

This is the peer reviewed version of the following article:

Continual Learning in Real-Life Applications / Graffieti, G., Borghi, G., Maltoni, D.. - In: IEEE ROBOTICS AND AUTOMATION LETTERS. - ISSN 2377-3766. - 7:3(2022), pp. 6195-6202. [10.1109/LRA.2022.3167736]

*Terms of use:*

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

13/06/2026 21:27

(Article begins on next page)

# Continual Learning in Real-Life Applications

Gabriele Graffieti , Guido Borghi , and Davide Maltoni 

**Abstract**—Existing Continual Learning benchmarks only partially address the complexity of real-life applications, limiting the realism of learning agents. In this paper, we propose and focus on benchmarks characterized by common key elements of real-life scenarios, including temporally ordered streams as input data, strong correlation of samples in short time ranges, high data distribution drift over the long time frame, and heavy class unbalancing. Moreover, we enforce online training constraints such as the need for frequent model updates without the possibility of storing a large amount of past data or passing the dataset multiple times through the model. Besides, we introduce a novel hybrid approach based on Continual Learning, whose architectural elements and replay memory management proved to be useful and effective in the considered scenarios. The experimental validation carried out, including comparisons with existing methods and an ablation study, confirms the validity and the suitability of the proposed approach.

## I. INTRODUCTION

Recently, many efforts have been devoted to improving the realism of learning paradigms in *real-life* applications. Indeed, in the classical Machine Learning (ML) paradigm, the “life” of the learning agent is unrealistically divided into the distinct phases of *learning* and *deployment*: training data are supposed to be comprehensive of all the nuances of the real world and collected before starting the learning phase. On the other hand, in real-life applications, data are usually not available at the same time, and often many chunks of data are not fully representative of the whole data distribution.

To address this discrepancy, Continual Learning (CL) [1] strategies have been proposed to make agents able to continuously learn from newly acquired data, without (or with very limited) access to past data. Specifically, CL algorithms aim to contrast the so-called *catastrophic forgetting* problem [2], which results in a model that forgets previously acquired knowledge when it is trained on new or unseen data without access to past learning experiences.

In this paper, we propose to improve the level of realism of real-life learning agents, extending the above-mentioned considerations of the CL paradigm. Specifically, a real-life learning agent should sense the world through a continuous acquisition, *i.e.* a temporally ordered stream of input data, without the possibility of storing a large amount of data for an asynchronous and delayed learning procedure. Moreover, we argue that real-life learning scenarios should be characterized by an heavy class unbalancing. Since input data may not be independent and identically distributed (i.i.d.), the high

correlation of samples in short time ranges, and high data distribution drift over the long time frame, are prominent issues. This is the case, for instance, of a learning system placed on an autonomous vehicle that continuously learns relying on the streaming of images captured through a frontal camera placed on the top of the car roof. Presumably, training data contains an unbalanced amount of vehicle typologies, depending, for instance, by the road typology, that are visually similar in the same driving activity but can strongly drift on temporally different driving activities, especially in presence of light and weather changes. Another example may be a domestic robot that continuously learns through the interaction with everyday objects, whose number of samples is heavy unbalanced with respect to the most common items. Besides, frames belonging to a certain object appear similar during the interaction but they can visually vary between different interactions, depending on indoor or outdoor acquisition settings.

This paper investigates the suitability of Continual Learning algorithms to deal with the aforementioned challenging aspects of a real-life learning agent and proposes a benchmarks and a novel effective hybrid approach based on the Continual Learning paradigm. We start our experimental validation from the *Soda10M* [3] dataset, introduced in the recent organized *ICCV Self-supervised learning for next-generation industry-level Autonomous Driving* (SSLAD) competition [4] and acquired for the automotive field. Moreover, we propose and release a new benchmark based on the *CORE50* [5] dataset, specifically designed for the continual object recognition in the field of robotic vision from the egocentric view.

In both of the proposed benchmarks (see Figure 1), a learning agent has to learn classes from small non-IID *experiences* composed of a few consecutive highly-correlated frames. Experiences are then aggregated in *macro-experiences* characterized by similar conditions (*e.g.*, driving in the same city on a sunny day or acquiring images in the same setting). When switching from one macro-experience to another one, the conditions can completely change (*e.g.* city streets vs highways or different backgrounds) and the class knowledge needs to be properly adjusted without forgetting since old conditions are likely to return in the future. Furthermore, class-unbalancing heavily affects both experiences (close temporally correlated frames usually refer to a single or a few classes) and macro-experiences (*e.g.*, in urban scenarios cars and buses, are more frequent than trucks or tractors).

On the algorithm side, the proposed CL approach is designed to address the above issues, getting and expanding the intuitions of the algorithm that we submitted to SSLAD competition winning the first prize. We also highlight aspects to be considered to set up an efficient solution, a desired feature in real-life applications potentially running on embedded boards with limited hardware resources.

Manuscript received: February, 4, 2022; Accepted March, 28, 2022.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers’ comments.

The authors are with the Department of Computer Science and Engineering, University of Bologna, Italy [gabriele.graffieti@unibo.it](mailto:gabriele.graffieti@unibo.it), [guido.borghi@unibo.it](mailto:guido.borghi@unibo.it), [davide.maltoni@unibo.it](mailto:davide.maltoni@unibo.it)

Digital Object Identifier (DOI): see top of this page.

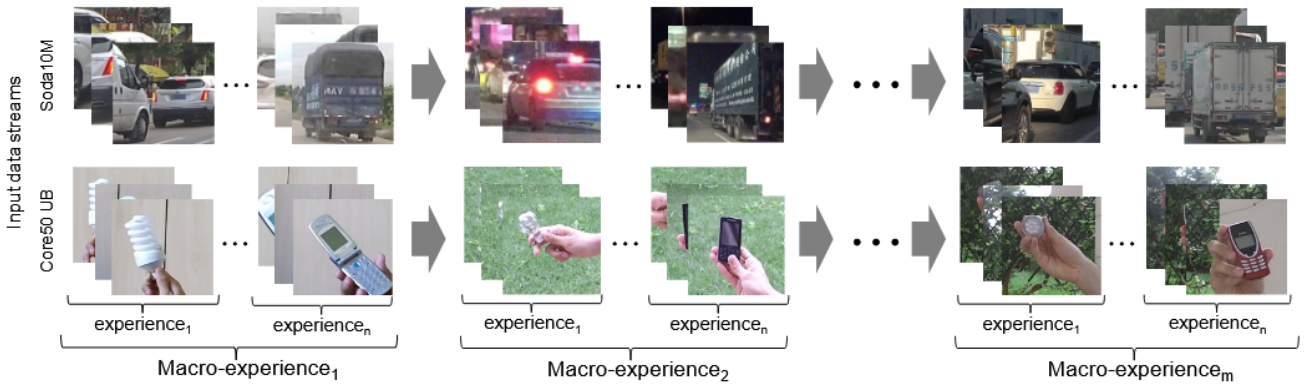


Fig. 1. A number of macro-experiences are identified along the input stream of both datasets. In each macro-experience, the learning phase is organized into several small experiences, whose frames belong to one or more classes and are highly correlated. These conditions can abruptly change across different macro-experiences, due to, for instance, the transition between day and night (*Soda10M*) or the change of backgrounds and lighting (*CORE50 UB*).

## II. RELATED WORK

### A. Continual Learning Strategies

For the sake of clarity, we adopt the three-way categorization adopted in the Continual Learning field, introduced in [1], [6], to describe the strategies considered in our experiments.

The first category is represented by the *architectural strategies*, based on specific architectures, layers, and other architectural elements to contrast the forgetting. Notable architectural strategies are PNNs [7] and CWR [6]. The second category groups the *regularization strategies*, including weight sparsification, early stopping, dropout, and the use of particular loss functions to contrast the weights overwriting. Notably, both EWC [8] and SI [9] exploit a quadratic regularization that aims to preserve the weights essential to the performance on old data. Finally, the third category contains the *rehearsal strategies* [10], mainly based on the storage of a subset of past data that are periodically replayed to the learning model.

The boundaries among these categories are not rigid, since some hybrid methods can be categorized at the intersection of the aforementioned categories. This is the case of iCaRL [11], where a regularization based on distillation is combined with a specific selection procedure to store in the replay memory the most important samples for old experiences. Also the GEM [12] and the improved version A-GEM [13] approaches combine the use of a fixed memory and regularization constraints to improve the accuracy on past experiences while contrasting the forgetting.

In this paper, we propose a *hybrid* method that belongs both to the architectural and replay categories. In particular, we investigate how the replay memory can be used to contrast the unbalanced input data stream and data drifts.

Only a few works specifically address a real-life unbalanced continual learning scenario. In [14], authors propose an online strategy based on a specific memory management based on a class-balancing reservoir sampling and a weighted random replay. Unfortunately, the method is tested on a set of well-known but simple datasets with simulated imbalances and no temporal coherence across different samples. Hayes *et al.* in [15] presents a method to maintain prototype buffers by storing an incoming feature and merging it with others in the

buffer if full. The main limitations of this method are that it only updates the final fully connected layer of a deep neural network and does not achieve real-time performance with a small batch size and considerable buffers dimensions. The same authors propose SLDA [16], in which they exploit the *streaming linear discriminant analysis* algorithm effectively used in the data mining research field, to compute the final prediction from feature vectors previously extracted through a pretrained ResNet-18 model. The work which is the most similar to the proposed approach is PRS [17], whose authors specifically investigate intra- and inter-task imbalances, using two multi-label classification datasets adapted for the CL scenario. Similarly to [14], the method is based on a *Partitioning Reservoir Sampling* (PRS) that allows the model to maintain a balanced knowledge of both well-represented and underrepresented classes. However, PRS does not take into account the temporal coherence of the data, and only focuses on the unbalance problem.

### B. Continual Learning Scenarios

Several CL scenarios have been proposed in the literature, mainly focusing on balanced supervised classification [5], [18], [19]. Following the nomenclature introduced in [6], CL scenarios can be divided into two main categories. *Multi-Task* category, requires to learn different disjoint tasks, and the task label, *i.e.* the id of the specific task is available both during training and evaluation. Differently, in the *Single-Incremental-Task* scenario only a single task is present, but the learning experience is incremental. This is the case where new classes or new instances of known classes are discovered during learning experiences. No task information is provided during evaluation, so the model has to classify the data among all the classes seen in the past. This scenario can be further split into three settings: the first one is the *Domain Incremental* [20] (Domain-IL) in which new instances of the same classes become available during the training experiences [18]. The second is the *Class-incremental learning* (Class-IL) [20] where new classes become available during the training procedure [9]. The last setting is a mixed version of the previous ones and it is referred to as *Class-Incremental with Repetitions*,

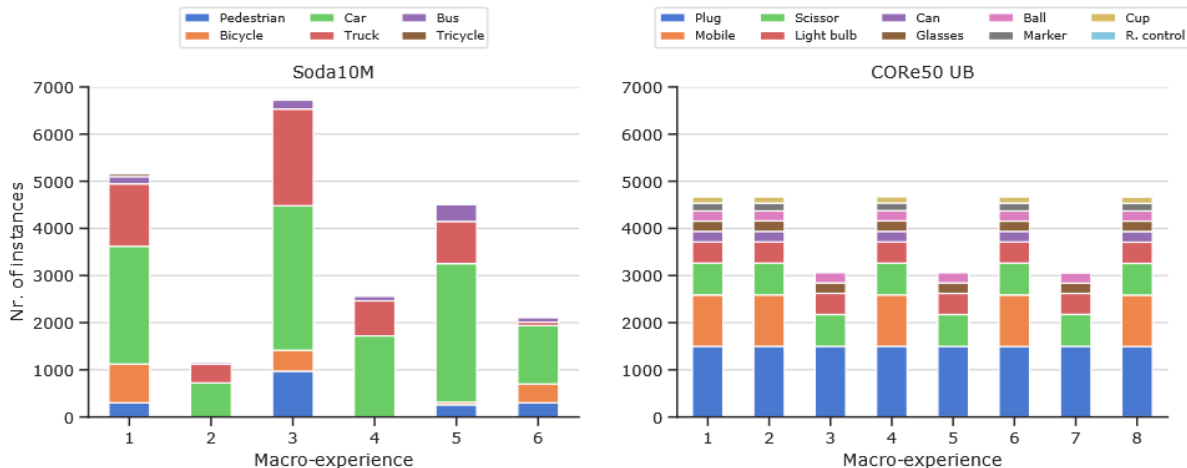


Fig. 2. Data distribution for each macro-experience for Soda10M and CORE50 UB datasets. As shown, both datasets are characterized by unbalanced distributions, with respect to data and classes available in each macro-experience. Specifically, Soda10M is focused on a great variety of sample availability while CORE50 UB focuses on classification challenges, consisting of more classes (10 instead of 6).

in which both new training patterns and classes are available in learning experiences [5].

Most of the current CL approaches are focused on Task-IL or Class-IL scenarios [20] while the Domain-IL scenario with a single task and all classes known is usually considered a simpler problem. The benchmarks considered in this paper belong to the Domain-IL scenario because all the classes are known since the beginning. However, some difference exists with respect to the conventions used by other researchers, since we do not assume that all the classes are present in each learning experience. The heavy class unbalancing, the high correlation of samples and the high distribution drift over the long time frame, make the considered scenarios quite challenging for a real-life agent, especially if online learning is enforced (*i.e.*, frequent model updates, limited storage, etc.).

### III. PROPOSED BENCHMARKS

#### A. Soda10M dataset

Soda10M [3] is a large-scale dataset collected for self-supervising learning in autonomous driving. The original frames are collected (1 every 10 seconds) from the front camera of a vehicle driven across 32 Asiatic cities under different conditions. A supervised classification problem is set up by considering the following 6 classes: pedestrian, bicycle, car, truck, tram, and tricycle. Manually defined bounding boxes allow cropping the objects of interest leading to a total of 22k object-centered and chronologically ordered images, which are rescaled to  $64 \times 64$  pixels. This dataset is challenging since many objects are occluded, even by other objects belonging to different classes. In addition, objects that are far from the camera may appear with a very low resolution. Finally, images captured at night and/or under bad weather conditions are often difficult to recognize. The unbalanced data distribution of the dataset across all training experiences is depicted in Figure 2: the class “car” represents the large majority of available samples, followed by the class “trucks”

and “pedestrian”, while the class “tricycle” is largely under-represented (and almost not visible in Fig. 2). Each macro-experience contains scenes acquired in different parts of the day and night, different road types, and weather conditions.

#### B. CORE50 dataset

CORE50 [5] consists of a total of 50 different objects belonging to 10 categories, acquired in 11 distinct sessions (8 indoor and 3 outdoor) with different lighting and backgrounds. The total amount of frame is about 164k (11 session  $\times$  50 objects  $\times$   $\sim$ 300 frames per session), and rescaled to  $128 \times 128$  pixels. In our experimental validation, we consider categories as classes, resulting in a total of 10 classes. Following the original paper, 3 sessions are used for the testing phase while the remaining 8 are used for the training procedure. Similar to the previous dataset, CORE50 presents several challenges, such as the low level of data variability during the same experience, since the same object is presented, through a single experience, without varying backgrounds and light sources. The original CORE50 dataset is highly balanced, and in the native Domain-IL benchmark, all the experiences have almost the same number of images of each class.

In this paper, we propose and release a new Domain-IL benchmark, referred to as CORE50 UB, whose experiences and macro-experiences enforce unbalancing and distribution drift. The data distribution is depicted in Figure 2. In this benchmark, the data amount across all macro-experiences is more stable while the number of classes is greater w.r.t. Soda10M dataset. Moreover, the number of patterns, *i.e.* data points, of a certain class, if present, is constant in all the macro-experiences. Finally, each macro-experience contains frames acquired in different settings (indoor and outdoor) and light conditions. For the sake of reproducibility, we release the configuration files<sup>1</sup> for recreating the dataset starting from the original CORE50.

<sup>1</sup>[https://miatbiolab.csr.unibo.it/?page\\_id=716](https://miatbiolab.csr.unibo.it/?page_id=716)

### C. Benchmark Settings

The settings and constraints adopted for the proposed benchmarks reflect the desired features to improve the realism of real-life continual learning agents described in Section. I.

- **Unbalanced instance distribution:** as shown in Figure 2, the amount of available training data is heavily unbalanced w.r.t. the available classes in each dataset. The most represented classes have a share of 55% and the 37% in Soda10M and CORE50 UB datasets, respectively, while the least represented class of both datasets contains about 1% of training data. For instance, in the Soda10M dataset, the class “tricycle” has only 85 instances in total, while the more common class “car” has about 15k samples.
- **Experience:** an experience is a chunk of data that the continual learning algorithm can accumulate before triggering a model update. Data from previous experiences are not available, except for a size-limited replay memory (see below). Following the SSLAD competition rules, each experience consists of just 10 patterns.
- **Macro-experience:** is an arbitrary long stream of data acquired in the same conditions, for instance, a single road trip, or a single manipulation of one or more objects by a domestic robot. Data from the same macro-experience presents similar characteristics, while data from different macro-experiences may greatly differ from each other (e.g., different weather conditions, different environments, different backgrounds, etc.). It is important to note that not all the classes are available in each macro-experience, as visible in Figure 2. Differently from Soda10M, CORE50 UB tends to preserve the total amount of data available in each macro experience (around 4, 5k and 3k samples when all classes are present or not, respectively) while there is a high variance in the number of instances belonging to different classes in the same macro-experience. The number of macro-experiences is 6 for Soda10M and 8 for CORE50 UB.
- **Memory constraints:** due to the streaming nature of the benchmarks, specific constraints are applied to the memory. First, the selection of patterns to be included or removed from the replay memory need to be performed at experience level. Moreover, patterns cannot be revisited within longer time frames. Second, the number of patterns in the macro-experiences and their class distribution are not known in advance (but it is known when a macro-experience starts or ends). Finally, in order to reduce hardware requirements, a global limit is enforced for the total replay memory storage, including any temporary storage. The limit is 1000 patterns for Soda10M ( $\sim 4.5\%$  of total data) and 1650 ( $\sim 5\%$ ) for CORE50 UB due to the higher number of classes of the latter.

## IV. PROPOSED METHOD

### A. Head Protection

The “learning in isolation” issue, which is one of the main causes of the catastrophic forgetting [21], [22], arises when only a limited number of classes is present in each experience, or when some classes are underrepresented, such as in the

proposed benchmarks. Several researchers argued that the forgetting is mostly localized in the classification head, *i.e.* the last fully connected layer of our adopted neural network [22].

The proposed method addresses this issue by protecting the classification head with the *Copy Weight with Reinit* (CWR) algorithm firstly proposed in [6]. CWR maintains two sets of weights for the classification layer, namely *Temporary Weights* ( $tw$ ) and *Consolidated Weights* ( $cw$ ):

- $tw$ : contains the weights used to train the model in the current experience. The weights are initialized to 0 at the beginning of the experience for the classes never seen before and loaded from  $cw$  for the already known classes.
- $cw$ : contains the weights from the previous experience that are used in the consolidation phase. In this phase, the  $cw$  weights are merged with the weights of the current experience  $tw$ , weighting the contribution based on the number of examples of each class seen in the past and current experiences.

Differently from the original paper [23], we do not freeze the weights of the feature extractor. Indeed, the forgetting in feature extractor is here mitigated by the replay memory and the natural replay implicit in a Domain-IL scenario.

### B. Replay Memory Management

Replay is known to be one of the most effective ways to contrast the catastrophic forgetting problem [10], [24], especially in complex continual learning scenarios. When combined with CWR it provides a very robust CL approach [23]. The proposed method is based on a per-class memory balancing and two-levels memory management specifically designed to address the relevant distribution drifts induced by macro-experiences. In particular, replay memory updates are triggered only at the end of a macro-experience while the selection of patterns (that will be later used for the update) is performed with higher granularity at the end of each experience.

To this purpose, we split the replay memory into two parts: the first one (60% of the total size) is the main replay memory  $\mathcal{M}$  and is used to provide replay data during the current macro-experience; the second one (remaining 40%), denoted as temporary replay memory  $\mathcal{T}$ , is used to store some of the original patterns from the current macro-experience that will be used in the future. In other words, reading and storing are performed separately on the two memories without interferences: in this manner, the main memory is not contaminated with data from the current macro-experience, and only contains past data.

The main memory  $\mathcal{M}$  is divided into  $N$  buffers  $\mathcal{M}_i$  of equal size, where  $N$  is the total number of classes. This results in a size of 100 samples for each class with both datasets. We experimentally observed that reducing the number of classes memorized or putting only some classes in memory, did not produce any particular benefit on both validation datasets. We also noticed that setting a different buffer size for each class (including, for instance, a larger number of samples) does not produce appreciable improvements. This is probably due to the following reasons: i) the total number of instances belonging to the most underrepresented class available in the training set is lower than the class buffer’s size; ii) it is important to limit

the number of replay samples for the most represented classes in the input. Analogously to the main memory, the temporary memory  $\mathcal{T}$  is divided into  $N$  buffers  $\mathcal{T}_i$ , but here the buffer sizes are not fixed and decrease across the macro-experiences. In particular, if  $m$  is the index of the current macro-experience, each class buffer  $\mathcal{T}_i$  has size  $100/m$ .

The memory management is performed according to the Algorithm 1. The main memory  $\mathcal{M}$  is used to draw samples for replay (line 4), while the storage takes place on  $\mathcal{T}$  (procedure *SaveReplayPatterns*). At the end of a macro-experience  $m > 1$ , for each class, the samples from  $\mathcal{T}_i$  replace (random) samples in  $\mathcal{M}_i$  (procedure *Replace* in the code). The temporary memory  $\mathcal{T}$  is then emptied. For  $m = 1$  the patterns are directly saved in the main memory  $\mathcal{M}$  since replay is not used in the first macro-experience. In the memorization procedure, the patterns are inserted in the  $\mathcal{T}_i$  buffers according to the procedure based on Reservoir Sampling [25], detailed in line 13. In particular, if a buffer for a specific class is not full and the current experience includes samples of the corresponding class, all the available samples are memorized in the buffer. When a buffer is full, the new candidates are memorized according to the reservoir sampling algorithm [25]. This requires maintaining a class occurrence vector  $\mathcal{C}$  at macro-experience level. Reservoir sampling guarantees that every class sample in a macro-experience has the same probability to be stored in the corresponding buffer at the end of each macro-experience.

### C. Model and Training Procedure

As a feature extractor backbone, we use a ResNet-50 [26] model, pretrained on the ImageNet dataset [27]. A last fully connected layer is added with the proper number of neurons, *i.e.* 6 and 10 for Soda10M and CORE50 UB datasets, respectively. The input of the model is first defined as a tensor of size  $(b, c, w, h)$ , where  $b = 10$  (batch size),  $c = 3$  (RGB image channels), and  $w = h = 64$ , in which  $w$  and  $h$  are the width and the height of the image, as introduced in Section III. Then, the width and the height of the tensor are resized through the *Nearest Neighbour* interpolation, from  $64 \times 64$  to  $224 \times 224$  to resemble the original image size of the ImageNet dataset on which the model has been pretrained. Even though the appearance of the patches may be degraded by this operation (the original size of the input patches is about  $1/12$  of the final size), we note that the convolutional kernels of the model respond better in resized images achieving improved performance. For the training phase across all the experiences, we use the *Stochastic Gradient Descent* (SGD) optimizer with a learning rate of  $10^{-2}$  without momentum and weight decay. As loss function, we use the *Cross Entropy* loss. For each macro-experience (except the first one, where replay is not used), we split each experience (whose size is 10 samples) into two mini-batches: each mini-batch also includes 10 samples, but 5 come from the experience and 5 are randomly sampled (without replacement) from the main memory  $\mathcal{M}$ .

Additionally, an “on-the-fly” data augmentation technique is applied, in order to increase the variety of the input tensors. We recall that, due to the streaming characteristics of the

---

### Algorithm 1 Memory Management

---

```

1: for each macro-experience  $m$  do
2:   RESET  $\mathcal{C}$ 
3:   for each experience  $e$  do
4:     TRAIN the model on  $e$  using replay memory  $\mathcal{M}$ 
5:     if  $m = 1$  then
6:       SAVEDREPLAYPATTERNS( $e, \mathcal{M}, \mathcal{C}, 100/m$ )
7:     else
8:       SAVEDREPLAYPATTERNS( $e, \mathcal{T}, \mathcal{C}, 100/m$ )
9:     if  $m > 1$  then
10:      REPLACE( $\mathcal{M}, \mathcal{T}, 100/m$ )
11:    RESET  $\mathcal{T}$ 
12:
13: procedure SAVEDREPLAYPATTERNS( $e, \mathcal{B}, \mathcal{C}, mb$ )
14:   for  $d, c$  in experience  $e$  do  $\triangleright d = \text{data}, c = \text{class}$ 
15:      $\mathcal{C}_c = \mathcal{C}_c + 1$ 
16:     if  $\mathcal{C}_c \leq mb$  then
17:       INSERT  $d$  in  $\mathcal{B}_c$ 
18:     else
19:        $j = \text{RANDOMINT}(1, \mathcal{C}_c)$ 
20:       if  $j \leq mb$  then
21:          $\mathcal{B}_c[j] \leftarrow d$ 

```

---

benchmarks, the input variety is highly limited, since many similar images of the same objects are provided in the same experience. Therefore, before passing the tensor to the model, horizontal flipping is applied and both the original and the flipped version are concatenated in the same mini-batch.

Finally, only during the first macro-experience, the resulting mini-batch is put in temporary memory (of size 10) and passed twice through the model. After each forward step, the loss and the gradients are computed, and then the optimization procedure is applied. This operation is motivated by the observation that, in the first macro-experience, the learning of the model can be boosted by seeing a larger number of (even similar) samples.

## V. EXPERIMENTAL EVALUATION

### A. Adopted Metrics

To assess the performance of the tested solutions, we rely on two metrics. The first one is the *Average Mean Class Accuracy* (AMCA) defined as:

$$AMCA = \frac{1}{N_c + N_{me}} \sum_{c=1}^{N_c} \sum_{e=1}^{N_{me}} \frac{corr_e^c}{tot_e^c}, \quad (1)$$

where  $N_c$  and  $N_{me}$  are the number of classes and macro-experience, respectively, while  $corr_e^c$  and  $tot_e^c$  are the number of correctly classified test images of class  $c$  on macro-experience  $e$ , and the total number of test images of class  $c$  in macro-experience  $e$ , respectively. The AMCA metric is useful to understand the learning capabilities and improvements across the macro-experiences of the tested model: indeed, the mean class accuracy is computed after each training macro-experience and then averaged to get the final results. Therefore,  $N_{me} = 6$  and  $N_c = 6$  for the Soda10M dataset, while  $N_{me} = 8$  and  $N_c = 10$  for CORE50 UB. We note that this metrics especially rewards solutions with stable incremental learning performances across all the macro-experiences, able to limit the effects of the catastrophic forgetting, even in

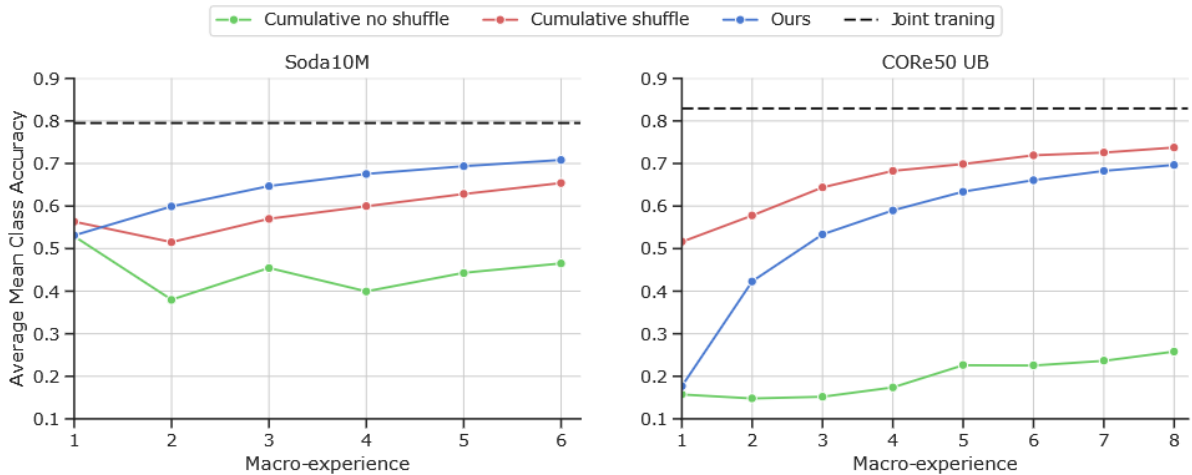


Fig. 3. Investigation about the complexity of the proposed benchmarks. The “joint training” algorithm, which is not a Continual Learning solution, represents the upper bound and it is reported with the dashed line. The “cumulative” algorithm reveals the importance of the shuffle operation on training data.

presence of highly unbalanced data. Moreover, since all the classes are weighted equally, a low performance of the model on underrepresented classes greatly affect the AMCA, making the metric particularly suitable to assess the performance on unbalanced datasets.

The second metric used, here referred to as top-1 accuracy, is calculated as the ratio between the test example classified correctly under the total number of test examples. The top-1 is different from the AMCA since the former weight each example the same, while the latter weight each class the same. The top-1, in fact, focuses more on the top performance of the model instead of the contrast to forgetting and the learning of the underrepresented data.

We test the model at the end of each training macro experience, using the validation set available in the competition for Soda10M and the full test set on CORE50 UB. In both cases, the test sets are fixed, and they contain all the classes present in the training set, even if some classes are not present in the current macro-experience, making the control of incremental learning statistics easier [6].

### B. Assessing the problem complexity

First, we propose to assess the overall complexity of the benchmarks proposed, comparing also a common machine learning approaches against our proposal. Experimental results are reported in Figure 3 for both Soda10M and CORE50 UB datasets. Specifically, we implement a strategy, referred to as *joint training*, which reflects the standard machine learning paradigm, where all the training data are available at the same time, making the model able to iterate through them many times (epochs) after shuffling the data. The motivation behind this experiment is to understand the theoretical performance of a deep learning-based solution in ideal conditions: this represents the upper bound solution and it is depicted with a dashed line in Figure 3. Experimental results reveal that it is possible to achieve a high AMCA ( $\sim 83\%$  on CORE50 UB and  $\sim 80\%$  on Soda10M) despite the data unbalancing.

In Figure 3 is also reported the accuracy of a test (referred as *cumulative*) that maintains the streaming nature of the benchmarks, while supposing infinite memory to store all the past data. In this way, experiences are composed of 10 images and one image can be seen only once, as stated in the proposed setting. Despite the availability of unlimited memory, the high similarity of data in each experience negatively affects the performance on both benchmarks.

To further investigate the previous findings, we set up a similar solution (referred to as *cumulative shuffle*), in which the whole data stream is shuffled so any single experience is composed of nearly IID data. A gain of about 50% on CORE50 UB and about 20% on Soda10M w.r.t. the cumulative strategy confirms that a strong correlation of samples in short time ranges poses a significant challenge.

Interestingly, our proposal overcomes the cumulative strategy on Soda10M: this can be explained by the distribution of classes in the first and second experience (see Fig. 2). The data merge of the first two macro-experiences produces an even more unbalanced data distribution, leading to a drop in accuracy of the cumulative strategies. Our solution, based on a balanced memory, helps to effectively mitigate this issue.

### C. Comparison with Continual Learning approaches

The second stage of the experimental validation is the comparison between our approach and other CL algorithms available in the literature. For each competitor, we followed the implementation details of the original publications or the provided source code. Moreover, if an algorithm does not explicitly rely on replay memory, we extend it to use an additional replay memory, reporting both results.

For all the competitors, we use the same hyper-parameters (e.g. optimizer, learning rate, etc.) and, following constraints introduced in Section III, we trained the model by performing the update after every experience, without any data shuffling.

The results are shown in Figure 4: on both Soda10M and CORE50 UB, our approach outperforms all the other

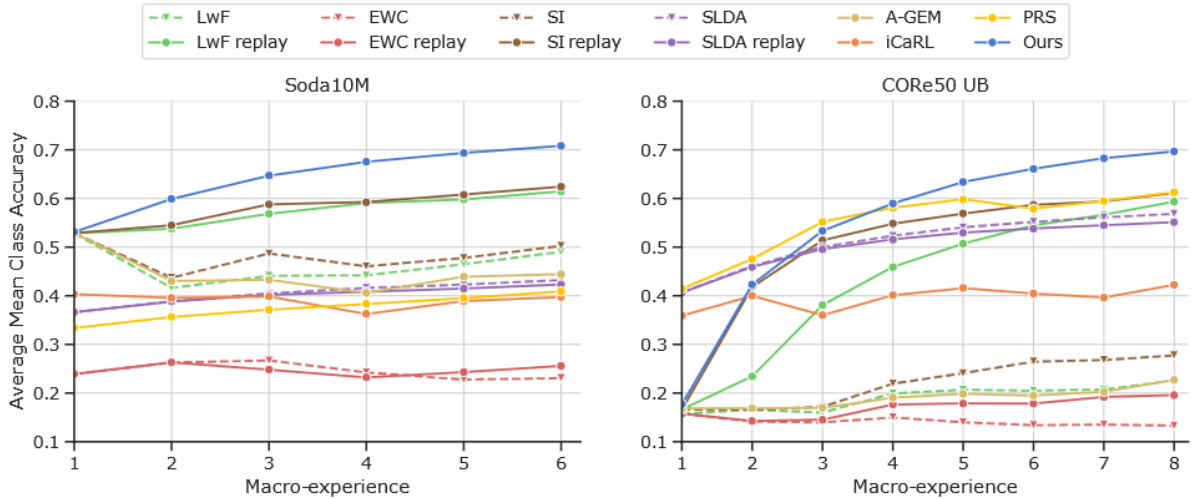


Fig. 4. Comparison between the proposed approach and other Continual Learning algorithms available in the literature obtained on both datasets (Soda10M and CORE50 UB). As shown, our approach largely overcomes existing solutions applied on unbalanced data distribution.

continual learning strategies by a large margin ( $\sim 10\%$  in both the experiments), while the best performance among the competitors are obtained by LwF and SI algorithms with the use of the replay memory. These competitors are able to learn incrementally, while natively rehearsal-based algorithms, such as A-GEM and iCaRL, do not perform well in this scenario. It is important to note that we adapt A-GEM to work also in the Single Incremental Task (SIT) scenario since the strategy was originally developed for the Multi-Task setting. We also adapted iCaRL to work in a Domain-IL scenario, instead of the original Class-IL scenario. We note that the use of the memory in iCaRL and A-GEM is not effective for the proposed unbalanced scenario, despite iCaRL always occupies all the storage available, while A-GEM incrementally fills the memory. PRS, which is specifically designed for unbalanced continual learning, shows good performance on CORE50 UB, starting better than any other competitor, but losing performance as the training proceeds. In Soda10M performs poorly: probably, the reason is that PRS directly stores samples into the main memory, unlike our approach in which temporary buffers are used, and its memory achieves a balanced partitioning only after some macro-experiences. SLDA performs fairly on both the datasets, despite it is not gradient-based and the model is updated after each example.

From Figure 4 is noticeable how the two benchmarks focus on two different aspects: on CORE50 UB, the AMCA after the first experience is less than 20% for the majority of the strategies, while in Soda10M, the most competitive strategies started from about 55%. This discrepancy can be explained by the fact that: i) Soda10M classes are more similar to ImageNet ones, used to pretrain the models on which our approach and the competitors are based; ii) the classification task is harder on CORE50 UB, since it contains more classes than Soda10M, also considering that in the first macro-experience the total amount of input data is similar.

In Figure 5 is reported the performance in terms of final AMCA versus training time, averaged on macro-experiences

TABLE I  
AMCA AND TOP-1 ACCURACY OF THE PROPOSED METHOD.

Method	Soda10M		CORE50 UB	
	Top-1	AMCA	Top-1	AMCA
<i>Ours (Baseline)</i>	89.88	70.84	79.47	69.65
Frozen feature extractor	73.88	47.90	67.24	57.47
No horizontal flip	88.97	68.83	73.33	65.40
Adam optimizer	74.15	33.81	26.13	18.23
No replay memory	81.59	56.62	46.68	26.61
No balanced memory	88.06	66.01	76.51	63.71
No temporary memory	86.00	69.25	79.73	67.28

calculated on Soda10M. As shown, our method combines high accuracy with a limited computational load.

#### D. Ablation Study

Finally, we conducted an ablation study in order to investigate the contribution of the main choices made in the implementation of the proposed method.

We identified three main directions of investigation: model, replay memory, and training procedure. Experimental results are shown in Table I in terms of AMCA and top-1 accuracy for both datasets. For the model, we show that the “freezing” of the feature extractor, as originally proposed in [6], limits the performance of the proposed approach by a large margin. On the training procedure side, we note that the data augmentation based on the horizontal flip, and the SGD optimizer probably represent an effective solution for our scenario. Specifically, we observe that the use of an adaptive learning rate algorithm, such as Adam [28], totally disrupts the learning dynamic, leading to very low performance. To further investigate this point we run the top four competitors (LwF with replay, SI with replay, A-GEM, and iCaRL) using the Adam optimizer. For Soda10M the final AMCA drops to 26.7, 26.6, 20.3, and 16.2 for LwF, SI, A-GEM, and iCaRL respectively, while for CORE50 UB it drops to 15.6, 12.6, 9.8, and 11.7 for the same methods.

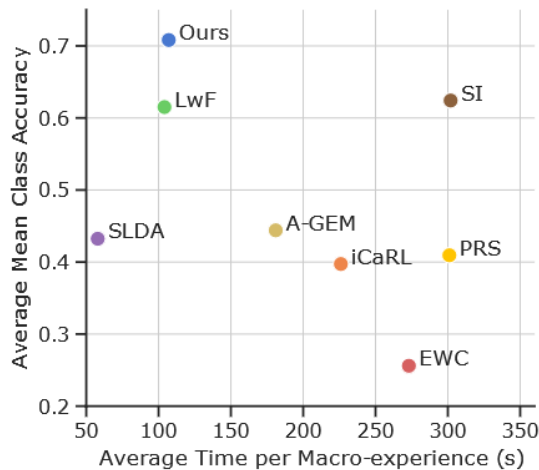


Fig. 5. Comparison of CL strategies with replay w.r.t. AMCA on average training time per macro-experience. Top-left corner denotes best performance.

Then, we conduct some experiments on the use of the replay memory. As generally reported in the literature [29], our results confirm that the replay memory plays a crucial role in order to increase the performance of the whole system even in presence of unbalanced data. Moreover, we observe that in our heavy unbalanced benchmarks with similar samples within an experience and strong data drift between different macro-experiences, a balancing memory allocation mechanism is needed, since using the same amount of memory without balancing leads to a loss of several points in both AMCA and top-1 accuracy. With the last line of Table I, we show how the use of the temporary replay memory  $\mathcal{T}$  leads to better AMCA performance in comparison with the same system without these additional buffers.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an hybrid approach based on Continual Learning applied to a real-life scenario in which data arrive in a streaming manner and are characterized by a strong correlation of samples in short time ranges, and high data distribution drift over the long time frame. Besides, a new benchmark based on the CORE50 dataset has been introduced and exploited, in combination with Soda10M, to analyze the performance of the proposed method. A variety of future work can be planned to further improve the realism in the proposed scenario, including the acquisition of additional benchmarks with more macro-experiences, a larger number of classes, and an extended temporal acquisition range.

## REFERENCES

- [1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.
- [2] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.
- [3] J. Han, X. Liang, H. Xu, K. Chen, L. Hong, C. Ye, W. Zhang, Z. Li, C. Xu, and X. Liang, "Soda10m: Towards large-scale object detection benchmark for autonomous driving," *arXiv preprint*, 2021.
- [4] "Iccv sslad competition," <https://sslad2021.github.io/>.
- [5] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 17–26.
- [6] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, vol. 116, pp. 56–73, 2019.
- [7] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.
- [8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, 2017.
- [9] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.
- [10] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *arXiv preprint*, 2021.
- [11] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *IEEE conference on Computer Vision and Pattern Recognition*, 2017.
- [12] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *International Conference on Learning Representations*, 2018.
- [13] A. Chaudhry, R. Marc'Aurelio, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR, 2019.
- [14] A. Chrysakis and M.-F. Moens, "Online continual learning from imbalanced data," in *International Conference on Machine Learning*, 2020.
- [15] 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)*. IEEE, 2019, pp. 9769–9776.
- [16] T. L. Hayes and C. Kanan, "Lifelong machine learning with deep streaming linear discriminant analysis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020.
- [17] C. D. Kim, J. Jeong, and G. Kim, "Imbalanced continual learning with partitioning reservoir sampling," in *European Conference on Computer Vision*. Springer, 2020, pp. 411–428.
- [18] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, F. Qiao, and R. H. M. Chan, "Openloris-object: A robotic vision dataset and benchmark for lifelong deep learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4767–4773.
- [19] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Are we done with object recognition? the icub robot's perspective," *Robotics and Autonomous Systems*, vol. 112, pp. 260–281, 2019.
- [20] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.
- [21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [22] E. Belouadah, A. Popescu, and I. Kanellos, "A comprehensive study of class incremental learning algorithms for visual tasks," *Neural Networks*, vol. 135, pp. 38–54, 2021.
- [23] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," in *IEEE/RSSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [24] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," *Nature communications*, vol. 11, no. 1, pp. 1–14, 2020.
- [25] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, p. 37–57, Mar. 1985.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.
- [29] G. I. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization," *Frontiers in neurobotics*, vol. 12, p. 78, 2018.