory and then computes their respective diversity level. It then keeps the memory state that has the highest diversity level as the new memory. To compute the diversity level of each possible memory state, the average Q-statistics ($Q_{av}$) over all pairs of models is used:

$$Q_{av} = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{k=i+1}^{L} Q_{i,k} \tag{5}$$

The time complexity of Eq. 5 is $O(L^2M_p)$. As the number of possible memory states is $L$, the overall time complexity of the exhaustive search is $O(L^3M_p)$, which is less efficient than the $O(LM_p)$ time complexity of the proposed memory management strategy.

## IV. Experiments To Evaluate The Predictive Performance of CDCMS

This section presents the experiments performed to evaluate the predictive performance of CDCMS and provide a detailed understanding of the reasons behind it, answering RQ2 and RQ3 posed in Section I. The source code of CDCMS is available at https://github.com/michaelchiucw/CDCMS.

### A. Data Streams Used in The Experiments

Twenty synthetic data streams were created using the Agrawal [31], SEA [32], Sine [15] and STAGGER [33] generators from the Massive Online Analysis (MOA) framework [34]. Synthetic data allows us to specify the severity of the concept drifts and the order of the concepts, enabling us to obtain a detailed understanding of the predictive performance of CDCMS in different situations.

Streams Sine1-2 consist of two numerical input attributes. Their concepts f1-f4 are defined in MOA, and involve the mathematical sine function in its calculation. Streams Agr1-4 consist of nine numerical input attributes. Their concepts f1-f10 are the same as those in [31]. Streams SEA1-2 consist of three numerical input attributes, where the last of them is irrelevant. Their concepts f1-f4 are the same as those in [32], where the thresholds are 8, 9, 7, and 9.5 respectively. We added a SEA function f5 with threshold 4, to enable the investigation of drifts with relatively higher severity. Streams STA1-2 consist of three categorical input attributes. Their input attributes (size, colour and shape) are the description for a domain of objects. The STAGGER concepts are defined in

TABLE I
DATA STREAMS

| Data Stream | Concept Drift Sequence |
|---|---|
| Sine1 | f3→f4→f3 |
| Sine2 | f1→f2→f3→f4→f1 |
| Agr1 | f1→f3→f4→f7→f10 |
| Agr2 | f7→f4→f6→f5→f2→f9 |
| Agr3 | f4→f2→f1→f3→f4 |
| Agr4 | f1→f3→f6→f5→f4 |
| SEA1 | f5→f3→f1→f2→f4 |
| SEA2 | f5→f1→f4→f3→f2 |
| STA1 | f1→f2→f3→f2 |
| STA2 | f2→f3→f1→f2 |

Coloured rows (lime / light grey) highlight data streams with recurring concepts. $fn$ represents the n-th function of the stream type, i.e., f1 in Agr1 is referring to the first function of the Agrawal generator.

TABLE II
PERCENTAGE DIFFERENCE OF CONCEPTS

| | Sine | | | SEA | | | | STAGGER | |
|---|---|---|---|---|---|---|---|---|---|
| | f1 | f2 | f3 | f1 | f2 | f3 | f4 | f1 | f2 |
| f2 | 100.0% | - | - | 8.5% | - | - | - | 59.3% | - |
| f3 | 26.8% | 73.2% | - | 7.4% | 16.0% | - | - | 77.8% | 48.1% |
| f4 | 73.2% | 26.8% | 100.0% | 13.1% | 4.6% | 20.6% | - | - | - |
| f5 | - | - | - | 23.9% | 32.5% | 16.5% | 37.1% | - | - |

| | Agrawal | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 |
| f2 | 53.9% | - | - | - | - | - | - | - | - |
| f3 | 53.1% | 50.8% | - | - | - | - | - | - | - |
| f4 | 53.9% | 20.5% | 50.8% | - | - | - | - | - | - |
| f5 | 53.4% | 47.6% | 50.7% | 47.7% | - | - | - | - | - |
| f6 | 69.9% | 28.9% | 51.2% | 35.5% | 48.1% | - | - | - | - |
| f7 | 50.5% | 53.3% | 50.1% | 53.5% | 60.1% | 57.2% | - | - | - |
| f8 | 33.5% | 60.4% | 46.5% | 59.6% | 59.6% | 59.8% | 49.8% | - | - |
| f9 | 50.4% | 53.3% | 50.2% | 53.5% | 59.9% | 57.3% | 6.0% | 49.5% | - |
| f10 | 32.9% | 61.3% | 46.5% | 61.3% | 60.0% | 59.9% | 51.1% | 1.8% | 51.1% |

All percentage differences were calculated using Eq. 6 based on one million random generated examples.

MOA, where any of one or more input attributes are taken into account and the remaining are irrelevant. All synthetic data streams are binary classification problems.

Concept drifts were simulated by connecting two data streams with different concepts at regular intervals, using a sigmoid function to decide the probability of data coming from the old and new concepts during the transitional period. The order of concepts of the synthetic data streams are presented in Table I. Each data stream from that table has been investigated using all gradual and all abrupt drifts. For gradual drift streams, the width of the drifts is 2,000 and the central drift point is at the middle of this transitional period. For abrupt drift streams, the width of the drifts is 1. These data sets is available at https://github.com/michaelchiucw/CDCMS.

Regarding the severity of drifts in the synthetic data streams, Table II presents the percentage difference of each pair of concepts used in the experiments, calculated as follows:

$$diff(f_a, f_b) = \frac{\sum_{i=1}^{n} |y^i_{f_a} - y^i_{f_b}|}{n} \tag{6}$$

where $y^i_{f_a}$ and $y^i_{f_b}$ are the class labels determined by the $a$-th and $b$-th functions of a generator, respectively, and $n$ is the total number of examples generated uniformly at random to calculate the severity. We consider a concept drift to be severe when the concepts before and after the drift have at least around 50% difference, while around 25% difference is referred to as a mild severity drift.

Three real-world data streams (KDDcup99, Power Supply stream, and Sensor) have also been used, allowing us to have a general idea of the predictive performance of CDCMS in practical applications. These data streams are available at [35]. KDDcup99 contains data that was produced based on simulated attacks to a network. So, even though it is composed of real-world data, its drifts can be considered synthetic.

## *B. Experiment Setup*

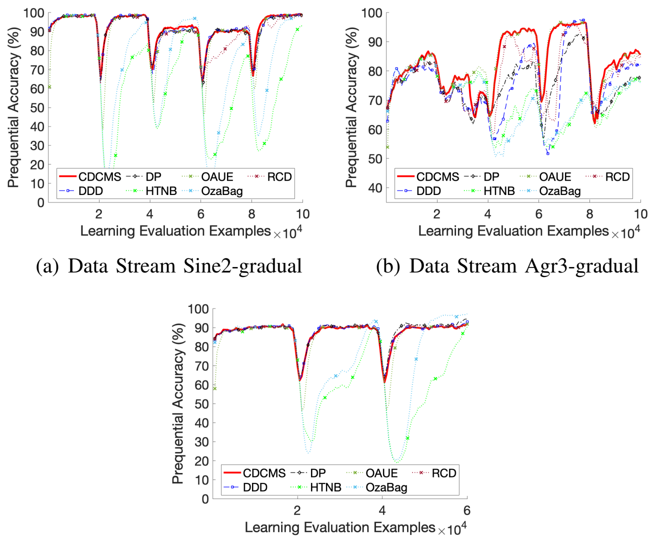
To answer RQ2 posed in Section I, CDCMS's predictive performance was quantitatively compared against other existing approaches. All approaches used Hoeffding Tree with Naïve Bayes at the leaves (HTNB) as the base learner. The approaches and the reasoning behind their choice are listed below:

- *HTNB* [36]: to compare against the base learner.
- *OzaBag (Online Bagging)* [37]: to compare against a baseline ensemble approach without any concept drift handling mechanism.
- *OAUE* [18]: to compare against a passive ensemble approach.
- *DDD* [11]: to compare against an active ensemble approach which uses a highly diverse ensemble to handle gradual and mild severity drifts.
- *DP* [10]: to compare against an approach that uses a diverse memory to handle recurrent drifts.
- *RCD* [16]: the compare against an approach designed to handle recurrent drifts.

Accuracy was measured prequentially, i.e., each example in the data stream was used to test the approach before training on it. A fading factor of 0.999 was used to make the past examples less important to the current accuracy [38]. The prequential accuracies were sampled at every 500th example.

To tune the hyper-parameters of each approach for experiments with synthetic data streams, four synthetic data streams were created (one for each stream type: Sine, Agrawal, SEA, STAGGER) with random orders of concepts, drift width and central drift point. The hyper-parameter values that lead to the highest average prequential accuracy on the randomised data stream for Sine, Agrawal, SEA, and STAGGER were then chosen to be used in the experiments with data streams Sine1-Sine2, Agr1-Agr4, SEA1-SEA2, and STA1-STA2, respectively. For experiments with real-world data streams, the first 10% examples of each stream were used for hyper-parameter tuning.

The hyper-parameter options are as follows. Hyper-parameters related to batch size and evaluation period are in {200, 400, 500, 600, 800, 1000}. Ensemble size is in {5, 10, 15, 20}. Memory size multiplier for CDCMS is in {4, 6, 8, 10}. Drift detection method is in {DDM [15], EDDM [24], ADWIN [25]}. Statistical significance threshold used in RCD is in {0.01, 0.05}. The diversity measure used in DP is in {Entropy Measure, Q-Statistic}. The $\lambda$ value to control the diversity level of the highly diverse ensemble in DDD is in {0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5}, which are the same options used in [11] for hyper-parameter tuning. The base learner, HTNB, and drift detection methods ran with default MOA hyper-parameter values.



(a) Data Stream Sine2-gradual (b) Data Stream Agr3-gradual

(c) Data Stream Sine1-gradual

Fig. 1. Prequential Accuracy with Synthetic Gradual Drift Streams

Each approach ran for thirty times on each data stream. For each data stream, the Friedman test is used to determine if there is a statistically significant difference in the average prequential accuracy among all approaches. If there is, Nemenyi post hoc test is used to determine which approach is significantly different from the top-ranked approach. Based on the comparison, a few representative runs from experiments with synthetic data streams were chosen to analyse CDCMS's behaviour qualitatively in detail. This will complement the answer to RQ2 and provide an in-depth understanding of the reasons behind the results, answering RQ3. Other cases are omitted due to page limitations and can be explained similarly.

## C. *Results with Synthetic Gradual Drift Data Streams*

The left part of Table III shows that CDCMS achieved similar or better average accuracy in most cases, as depicted by bold text cells. CDCMS only performed significantly worse than DDD and DP in Sine1-gradual. In other cases, even though CDCMS was not always the top-ranked, its accuracy had no significant difference with the top-ranked approaches. Over the next paragraphs, we provide a detailed analysis of these results based on the first single run from data streams with representative predictive performance results.

*1) Cases where CDCMS performed better:* Sine2-gradual and Agr3-gradual are representative cases where CDCMS performed significantly better than most other approaches.

Sine2-gradual is composed of four gradual severe drifts (see Tables I and II). Figure 1(a) shows that CDCMS was usually among the best approaches in Sine2-gradual both in terms of fast reaction and recovery from all drifts. It archived particularly good accuracy at the rear of the data stream, which is a recurring concept. It also archived good stable accuracy after drifts. The behaviour log of CDCMS shows that it successfully identified the first drift as a drift leading to a new concept. However, it mistakenly considered the second drift as a recurrent drift and recovered a set of models that mainly held the knowledge of concept f1. This was because concepts f1 and f3 are quite similar (see Table II). Thus, CDCMS performed the best during the period of concept f3, as shown in Figure 1(a) (40k-60k time steps). At the third drift, CDCMS also considered this drift as a recurrent drift and recovered a set of models holding the knowledge of concepts f1 and f2. Despite this new low diversity ensemble holding a partially different concept (*diff*(f1,f4): 53.9%, *diff*(f2,f4): 20.5%), it still managed to be one of the fastest approaches in reacting to this drift and adapting to concept f4. At the fourth drift, CDCMS successfully identified the recurring concept and managed to recover corresponding models from the memory. That is why it recovered from this drift faster than the other approaches. DP and RCD could not identify this recurring concept because their methods for identifying recurring concepts are conservative.

This analysis shows that the clustering in the model space strategy together with the diversity-based memory management strategy worked better than DP and RCD in handling gradual severe recurrent drifts appearing after several gradual severe drifts. Interestingly, despite OzaBag having an initially slower recovery (as it is not designed for non-stationary data streams), it eventually managed to recover its accuracy from this drift and surpass all approaches' accuracies at around 50,000th time steps. This may indicate that OzaBag is better for stationary periods, as it will not be negatively affected by false-positive drift detections, nor by new base models.

Agr3-gradual's first drift has mild severity, whereas its other three drifts are severe (see Tables I and II). Although Figure 1(b) shows that there is a slight drop in CDCMS's accuracy at around the 32,450th time step, the predictive performance of CDCMS and OAUE generally are similar, which is confirmed by the statistical test (see Table III). Looking back to Figure 1(b), CDCMS had a better recovery rate than other approaches from the last drift, which is a recurrent drift. It also recovered quite well from the second and the third drifts. Surprisingly, the behaviour log shows that CDCMS only had a delayed drift detection for the first drift but no drift detection at the second and the third drifts. Yet, it could still adapt to the changes at these drifts. This is because the ensemble committee substitution mechanism at every fixed time interval gradually and continuously updates the knowledge held in the representative ensemble, enabling it to recover from missed drift detections. In fact, OAUE, which also uses a similar mechanism, obtained similar overall accuracy to CDCMS, suggesting that this type of mechanism is the most suitable for this data stream. Besides, a few false-positive drift detections from the 32,450th time step to the 33,278th time step were given, causing CDCMS to have a drop in accuracy. For the last drift, CDCMS successfully identified the recurring concept and recovered a set of models corresponding to concept f4. This shows that the clustering in the model space strategy was able to distinguish two sets of models holding similar concepts, which echoes the analysis of the experiment with Sine2-gradual.

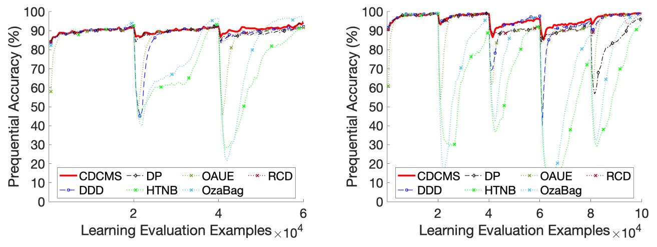
*2) Cases where CDCMS performed worse:* Table III shows that Sine1-gradual is the case where CDCMS performed significantly worse than at least one other existing approach.

Tables I and II show that Sine1-gradual consists of two extremely severe drifts where the second drift is a recurrent

TABLE III
AVERAGE PREQUENTIAL ACCURACY (STANDARD DEVIATION) ACROSS 30 RUNS FOR THE SYNTHETIC DATA STREAMS

| Data Stream | Synthetic Data Stream with **Gradual** Drifts | | | | | | | Synthetic Data Stream with **Abrupt** Drifts | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDCMS | HTNB | OzaBag | DP | OAUE | RCD | DDD | CDCMS | HTNB | OzaBag | DP | OAUE | RCD | DDD |
| Sine1 | 87.737% (6.109%) | 67.745% (22.178%) | 75.283% (23.252%) | **88.3% (5.958%)** | 85.92% (9.998%) | 87.806% (6.026%) | **87.786% (7.341%)** | **90.248% (1.586%)** | 74.35% (18.25%) | 78.58% (20.238%) | 89.374% (1.877%) | 86.846% (9.164%) | **89.527% (1.827%)** | 88.54% (6.961%) |
| Sine2 | **92.692% (7.666%)** | 65.555% (26.393%) | 74.39% (26.552%) | **91.689% (7.935%)** | 89.968% (12.008%) | 90.39% (8.643%) | **91.632% (9.236%)** | **95.894% (3.055%)** | 65.559% (27.081%) | 76.702% (25.508%) | 91.557% (7.497%) | 91.593% (9.918%) | **94.061% (4.032%)** | 93.755% (7.22%) |
| Agr1 | **89.53% (8.576%)** | 74.003% (11.758%) | 71.998% (12.03%) | 85.268% (10.152%) | **89.707% (9.193%)** | 85.634% (8.302%) | 87.293% (10.867%) | **90.31% (8.367%)** | 76.283% (12.768%) | 76.282% (10.802%) | 87.209% (8.58%) | **90.666% (8.733%)** | 86.706% (8.735%) | 88.805% (9.617%) |
| Agr2 | **78.494% (11.311%)** | 67.301% (12.327%) | 68.304% (12.175%) | 71.877% (9.767%) | **80.458% (9.153%)** | 73.954% (10.229%) | 74.09% (11.955%) | **79.528% (10.862%)** | 67.575% (12.765%) | 68.871% (12.383%) | 72.668% (10.878%) | **81.807% (8.609%)** | 74.711% (10.684%) | 75.038% (12.379%) |
| Agr3 | **82.895% (9.108%)** | 70.068% (8.178%) | 70.295% (8.731%) | 75.833% (7.616%) | **83.413% (9.116%)** | 78.58% (9.037%) | 77.759% (10.457%) | **83.888% (9.128%)** | 69.972% (8.402%) | 70.404% (8.892%) | 76.95% (8.428%) | **84.499% (8.921%)** | 77.986% (9.576%) | 79.789% (10.644%) |
| Agr4 | **81.187% (11.45%)** | 69.749% (14.208%) | 67.592% (14.072%) | 75.46% (12.894%) | **82.505% (10.119%)** | 77.783% (11.093%) | 77.647% (13.051%) | **80.775% (12.21%)** | 71.867% (15.6%) | 70.141% (14.376%) | 78.063% (12.118%) | **82.984% (10.584%)** | 77.679% (11.969%) | 78.324% (13.253%) |
| SEA1 | **87.591% (1.488%)** | 86.117% (2.239%) | 86.581% (2.777%) | 86.879% (1.709%) | 87.168% (2.505%) | 86.71% (1.855%) | **87.525% (1.652%)** | **87.569% (1.528%)** | 86.136% (2.467%) | 86.58% (2.974%) | 86.434% (1.893%) | **87.378% (2.551%)** | 86.527% (1.633%) | **87.538% (1.701%)** |
| SEA2 | **86.841% (1.652%)** | 85.036% (3.392%) | 85.733% (3.62%) | 86.296% (1.854%) | **86.713% (2.579%)** | 86.033% (1.965%) | **86.738% (1.964%)** | **86.935% (1.761%)** | 85.05% (3.605%) | 85.773% (3.856%) | 85.976% (2.514%) | **86.953% (2.667%)** | 85.846% (2.105%) | **86.884% (2.08%)** |
| STA1 | **98.268% (4.193%)** | 86.462% (14.325%) | 86.782% (13.774%) | **98.183% (4.306%)** | 96.898% (7.258%) | 97.783% (4.91%) | **98.132% (4.466%)** | **99.936% (0.223%)** | 90.324% (11.997%) | 90.142% (11.639%) | 99.904% (0.291%) | 98.129% (5.867%) | **99.945% (0.203%)** | 98.964% (3.343%) |
| STA2 | **98.155% (4.623%)** | 78.25% (16.809%) | 78.511% (16.635%) | **98.079% (4.65%)** | 96.842% (7.484%) | 97.713% (5.402%) | 97.949% (5.028%) | **99.953% (0.156%)** | 86.898% (15.764%) | 87.342% (16.241%) | 99.893% (0.3%) | 98.004% (6.281%) | **99.946% (0.161%)** | 99.024% (3.2%) |

Lime or light grey in the column of Data Stream means that synthetic data stream consists of recurring concepts. The p-values of Friedman tests are all <2.2E-16. Cells with bold text represent the best accuracy (or approaches have no significant difference with best one) in the data stream, based on the p-value ($\geq$ 0.05) in Nemeyi post-hoc test.



(a) Data Stream Sine1-abrupt (b) Data Stream Sine2-abrupt

Fig. 2. Prequential Accuracy with Synthetic Abrupt Drift Data Streams

drift. The best approaches for Sine1-gradual were DDD and DP. Figure 1(c) shows that DDD achieved the best recovery at the first drift but the magnitude of the improvement was relatively small. Even though CDCMS and DDD both have a highly diverse ensemble for prequential accuracy recovery, DDD enforces low diversity learning on the highly diverse ensemble after the drift, whereas CDCMS uses low diversity only in the representative ensemble. This potentially enabled DDD to recover better from the drift. The same figure also shows that DP achieved the best recovery from the second drift, which takes the stream to a concept that is the same as the first concept. Both DP and CDCMS successfully identified this recurrent drift. DP correctly recovered a model that purely learnt from the corresponding concept (f3) while CDCMS recovered a set of models that had been learning since the first stationary period of f3 and a set of models that had learnt during the transitional period of the first drift. The reason behind such a model recovery of CDCMS at this drift can be explained as follows.

CDCMS considered a false-positive drift detection during the transitional period of the first drift (f3$\rightarrow$f4) as a recurrent drift and then recovered a set of models corresponding to concept f3 and a few models created during the transitional period to handle it. As this set of mixed concepts models has learnt from the examples of this transitional period, the clustering in the model space strategy treated it as an intermediate concept between f3 and f4. As the transitional period of the second drift (f4$\rightarrow$f3) is also composed of these two concepts, CDCMS considered this drift as the recurrence of such intermediate concept, instead of the recurrence of f3. In other words, the clustering in the model space strategy precisely distinguished models with a significant difference in the underlying concept. While this distinction would normally be advantageous, it was detrimental because the new concept was f3, and no new drift detection was triggered when moving from the intermediate concept of f3-f4 to the pure f3 concept. Besides, we can again see that OzaBag performed better than all concept drift handling approaches at the end of the concepts, which also happened in Sine2-gradual.

*Summary: CDCMS usually achieved similar or better results than other approaches on synthetic gradual drift data streams, being more robust to and fast recover from multiple types of gradual drift (RQ2). This is because the clustering in the model space strategy managed to recover relevant past models to handle concept drifts, in particular, severe recurrent drifts and severe drifts taking to a concept that shares similarities with past concepts. However, in cases where there are extremely severe gradual recurrent drifts, CDCMS can retrieve models that learnt from intermediate concepts, slightly hindering the accuracy (RQ3).*

## D. Results with Synthetic Abrupt Drift Data Streams

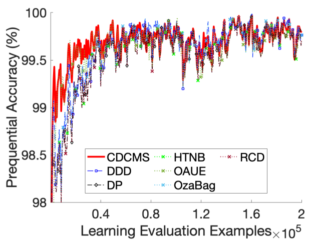
The right part of Table III shows that CDCMS achieved similar or better accuracy than other approaches in all abrupt drift data streams. Thus, the analysis done in this section will focus on why CDCMS performed significantly better than DP and DDD for Sine1-abrupt, given that it had performed significantly worse for Sine1-gradual. Sine2-abrupt has been chosen as an additional representative case.

Sine1-abrupt consists of two extremely severe drifts, where the second drift is a recurrent drift (see Tables I and II). Figure 2(a) shows that CDCMS was the best approach both in terms of reaction to drifts and recovery speed. At the first drift, CDCMS mistakenly considered it as a recurrent drift and recovered only one model which was created 200 time steps before the drift. Relative to other models in memory, this model did not learn much from the old concept. Thus, we can consider that the new representative ensemble consists of new models only, which enabled better adaptation to the new concept. A further investigation of the ensembles' weights right after this drift shows that the new representative ensemble obtained a weight of around 56% while the highly diverse ensemble helped to recover the accuracy, with the weight of around 38%. In contrast, the old representative ensemble performed poorly, having a very low weight of around 5.6%. As this drift is abrupt, it is reasonable that training new models is the best strategy [11]. CDCMS successfully benefited from that and recovered well from this drift.

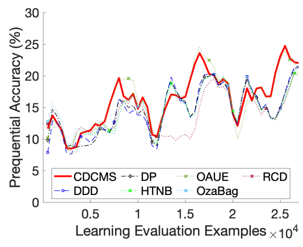
On the contrary, OAUE and DDD had a great drop in accuracy at the first drift. For OAUE, the main reason was that it is a passive approach, which usually has an adaptation lag in handling abrupt drifts. For DDD, it was because its highly diverse ensemble was not diverse enough to have a better generalisation in handling the drift, i.e., the $\lambda$ value suggested by the hyper-parameter tuning procedure was not small enough. Thus, the highly diverse ensemble did not perform well and could not draw a higher weight. This indirectly led DDD to put too much emphasis on the old representative ensemble, being unable to recover well from this drift. An extra experiment using a smaller $\lambda$ value showed that DDD could place a higher emphasis on the highly diverse ensemble, performing more similarly to CDCMS.

For the second drift, which is a recurrent drift, CDCMS was the best one in adapting to this change. This was because all the drifts in Sine1-abrupt are abrupt, resulting in no transitional period where intermediate concepts can be formed. Thus, CDCMS managed to identify the recurring concept and recover corresponding models more easily. Besides, its highly diverse ensemble also helped to prevent a great drop in accuracy right after the drift by participating in making predictions with a weight of 49% (the new representative ensemble weighted 46%). OAUE performed poorly for the same reason as the first drift, while DDD performed satisfactorily despite not being designed for handling recurring concepts. The results obtained for STA1-abrupt can be explained similarly to Sine1-abrupt.

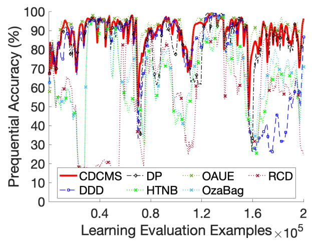
Sine2-abrupt is the representative case that shows CDCMS performed significantly better. Tables I and II show that all the concept drifts in Sine2 are severe. Even though the statistical test result did not capture any significant difference in the overall accuracy between CDCMS and DDD, Figure 2(b) shows that CDCMS started to perform considerably better after the second drift (around 41,000th time step). The details of CDCMS's behaviour reveal that it considered this drift as a recurrent drift. It then recovered a vast majority of models holding the knowledge of concept f1, despite the actual underlying concept being f3. CDCMS behaved similarly at the third drift. A vast majority of models holding the knowledge from f2 were recovered, despite the actual underlying concept being f4. This happened because the concepts of f1 and f3 are quite similar and the concepts of f2 and f4 are also quite similar (see Table II). CDCMS had not seen the concepts of f3 and f4 by the time it encountered them but managed to exploit relevant past knowledge and successfully recover its accuracy from these two drifts. This reflects CDCMS's ability in dealing with mild severity abrupt drifts.



(a) KDDcup99' (first 200k time steps)



(b) Power Supply



(c) Sensor (first 200k time steps)

Fig. 3. Prequential Accuracy with Real-World Data Streams

TABLE IV
AVERAGE PREQUENTIAL ACCURACY (STANDARD DEVIATION) ACROSS 30 RUNS FOR THE REAL WORLD DATA STREAMS

| Data Stream | CDCMS | HTNB | OzaBag | DP | OAUE | RCD | DDD |
|---|---|---|---|---|---|---|---|
| KDD Cup99 | **99.738% (0.191%)** | 99.65% (0.291%) | **99.728% (0.278%)** | 99.65% (0.291%) | 99.663% (1.073%) | 99.65% (0.291%) | 99.717% (0.277%) |
| Power Supply | **16.247% (4.115%)** | 14.833% (3.359%) | 14.907% (3.28%) | 14.637% (3.671%) | **16.237% (4.523%)** | 13.84% (3.226%) | 14.838% (3.503%) |
| Sensor | **89.38% (8.867%)** | 56.283% (17.35%) | 71.126% (15.96%) | 83.504% (16.628%) | **92.332% (7.306%)** | 53.402% (18.793%) | 85.838% (12.99%) |

The p-values of Friedman tests are all <2.2E-16. Cells with bold text represent the best accuracy (or approaches have no significant difference with best one) in the data stream, based on the p-value ($\geq$ 0.05) in Nemeyi post-hoc test.

*Summary: CDCMS always performed similar or better than other approaches on the abrupt drift data streams, indicating that it was more robust to and fast recover from multiple types of abrupt drift (RQ2). This is because the highly diverse ensemble, containing relevant knowledge to the concept after the drift, helped to prevent a great drop in accuracy. Besides, CDCMS handled abrupt drifts better than handling gradual drifts because it would not retrieve models that learnt from intermediate concepts (RQ3).*

## *E. Results with Real-World Data Streams*

Table IV shows that CDCMS performed similarly or significantly better than all other approaches on the real-world data streams. Due to the length of KDDcup99 and Sensor, Figure 3(a) and 3(c) only show their initial 200k time steps.

Despite Table IV showing that OzaBag is the most competitive against CDCMS, Figure 3(a) shows that the accuracy of OAUE is actually more competitive against CDCMS at the beginning of KDDcup99 stream. Figure 3(a) shows that CDCMS performed well and relatively more stable than other approaches at the beginning of the KDDcup99 stream. The standard deviation of CDCMS's accuracy shown in Table IV also supports this observation. The behaviour of CDCMS on this data stream was similar to its behaviour in the second and third drifts of Agr3-gradual. Thus, in principle, the accuracy of CDCMS and OAUE should have no significant difference in this data stream but Table IV shows that CDCMS performed significantly better. This may suggest that using prequential accuracy as done by CDCMS may be a better weighting or comparison metric in this data stream than using Mean Square Error as done by OAUE.

For Power Supply, Table IV shows that no significant difference has been found between the accuracies of CDCMS and OAUE. Figure 3(b) shows that the accuracies of CDCMS and OAUE are indeed quite similar throughout time. We believe that this is mainly because CDCMS and OAUE both keep updating their representative ensemble over time. When comparing against other approaches, the same Figure shows that there are periods of time when HTNB and OzaBag won against CDCMS by a small magnitude. However, when CDCMS won, it was by a large magnitude.

For Sensor, Table IV shows that CDCMS performed significantly better than other approaches and similar to OAUE. When looking closer at Figure 3(c), CDCMS could not achieve better average accuracy than OAUE because it had frequent drops in accuracy, caused by drift detection. Given the frequent drops in accuracy, these drift detections may have been false-positives. CDCMS can normally deal with false-positives by emphasising old models after drift detection. However, the large number of false-positives may have caused none of the existing models to represent the current concept well enough, causing the approach not to emphasise them.

## V. Experiments to Analyse the Time-Memory-Accuracy Relationship in CDCMS

This section extends the time complexity analysis of CDCMS presented in Section III-D with empirical results. CDCMS was evaluated prequentially with different hyper-parameter values that control its memory requirements and may affect its runtime, based on the five representative synthetic data streams that were analysed in Sections IV-C and IV-D: Sine1-(gradual, abrupt), Sine2-(gradual, abrupt), Agr3-gradual. We recorded the information about the average prequential accuracy, total runtime (CPU seconds) and model cost (RAM-hours) across thirty runs for each hyper-parameter combination per data stream. Each run was performed using the University of Birmingham's BlueBEAR HPC with 8 CPU cores and 8GB memory in a single computing node.

### A. Experimental Setup

The hyper-parameters of CDCMS that control its memory restrictions and may affect its runtime are the maximum total number of models that CDCMS can hold at a time ($3e + e \times r$) and the size of the most recent window of data ($b$). The options for these hyper-parameters were the same as those in Section IV-B, except that we selected a subset of ensemble size-memory size multiplier ($e$-$r$) combinations that lead to roughly equal intervals for the maximum total number of models. In summary, the $e$-$r$ combinations chosen to control the maximum total number of models were 5-4 (35), 5-10 (65), 10-6 (90), 10-8 (110), 10-10 (130), 15-8 (165), 15-10 (195), 20-8 (220), 20-10 (260), where the value in brackets is the resulting maximum number of models. Other non-memory related hyper-parameters were kept with the values previously selected through hyper-parameter tuning in Section IV-B

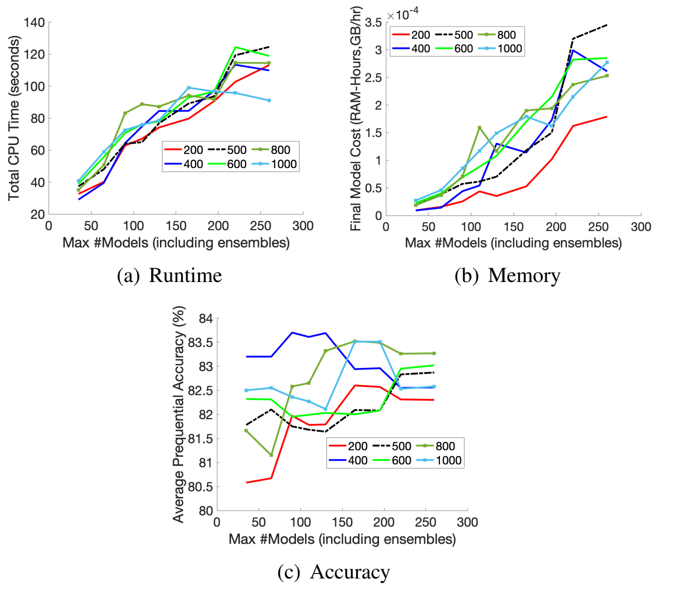


Fig. 4. Overall runtime, memory and accuracy of CDCMS for different maximum total numbers of models in Agr3-gradual. The different coloured lines represent different sliding window sizes.

### B. Results with Synthetic Data Streams

Figure 4 shows representative plots of the time, memory and accuracy obtained when restricting the maximum total number of models to different values. Other plots were omitted due to space restrictions, but present similar trends unless stated otherwise. Figures 4(a) and 4(b) show that increasing the maximum number of models ($3e + e \times r$) in general leads to an increase in CDCMS's memory cost (RAM-hours) and total runtime (CPU seconds). This is because CDCMS can hold and is required to process more information. However, the relationship between sliding window size ($b$), the memory cost and the total runtime was less clear, varying and depending on the data stream and the maximum number of models.

Despite the higher memory cost and runtime, increasing the maximum number of models was in general beneficial to the overall prequential accuracy (see Figure 4(c)). This could be because a larger memory can store more concepts, increasing the chances that various past knowledge is kept. So, it can better handle recurrent drifts and drifts leading to a concept that shares similarities with past concepts. Larger sliding window sizes did not necessarily lead to better overall prequential accuracy as expected, because different sliding window sizes will be better for different types and frequencies of drift.

In summary, the analysis above shows that increasing the maximum total number of models could slightly improve the prequential accuracy but the trade-offs are the increase in the memory cost and the total runtime.

## VI. Conclusion

Multiple types of concept drift could potentially coexist in real-world applications. However, existing data stream mining algorithms tend to be better at handling a small subset of types of drift, which may not be sufficient for real-world applications. This paper proposes a novel diversity framework called CDCMS that can handle multiple types of drift well. CDCMS creates highly diverse ensembles through clustering in the model space and uses a diversity-based memory management strategy (RQ1). CDCMS achieved similar or better prequential accuracy than existing approaches in both synthetic and real-world data streams, doing particularly well with abrupt and recurrent drifts in terms of accuracy recovery speed and the overall predictive performance (RQ2). The main reason behind is that CDCMS's novel clustering in the model space strategy and diversity-based memory management strategy were able to keep, identify and recover relevant past knowledge to handle concept drifts. The highly diverse ensemble created based on such strategies also helped to prevent great drops in accuracy after concept drifts (RQ3).

Future work includes the investigation on how to improve the accuracy of CDCMS on data streams with multiple extremely severe gradual drifts and the extension of this diversity framework to online class imbalanced learning.

## Acknowledgment

This work was supported by EPSRC Grant No. EP/R006660/2.

## References

[1] G. Ditzler, M. Roveri, C. Alippi and R. Polikar, "Learning in Nonstationary Environments: A Survey," *IEEE CIM*, vol. 10, pp. 12–25, 11 2015.

[2] G. Cabral, L. L. Minku, E. Shihab and S. Mujahid, "Class Imbalance Evolution and Verification Latency in Just-in-Time Software Defect Prediction," *ICSE*, pp. 666–676, 05 2019.

[3] L. L. Minku and S. Hou, "Clustering Dycom: An Online Cross-Company Software Effort Estimation Study," *PROMISE*, pp. 12–21, 11 2017.

[4] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," *IEEE TNNLS*, pp. 1–14, 09 2017.

[5] S. Delany, P. Cunningham, A. Tsymbal and L. Coyle, "A Case-Based Technique for Tracking Concept Drift in Spam Filtering," *Knowl. Based Syst.*, vol. 18, pp. 187–195, 08 2005.

[6] P. Kadlec, B. Gabrys and S. Strandt, "Data-Driven Soft Sensors in the Process Industry," *Comput. Chem. Eng.*, vol. 33, pp. 795–814, 04 2009.

[7] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski and M. Wozniak, "Ensemble Learning for Data Stream Analysis: A Survey," *Information Fusion*, vol. 37, p. 132–156, 09 2017.

[8] I. Žliobaitė, "Learning under Concept Drift: an Overview," 2010.

[9] L. L. Minku, A. White and X. Yao, "The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift," *IEEE TKDE*, vol. 22, pp. 730–742, 05 2010.

[10] C. W. Chiu and L. L. Minku, "Diversity-Based Pool of Models for Dealing with Recurring Concepts," *IJCNN*, pp. 2759–2766, 07 2018.

[11] L. L. Minku and X. Yao, "DDD: A New Ensemble Approach for Dealing with Concept Drift," *IEEE TKDE*, vol. 24, pp. 619 – 633, 05 2012.

[12] C. Alippi, G. Boracchi and M. Roveri, "Just-In-Time Ensemble of Classifiers," *IJCNN*, pp. 1–8, 06 2012.

[13] G. B. C. Alippi and M. Roveri, "Just-In-Time Classifiers for Recurrent Concepts," *IEEE TNNLS*, vol. 24, pp. 620–634, 04 2013.

[14] N. Sun, K. Tang, Z. Zhu and X. Yao, "Concept Drift Adaptation by Exploiting Historical Knowledge," *IEEE TNNLS*, 02 2017.

[15] J. Gama, P. Medas, G. Castillo and P. Rodrigues, "Learning with Drift Detection," *Intell. Data Anal.*, vol. 8, pp. 286–295, 09 2004.

[16] J. Gonçalves Jr and R. Barros, "RCD: A Recurring Concept Drift Framework," *Patt. Recognit. Lett.*, vol. 34, pp. 1018–1025, 07 2013.

[17] J. Z. Kolter and M. A. Maloof, "Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift," *IEEE ICDM*, pp. 123–130, 12 2003.

[18] D. Brzezinski and J. Stefanowski, "Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams," *Information Sciences*, vol. 265, p. 50–67, 05 2014.

[19] A. Bosch and A. Zisserman and X. Munoz, "Scene Classification Using a Hybrid Generative/Discriminative Approach," *IEEE TPAMI*, vol. 30, no. 4, pp. 712–727, 2008.

[20] K. H.Brodersen, T. M. Schofield, A. P. Leff, C. S. Ong, E. I. Lomakina, J. M. Buhmann, and K. E. Stephan, Klaas E., "Generative Embedding for Model-Based Classification of fMRI Data," *PLOS Computational Biology*, vol. 7, no. 6, pp. 1–19, 06 2011.

[21] R. K. T. Jebara and A. Howard, "Probability product kernels," *JMLR*, vol. 5, p. 819–844, 12 2004.

[22] H. Chen, F. Tang, P. Tino and X. Yao, "Model-Based Kernel for Efficient Time Series Analysis," in *ACM SIGKDD*, 2013, p. 392–400.

[23] H. Chen, P. Tino, A. Rodan and X. Yao, "Learning in the Model Space for Cognitive Fault Diagnosis," *IEEE TNNLS*, vol. 25, no. 1, pp. 124–136, 2014.

[24] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo and A. Bifet, "Early Drift Detection Method," in *IWKDDS*, 2006, pp. 77–86.

[25] A. Bifet and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing," *SIAM ICDM*, vol. 7, 04 2007.

[26] R. Barros, J. Gonzalez Hidalgo and D. Cabral, "Wilcoxon Rank Sum Test Drift Detector," *Neurocomputing*, vol. 275, pp. 1954–1963, 01 2018.

[27] T. Escovedo, A. Koshiyama, A. Abs da Cruz and M. Vellasco, "DetectA: Abrupt Concept Drift Detection in Non-stationary Environments," *Appl. Soft Comput.*, vol. 62, 10 2017.

[28] A. Pesaranghader and H. Viktor, "Fast Hoeffding Drift Detection Method for Evolving Data Streams," in *ECML PKDD*, 2016, pp. 96–111.

[29] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM algorithm," *J R Stat. Soc. B*, vol. 39, pp. 1–38, 01 1977.

[30] L. Kuncheva and C. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy," *Mach. Learn.*, vol. 51, pp. 181–207, 05 2003.

[31] R. Agarwal, T. Imielinski and A. Swami, "Database mining: A performance perspective," *IEEE TKDE*, vol. 5, no. 6, pp. 914–925, 1993.

[32] W. Street and Y. Kim, "A Streaming Ensemble Algorithm (SEA) for Large-scale Classification," in *ACM SIGKDD*, 2001, pp. 377–382.

[33] J. Schlimmer and R. Granger, "Incremental Learning from Noisy Data," *Mach. Learn.*, vol. 1, pp. 317–354, 09 1986.

[34] A. Bifet, G. Holmes, R. Kirkby and B. Pfahringer, "MOA: massive online analysis," *JMLR*, vol. 11, 05 2010.

[35] X. Zhu, "Stream Data Mining Repository," http://www.cse.fau.edu/~xqzhu/stream.html, accessed: 2019-05-30.

[36] P. Domingos and G. Hulten, "Mining High-Speed Data Streams," *ACM SIGKDD*, 11 2002.

[37] N. C. Oza, "Online Bagging and Boosting," *ICSMC*, vol. 3, pp. 2340–2345, 11 2005.

[38] J. Gama, R. Sebastião and P. Rodrigues, "On Evaluating Stream Learning Algorithms," *Mach. Learn.*, vol. 90, pp. 317–346, 10 2013.

**Chun Wai Chiu** received his BSc in Computer Science from the University of Leicester, UK, in 2017. He is currently working towards the Ph.D. degree in Computer Science at the University of Birmingham, UK. His research interests include data stream mining, online learning, concept drift, recurring concepts and class imbalanced learning.

**Leandro L. Minku** is a Lecturer in Intelligent Systems at the School of Computer Science, University of Birmingham, UK. Prior to that, he was a Lecturer in Computer Science at the University of Leicester, UK. He received the PhD degree in Computer Science from the University of Birmingham, UK, in 2010. Dr. Minku's main research interests are machine learning for non-stationary environments / data stream mining, class imbalance learning, ensembles of learning machines and computational intelligence for software engineering.