Applied Intelligence
https://doi.org/10.1007/s10489-024-05493-z

A rehearsal framework for computational efficiency in online continual learning

Charalampos Davalas[1] · **Dimitrios Michail**[1] · **Christos Diou**[1] · **Iraklis Varlamis**[1] · **Konstantinos Tserpes**[1]

Accepted: 29 April 2024

Abstract
In the realm of online continual learning, models are expected to adapt to an ever-changing environment. One of the most persistent hurdles in this adaptation is the mitigation of a phenomenon called "Catastrophic Forgetting" (CF). This critical condition occurs when models trained on non-identically distributed data lose performance in previously learned tasks. Rehearsal methods, leveraging the ability to replay older samples, aim to address this challenge by incorporating a buffer of past training samples. However, the absence of known task boundaries complicates the adaptation of current CF mitigation methods. This paper proposes a method attuned to data stream characteristics and online model performance in a resource-constrained environment. The number of training iterations and learning rate emerges as crucial hyperparameters, impacting the efficacy and efficiency of online continual learning. Up to this point, we propose a combination of Experience Replay methodologies, a Drift Detector, and various training convergence policies, specially tailored for scenarios with unknown task boundaries. Experimental results demonstrate the effectiveness of our approach, maintaining or enhancing performance compared to baseline methods, while significantly improving computational efficiency.

Keywords Catastrophic forgetting · Continual learning · Online learning · Image classification experience replay · Rehearsal methods

1 Introduction

Continual learning is a rapidly evolving field of machine-learning research, with several applications such as lifelong learning for robotic vision [1] and multi-task model learning [2, 3].

Continual learning methods aim to mitigate *catastrophic forgetting* [4] which occurs when training data drawn from different distributions are incrementally presented to the model , essentially violating the i.i.d. assumption [5]. Its immediate effect is the decrease of model effectiveness on previously learned tasks, as the model learns with the newly presented training data. The same issue has also been known as the "stability-plasticity dilemma" [6] for machine learning models [7], which refers to the trade-off between adapting a model's parameters to new information and maintaining its effectiveness on previously seen samples.

✉ Charalampos Davalas
cdavalas@hua.gr

[1] Department of Informatics and Telematics, Harokopio University of Athens, Athens 17778, Greece

Continual learning remains a challenging problem for deep neural networks and has attracted significant research interest recently [7], with methods addressing several variations of learning scenarios, depending on the information available to the learning algorithm. Some of the methods have been proposed to address these issues under the assumption that (i) the model learns a sequence of well-defined, disjoint tasks and (ii) the task boundaries are known [8]. This is usually simulated by splitting a dataset into smaller, disjoint subsets (e.g., corresponding to different classes, or different appearances of the same classes) [3]. Other methods aim to address the more challenging online continual learning problem [9], where training data arrive as a stream, without well-defined tasks or with unknown task boundaries.

Online continual learning introduces additional challenges. For a specific task, the model does not have access to all the task data before training and cannot plan the training process. In the case of a continuum of data, the types of tasks and the task boundaries are unknown. Furthermore, when operating in constrained resource setups (e.g. in edge computing and critical application environments) there is a need to keep the computational complexity low, which in turn

Published online: 13 May 2024

imposes significant memory and computational constraints [10].

In this work, as a part of recent advances such as [11], we propose a set of training strategies for rehearsal-based online continual learning, aiming to improve computational efficiency without compromising model effectiveness. These strategies involve decisions on ***when*** to train, as stream training data become available, as well as on ***how*** to train, in terms of the number of required training iterations and learning rate.

This work expands significantly our initial work in the field [11] by testing all baselines and method combinations with additional datasets such as the permutations of the MNIST digits dataset [12]. In this work, also the CIFAR10 dataset [13] is used with different buffer sizes as an extension of the experiments. Additionally, extended hyperparameter and heuristic rules testing are included in the Ablation Study (Section 6). These additions provide a more transparent view of each component and its contribution to the final results.

Experiments on image classification tasks demonstrate the importance of selecting an appropriate training strategy in online continual learning scenarios. Results show that the proposed methods achieve higher classification accuracy with lower computational complexity compared to simpler baseline rehearsal strategies, such as fixed-iteration training.

The structure of the paper is as follows: Section 2 describes the online continual learning setting and the implications that may arise when samples arrive at a stream with unknown boundaries between tasks. It surveys the main rehearsal methods and their computational limitations and summarizes the contributions of our work. Section 3 describes the online continual learning setting used in this work and gives an algorithmic description of our suggested framework. Section 4 details the heuristics that help our method to determine *how* to train the model *when* rehearsal is needed. Section 5 illustrates the experimental setup and our results, whereas Section 6 performs an ablation study demonstrating the value of individual heuristics and strategies described in Section 4. Finally, Section 7 summarizes the main findings and discusses the future steps of our work.

2 Related work

Continual learning approaches can be broadly categorized into *regularization*, *rehearsal (or replay)*, and *parameter isolation* methods [7]. Regularization methods, use a custom loss function with a regularization term that helps avoid catastrophic forgetting when learning with new data. Replay methods store examples or generate synthetic ones in order to train the model using a mixture of old and new samples, or use them to constrain the optimization. Finally, parameter isolation methods dedicate different parts of the neural network to different tasks. Consequently, they try to support all tasks by either dynamically adjusting the architecture or by re-adjusting the per-task parameters [2].

Rehearsal methods might require memory for buffer storage unlike other methods, such as Elastic Weight Consolidation (EWC) [14] or Learning-Without-Forgetting (LwF). The work of [15] summarizes and tests all known Rehearsal-Free methods in that area. An issue with EWC is that recomputing the Fisher Information Matrix can become computationally demanding. Calculation of the Fisher Diagonal requires all the model weights and in an online training setup this can become prohibitive. Another important issue is that the model distillation proposed in the baselines of [15] requires additional methods for providing a decent result [7]. In most works such as [16] Rehearsal-Free methods are best used as additional components for increasing classification accuracy. These additions often result in more computations and memory. In the case of model distillation, an extra copy of the model will be stored and used for additional model inferences in each training step.

Rehearsal methods are promising candidates for real-life incremental learning scenarios, such as *online continual learning*, since they have relatively simple implementations [7, 17, 18] which can be adapted in terms of computational and memory demands. Their effectiveness, however, is largely affected by two distinguishing factors: i) the size of the buffer that stores training samples from previous tasks, and ii) the strategy of mixing old and new task samples during rehearsal.

One of the most popular replay-based methods is *iCaRL* by Rebuffi et al. [19]. iCaRL is a class incremental method for image classification without forgetting. It is based on the use of a set of image samples that is dynamically updated to include the samples nearest to the class mean in the learned representation space. This set is used both for rehearsal and classification via the Nearest Class Mean (NCM) method. This dynamic buffer is updated via a combination of herding selection of new exemplars and priority-list-based removal of less representative exemplars. iCaRL has been created for solving the class incremental learning problem, where the task identities are unknown, but is not specifically designed for online learning.

Experience Replay (ER) [20] is another approach that is based on the concept of mixing old and new task samples in order to mitigate catastrophic forgetting. ER, which has been successfully employed in reinforcement learning and supervised learning tasks [21], maintains a dynamic buffer that is updated at every time step. Samples are randomly picked by the buffer and mixed with the new samples that arrive in a stream to create a synthetic batch, which is then

used for training in the next iteration. Experiments show that even a small number of samples from the buffer can make a significant difference to mitigate forgetting.

The Greedy Sampler and Dumb Learner *GDUMB* [17] method is the online equivalent of experience replay, meaning that it can be adapted on a more stream-based approach a simple rehearsal strategy and buffer update that focuses on the greedy sampling of new knowledge and is a proof of the simplicity of memory-based approaches. The methods proposed in this paper, follow a similar principle in terms of adapting to stream-based learning of new data.

Similarly, the *Gradient Episodic Memory* (GEM) method [22] regularizes the gradients employed for back-propagation with the use of buffer samples, which are called *episodic memory*. A common setting for testing the rehearsal approaches is by streaming samples in small batches, each one containing complete sets of samples for new tasks, with known tasks and clear task boundaries [7, 19]. This task-incremental setting is a simplification of the general continual online learning setting, in which the tasks and their boundaries are completely unknown.

Between the two ends, the relaxation approach proposed in [23] employs a Bayesian method to infer the task context. Similarly, an algorithm that uses the Shannon entropy as a measure to select representative samples from previously seen classes, without being affected by the fact that task boundaries are unknown is presented in [24]. However, the applicability of these solutions in resource-constrained applications may be limited by their computational complexity.

Another prominent approach for continual online learning is based on a combination of model distillation and iCaRL [9]. This method proposes the use of a custom distillation loss with an offline baseline model for retraining and the customization of the iCaRL method for online learning. This combination of proven methods shows promising results although its implementation is multi-faceted and can be resource-intensive. In its base, the *Incremental Learning Online Scenario* adopts the iCaRL principles in terms of exemplar use and NCM classification, and LwF-based distillation. However, it improves some intermediate steps and categorizes the training itself into various phases in a combination of online and offline training.

In all the above-mentioned methods, little emphasis is placed on the evaluation of their performance in terms of memory requirements and timeliness (i.e. the time to embody new knowledge once it arrives) [25]. A rather small body of literature is dealing with these issues. A representative case is *Latent Replay* [10], which emphasizes storing and using the output of intermediate layers for optimizing computational complexity and memory usage. As in the previously reported cases, this approach also assumes a priori known task boundaries.

Another case that investigates the computational cost of the training procedure is [26]. In this work, the authors propose the replacement of the Softmax activation function with a Balanced Softmax, which reduces the need for an additional fine-tuning step during training. Although this approach results in a bias towards new tasks, it tackles the important, real-world, problem of memory size limitations.

Recent works on rehearsal-based methods such as [27], suggest a prototype-based selection of samples to update the buffer. The buffer update is based on the cosine distance of the prototypes to achieve a balance between samples that are easier to classify correctly before training and samples that are further from a model's knowledge. This method is best used in the scenario where a model is updated frequently to incorporate new knowledge, which is not always optimal due to possible hardware restrictions. Also, on some occasions, the data stream might contain samples that can be classified correctly without training, therefore some training cycles might be unnecessary.

Another recent work is [28] where repeated data augmentation is used, for rehearsal training to maintain variety and bias-resistance during training. This method makes use of data augmentation which adds extra memory requirements for the new samples. Also, the constant requirement of repeating this process adds significant memory and computational overhead, thus requiring more resources.

The work of [16] is a stronger alternative to experience replay tested on task-incremental and domain-incremental experimental conditions. This alternative maintains two models for model distillation. This work is mostly focused on incremental learning in the same manner as the majority of the continual learning literature, excluding the stream-based, task-agnostic scenarios that often occur in real events.

All the aforementioned methods do not consider a model's ability to classify without errors at any given time step. This means that a model is unnecessarily over-trained thus adding to the time complexity, which can be proven critical in a resource-constrained setup. Also, in practice, the changes between tasks and the distribution of data within the stream may be unknown. Most of the aforementioned methods assume that we are able to have a beforehand knowledge of when a task will be altered. This task agnosticism can be an additional cause of the degradation of the model performance, due to possible i.i.d corruption.

This work is targeting the general online continual learning setting, where input samples are provided by a stream and correspond to tasks with unknown boundaries, while timeliness and memory efficiency are of equal importance to complexity, plasticity, scalability and accuracy. Those

characteristics allow our work to move ahead of previous research and deliver the following important contributions:

- The introduction of a *Drift Decision Mechanism* for determining when training is needed. This mechanism offers a significant advantage since it avoids unnecessary training and reduces the overall training time. Another advantage is that this mechanism is suitable for detecting any transitions between classification tasks (i.e. task boundaries), and therefore dealing with the issue of task agnosticism.
- The proposition of an *Adaptive Rehearsal Tuning* mechanism as a solution for customizing the rehearsal correction to the current model status, in a setup on which the model status is ever-changing in unforeseen ways. Our approach solves the problem of fine-tuning the rehearsal training at very precise moments in time, in an online data stream, where the distribution of data in terms of classes and tasks is unknown. Another important feature is the additional dynamic heuristics which allow for higher-level adaptation in different datasets and resources.

3 Online continual learning

3.1 Scenario

The working scenario considers a constant stream S of annotated data that is used for model training in an online fashion (Fig. 1). The stream of data is not i.i.d. (independent and identically distributed), but it is sampled in each period from different tasks. The task boundaries and the task identities are not known in advance. This adds difficulty compared to a scenario where batch training takes place with all data available from each task. We focus on classification problems and therefore each task is a sequence of annotated samples from a set of classes.

For simplicity, we assume that the data stream comes in batches of constant size (e.g., 32) comprising samples of any task. This setup is general enough to accommodate various scenarios, such as a stream composed of large and few tasks, or a stream with a fast succession of multiple small tasks. In all cases the stream is seen as a sequence of batches, each one comprising a set of samples $B_t = (X_t, Y_t)$, where $t \geq 1$ represents the batch at time step t. The task corresponds to a subset of batches that are observed sequentially. When the task changes, batches of samples that belong to new classes appear in the stream.

Our objective in this scenario is to continuously train a model, where the model snapshot h_{t+1} results from training h_t with the corresponding batch B_t of time step t. As the model learns using data from new tasks through this procedure, the final goal is to mitigate forgetting of old tasks, while at the same avoiding unnecessary training iterations (i.e., rehearsals), to keep computational complexity minimal.

3.2 An online rehearsal framework

In this section, we describe a general rehearsal framework, which is shown in Fig. 2 and is more formally defined in Algorithm 1.

Assume we are at time step t. Let $B_t = (X_t, Y_t)$ be the batch of newly acquired samples from the stream and H_t be the model from the previous time step. We denote as S_t a possible collection of local state variables that the algorithm needs to maintain during its execution (e.g. counters, etc.) in order to make informed decisions.

During training, we maintain two buffers, a *postponed buffer* P_t that keeps samples that have arrived since the last rehearsal, and a *rehearsal buffer* R_t that stores a mix of samples used for rehearsal. The postponed buffer is initially empty and the rehearsal buffer initially contains a user-selected fixed number of q samples per class where $q = \mid R_t \mid /n$ with n the number of classes. The samples

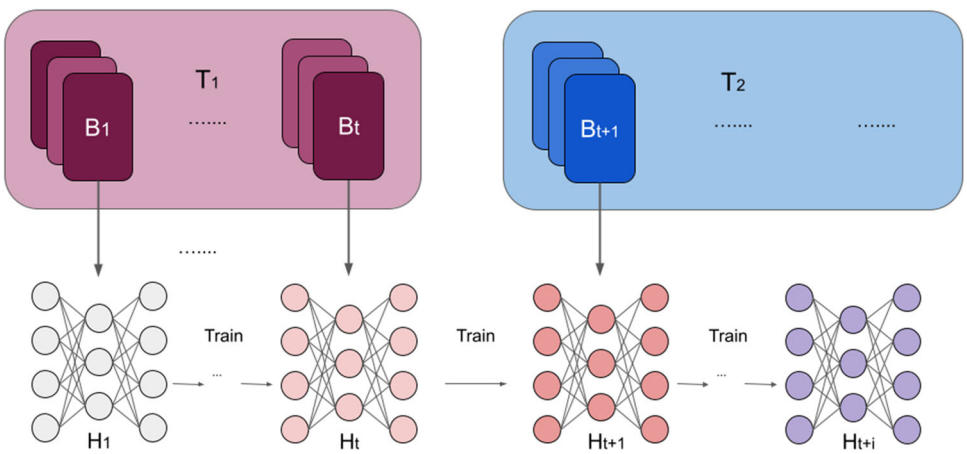

Fig. 1 Online continual learning scenario. Each batch $B_1 \ldots B_t, B_{t+1}, \ldots$ belongs to a specific classification task $T_1, T_2, \ldots$ and $H_1 \ldots H_t, H_{t+1} \ldots$ is the sequence of produced models after each step

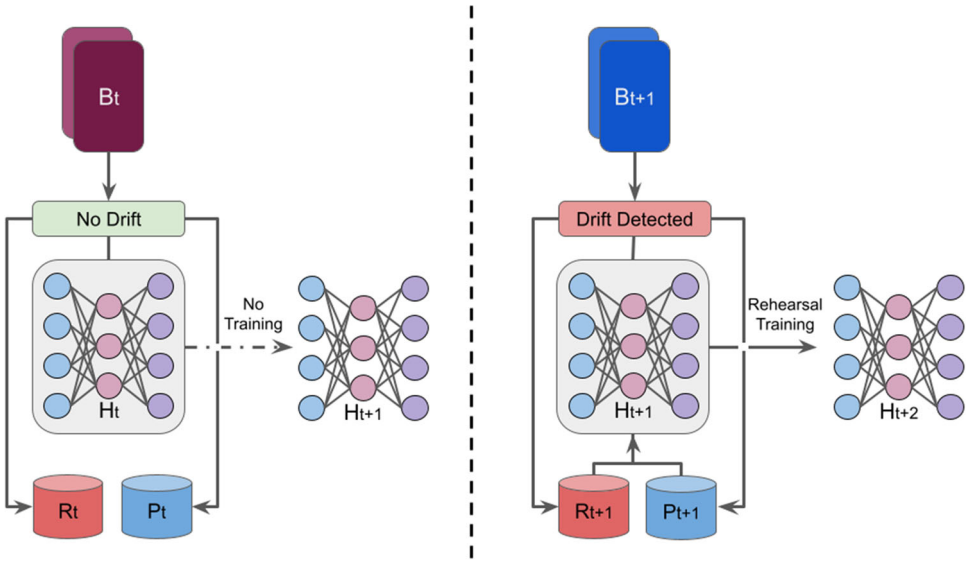

Fig. 2 Proposed strategy outline. If there is no change between tasks (Left), the drift detection mechanism stays idle and the new batch B_t is added to P_t, R_t. When training is needed (Right) the next batch B_{t+1} is used together with both the new rehearsal buffer R_{t+1} and the new postponed buffer P_{t+1}. All the aforementioned data are mixed for rehearsal training

of R_t are initially selected uniformly at random, while the size remains constant.

At each time step, the algorithm first applies a change detection mechanism to decide whether to train with the new batch of samples or not. If not, the new batch is simply appended to the postponed buffer P_t and the algorithm returns. Assume now that a change is detected and the algorithm decides to train the model at the current time step. The postponed buffer may be empty if training took place in the previous time step, or it may contain multiple batches in case the last training took place in previous time steps. The algorithm then employs a *rehearsal strategy* (see Section 4 for the supported options). Part of the strategy is to create training batches by combining the newly collected samples $P_t \cup B_t$ with samples from the rehearsal buffer R_t.

After each training step, the rehearsal buffer R_t is updated by deciding which samples to replace with samples from P_t or B_t, while the whole postponed buffer is cleared. In the proposed framework the update of the rehearsal buffer is performed using the method proposed by He et al. [9]. This method is a simplified variant of the *prioritized example selection* algorithm [19] that is based on herding ,also known as Herding Selection. The main difference between the two, which is in line with our performance requirements, is that the latter maintains a running average estimation for each class, instead of recalculating the sample average at each time step.

Herding Selection updates a buffer based on a model's representation average to ensure a more consistent set of representatives from each class. An extra feature is the ability to limit the number of the aforementioned representatives, thus controlling the buffer size whilst maintaining class balance during training cycles.

Based on the above, at each time t the rehearsal buffer contains a set of q samples $(E_1^y, \ldots, E_q^y)$, $y \in [1, \ldots, n]$, where n is the number of classes seen so far. The set is updated at each time step to always contain $q = |R|/n$ samples per class. When classes are encountered for the first time, the number of samples per class is dynamically updated to keep the total memory requirements constant.

Algorithm 1 Model and buffer update mechanism (time step t).

Input: model H_t, batch $B_t = (X_t, Y_t)$, state S_t (global variables from previous time steps), postponed buffer P_t, rehearsal buffer R_t

Output: updated H_{t+1}, S_{t+1}, P_{t+1}, and R_{t+1}

$Alarm \leftarrow detectChange(\{H_t, S_t, P_t \cup B_t, R_t\})$

if $Alarm = False$ **then**
    $P_{t+1} \leftarrow P_t \cup B_t$
    Update to a new state S_{t+1}
    $H_{t+1} \leftarrow H_t$
    $R_{t+1} \leftarrow R_t$
else:
    $H_{t+1}, S_{t+1} \leftarrow AdaptiveRehearsalTraining(\{H_t, S_t, P_t \cup B_t, R_t\})$
    Update R_{t+1} from R_t, $P_t \cup B_t$
    Reset postponed buffer: $P_{t+1} \leftarrow \emptyset$
end if

return $\{H_{t+1}, S_{t+1}, P_{t+1}, R_{t+1}\}$

Algorithm 2 describes a generic training and buffer update mechanism that is applied when training takes place. The number of training iterations (method $iter$) and the learning rate (method lrs) are being chosen using the methods described in Section 4. After redefining its hyper-parameters the model is ready for the required stochastic gradient descent steps (method SGD).

Algorithm 2 AdaptiveRehearsalTraining.

1: **Input**: model H, postponed buffer P, rehearsal buffer R, state S (global variables)
2:
3: **Output**: updated model H' and state S'
4:
5: Set Rehearsal Iterations: $N = iter(S, P, R)$
6: Set new Learning Rate Schedule: $\eta = lrs(S, P, R)$
7:
8: **for all** $i \in 1, \ldots, N$ **do**
9: Create a set of new batches D by mixing: $D = mix(i, S, P, R)$
10: **for all** $d \in D$ **do**
11: $H', S' = SGD(H, d, eta, lrs)$
12: **end for**
13: **end for**
14: **return** Updated model H' and state S'

To make this abstract framework applicable in practice, we need methods that allow us to determine (a) if additional training is currently required, (b) how samples from the postponed and rehearsal buffers should be mixed for training, (c) how many iterations to use for training the model at the current time step, (d) what learning rate to use. We address these questions, in the section that follows.

4 Rehearsal strategies

This section presents the alternative ways to decide *when* (line 5, Alg. 1) and *how* (line 13, Alg. 1) to perform the rehearsal during training. Algorithm 2 gives a high-level overview of the online training process. Note that all suggested methods and heuristics can be tuned and switched in various independent setups offering an adequate palette of different online learning setups, thus creating a more flexible forgetting mitigation plan.

4.1 Continuous rehearsal and experience replay

The baseline strategy on which we make our comparative study is a variant of Experience Replay [18]. This method is based on the online application of experience replay similar to the GDUMB [17]. The only difference is the usage of herding selection instead of greedy sampling (as proposed by GDUMB). The herding selection buffer ensures class balance and is commonly used in the most well-established continual learning methods. Our baseline does consider recent works and additions to make an even more challenging testing setup for our methods.

This method assumes that training takes place in each time step. At each time step t, training happens for a fixed number of iterations (epochs) using the latest batch $P_t \cup B_t = B_t$ (for this strategy the postponed buffer P_t is always empty) and batches from the rehearsal buffer R_t. In each iteration, the latest batch B_t is combined with a different batch $r_j \in R_t$ in order to produce two new batches containing 50% samples from B_t and 50% samples from r_j each. In order to make use of the entire rehearsal buffer, the position of the batch used in each time step is kept in a global variable and the rehearsal batches (i.e. R_t) are employed in a round-robin fashion. In the simplest case, only one iteration is used per batch [18]. In what follows we use `ER-n` to denote this baseline method with n iterations (`ER-1` for a single iteration).

4.2 Drift activated rehearsal

In the general online continual learning scenario defined in Section 3.1, the task boundaries are unknown. A solution for deciding when to train (instead of applying a constant training schedule) is to use a *concept drift detector* [29]. From the concept Drift Detectors that are available in the literature, we choose the ECDD detector [30] which uses exponentially weighted moving average charts (EWMA) as an indicator of divergence between samples. It is a single pass method with $\mathcal{O}(1)$ update in each time step, which makes it suitable for performance-critical, streaming applications. The ECDD detector is a simple and popular solution [31] for drift detection, so in the following, we describe the internal mechanism of this particular Drift Detector and leave as future work the task of studying different and more sophisticated drift detection mechanisms.

Samples in the stream are sequentially presented to the classifier, and at each time step we examine whether the predicted class label was correct (i.e. $X_t = 0$), or incorrect (i.e. $X_t = 1$). The ECDD detector perceives $\{X_t\}$ as a sequence of observations from a Bernoulli distribution. Detecting concept drift becomes the problem of detecting an increase in the Bernoulli parameter p corresponding to the probability of misclassification. The ECDD detector maintains an estimate $Z_t = (1-\lambda)Z_{t-1} + \lambda X_t$ of the current mean μ_t along with a second estimator $\hat{p}_{0,t} = \frac{1}{t}\sum_{i=1}^{t} X_i$, which changes more slowly and better estimates the probability before the change (event). Given estimates of $\hat{\sigma}_{X_t}$ and $\hat{\sigma}_{Z_t}$ ECDD first computes a control limit, L_t based on the methodology presented in [30]. If $Z_t > \hat{p}_{0,t} + L\hat{\sigma}_{Z_t}$, then ECDD raises a concept drift flag.

During time step t all samples of batch B_t are given to the Drift Detector which updates the estimators. Since new samples arrive in a batch B_t, $\{X_t\}$ can be viewed as a sequence of per-batch observations that contains information about how many samples from B_t when classified incorrectly. The following heuristic rules are used to decide whether to train: (i) $Z_t > \hat{p}_{t,0} + L_t\hat{\sigma}_{Z_t}$, which is the original rule of ECDD, (ii) $U_t > \hat{p}_{t,0} + 2\hat{\sigma}_{U_t}$ i.e., the current batch error U_t must not be too high (two standard deviations above the estimated mean), (iii) $Z_t > \epsilon$, the running average of the error must not exceed a user-defined limit, and (iv) no training during the last μ time steps, where ϵ and μ are user-defined thresholds.

Algorithm 3 detectChange.

Input: model logits m_{out}, ground truths, y, lambda lam, average run length ARL
Output: Drift Flag value $isDrift$

$isDrift = False$, $Z_0 = 0$, $\hat{p}_{0,0} = 0$

for all m_{out}, y **do**
  $X_t = 0$ if $argmax(m_{out}) = argmax(y)$ else $X_t = 1$
  $\hat{p}_{0,t} = \frac{t}{t+1}\hat{p}_{0,t-1} + \frac{1}{t+1}X_t$
  Compute estimates of $\hat{\sigma}_{X_t}$ and $\hat{\sigma}_{Z_t}$
  $Z_t = (1-\lambda)Z_{t-1} + \lambda X_t$
  Compute L_t (Method precomputed tables)
  $isDrift = True$ if $Z_t > \hat{p}_{0,t} + L_t\hat{\sigma}_{Z_t}$
end for
return $isDrift$

Reasonable choices are $\epsilon = 0.2$ for the error threshold, and $\mu = 20$ for the no-training time steps threshold. Assuming a constant number of iterations for training, the main difference with the `ER-n` method is that, when it is triggered, the postponed buffer P_t will likely contain multiple batches. In each training iteration, we scan over all batches of P_t. For each batch $p_i \in P_t$ we read a batch $r_j \in R_t$ and create two batches which contain 50% from each. The buffer R_t is used in a round-robin fashion using the updated position pointer after each time step. We use `DRIFTA-n` to name the subset of methods which performs a constant number of n iterations when the Drift Detector is triggered.

4.3 Drift detection with buffer samples (double drift detector strategy)

Depending on the size of the rehearsal buffer and the average length of the tasks, a possible adaptation to the previous strategy is to include a second Drift Detector. This additional detector monitors the classification failure rate from samples of the rehearsal buffer. Thus, the detection happens on an artificially created stream of samples, which are drawn uniformly at random from the rehearsal buffer, in each time step. Training takes place when any of the two detectors are triggered, based on the previously mentioned heuristics. The only difference is that two versions of each estimate are kept, for example, Z_t for the new samples and $\hat{Z}_t$ for the rehearsal samples. The number of random elements which are used for drift detection can change as a hyper-parameter. This mode is more sensitive in terms of corruption detection at the cost of a slightly more computationally expensive implementation. We denote this strategy which uses a constant number of n iterations for training as `2DRIFTA-n`.

4.4 Setting the number of iterations dynamically

Instead of a fixed number of training iterations in each step, we can dynamically adapt the number of iterations based on the misclassification rate Z_t. We use the following heuristic rule where n is a hyperparameter.

$$n_t = \lceil 2 * n * \log_2(1 + Z_t) \rceil \tag{1}$$

The heuristic rule behind dynamic iterations is a simple binary logarithm equation that shifts between zero and $2 * n$ iterations based on the misclassification rate Z_t. We used the binary logarithm to achieve smoother adaptations when Z_t rate is close to zero or one. We denote this strategy that associates the estimator Z_t values with the number of iterations as `DRIFTA-DYN-n`. Similarly, `2DRIFTA-DYN-n` do the same getting the output of both Drift Detectors and using $max(Z_t, \hat{Z}_t)$ for the final decision.

4.5 Rehearsal based on convergence

A different approach is to determine the number of necessary rehearsal iterations based on the model convergence. In this approach, we monitor the model's loss $\mathcal{L}$ by keeping two exponential moving averages, one short where:

$$\mathcal{EMA}_{short} = (1 - \alpha_{short})\mathcal{EMA}_{short} + \alpha_{short}\mathcal{L} \tag{2}$$

and one long, where:

$$\mathcal{EMA}_{long} = (1 - \alpha_l)\mathcal{EMA}_{long} + \alpha_{long}\mathcal{L}, \tag{3}$$

with $\alpha_{short} = 0.5$ and $\alpha_{long} = 0.05$ respectively. We stop training when the two values converge, i.e. $\|(\mathcal{EMA}_{long} - \mathcal{EMA}_{short})\| < \epsilon$ for some hyper-parameter value ϵ. Note that this approach can be used for any rehearsal strategy. We denote this strategy as `ER-CONV`, `DRIFTA-CONV` and `2DRIFTA-CONV` for a constant, drift-activated and two-Drift Detector setup respectively.

4.6 Adjusting the learning rate

A last action that can affect the efficiency of the rehearsal strategy is a learning rate schedule across iterations based on a model's state. This is more important in the drift-activated methods, where multiple batches are collected before initiating the training. The simplest approach is to keep the learning rate η constant, however as in regular training with i.i.d. data, this is often suboptimal. Another approach is to initialize the learning rate with a user-defined value at the beginning of each time step and use a decay mechanism that modifies the learning rate through training iterations. Finally, the drift-activated methods can also use Z_t and a predefined η value to dynamically adjust the initial learning rate. A combination of the aforementioned approaches can help to further improve the performance. Z_t is used as an exponent to a variation of the exponential function. The reason behind this

is a fast adaptation of the learning rate if Z_t is high. For the initialization of the learning rate (LR) we use a simple rule that sets it to $LR_{new} = LR_0 * \min(100, 5 * e^{3Z_t})$ where LR_0 is a pre-defined learning rate. For the decay of LR a simple used-defined constant is enough as shown in our experiments.

5 Experiments

In this section, we define a robust experimental benchmark, for testing both our propositions alongside the standard rehearsal in an online setup based on Experience Replay. The purpose of the experiments is to provide insight into how ER-based methods can be improved and test the limitations and benefits of each algorithmic setup. The methods will be tested with an improved version of the Experience Replay algorithm based on GDUMB [17], which can be viewed as an online application of Experience Replay.

5.1 Experimental setup

5.1.1 Datasets

We use the CIFAR-10 image classification dataset [13] and the MNIST digits dataset [8, 32] to evaluate the online continual learning strategies of Section 4. CIFAR-10 consists of 60,000 images (50,000 training, 10,000 testing), sampled uniformly from ten classes $\{0, \cdots, 9\}$ and MNIST consists of 70,000 images (60,000 training, 10,000 testing), also sampled uniformly from digits $\{0, \cdots, 9\}$. Following the online continual learning scenario described in Section 3.2, we split the training data into five tasks, each one containing images from two classes, i.e.,

$$T_i = [(\mathbf{x}, y), y \in [2i, 2i+1], i = 0, 1, 2, 3, 4] \tag{4}$$

An online annotated image stream, S, is generated by first sampling multiple images from the first task, then the second task and so on

$$S = (T_0^{(0)}, T_1^{(0)}, T_2^{(0)}, T_3^{(0)}, T_4^{(0)}, T_0^{(1)}, T_1^{(1)}, \cdots) \tag{5}$$

Each set $T_i^{(j)}$ has a fixed size of 3,200 images, grouped into batches of 32 images, for a total of 100 batches per task. Images in each batch are sampled uniformly at random from the classes of the current task, T_i. For each experiment, each task appears three times (i.e., $j = 0, 1, 2$). All experiments use the same random seed for sampling and for model parameter initialization, to ensure that experiments are comparable.

5.1.2 Model and pre-training

The base model in our experiments is the adaptation of ResNet32 for the CIFAR dataset, as described in [33]. The model is first trained offline for 100 epochs with a subset of the CIFAR dataset consisting of 1500 images for each of the 10 classes. As a result, the model used in the experiments has gone through a "warm-up" training stage but has not been fully trained. It achieves accuracies of [0.805, 0.602, 0.732, 0.844, 0.854] for tasks T_0-T_4 respectively. This model is used at the start of online training in all experiments. In most occasions, in the continual learning literature, a method keeps either one or two model weight copies (two, in the case of model distillation techniques). In this setup we only keep and update one copy of the model weights.

5.1.3 Metrics

Each training strategy is evaluated using the following metrics, computed at each time step t. We denote as $H_t(x)$ the output of the model at time step t when the input is sample x. The ground truth is denoted as $\hat{y}$

- *Accuracy* (A_t): Accuracy of the model evaluated in the held-out test set D, averaged across all tasks, this metric is measured on each time step For time step t the accuracy is:

$$A_t = \frac{1}{\|D\|} \sum_{\forall x \in D} \begin{cases} 0, & H_t(x) \neq \hat{y} \\ 1, & H_t(x) = \hat{y} \end{cases}$$

- *Current task accuracy* (C_t): Accuracy of the model evaluated in the held-out test set, but only for the images belonging to classes of the current task, T_t. For time step t the Current accuracy for T_t is:

$$C_t = \frac{1}{\|D_{T_t}\|} \sum_{\forall x \in D_{T_t}} \begin{cases} 0, & H_t(x) \neq \hat{y} \\ 1, & H_t(x) = \hat{y} \end{cases}$$

- *Online accuracy* (O_t): Accuracy in each batch B_t of stream S at time step t, evaluated right before it is used for training. This approach was also used in [9]. For time step t the Online accuracy at time step t is:

$$O_t = \frac{1}{\|B_t\|} \sum_{\forall x \in B_t} \begin{cases} 0, & H_t(x) \neq \hat{y} \\ 1, & H_t(x) = \hat{y} \end{cases}$$

- *Training iterations* (N_t): Cumulative number of iterations performed during stochastic gradient descent optimization for every time step t. Less training iterations show

a more efficient approach, although classification accuracy needs to be taken into account as well. Given that the model and the batch size are the same across all experiments, this metric can be used to compare the computational complexity of different training strategies. The tables below will include this metric as a ratio $N_t / \max(N_t)$ where $\max(N_t)$ is the training iterations count of the most computationally intensive method. Unlike the rest of the aforementioned metrics, we aim to reduce the amount of training within all experiments.

Each metric evaluates a different aspect of the model's effectiveness. C_t and O_t evaluate the effectiveness of the model in the current task and data, respectively, while A_t evaluates the overall effectiveness of the model (including previous tasks). In addition to metrics computed at each t, we also report averages across t, such as $\bar{A}_t$.

5.2 Experiment 1: motivation

This experiment demonstrates the benefits of an adaptive training strategy in an online environment, as discussed in Section 4. We compare three training strategies, which include (i) online training without any further action, (ii) experience replay with 1 training iteration [20] (`ER-1`), and (iii) experience replay with 50 training iterations (`ER-50`). In all cases, the learning rate was fixed to $\eta = 0.01$. Results are visually illustrated in Fig. 3 and show the limitations of training with continuous rehearsal methods in an online setting.

It is clear that a static strategy of dealing with online learning is either subpar in terms of accuracy when iterations are a few and in the case of (`ER-50`) the accuracy is achieved by adding a very high cost in terms of computation, therefore rendering this implementation insufficient

These results show that `ER-1` is similar to not performing any training at all. Performing several iterations per time step (`ER-50` in this case) leads to a model that performs well only for the current task, at the expense of the overall accuracy (due to forgetting previous tasks).

It is apparent that between task changes, we often see sharp drops in the Current Accuracy metric, as shown in Fig. 3. These fluctuations are caused by overfitting the model with a specific task. The more per-step iterations a method uses the harder is to recover between task training, the `ER-50` method is the more prone to this trend due to the fact that it trains the model at every step and every training step iterates constantly 50 times, causing overfitting as a result.

All the experiments conducted, actually introduce tasks that are occasionally radically different, in order to provide a better insight of the limitations per method.

5.3 Experiment 2: comparison of training strategies

The goal of the second experiment is to assess the improvement of the proposed training strategies in terms of both effectiveness and efficiency. Figures 4 and 5 illustrate the results for the MNIST and CIFAR-10 datasets respectively. One can observe that `DRIFTA-DYN-50` manages to maintain high accuracy in the current task, C_t, while requiring significantly lower computation compared to `ER-50`.

Tables 1 and 2 show the average metrics and the 95% confidence intervals of various `ER-n` methods, for $n = 1, 10, 25, 50$, the proposed methods as well as the computational requirements of each method (in terms of the number of iterations). All methods are compared in terms of the number of training iterations, N_t, as well as in terms of the average accuracy metrics $\bar{A}_t$, $\bar{C}_t$ and $\bar{O}_t$.

Results demonstrate the benefit of using the proposed drift-activated, dynamic and convergence-based rehearsal strategies especially `2DRIFTA-DYN-n` and `2DRIFTA-CONV`. In the MNIST dataset, `DRIFTA-DYN-n` and `2DRIFTA-DYN-n` achieve approximately the same average accuracy metrics as the best experience replay methods, for a significantly smaller number of iterations (i.e., computational cost). Similar observations can be made for the CIFAR-10 dataset, and especially for convergence-based methods.

5.4 Evaluation of the permuted MNIST digits problem

In previous experiments, the experiments involved model training with a sequence of tasks that included disjoint classes. In this section, we evaluate our model on a different scenario, where each class may have different appearances which are presented to our model sequentially. In practice, we use the *Permuted MNIST* dataset [12]. Permuted MNIST expands over the MNIST by creating tasks via fixed pixel permutations. The first task is classifying all original digit images, and each new task is a pixel-wise permutation of the first task. Note that every permutation is constant for all images of a task. The model has to classify correctly each digit in the $\{0, \cdots, 9\}$ classes without any known correspondence between images and tasks.

The ResNet32 model of previous experiments is also used in this case. The model is first trained offline for 50 epochs only with the original MNIST dataset, using only 500 samples per MNIST class. This model is then used for online training in all experiments. Note that the model is originally effective only for non-permuted digits. The buffer contains samples from the 10 classes and 50 images per class. Each task is created with 100 batches of 32 elements (i.e., not all permuted MNIST digits are presented to the model during a task).

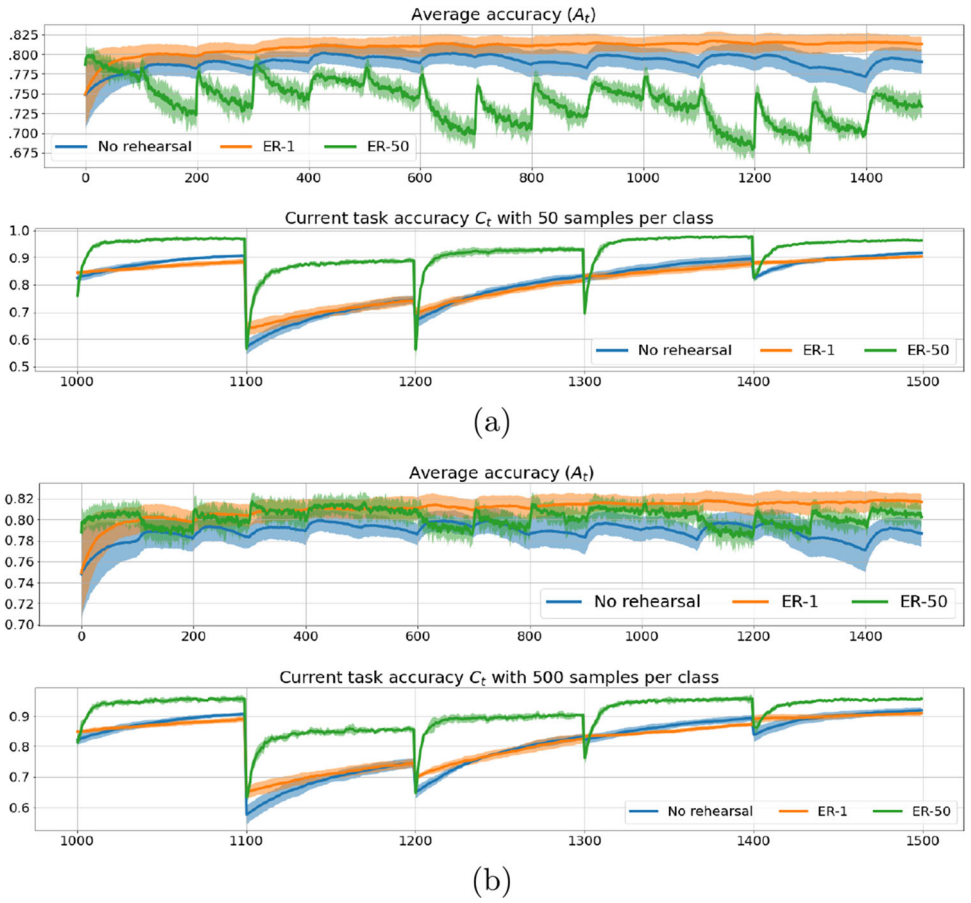

Fig. 3 Results from the CIFAR-10 test on accuracy (A_t) and current task accuracy (C_t) with (a) 50 samples per class in the buffer, and (b) 500 samples per class. Continuous rehearsal with 1 iteration (`ER-1`) performs similarly to having no rehearsal at all. `ER-50` iterations perform better for 500 samples per class but require 50 times higher computational cost. It also does not perform well for 50 samples per class. None of these methods seems to be satisfactory in a continuous online learning setting

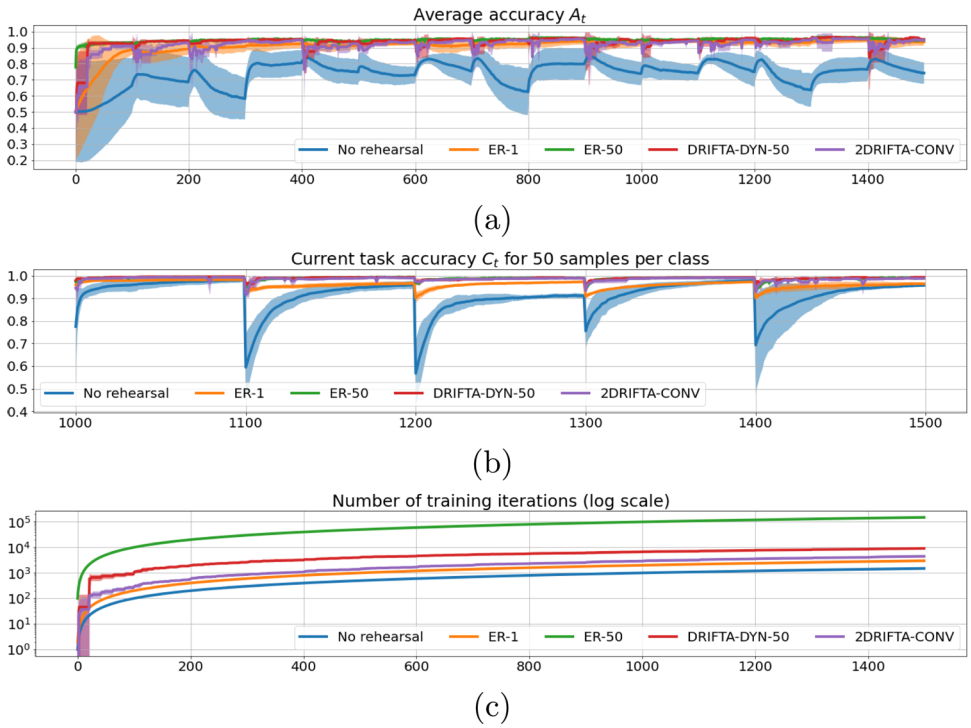

Fig. 4 Experiment on the MNIST dataset measuring: (a) average task accuracy A_t, (b) current task accuracy C_t on the range of $t = 1000, \cdots, 1500$, and (c) number of batches N_t for $t = 0, \cdots, 1500$ of the stream for a rehearsal buffer with 50 samples per class. `DRIFTA-DYN-50` is as effective as `ER-50` with a fraction of the computational requirements

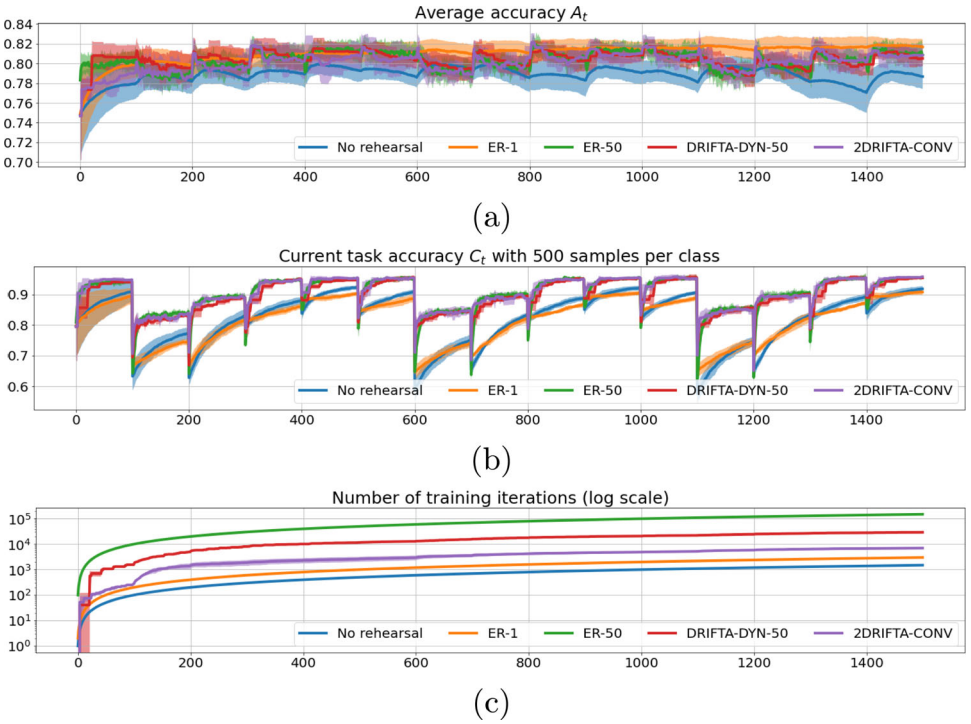

Fig. 5 Experiment on the CIFAR-10 dataset measuring: (a) average task accuracy A_t, (b) current task accuracy C_t on the range of $t = 1000, \cdots, 1500$, and (c) number of batches N_t for $t = 0, \cdots, 1500$ of the stream for a rehearsal buffer with 500 samples per class. `2DRIFTA-CONV` is consistently more effective and efficient in this experimental setup

The stream is built as:

$$S = (T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_0) \tag{6}$$

with T_0 corresponding to the classification of the original dataset and each T_i, $i = 1, \cdots, 9$, corresponding to a permutation (9 permutations of the original dataset in total).

Figure 6 illustrates the results in terms of effectiveness, comparing the `ER-1` , `ER-50` , `DRIFTA-DYN-50` and `2DRIFTA-CONV` rehearsal strategies. Furthermore, Table 3 shows average accuracy values, as well as the number of iterations required by each method. Results indicate that for a stream constructed using the permuted MNIST scenario, the proposed methodology improves effectiveness in terms of A_t, C_t as well as O_t, while at the same time significantly reducing the required number of training iterations.

6 Ablation study

The proposed methodology addresses various aspects of online continual learning with rehearsal, including the use of a Drift Detector to determine when to train, as well as strategies for determining the number of training iterations

Table 1 Comparison of the different online training strategies for the MNIST digit dataset by using average values of the metrics are shown across the entire stream (bold indicates the best and underline indicates the second best result for each metric)

MNIST (50/class)				
Strategy	$N_t / \max(N_t)$	A_t	C_t	O_t
`NO REHEARSAL`	**0.01 ± 0.0**	0.741 ± 0.083	0.904 ± 0.028	0.905 ± 0.027
`ER-1`	<u>0.02 ± 0.0</u>	0.912 ± 0.037	0.948 ± 0.011	0.947 ± 0.01
`ER-10`	0.2 ± 0.0	0.942 ± 0.009	0.981 ± 0.004	0.979 ± 0.003
`ER-25`	0.5 ± 0.0	<u>0.947 ± 0.004</u>	<u>0.984 ± 0.002</u>	<u>0.983 ± 0.002</u>
`ER-50`	1.0 ± 0.0	**0.949 ± 0.002**	**0.985 ± 0.001**	**0.985 ± 0.002**
`DRIFTA-DYN-50`	0.061 ± 0.003	0.936 ± 0.009	**0.985 ± 0.001**	**0.985 ± 0.001**
`2DRIFTA-DYN-50`	0.066 ± 0.004	0.943 ± 0.006	**0.985 ± 0.002**	**0.985 ± 0.001**
`DRIFTA-CONV`	0.025 ± 0.001	0.899 ± 0.025	0.977 ± 0.004	0.979 ± 0.002
`2DRIFTA-CONV`	0.03 ± 0.001	0.92 ± 0.015	0.979 ± 0.003	0.98 ± 0.002

Each class is represented by 50 samples per class within the rehearsal buffer. The proposed training strategies achieve similar accuracy to the best experience replay methods at a fraction of the training cost

Table 2 Comparison of the different online training strategies in terms of average values of the metrics across the entire stream (bold indicates the best and underline indicates the second best result for each metric)

CIFAR-10				
Strategy	$N_t / \max(N_t)$	A_t	C_t	O_t
(50 samples per class)				
NO REHEARSAL	**0.01 ± 0.0**	0.789 ± 0.013	0.82 ± 0.012	0.866 ± 0.007
ER-1	0.02 ± 0.0	**0.808 ± 0.011**	0.813 ± 0.013	0.865 ± 0.007
ER-10	0.2 ± 0.0	0.783 ± 0.01	0.893 ± 0.005	0.926 ± 0.002
ER-25	0.5 ± 0.0	0.768 ± 0.01	0.911 ± 0.005	0.939 ± 0.002
ER-50	1.0 ± 0.0	0.756 ± 0.01	0.917 ± 0.005	0.944 ± 0.002
DRIFTA-DYN-50	0.172 ± 0.003	0.742 ± 0.006	**0.921 ± 0.005**	**0.947 ± 0.002**
2DRIFTA-DYN-50	0.169 ± 0.006	0.749 ± 0.015	**0.921 ± 0.005**	**0.947 ± 0.002**
DRIFTA-CONV	0.038 ± 0.004	0.768 ± 0.01	0.916 ± 0.005	0.942 ± 0.003
2DRIFTA-CONV	0.04 ± 0.004	0.767 ± 0.01	0.916 ± 0.006	0.942 ± 0.003
(500 samples per class)				
NO REHEARSAL	**0.01 ± 0.0**	0.789 ± 0.013	0.82 ± 0.012	0.866 ± 0.007
ER-1	0.02 ± 0.0	**0.81 ± 0.011**	0.816 ± 0.013	0.868 ± 0.007
ER-10	0.2 ± 0.0	0.806 ± 0.009	0.89 ± 0.006	0.925 ± 0.003
ER-25	0.5 ± 0.0	0.805 ± 0.01	0.902 ± 0.005	0.936 ± 0.002
ER-50	1.0 ± 0.0	0.803 ± 0.01	0.905 ± 0.006	0.939 ± 0.003
DRIFTA-DYN-50	0.195 ± 0.008	0.804 ± 0.01	0.898 ± 0.005	0.938 ± 0.003
2DRIFTA-DYN-50	0.196 ± 0.011	0.805 ± 0.01	0.899 ± 0.007	0.937 ± 0.004
DRIFTA-CONV	0.034 ± 0.003	0.796 ± 0.009	**0.909 ± 0.006**	**0.94 ± 0.004**
2DRIFTA-CONV	0.047 ± 0.004	0.801 ± 0.01	0.907 ± 0.006	0.939 ± 0.002

Results are provided for the CIFAR-10 with a rehearsal buffer of 50 and 500 samples. The statistics of the proposed training strategies are as high as those of the baseline methods but at a fraction of their computational cost

and the learning rate. In this section, we progressively evaluate the added value of these different components of the proposed training framework. Experiments use the CIFAR-10 dataset and the ResNet32 pre-trained model, as described in Section 5.1.1.

6.1 Dynamic choice of iterations

In this experiment, we only use the option of dynamic iterations parameter, as described in Section 4.4, to assess the value of adaptive vs constant number of training iterations per

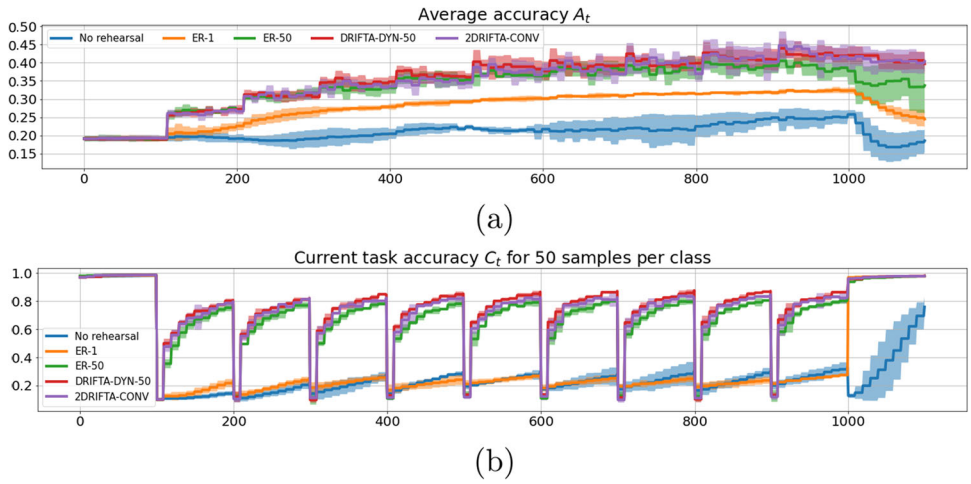

Fig. 6 (a) Average accuracy A_t, and (b) current task accuracy C_t for the MNIST digits dataset and nine permutations, for ER-1 , ER-50 , DRIFTA-DYN-50 and 2DRIFTA-CONV rehearsal strategies

Table 3 Comparison of the different online training strategies in terms of average values of the metrics across the entire stream from the MNIST digits and their permutations

Permuted MNIST digits (50/class)				
Strategy	$N_t / \max(N_t)$	A	C_t	O_t
NO REHEARSAL	**0.01 ± 0.0**	0.21 ± 0.012	0.294 ± 0.009	0.297 ± 0.008
ER-1	0.02 ± 0.0	0.277 ± 0.001	0.352 ± 0.003	0.354 ± 0.002
ER-10	0.2 ± 0.0	0.319 ± 0.012	0.618 ± 0.02	0.629 ± 0.022
ER-25	0.5 ± 0.0	0.328 ± 0.015	0.687 ± 0.01	0.702 ± 0.01
ER-50	1.0 ± 0.0	0.336 ± 0.016	0.708 ± 0.007	0.731 ± 0.006
DRIFTA-DYN-50	0.575 ± 0.014	0.354 ± 0.012	0.76 ± 0.007	0.787 ± 0.005
2DRIFTA-DYN-50	0.574 ± 0.019	**0.358 ± 0.017**	**0.762 ± 0.008**	**0.788 ± 0.007**
DRIFTA-CONV	0.283 ± 0.012	0.347 ± 0.006	0.745 ± 0.004	0.767 ± 0.003
2DRIFTA-CONV	0.292 ± 0.022	0.349 ± 0.012	0.742 ± 0.007	0.763 ± 0.007

Bold indicates the best and underline indicates the second result for each metric

batch. In all experiments, we use the same learning rate, and there is no learning rate decay. Results are shown in Table 4.

The dynamic choice of iterations reduces the required number of iterations and consequently decreases computational complexity. Based on the results, the convergence-based methods are the most suitable for this purpose.

6.2 Dynamic learning rate schedule

We further evaluate the added value of dynamically selecting the learning schedule based on the output of the Drift Detector (as described in Section 4), compared to a constant learning rate. The experimental results in Fig. 7 show the benefit of using a dynamic learning rate schedule.

Table 5 shows the average performance of both the static and the dynamic learning rate schedules.

Experiments show the value of adapting the learning rate using the strategies described in Section 4.6. They also demonstrate that convergence-based methods are more sensitive to changes in the learning rate. Furthermore, it seems that the 2DRIFTA-CONV and 2DRIFTA-DYN-50 methods require fewer iterations when the learning rate is tuned based on drift detector statistics. Note that there is both a drop in N_t and an increase in overall accuracy for both DRIFTA-CONV and 2DRIFTA-CONV methods.

Table 4 Comparison of dynamic iteration strategies in terms of average values of the metrics across the entire stream

CIFAR-10 (500/class)				
Strategy	$N_t / \max(N_t)$	A_t	A_{0-500}	$A_{1000-1500}$
NO REHEARSAL	**0.01 ± 0.0**	0.789 ± 0.013	0.787 ± 0.015	0.788 ± 0.013
ER-1	0.02 ± 0.0	**0.81 ± 0.011**	0.802 ± 0.013	**0.816 ± 0.009**
ER-10	0.2 ± 0.0	0.806 ± 0.009	0.8 ± 0.01	0.81 ± 0.009
ER-25	0.5 ± 0.0	0.805 ± 0.01	0.801 ± 0.01	0.807 ± 0.009
ER-50	1.0 ± 0.0	0.803 ± 0.01	0.802 ± 0.011	0.803 ± 0.009
DRIFTA-DYN-50	0.195 ± 0.008	0.804 ± 0.01	**0.805 ± 0.011**	0.803 ± 0.009
2DRIFTA-DYN-50	0.196 ± 0.011	0.805 ± 0.01	**0.805 ± 0.011**	0.803 ± 0.009
DRIFTA-CONV	0.034 ± 0.003	0.796 ± 0.009	0.79 ± 0.009	0.8 ± 0.009
2DRIFTA-CONV	0.047 ± 0.004	0.801 ± 0.01	0.796 ± 0.012	0.804 ± 0.009
	$\bar{C}_t$	$\bar{C}_{0-500}$	$\bar{C}_{1000-1500}$	$\bar{O}_t$
NO REHEARSAL	0.82 ± 0.012	0.823 ± ± 0.022	0.817 ± 0.008	0.866 ± 0.007
ER-1	0.816 ± 0.013	0.813 ± 0.019	0.819 ± 0.01	0.868 ± 0.007
ER-10	0.89 ± 0.006	0.886 ± 0.007	0.893 ± 0.006	0.925 ± 0.003
ER-25	0.902 ± 0.005	0.897 ± 0.006	0.906 ± 0.006	0.936 ± 0.002
ER-50	0.905 ± 0.006	0.9 ± 0.005	0.908 ± 0.006	0.939 ± 0.003
DRIFTA-DYN-50	0.898 ± 0.005	0.889 ± 0.005	0.903 ± 0.005	0.938 ± 0.003
2DRIFTA-DYN-50	0.899 ± 0.007	0.891 ± 0.007	0.904 ± 0.009	0.937 ± 0.004
DRIFTA-CONV	**0.909 ± 0.006**	**0.903 ± 0.006**	**0.913 ± 0.007**	**0.94 ± 0.004**
2DRIFTA-CONV	0.907 ± 0.006	0.902 ± 0.006	0.911 ± 0.006	0.939 ± 0.002

Results are provided for the case of a rehearsal buffer with 500 samples per class. Bold indicates the best and underline indicates the second result for each metric

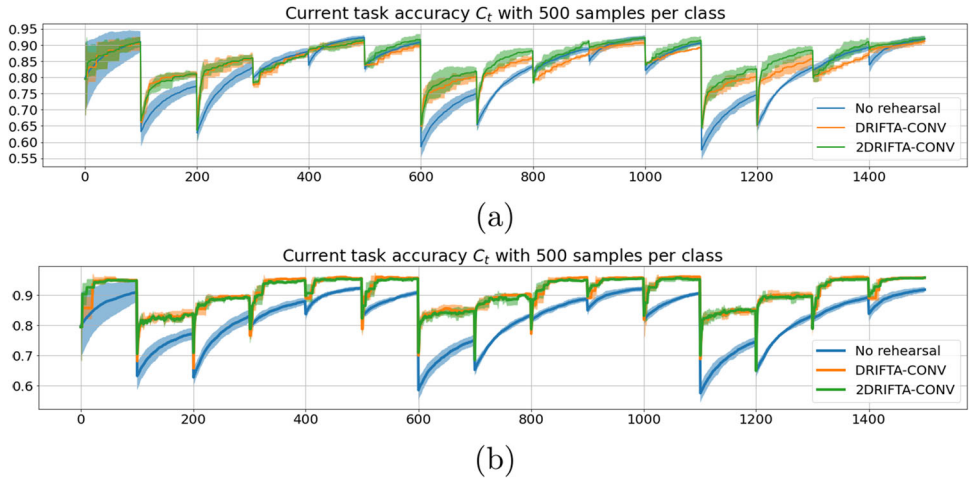

Fig. 7 Current task accuracy C_t for (a) constant learning rate and (b) dynamic learning rate schedule in convergence rehearsal strategies

7 Conclusions and future work

This study presented a framework for utilizing Rehearsal-based methods into stream-based scenarios in order to facilitate online continual learning. Our primary objective is to tackle the constraints of limited computational and memory resources. The key considerations revolve around optimizing training cycles and adapt the training procedure at each step.

To address these challenges, we propose the incorporation of a drift detection mechanism, which initiates model training in response to changes in data distribution. Additionally, we propose several strategies for determining the appropriate number of training iterations and creating an dynamic learning rate schedule, based on model misclassification statistics.

Our proposed approach, according to our experimental settings, demonstrates comparable and even increased effectiveness compared to the conventional rehearsal strategies while utilizing fewer training iterations and balancing between accuracy and overfitting. Baseline methods, such as GDUMB, are tested with various iterations per training step to fix the issues that arise with the original Experience Replay methods. Baseline methods with fewer iterations (such as ER-1, ER-10 (Original Experience Replay) do not overfit, but at the cost of decreased classification accuracy. More computationally intensive baseline methods such as ER-50 (50 iterations per batch) are more accurate per task, but between tasks, they tend to show decreased average accuracy and "sharp changes" in between tasks (ER-50 shows clear signs of overfitting). These observations validate that our proposed methods keep the balance between accuracy per task and overfitting. Also, our methods use only the least necessary training times to achieve these results. The final result shows a significant advantage in terms of computa-

Table 5 Comparison of dynamic versus constant iteration strategies in terms of average values of the metrics across the entire stream

CIFAR-10 (500/class)					
Strategy	N_t / max(N_t)	C_t	C_{0-500}	$C_{1000-1500}$	O_t
Constant LR					
DRIFTA-DYN-50	0.214 ± 0.004	0.894 ± 0.005	0.889 ± 0.007	0.897 ± 0.005	0.929 ± 0.003
2DRIFTA-DYN-50	0.21 ± 0.007	0.895 ± 0.006	0.891 ± 0.008	0.899 ± 0.005	0.93 ± 0.003
DRIFTA-CONV	0.037 ± 0.004	0.842 ± 0.009	0.845 ± 0.015	0.839 ± 0.008	0.887 ± 0.005
2DRIFTA-CONV	0.056 ± 0.005	0.855 ± 0.012	0.846 ± 0.015	0.86 ± 0.013	0.898 ± 0.006
Dynamic LR					
DRIFTA-DYN-50	0.195 ± 0.008	0.898 ± 0.005	0.889 ± 0.005	0.903 ± 0.005	0.938 ± 0.003
2DRIFTA-DYN-50	0.196 ± 0.011	0.899 ± 0.007	0.891 ± 0.007	0.904 ± 0.009	0.937 ± 0.004
DRIFTA-CONV	**0.034 ± 0.003**	**0.909 ± 0.006**	**0.903 ± 0.006**	**0.913 ± 0.007**	**0.94 ± 0.004**
2DRIFTA-CONV	0.047 ± 0.004	0.907 ± 0.006	0.902 ± 0.006	0.911 ± 0.006	0.939 ± 0.002

The dynamic-iterations-based strategies have a significant benefit in computational complexity. Bold indicates the best and underline indicates the second best result for each metric

tional complexity. Consequently, this approach emerges as a viable solution for continual learning in online applications dealing with stream-based data.

A reasonable future insight is the application of such methods in real-life scenarios where Edge devices are utilized, such as automotive solutions, search & rescue operations and the health industry. In all the aforementioned domains, smart devices have access to an infinite amount of data whilst having a limited amount of memory and processing power. Our methods could provide a decent balance of computational efficiency and classification accuracy.

Our work does come with some limitations. We have yet to outperform some of the methods which work with known task boundaries and some of the batch processing methods. The decrease of computations and memory usage do play a significant part in the classification accuracy. Another issue is the fact that some Edge devices cannot provide even a small amount of memory or support training computations.

Some practical hurdles might also occur in terms of training with questionable ground truth, such as driver input in the case of recommendation systems in driving. Finally safe-critical applications such as medical diagnosis do not allow the same margins of misclassification as of our methods, since medical accuracy is more important than hardware usage optimizations.

Acknowledgements The publication of the article in OA mode was financially supported by HEAL-Link.

Author Contributions All authors contributed to the study conception, design, material preparation, data collection and analysis

Funding Open access funding provided by HEAL-Link Greece.

Data Availability The data that support the findings of this study are available in: https://www.cs.toronto.edu/ kriz/cifar.html, https://www.tensorflow.org/datasets/catalog/cifar10, http://yann.lecun.com/exdb/mnist,https://www.tensorflow.org/datasets/catalog/mnist with the identifier(s) https://doi.org/10.1109/5.726791 for the MNIST digits dataset.

Declarations

Competing of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical and informed consent for data used All data have been used with ethical and informed consent

References

1. She Q, Feng F, Hao X, Yang Q, Lan C, Lomonaco V, Shi X, Wang Z, Guo Y, Zhang Y et al (2020) Openloris-object: a robotic vision dataset and benchmark for lifelong deep learning. In: 2020 IEEE international conference on robotics and automation (ICRA), pp 4767–4773. https://doi.org/10.1109/ICRA40945.2020.9196887. IEEE
2. Fernando C, Banarse D, Blundell C, Zwols Y, Ha D, Rusu AA, Pritzel A, Wierstra D (2017) Pathnet: evolution channels gradient descent in super neural networks. https://doi.org/10.48550/arXiv.1701.08734
3. Li Z, Hoiem D (2017) Learning without forgetting. IEEE Trans Pattern Anal Mach Intell 40(12):2935–2947. https://doi.org/10.48550/arXiv.1606.09282
4. McCloskey M, Cohen NJ (1989) Catastrophic interference in connectionist networks: the sequential learning problem. Psychol Learn Motiv 24:109–165. https://doi.org/10.1016/S0079-7421(08)60536-8
5. Bottou L, Bousquet O (2011) The tradeoffs of large-scale learning: optimization for machine learning. The MIT Press. https://doi.org/10.7551/mitpress/8996.003.0015
6. Grossberg ST (2012) Studies of mind and brain: neural principles of learning, perception, development, cognition, and motor control. Springer. https://doi.org/10.1007/978-94-009-7758-7
7. De Lange M, Aljundi R, Masana M, Parisot S, Jia X, Leonardis A, Slabaugh G, Tuytelaars T (2021) A continual learning survey: defying forgetting in classification tasks. IEEE Trans Pattern Anal Mach Intell 44(7):3366–3385. https://doi.org/10.1109/TPAMI.2021.3057446
8. Van de Ven GM, Tolias AS (2019) Three scenarios for continual learning. https://doi.org/10.48550/arXiv.1904.07734
9. He J, Mao R, Shao Z, Zhu F (2020) Incremental learning in online scenario. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13926–13935 . https://doi.org/10.1109/CVPR42600.2020.01394
10. Pellegrini L, Graffieti G, Lomonaco V, Maltoni D (2020) Latent replay for real-time continual learning. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 10203–10209. https://doi.org/10.1109/IROS45743.2020.9341460. IEEE
11. Davalas C, Michail D, Diou C, Varlamis I, Tserpes K (2022) Computationally efficient rehearsal for online continual learning. In: International conference on image analysis and processing, pp 39–49. https://doi.org/10.1007/978-3-031-06433-3_4. Springer
12. Goodfellow I.J, Mirza M, Xiao D, Courville A, Bengio Y (2013) An empirical investigation of catastrophic forgetting in gradient-based neural networks. https://doi.org/10.48550/arXiv.1312.6211
13. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
14. Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A et al (2017) Overcoming catastrophic forgetting in neural networks. Proc Natl Acad Sci 114(13):3521–3526. https://doi.org/10.1073/pnas.1611835114
15. Smith JS, Tian J, Halbe S, Hsu Y-C, Kira Z (2023) A closer look at rehearsal-free continual learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2409–2419. https://doi.org/10.1109/CVPRW59228.2023.00239

16. Zhuo T, Cheng Z, Gao Z, Fan H, Kankanhalli M (2023) Continual learning with strong experience replay. https://doi.org/10.48550/arXiv.2209.13917
17. Prabhu A, Torr PH, Dokania PK (2020) Gdumb: a simple approach that questions our progress in continual learning. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pp 524–540. https://doi.org/10.1007/978-3-030-58536-5_31. Springer
18. Chaudhry A, Rohrbach M, Elhoseiny M, Ajanthan T, Dokania P, Torr P, Ranzato M (2019) Continual learning with tiny episodic memories. In: Workshop on multi-task and lifelong reinforcement learning. https://doi.org/10.48550/arXiv.1902.10486
19. Rebuffi S-A, Kolesnikov A, Sperl G, Lampert CH (2017) icarl: incremental classifier and representation learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2001–2010. https://doi.org/10.1109/CVPR.2017.587
20. Rolnick D, Ahuja A, Schwarz J, Lillicrap T, Wayne G (2019) Experience replay for continual learning. Adv Neural Inf Process Syst 32. https://doi.org/10.48550/arXiv.1811.11682
21. Adam S, Busoniu L, Babuska R (2011) Experience replay for real-time reinforcement learning control. IEEE Trans Syst, Man, Cybern, Part C (Appl Rev) 42(2):201–212. https://doi.org/10.1109/TSMCC.2011.2106494
22. Lopez-Paz D, Ranzato M (2017) Gradient episodic memory for continual learning. Adv Neural Inf Process Syst 30. https://doi.org/10.48550/arXiv.1706.08840
23. Milan K, Veness J, Kirkpatrick J, Bowling M, Koop A, Hassabis D (2016) The forget-me-not process. Adv Neural Inf Process Syst 29. https://doi.org/10.5555/3157382.3157512
24. Wiewel F, Yang B (2021) Entropy-based sample selection for online continual learning. In: 2020 28th European signal processing conference (EUSIPCO), pp 1477–1481. https://doi.org/10.23919/Eusipco47968.2020.9287846. IEEE
25. Belouadah E, Popescu A, Kanellos I (2021) A comprehensive study of class incremental learning algorithms for visual tasks. Neural Netw 135:38–54. https://doi.org/10.1016/j.neunet.2020.12.003
26. Jodelet Q, Liu X, Murata T (2021) Balanced softmax cross-entropy for incremental learning. In: International conference on artificial neural networks, pp 385–396. https://doi.org/10.1016/j.cviu.2022.103582. Springer
27. Harun M.Y, Gallardo J, Kanan C (2023) Grasp: a rehearsal policy for efficient online continual learning. https://doi.org/10.48550/arXiv.2308.13646
28. Zhang Y, Pfahringer B, Frank E, Bifet A, Lim NJS, Jia Y (2022) A simple but strong baseline for online continual learning: repeated augmented rehearsal. Adv Neural Inf Process Syst 35:14771–14783. https://doi.org/10.48550/arXiv.2209.13917
29. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23:69–101. https://doi.org/10.1016/j.cviu.2022.103582
30. Ross GJ, Adams NM, Tasoulis DK, Hand DJ (2012) Exponentially weighted moving average charts for detecting concept drift. Pattern Recogn Lett 33(2):191–198. https://doi.org/10.1016/j.patrec.2011.08.019
31. Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2018) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363. https://doi.org/10.1109/TKDE.2018.2876857
32. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
33. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778. https://doi.org/10.13140/RG.2.2.33865.52329

Charalampos Davalas received his B.Sc. degree in Mathematics for the National and Kapodistrian University of Athens in 2016 and a M.Sc. degree in Web Engineering from Department of Informatics and Telematics, Harokopio University of Athens in 2018 He is currently a PhD Candidate and Research Associate with the Department of Informatics and Telematics, Harokopio University of Athens, Athens, Greece. His research is focused in the fields of Deep Learning and Big Data analysis. Research interests include Social Network de-anonymization testing with Big Data Graph Mining, Online Continual Learning in Computer Vision and Graph Neural Network Engineering for fire prediction on a global scale.

Dimitrios Michail is an Associate Professor at the Department of Informatics and Telematics at the Harokopio University of Athens. He holds a Diploma in Electronics and Computer Engineering from the Technical University of Crete, and an MSc (Computer Science) and PhD (Algorithms) from the Max-Planck Institute for Informatics, all with distinction. He has also conducted post-doctoral research at the Max- Planck Institute for Informatics in Germany as well as at the INRIA research institute in Sophia-Antipolis, France. His main research focuses on graph algorithms, graph mining, and graph representation learning. His expertise extends to machine learning techniques with an emphasis on computer vision and remote sensing. He has published numerous articles in top-rank international journals and conferences and has participated as a researcher in a number of R&D projects at European and national level.

Christos Diou received the Diploma in Electrical and Computer Engineering and the Ph.D. degree on the analysis of multimedia with machine learning from the Aristotle University of Thessaloniki, Thessaloniki, Greece. He is currently an Assistant Professor of Artificial Intelligence and Machine Learning with the Department of Informatics and Telematics, Harokopio University of Athens. His main research interests include robust representation learning capable of out-of-distribution generalization, methods for fair and interpretable machine learning, as well as the use of machine learning for the estimation of causal effects from observational data. He has published over 100 articles in international scientific journals and conferences and has more than 15 years of experience participating and leading European and national research projects, focusing on applications of artificial intelligence in healthcare.

Iraklis Varlamis received an M.Sc. degree in Information Systems Engineering from UMIST, U.K., and a Ph.D. from the Athens University of Economics and Business, Greece. He is a Professor in data management with the Department of Informatics and Telematics, at Harokopio University of Athens (HUA). His research interests range from data mining and social network analytics to recommender systems for social media and real-world applications. He has more than 200 articles published in international journals and conferences and more than 5000 citations on his work. He holds 2 patents from the Greek Patent Office for systems that crawl the web and thematically group web documents using content and links. He is the scientific coordinator for HUA in several EU (H2020, ECSEL, REC) projects and in national projects.

Konstantinos Tserpes is currently an Associate Professor with the Department of Informatics and Telematics, Harokopio University of Athens. His research interests revolve around intelligent methods for resolving resource allocation issues. He applies the findings of this research in designing and implementing software computing platforms that enable classes of novel applications with stringent requirements, such as ultra-low latency, big data stream processing, resourceconstrained execution, real-time performance, resilient and adaptive operation, etc. He has co-authored more than 100 publications in international scientific conferences and journals, and his work has been cited more than 2.5K times according to Google Scholar as of May 2024.He has been involved in several EU- and nationally-funded projects tackling research challenges in application domains such as multimedia, e-governance, post-production, finance, e-health, and others.