Class-Incremental Experience Replay for Continual Learning under Concept Drift

Łukasz Korycki
Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA

koryckil@vcu.edu

Bartosz Krawczyk
Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA

bkrawczyk@vcu.edu

Abstract

Modern machine learning systems need to be able to cope with constantly arriving and changing data. Two main areas of research dealing with such scenarios are continual learning and data stream mining. Continual learning focuses on accumulating knowledge and avoiding forgetting, assuming information once learned should be stored. Data stream mining focuses on adaptation to concept drift and discarding outdated information, assuming that only the most recent data is relevant. While these two areas are mainly being developed in separation, they offer complementary views on the problem of learning from dynamic data. There is a need for unifying them, by offering architectures capable of both learning and storing new information, as well as revisiting and adapting to changes in previously seen concepts. We propose a novel continual learning approach that can handle both tasks. Our experience replay method is fueled by a centroid-driven memory storing diverse instances of incrementally arriving classes. This is enhanced with a reactive subspace buffer that tracks concept drift occurrences in previously seen classes and adapts clusters accordingly. The proposed architecture is thus capable of both remembering valid and forgetting outdated information, offering a holistic framework for continual learning under concept drift.

1. Introduction

Contemporary real-world problems generate challenging and ever-growing data with dynamic properties. This kick-started exciting developments of novel machine learning algorithms capable of constant accumulation of new information [1], aggregating useful data [21], and handling its non-stationary properties [16]. Two fields are being developed in parallel – continual learning [18] and data stream mining [13]. The former focuses on how to retain useful

knowledge within the model, while allowing its growth and accumulation of new information. The latter focuses on adaptation to the current state of data, detecting the phenomenon known as concept drift, and swift adaptation to any changes taking place [16]. One must notice that those two approaches offer complementary views on the problem of continual learning from dynamic data and thus should be bridged together, leading us to develop robust and adaptive learning algorithms.

Research hypothesis. Class-incremental continual learning can be effectively extended to allow at the same time: (i) avoiding catastrophic forgetting by effectively accumulating knowledge from new classes; and (ii) monitoring changes in previously learned classes (revisiting) with automatic adaptation to concept drift.

Motivation. Existing continual learning methods assume that the once learned knowledge should be remembered and stored in the model. This assumes that information once learned stays permanently valid. This is not true, as modern dynamic data sources may be affected by concept drift, thus changing the properties of some classes, as depicted by the example of a binary recommendation system given in Fig. 1. This calls for developing methods that can bridge the gap between continual learning (knowledge retaining) and data stream mining (concept drift adaptation).

Overview. We propose a holistic approach to class-incremental continual learning, based on experience replay. The novelty of our work is that our algorithm allows for both avoiding catastrophic forgetting and automatic update of previously learned classes if they are affected by concept drift. We distinguish three vital aspects of the proposed framework: (i) capability for class-instrumental continual learning; (ii) capability for retaining useful knowledge to mitigate catastrophic forgetting; and (iii) capability of adaptation to changes by forgetting outdated knowledge and updating the model. Our approach combines centroid-driven
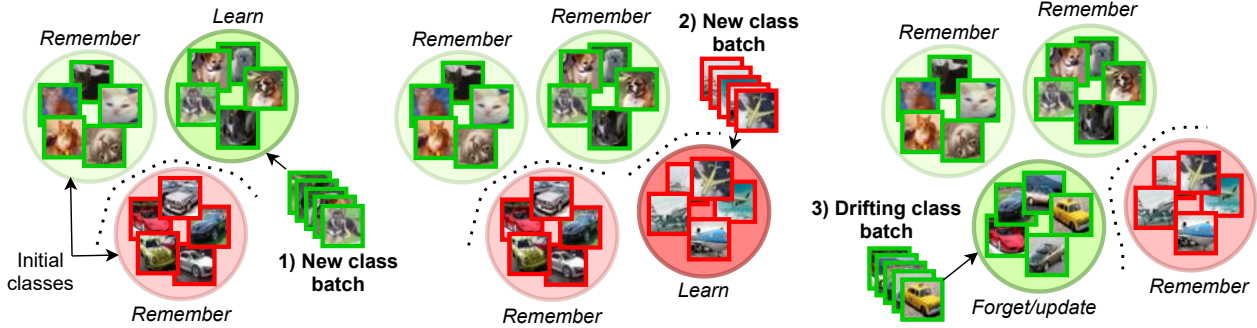
Figure 1: Three vital aspects of a holistic approach to continual learning: learning new classes, retaining previous knowledge, and adapting to concept drifts, illustrated by the example of a binary recommendation system (like or dislike).

memory for storing class-based prototypes with a reactive subspace buffer that can detect and react to concept drift affecting one or more classes. It traces the dominant class in each of the clusters, allowing for switching labels among clusters and splitting them whenever local changes are being detected. We simultaneously ensure the diversity of information stored within class buffers with their reactivity to concept drifts.

Main contributions. Our work offers the following advancements to the field of continual learning.

- **Combination of aggregation and adaptation.** We propose a unifying view on continual learning that combines avoiding catastrophic forgetting during class-incremental learning with adaptation to changes affecting previously learned classes.

- **Reactive subspace buffer.** We develop a novel experience replay approach that combines clustering-driven buffers for managing data diversity with cluster tracking, switching, and splitting for forgetting outdated information and automatic adaption to concept drift without a need for explicit change point detection.

- **Realistic continual learning scenario under concept drift.** We discuss a realistic and illustrative learning scenario – continual preference learning and recommendation. As both users may acquire new preferences and their old preferences may change over time, this problem touches both avoiding catastrophic forgetting in incremental learning and the need for handling concept drift.

2. Related works

Continual learning – robustness to forgetting. The field of continual learning focuses on incremental incorporation of new information into the model, while preserving the knowledge learned on previous classes or tasks [18]. This

leads to two challenges: how to grow model to make space for new knowledge and how to avoid catastrophic forgetting. Model growing is mainly discussed from the perspective of deep neural networks, where recent works suggest to either elastically add new neurons to the network [15], have a pretrained bigger structure that is incrementally populated [24], or use hypernetworks to control the accumulation of new data [26].

At the same time, while the model is accumulating new classes/tasks it may become biased towards this recent distribution [27]. Continual learning models must be robust to catastrophic forgetting, to maintain high accuracy in previous classes. Most common solutions to this problem use instance buffers from previous classes for experience replay [7], specialized parts of neural networks for solving each task while freezing the rest with masking [17], or use regularization and parametrization methods to debias the learning procedure [4, 11].

Data stream mining – robustness to changes. The field of data stream mining focuses on continuous adaptation to the newest incoming data, with the assumption that they are the most representative [13]. This is dictated by the non-stationary data characteristics and presence of concept drift that may affect class boundaries, distributions, and features [12, 16]. Concept drift can be handled in either an explicit or implicit manner. The former approach uses drift detectors – external monitoring tools that evaluate selected properties of incoming instances and/or learning models to signal the moment of a drift [5]. Once the drift is detected, the learning model is replaced with a new one trained on most recent data, thus facilitating forgetting of outdated concepts. The letter approach uses adaptive learning techniques, assuming that the underlying model will smoothly follow the changes in a data stream. Here sliding windows [8] and online classifiers [3] are the most popular solutions, allowing for learning from incoming instances, while simultaneously forgetting the old information.

3. Unifying continual learning with concept drift adaptation

Need for unifying approach to continual learning. Both discussed approaches to learning from dynamic data focus on different aspects and are being developed in relative separation. One should notice that both raise vital issues that are present in many dynamic real-world problems [28, 30]. This allows us to postulate the need for a unifying approach that will bridge those two domains and offer a holistic view on the continual learning paradigm [2]. The questions that all continual learning models should answer are (i) how to accumulate new information and expand the capabilities of a learning model; (ii) how to remember important and useful information over time; and (iii) how to detect and forget outdated concepts. This proposed unified approach will allow continual learning models to offer leveraged robustness to both catastrophic forgetting and concept drift, making them a step closer to solving emerging real-world problems.

Realistic scenario for continual learning under concept drift. Most works on concept drift use artificial data generators [21] or forcefully inject drift into every class without considering the reason behind it (e.g., *apples* changing into *cars*). Those examples are not very convincing and thus there is a need for finding a way of creating concept drift problems that are rooted in reality. We propose to focus on user's preferences, as they are an excellent example of continual and streaming data, as depicted in Fig. 1.

Users are constantly processing new information given to us from social media, the internet, or news outlets, learning about new things we have not seen before. Those new things may become interesting to a user or not – but they still need to be processed in a continuous manner, calling for class-incremental mechanisms. A new topic does not become the major or only interest for the user; thus it cannot overshadow the previously seen ones. Therefore, catastrophic forgetting must be avoided to retain not only the most current, but all topics relevant to a given user. At the same time, our preferences and tastes are not static. We change our interests within the span of years, months, or even days. A concept that was interesting to the user at a given point cannot be assumed as interesting indefinitely. A continual learning system must thus be able of revisiting previously learned knowledge and updating it according to any shifts in preferences. This calls for concept drift adaptation approaches, as previously seen topics may evolve over time and the interest of users in them may either increase or decrease over time. Creating a true continual learning system over user preferences is a real-world and practical illustration explaining the need for a holistic approach capable of remembering new concepts and selective forgetting with adaptation to changes in the old ones.

4. Class-incremental experience replay under concept drift

The prevalent majority of the class-incremental methods based on experience replay focus on storing the most representative instances or prototypes for stationary data [19]. They rely on the assumption that classes of the observed and selected instances cannot change, therefore there is no need to control them. As a result, the instances picked for a given class will remain in its buffer for a very long time and the only criterion which may trigger their removal or replacement will be representativeness or diversity of the memory [6]. However, in many real-world applications the mentioned assumption does not hold true. In the presented example of a binary recommendation system, the preferences may change, invalidating some of the experiences stored in the buffer. In such a case, we have to address the concept drift problem and update our memory adequately.

In the following sections, we introduce two commonly used basic algorithms – class buffers and centroid-driven memory – in the context of the given problem. We also propose an adaptive experience replay approach capable of adapting to concept changes.

4.1. Class buffers

Standard experience replay methods tackle the catastrophic forgetting problem by storing a separate buffer per class. They assume that a label of an incoming instance is known, so they can perfectly balance the storage and reasonably diversify their available memory. Due to practical concerns, the class buffers have limited capacity, therefore, there is the necessity of selecting which (or if) previously captured instances should be replaced with the currently incoming ones.

The most simplistic approaches use basic algorithms like FIFO (queue) effectively acting as sliding windows [22]. The problem with such methods is that they may very quickly erase the memory of earlier examples, that may be representative for a given problem, leading to catastrophic forgetting [20]. Assuming that incoming instances may be somehow correlated in time, one possible modification mitigating the issue is to enforce a wider spread of the stored instances across time. To achieve that, we can sample instances stored in the buffer, using a simple formula:

$$r \sim \mathcal{U}(0, 1) < \tau, \tag{1}$$

where r is a random variable sampled from the uniform distribution and τ is a threshold specified by a user. By increasing the threshold we can enforce quicker replacements, while, on the other hand, by decreasing it we can make the buffer more conservative. Too low τ will lead to impaired learning from new data, while too high τ will inevitably lead to catastrophic forgetting. Although balanced thresholds should be preferable for classic stationary scenarios,

such approaches will fail when concept drift occurs, imposing unnecessary avoidance of forgetting and impeding adaptation to changes.

4.2. Centroid-driven memory

Usually, the simple class buffer methods are too simplistic, since they do not utilize any significant characteristics of the incoming data. This is especially important, when we have to deal with complex or high-level abstract classes (e.g. in recommendation), since it is very likely there are several different subspaces that should be covered by the maintained buffers. This is where the clustering methods can be found very useful [25]. They are usually utilized to diversify the replay buffer by grouping instances into differing groups, which should result in a better coverage of the decision space [14]. The centroids can be used as instances themselves (prototypes) [9, 23], or solely as representations of buffers to forward new examples to their similar memory cells [29]. In this work, we focus on the latter approach.

Although the centroid-driven approaches are one step further than the simple class buffers, they are still susceptible to concept drifts. The reason for that is the fact that they very often do not check whether previously created clusters are still valid for a given class. If a given subconcept cluster changes its label (e.g., from liking to disliking), the new incoming instances will start (slowly) updating class centroids for another class. However, they will not affect the old cluster for the previous class, since the new instances will not be identified as those belonging to it, leaving it obsolete and impeding the learning process when one samples from it. This will, once again, lead to the opposite of catastrophic forgetting, resulting in much slower or non-existent adaptation to the current concepts.

4.3. Reactive subspace buffer

To address the presented problem, we propose a modification of the clustering-driven replay buffers, called **Reactive Subspace Buffer (RSB)**, capable not only of efficient knowledge aggregation, but also of adequate forgetting when it is needed. The outline of the algorithm is given in Algorithm 1. More details can be found in our public repository: github.com/lkorycki/rsb.

In the given algorithm, for each new instance x with a label y, we first ensure that there are at least c_{min} centroids for the class. Then, we find the nearest cluster C_x for the given instance x. If the given cluster belongs to the class y of the instance, we simply update it, its buffer B_x of maximum size b_{max} and sliding window W_x of maximum size ω_{max}, where the last component is responsible for tracking the most current concepts for the given centroid. Otherwise, there is a risk that a concept drift appeared and instances of a different class have started appearing around the centroid. Therefore, if the instance x is sufficiently close (we use sim-

---

Algorithm 1: Reactive Subspace Buffer (RSB).

Data: min centroids c_{max}, max centroids c_{max}, buffer size b_{max}, window size ω_{max}
Result: replay buffers B at every iteration
repeat
  receive incoming instance x and its label y;
  if $c_y < c_{min}$ **then**
    add new centroid C_{new}, buffer B_{new} and window W_{new} for y;
    continue;

  find the closest centroid C_x for x;
  if $y_{C_x} == y$ **then**
    update centroid C_x, buffer B_x and its window W_x with (x, y);
  else if *x is within C_x* **then**
    update window W_x with (x, y);
    if *should switch C_x* **then**
      move C_x to centroids of y and update it using W_x;
  else
    find the closest centroid $C_{y,x}$ for (x, y);
    if *x is within $C_{y,x}$ or $c_y \geq c_{max}$* **then**
      update centroid $C_{y,x}$, buffer $B_{y,x}$ and its window $W_{y,x}$ with (x, y);
    else
      add new centroid C_{new}, buffer B_{new} and window W_{new} for y;
  check for splits and removals
until *stream ends*;

---

ple standard deviation rules), we update the sliding window of the centroid C_x, but not the cluster itself. Now, if we detect that there is a significant number of instances with labels different from the current label of the centroid, we switch it to the new majority class. By doing so, we allow the buffer to quickly react to a potential drift. Otherwise, we find the closest centroid $C_{y,x}$ belonging to the same class y as x and we either update it, if x is sufficiently close to the cluster and the maximum number of clusters c_{max} has not been reached, or create a new centroid for the given class y.

Finally, for each centroid C, after every n_s-th update of its sliding window W, we check whether it did not switch labels but is impure enough to be split into two separate classes. We apply a simple formula checking if $c_1/c_2 - 1.0 < \tau_s$, where c_1 and c_2 are the first and second most numerous classes in the cluster and τ_s is a threshold determined by a user. During this step we also get rid of minuscule clusters for which less than $\tau_r = \alpha_r \omega_{max}$ instances were registered, where α_r is set by a user.

The whole algorithm is then used as a part of the experience replay method, in which we attempt to sample one instance for each centroid C from its buffer B, based on the purity criterion:

$$\gamma_C = \tanh(\beta \frac{c_1 - c_2}{c_1 + c_2}) > r \sim \mathcal{U}(0, 1), \qquad (2)$$

where c_1 and c_2 are, once again, the most numerous classes

in the cluster, and $\beta = 4$. By doing so, we provide an additional mechanism preventing us from enhancing outdated or at least uncertain concepts. Finally, since by using probabilistic sampling we make the total number of sampled instances non-deterministic, we apply oversampling to balance the selected batch.

To summarize – by enabling: (i) tracking the current dominant classes in a given cluster, (ii) switching labels between clusters, (iii) splitting them, and (iv) sampling from the replay buffer based on clusters purity, we make the centroid-driven algorithm sensitive to concept changes. At the same time, by maintaining stable replay buffers for subspaces that do not change, we can still avoid catastrophic forgetting. As a result, we are able to obtain a method capable of both remembering what is valid and forgetting what is outdated. In addition, since our method is based on local buffers, it should be able to efficiently diversify more complex concepts without explicit knowledge of its subconcepts.

5. Experimental study

In the experimental study, we attempt to prove that our algorithm is capable of class-incremental learning from stationary and non-stationary data. We aim to show that it can both (i) avoid catastrophic forgetting by maintaining diversified subspace-oriented replay buffers, and (ii) adapt to concept drifts by forgetting outdated information. All of the presented experiments can be conducted using scripts provided in the mentioned repository.

5.1. Data

To evaluate the proposed algorithm we decided to simulate a binary recommendation system by assigning superclasses (0/1) to the classes from original data sets. By doing so we could simulate the situation in which a user likes or dislikes certain types of available images (subconcepts). We constructed two types of class-incremental data sets: **stationary** and **drifting**.
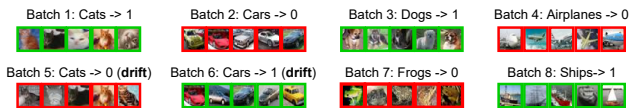

Figure 2: The general idea of the design for the drifting benchmark sequences.

For the former, we simply used five image benchmarks: MNIST, FASHION, SVHN, CIFAR10 and IMAGENET10, which is a subset of the 64x64 ImageNet set. During the evaluation we were feeding our models class after class, interleaving 0/1 assignments (for example, the first class from CIFAR10 was 1, the second one was 0, and so on). For the latter scenario, we were changing the 0/1 labels for two

consecutive classes after three or four stationary ones. As a result, we obtained 30 batches of classes for each data set, representing both stationary and concept drifting periods (for more details please refer to the repository). An example of our approach is depicted in Fig. 2.

5.2. Algorithms

We evaluated our algorithm as a part of the experience replay framework. The module consisted of a classifier (a neural net) and the replay buffer, which was used to sample additional instances for a given input batch. In order to compare our method (**RSB**) with other mentioned approaches, we run experiments using four additional classifiers: (i) offline neural network retraining after each batch (**OFFLINE**), (ii) a naively fine-tuning neural network (**NN**), which learned from the batches without any additional mechanisms for handling catastrophic forgetting, (iii) a simple class buffer (**CB**), which stored separate buffers for both recommendation classes, and a centroid-based method that utilized an on-line k-means algorithm to create representations of the original classes treated as subconcepts (or subspaces) of the recommendation space (**SB**).

While configuring our method, we used the following values of its parameters: $c_{min} = 0.5c_{max}$, where $c_{max} = 10$ for all of the data sets except for FASHION for which we set $c_{max} = 20$ based on preliminary experiments. Each buffer of the method could store at most $b_{max} = 100$ instances and equal was the size of each sliding window $\omega_{max} = 100$. We also empirically set $n_s = 1000$ and $\tau_s = 0.5$ for splitting, and $\alpha_r = 0.4$ for removals. These settings worked very well with all of the considered data sets. We used the same values of c_{max} and b_{max} for the SB algorithm. When it comes to the CB method, we set $b_{max} = 2000$ per class to provide similar memory resources compared with RSB and SB. Furthermore, we distinguished CB with $\tau = 0.0$ (**CB0**) and $\tau = 1.0$ (**CB1**) to check their performance in stationary and non-stationary scenarios.

All of the mentioned algorithms used pretrained convolutional feature extractors, from which we used representations returned by a middle layer of the classifier (we needed a high-level representation due to the nature of our task). For MNIST and FASHION we used a simple CNN with two convolutional layers consisting of 32 (5x5) and 64 (3x3) filters, interleaved with ReLU, batch normalization and max poolig (2x2). For SVHN, CIFAR10 and IMAGENET10 we utilized ResNet18. As a trainable classifier we chose a 3-layer fully connected net with 512, 256, 128 neurons in the hidden layers interleaved with ReLU, batch normalization and dropout ($p = 0.5$). During training we used the Adam optimizer. After each batch, the classifier learned for either 5 epochs (IMAGENET10) or 10 (the rest). Additionally, we initialized each algorithm with 10% of the first and the second class.

5.3. Evaluation

We evaluated the presented methods in a class-incremental setting, where each original class is treated as a subconcept of the binary recommendation space and comes as a whole in a form of a batch (Fig. 2). In our scenario, we assume that old subconcepts may become outdated and batches may change their labels. We measured the accuracy of a given algorithm after each batch (a new or updated class), utilizing holdout testing sets, and then, based on [10], used it to calculate the normalized average accuracy over the whole sequence:

$$\Omega_{all} = \frac{1}{T} \sum_{t=1}^{T} \frac{\alpha_t}{\alpha_{\text{offline},t}}, \qquad (3)$$

where α_t is the model performance after t classes and $\alpha_{\text{offline},t}$ is the optimal performance obtained by the offline learner.

To make our scenario more challenging, we assumed that we did not know the classes of the original set, only the recommendation labels. This allowed us to create a complex decision space without explicit knowledge of its subspaces and with a lot of potential for local concept drifts.

5.4. Results

Performance on stationary continual learning. Firstly, we evaluated the performance of RSB against the reference approaches in class-incremental continual learning with stationary properties. That means there was no concept drift present in the data and the main challenge lay in aggregating learned knowledge and avoiding catastrophic forgetting. We used this scenario first as an ablation study, to show that RSB is capable of learning newly arriving classes, without forgetting the previously seen ones. Tab. 1 shows the normalized average accuracy results over the five used benchmarks, while Fig. 3 depicts the changes in accuracy over time, calculated after each class (subconcept) batch. We omit the MNIST plot as it has identical characteristics as the FASHION plot.

In the presented results, we can see that all of the considered experience replay approaches were able to obtain satisfactory performance on the stationary sequences, slightly below the offline upper bound. They significantly improved upon the naive fine-tuning (NN), which severely suffered from catastrophic forgetting. The simple class buffers performed similarly on average. Holding instances of the earliest classes (CB0) turned out to be a bit better approach on simpler benchmarks, while giving a higher priority to the newer instances (CB1) resulted in higher accuracy on CIFAR10 and IMAGENET10. The more sophisticated centroid-driven experience replay (SB, RSB) provided even higher quality on all sequences by maintaining more diversified memory buffers per recommendation class. Finally,

the results indicate that our method is often capable of improving upon the simpler centroid-based method (SB), most likely by correcting partially inaccurate clusters.

Table 1: The normalized average accuracy (absolute values for the offline baseline) for stationary sequences.

Model	MNIST	FASH	SVHN	CIF10	IMGN10
OFFLINE	1.0	0.9865	1.0	1.0	1.0
NN	0.5529	0.5603	0.5529	0.5596	0.4886
ER-CB0	0.9537	0.9554	0.9414	0.9106	0.8828
ER-CB1	0.8754	0.8990	0.9235	0.9298	0.9349
ER-SB	0.9897	0.9739	0.9750	0.9675	**0.9513**
ER-RSB	**0.9967**	**0.9926**	**0.9816**	**0.9740**	0.9405

Table 2: The normalized average accuracy (absolute values for the offline baseline) for drifting sequences.

Model	MNIST	FASH	SVHN	CIF10	IMGN10
OFFLINE	1.0	0.9744	1.0	1.0	1.0
NN	0.5894	0.6043	0.5872	0.5884	0.4546
ER-CB0	0.5977	0.6473	0.5494	0.5635	0.6084
ER-CB1	0.7422	0.7931	0.7743	0.7918	0.8540
ER-SB	0.7268	0.7341	0.7267	0.7004	0.6696
ER-RSB	**0.9938**	**0.9745**	**0.9722**	**0.9545**	**0.9187**

In Fig. 3 we can clearly see that RSB displayed stable incremental learning capabilities and was not affected by catastrophic forgetting. This is especially visible on FASHION, SVHN and CIFAR10 data sets, where with the increasing number of classes reference methods displayed drops of performance, while RSB achieved stable results for all arriving classes. For SVHN, we can see that CB0 returned to similar performance as RSB after the 8-th class – but the intermediate learning process between classes no. 4 and 8 was significantly impaired. SB was much more resilient to forgetting, yet it performed slightly worse than RSB on 3 out of 4 sequences and on average. This allows us to conclude that RSB is robust to both catastrophic forgetting and false concept drift detection on stationary data.

Performance on continual learning under concept drift. After establishing that RSB displays robustness to catastrophic forgetting, we needed to evaluate its capability of simultaneous incremental learning and adaptation to drift. We used the same five benchmarks that now were injected with concept drift as discussed in Sec. 5.1. This way we should be able to see if RSB is able to detect changes on previously learned classes and correctly modify the underlying classifier to update its stored knowledge. Tab. 2 shows the normalized average accuracy results over five used benchmarks, while Fig. 4 depicts the changes in accuracy over time. Again, we omit MNIST plot as it has identical characteristics as the FASHION plot.
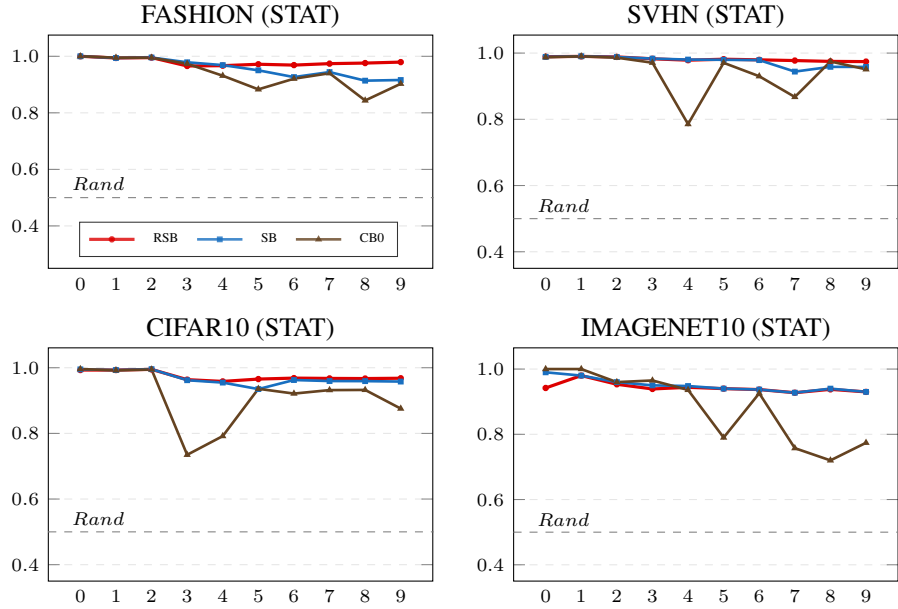
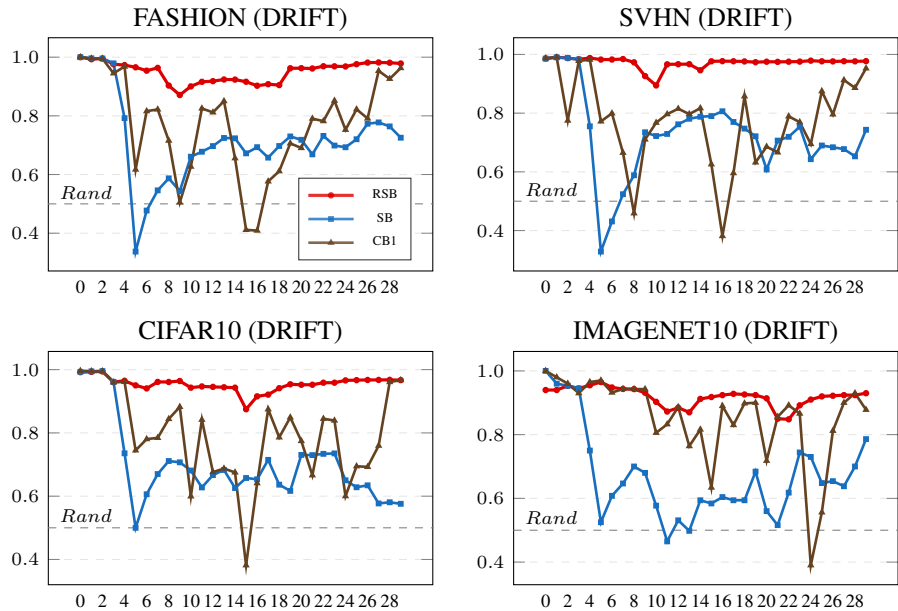Figure 3: Average accuracy over all classes for stationary class-incremental sequences.

Figure 4: Average accuracy over all classes for drifting class-incremental sequences. Drifts occur in batches 4, 5, 9, 10, 14, 15, 19, 20, 24 and 25.

For continual learning under concept drift we can see significant differences among the examined algorithms. Neither CB0, CB1 or SB were capable of keeping up with the presence of concept drift in the data. The main reason for that was the fact that CB0 and SB kept outdated instances in their buffers, impeding the adaptation process by forcing the model to retain obsolete concepts. On the

other hand, CB1 adapted to newer concepts much better than CB0, but it was not able to store instances for older classes, which inevitably led to catastrophic forgetting. For all five data sets the proposed RSB displayed the most stable performance, which is especially striking in the case of FASHION, SVHN and CIFAR10 sequences. By analyzing the plots we can see how the reference methods were sig-

nificantly impacted by the first occurrence of concept drift, often dropping to similar or lower performance levels as the random approach. Sometimes they were slowly recovering their performance over time, but this was happening at an unacceptable rate.

To gain further insights into the performance of the experience replay under concept drift let us look at Fig. 5 that depicts the accuracy over selected drifting classes. We can see that both CB1 and SB were highly sensitive to any drift in data. Even if sometimes they could spontaneously recover their performance (which usually was rather a coincidence), the next occurrence of concept drift could easily bring their performance back to the level of random decision (or even below). In the case of the MNIST class 0 we can see that the SB method could not recover at any point of time after the first drift. The extremely low accuracy was caused by obsolete centroids, which did not update their label and kept generating invalid instances for the recommendation class. These results clearly indicate that standard experience replay approaches cannot handle concept drifts, and that some of the occurring errors may even never be corrected. On the contrary, the proposed RSB is characterized by excellent robustness to concept drift, stable performance, and on-the-fly adaptation to changes in previously learned classes without any delay or loss in predictive power.
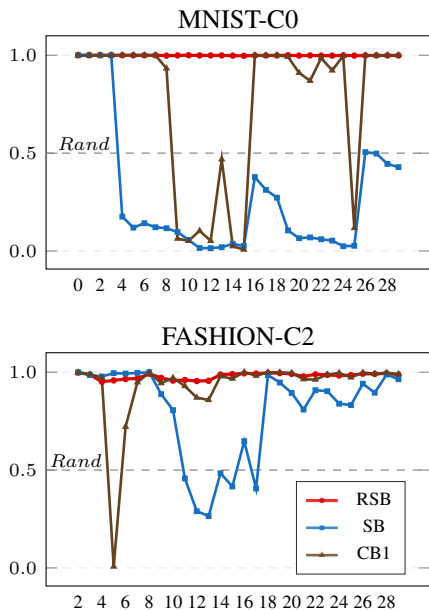

Figure 5: Accuracy for the selected classes under concept drift. C0 drifts in batches 4 and 5, and C2 drifts in 9 and 10.

Finally, we should be aware that concept drift may affect not only the performance of models on previously seen classes, but also their incremental learning capabilities. As the underlying neural network model tries to handle the

catastrophic forgetting by using instances from the buffer for experience replay, it utilizes instances coming from outdated concepts that may be contradicting the most current ones. Therefore, this may impact its ability to incorporate and retain new knowledge, resulting in a significant decrease in the model's predictive power. This allows us to conclude that continual learning under concept drift requires a strong interplay between avoiding catastrophic forgetting and adaptation to concept drift, as weaker performance on one will negatively affect the other. The proposed RSB offers an excellent balance between these two tasks, leading to a well-rounded and stable continual learning solution.

6. Conclusions and future works

Conclusions. In this paper, we have discussed a unified approach to continual learning that bridges the gap between avoiding catastrophic forgetting and data stream mining under concept drift. By pointing to the fact that these fields are two faces of the same coin, we showed that there is a need for developing holistic systems that are capable of incremental incorporation of new information, while offering adaptation capabilities by selective forgetting. This was illustrated by a practical example of continual learning of user's preferences that expand and evolve over time.

To address this challenging scenario, we have proposed an experience replay approach based on a reactive subspace buffer. It combines clustering-driven memory, storing diverse instances per class, with adaptation components that allow for dynamic monitoring, relabeling, and splitting of existing clusters. As a result, our method provides both the capability of accommodating new classes without catastrophic forgetting and the ability to react to concept drift affecting the previously learned classes. In our experimental study, we exhibited the effectiveness of our algorithm and proved that it is an effective and complete approach to continual learning that is not limited by either inability to accommodate new information, or by inability to adapt to changes.

Future works. We have shown that while existing standard experience replay approaches are able to handle the problem of avoiding catastrophic forgetting, they do not possess mechanisms allowing for adaptation on previously learned classes affected by concept drift. We suppose that similar issues can be identified in other continual learning algorithms. Therefore, our future works will focus on improving different approaches. This may involve, for example, introducing adaptive masking, reactive regularization and dynamic neural network structures capable of reacting to drifts. These will be important steps towards creating a holistic view of continual learning systems that can handle diverse challenges present in various real-life problems.

References

[1] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-Free Continual Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11254–11263. Computer Vision Foundation / IEEE, 2019. 1

[2] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. Online Fast Adaptation and Knowledge Accumulation (OSAKA): a New Approach to Continual Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16532–16545. Curran Associates, Inc., 2020. 3

[3] Alberto Cano and Bartosz Krawczyk. Kappa Updated Ensemble for drifting data stream mining. *Mach. Learn.*, 109(1):175–218, 2020. 2

[4] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating Forgetting in Online Continual Learning via Instance-Aware Parameterization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[5] Roberto Souto Maior de Barros and Silas Garrido Teixeira de Carvalho Santos. A large-scale comparison of concept drift detectors. *Inf. Sci.*, 451-452:348–370, 2018. 2

[6] Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1146–1155. PMLR, 2017. 3

[7] Scott Fujimoto, David Meger, and Doina Precup. An Equivalence between Loss Functions and Non-Uniform Sampling in Experience Replay. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[8] Xiangyang Gou, Long He, Yinda Zhang, Ke Wang, Xilai Liu, Tong Yang, Yi Wang, and Bin Cui. Sliding Sketches: A Framework using Time Zones for Data Stream Processing in Sliding Windows. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1015–1025. ACM, 2020. 2

[9] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory Efficient Experience Replay for Streaming Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776, 2019. 4

[10] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring Catastrophic Forgetting in Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. 6

[11] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 2

[12] Ł. Korycki and B. Krawczyk. Online Oversampling for Sparsely Labeled Imbalanced and Non-Stationary Data Streams. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. 2

[13] Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michal Wozniak. Ensemble learning for data stream analysis: A survey. *Inf. Fusion*, 37:132–156, 2017. 1, 2

[14] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David J. Crandall, and Dhruv Batra. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2119–2127, 2016. 4

[15] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3925–3934. PMLR, 2019. 2

[16] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.*, 31(12):2346–2363, 2019. 1, 2

[17] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2018. 2

[18] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 1, 2

[19] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 348–358, 2019. 3

[20] Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. Complementary Learning for Overcoming Catastrophic Forgetting Using Experience Replay. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial*

Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pages 3339–3345. ijcai.org, 2019. 3

[21] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. Online Deep Learning: Learning Deep Neural Networks on the Fly. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2660–2666. ijcai.org, 2018. 1, 3

[22] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 3

[23] Haobin Shi, Shike Yang, Kao-Shing Hwang, Jialin Chen, Mengkai Hu, and Heng-sheng Zhang. A Sample Aggregation Approach to Experiences Replay of Dyna-Q Learning. *IEEE Access*, 6:37173–37184, 2018. 4

[24] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. SpaceNet: Make Free Space For Continual Learning. *CoRR*, abs/2007.07617, 2020. 2

[25] Jérémie Sublime, Basarab Matei, and Pierre-Alexandre Murena. Analysis of the influence of diversity in collaborative and multi-view clustering. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4126–4133. IEEE, 2017. 4

[26] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2

[27] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large Scale Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[28] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic Drift Compensation for Class-Incremental Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6980–6989. IEEE, 2020. 3

[29] Mengmi Zhang, Tao Wang, Joo Hwee Lim, and Jiashi Feng. Prototype Reminding for Continual Learning. *CoRR*, abs/1905.09447, 2019. 4

[30] Shiyue Zhang and Mohit Bansal. Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2495–2509. Association for Computational Linguistics, 2019. 3