Repeated Augmented Rehearsal: A Simple but Strong Baseline for Online Continual Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Online continual learning (OCL) aims to train neural networks incrementally from a non-stationary data stream with a single pass through data. Rehearsal-based methods attempt to approximate the observed input distributions over time with a small memory and revisit them later to avoid forgetting. Despite its strong empirical performance, rehearsal methods still suffer from a poor approximation of past data's loss landscape with memory samples. This paper revisits the rehearsal dynamics in online settings. We provide theoretical insights on the inherent memory overfitting risk from the viewpoint of biased and dynamic empirical risk minimization, and examine the merits and limits of repeated rehearsal. Inspired by our analysis, a simple and intuitive baseline, Repeated Augmented Rehearsal (RAR), is designed to address the underfitting-overfitting dilemma of online rehearsal. Surprisingly, across four rather different OCL benchmarks, this simple baseline outperforms vanilla rehearsal by 9%-19% and also significantly improves state-of-the-art rehearsal-based methods MIR, ASER, and SCR. We also demonstrate that RAR successfully achieves an accurate approximation of the loss landscape of past data and high-loss ridge aversion in its learning trajectory. Extensive ablation studies are conducted to study the interplay between repeated and augmented rehearsal and reinforcement learning (RL) is applied to dynamically adjust the hyperparameters of RAR to balance the stability-plasticity trade-off online.

1 Introduction

Despite its recent success, deep learning largely relies on the assumption of independent and identically distributed (i.i.d.) data that can be repeatedly revisited during training. Non-i.i.d settings are challenging for neural networks due to catastrophic forgetting: previously learned knowledge can easily be overwritten when training on new data since this data may follow a different distribution [Li and Hoiem, 2017, Rebuffi et al., 2017, Delange et al., 2021]. Online continual learning (OCL or Online CL) studies how to enable deep learning in an online manner from a non-stationary data stream. As the data stream can be vast or even infinite, it is infeasible to store and shuffle the dataset for multiple epochs of training. Therefore, a fundamental assumption is that the data stream can only be accessed one batch at a time and training is performed with a *single pass* over the data [Aljundi et al., 2019].

Experience replay (ER), also known as rehearsal [Chaudhry et al., 2019, Delange et al., 2021], is a key method in OCL. It stores a subset of previously seen data $\mathcal{D}$ in a fixed-size memory $\mathcal{M}$ and revisits the memorized samples during training to mitigate forgetting of previous tasks. To update the model, a batch sampled from the memory is combined with the incoming batch from the stream to compute the gradient [Chaudhry et al., 2019]. Different variants of ER have been developed to improve

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

memory sampling policies and representation learning, achieving state-of-the-art performance in a number of standard OCL benchmarks [Aljundi et al., 2019, Mai et al., 2021, Shim et al., 2021].

However, whether rehearsal is appropriate for continual learning, considering the risk of overfitting the memory when using data from the memory to directly contribute to the gradient computation, has been debated vigorously [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2019, Verwimp et al., 2021]. The potential for overfitting has motivated the development of constraint-based replay methods [Lopez-Paz and Ranzato, 2017], e.g. GEM and A-GEM, which use memory samples solely to *constrain* the gradient direction. However, there is empirical evidence suggesting that rehearsal-based methods consistently outperform methods that do not train directly on the memory [Chaudhry et al., 2019]. This indicates rehearsal on the memory does not necessarily prevent effective generalization, possibly due to the implicit regularization effect of incoming data [Chaudhry et al., 2019]. Nevertheless, recent work analyzing the loss landscape when applying rehearsal to offline continual learning finds that memory samples indeed provide a poor approximation of the loss landscape past task's data, especially near a high-loss ridge region. As a result, "*instead of ending up near the high-loss ridge in perspective of the rehearsal memory, the solution in reality resides on the high-loss ridge for the training data*" [Verwimp et al., 2021]. This latest finding poses the question of how to better approximate the loss surface of past data $\mathcal{L}(\mathcal{D};\theta)$ with memory samples' loss $\mathcal{L}(\mathcal{M};\theta)$.

To better approximate past data's loss distribution, previous work studies which samples should be memorized for rehearsal. Instead, we study how to effectively perform rehearsal with the memorized samples. Focused on *online* CL, our study extends the previous understanding of rehearsal along two directions. First, we provide theoretical considerations that reveal two insights regarding the extent of overfitting to memory: it is a) related to the inherent attributes of the OCL problem concerned and b) varies across the different stages of continual learning. Second, we highlight the limits of applying rehearsal with multiple iterations—a trick used to maximally utilize the incoming batch [Aljundi et al., 2019]—and identify an underfitting-overfitting dilemma for online rehearsal.

Based on our analysis, we design a simple baseline to deal with the underfitting-overfitting dilemma in online CL problems, dubbed repeated augmented rehearsal (RAR), that can be easily integrated into existing rehearsal-based methods. Surprisingly, this simple baseline leads to a large performance boost for ER, as well as state-of-the-art ER-based approaches, across four OCL benchmarks. More importantly, the loss landscape analysis shows that RAR can help memory samples reliably approximate the distribution of past data and successfully avoids the high-loss ridge of past tasks. To better understand the behavior of RAR, we further investigate the interplay between repeated rehearsal and augmented rehearsal via an ablation study. We also propose a reinforcement learning-based method to dynamically adjust the hyperparameters of RAR and balance the stability-plasticity trade-off in an online manner.

2 Related Work

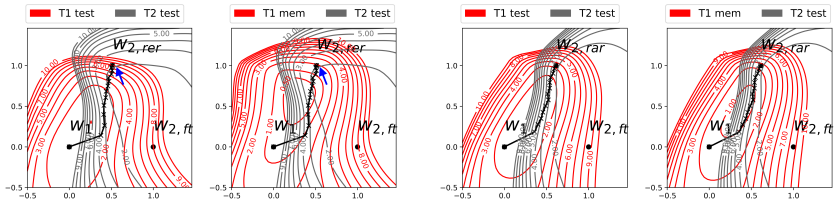
Online continual learning: We consider the online continual learning setting with a non-stationary (potentially infinite) stream of data: at each time step t, the continual learning agent receives an incoming batch of data samples $\mathcal{B}_t = \{\mathbf{x}_i, y_i\}_{i=1,..,|\mathcal{B}_t|}$ that are drawn from the current data distribution $\mathbb{D}_t$. The period of time where the data distribution stays the same is often called a *task* or *experience* in the continual learning literature. Abrupt change in the data distribution occurs when the task changes. The standard objective during training is to minimize the empirical risk on all the data seen so far:

$$\min_\theta \mathcal{R}(\theta) = \frac{1}{\sum_t |\mathcal{B}_t|} \Sigma_t \Sigma_{\mathbf{x},y\in\mathcal{B}_t} \mathcal{L}\left(f_\theta(\mathbf{x}), y\right) \tag{1}$$

with loss function $\mathcal{L}$, the CL network function f, and its associated parameters θ.

Metrics: A common metric is the end accuracy after training on T tasks, defined as $A_T = \frac{1}{T}\sum_{j=1}^{j=T} a_{T,j}$, where $a_{i,j}$ denotes the model's accuracy on the held-out test set of task j after training on task i. Other metrics are "forgetting" [Chaudhry et al., 2018], which is defined as $F_T = -\frac{1}{T-1}\sum_{i=1}^{T-1}\left(a_{T,i} - \max_{l\in 1...T-1} a_{l,i}\right)$ and the related metric "backward transfer" [Lopez-Paz and Ranzato, 2017]: $\mathrm{B_T} = \frac{1}{T-1}\sum_{i=1}^{T-1} a_{T,i} - a_{i,i}$.

Online setting: A key difference between online and offline CL is that the latter assumes full access to the whole training data for the task that is currently being processed. Therefore, it allows training on each single task with multiple epochs (e.g., 70-200 epochs) [Rebuffi et al., 2017, Wu et al., 2019,



(a) RER ($K = 10$) (b) RAR ($K = 10$)

Figure 1: Loss contours of RER and RAR. Memory data overfitting can be observed in (a) RER: Note how the loss contour of "T1 test" differs from "T1 memory" loss contour. At the CL solution points, test loss: 7.9 (left blue arrow) v.s. memory loss: 2.1 (right blue arrow).

Cha et al., 2021]. The online CL setting is more challenging because the agent can only access the current batch of incoming data and performs training with a single pass through the data.

Experience Replay (Rehearsal): Chaudhry et al. (2019) propose experience replay (ER), which performs joint training on memory samples and incoming samples. A simple but strong baseline approach to sampling in ER is reservoir sampling Vitter [1985]. Aljundi et al. (2019) propose Maximally Interfered Retrieval (MIR), which retrieves the samples that will be most negatively impacted by the foreseen parameters update. Shim et al. (2021) propose ASER, which selects samples to best preserve existing memory-based class boundaries. In terms of model training, Mai et al. (2021) propose to replace the cross-entropy loss with supervised contrastive loss to learn a better representation. We consider all these variants of ER in our experiments.

Augmentation: In the standard i.i.d. setting, data augmentation is a widely used method to improve deep learning [Cubuk et al., 2020]. In the offline CL setting, Mai et al. [2021] uses augmentation to construct a supervised contrastive loss and Bang et al. [2021] employs it together with an uncertainty-based memory management strategy. However, these papers apply augmentation together with other advanced techniques, and there is no ablation study on the effect of augmentation per se. Thus, it is unclear whether augmentation itself helps rehearsal or not, especially in the online setting.

Repeated Rehearsal (Multiple Iterations): Vanilla online continual learning employs a single gradient update given an incoming batch of data. To maximally utilize the current incoming batch, Aljundi et al. [2019] proposes to perform multiple gradient updates instead. Their experiment with CIFAR10 shows using MIR with 5 iterations leads to a slight accuracy improvement of 1.7%. In this paper, we systematically analyze the effect of multiple iterations on online rehearsal and provide theoretical and empirical insights on when this trick may improve or harm performance.

Hyperparameter Tuning for OCL: Hyperparameter tuning is a particular challenge in OCL due to the lack of a dedicated validation set and the constraint of a single pass through the data. Chaudhry et al. [2018] and Mai et al. [2022] employ a hyperparameter tuning protocol that uses an external validation data stream with a small number of tasks. Offline hyperparameter tuning is applied to this validation data with multiple passes to identify optimal values, which are then used for the actual online continual learning tasks. A limitation of this method is that it relies on external validation data.

3 Revisiting Online Rehearsal: Is Repeated Rehearsal a Good Idea?

We revisit rehearsal from two directions. First, while previous work demonstrates its strong empirical performance [Chaudhry et al., 2019] and provides conceptual analysis for *offline* CL [Verwimp et al., 2021], we focus on the *online* setting and provide theoretical insights through the lens of empirical risk minimization (ERM). Second, we examine the dynamics of employing rehearsal with multiple iterations. This has been proposed as a trick for online CL to maximally utilize the incoming batch already [Mai et al., 2022, Aljundi et al., 2019]; we investigate whether it is always better than rehearsal with a single iteration.

3.1 Empirical Risk Minimization in Online Rehearsal: a Biased and Dynamic Objective

For rehearsal in OCL, at each iteration t, a batch of data $\mathcal{B}_t$ is obtained from the incoming task, where $\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}$ and $\mathcal{B}_t = \{\mathbf{x}_i, y_i\}_{i=1,\dots|\mathcal{B}_t|}$, and a batch $\mathcal{B}_t^{\mathcal{M}}$ is sampled from memory, where $\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}^t$

and $\mathcal{B}_t^{\mathcal{M}} = \{\mathbf{x}_i, y_i\}_{i=1,\dots|\mathcal{B}_t^{\mathcal{M}}|}$. The gradient update rule of ER is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{|\mathcal{B}_t|}\Sigma_{\mathbf{x},y\in\mathcal{B}_t}\nabla\mathcal{L}\left(f_\theta(\mathbf{x}), y\right) - \frac{\eta}{|\mathcal{B}_t^{\mathcal{M}}|}\Sigma_{\mathbf{x},y\in\mathcal{B}_t^{\mathcal{M}}}\nabla\mathcal{L}\left(f_\theta(\mathbf{x}), y\right) \tag{2}$$

Given this update rule, we would like to establish the corresponding objective function, but this is not straightforward to derive because the memory is immediately updated after each incoming batch, which means $\mathcal{D}_{\mathcal{M}}^t$ changes all the time. For a widely used memory management policy, where the memory is updated using reservoir sampling [Vitter, 1985, Chaudhry et al., 2019], we prove in the appendix that the empirical risk for online ER follows Proposition 1.

Proposition 1 (ERM for online rehearsal): Assume an incoming task stream $\mathcal{D}_{\mathcal{T}}$ and an initial memory set $\mathcal{D}_{\mathcal{M}}^0$ with different data distribution $\mathbb{P}(\mathcal{D}_{\mathcal{T}}) \neq \mathbb{P}(\mathcal{D}_{\mathcal{M}})$. Assume further that the memory is updated at the end of each iteration using reservoir sampling. Then, Eq 2 implements unbiased stochastic gradient descent for the following loss function:

$$\mathcal{R}_t(\theta) = \Sigma_{\mathbf{x},y\in\mathcal{D}_{\mathcal{T}}}\mathcal{L}(f_\theta(\mathbf{x}), y) + \beta_t\lambda\Sigma_{\mathbf{x},y\in\mathcal{D}_{\mathcal{M}}^0}\mathcal{L}(f_\theta(\mathbf{x}), y) \tag{3}$$

where $\lambda := \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}^0|}$ and $|\mathcal{D}_{\mathcal{M}}^0|$ and $|\mathcal{D}_{\mathcal{T}}|$ are the memory size and incoming task data size; $\beta_t := 1/(1 + \frac{2N_{cur}^t}{N_{past}^{\mathcal{T}}})$, and $N_{cur}^t = \sum_{i=1}^{i=t}|\mathcal{B}_i|$ denotes the number of samples of the current task that have been seen so far and $N_{past}^{\mathcal{T}} = \sum_{j=1}^{j=\mathcal{T}}|\mathcal{D}_j|$ denotes the number of samples of past tasks. We immediately have $\beta_t \in (0, 1]$.

The proposition reveals several interesting properties: **1) Bias**: compared with the true objective in continual learning in Eq 1, the loss of online ER in Eq 3 is actually a biased approximation of the former, as it puts a different weight ($\beta_t\lambda$) on memory samples while the true objective treats all samples with equal weight. This bias, introduced by ER's objective function, can contribute to the risk of memory overfitting. **2) Problem-dependence**: given that $\beta_t \in (0, 1]$, the biased weight on the memory samples is heavily influenced by λ. In other words, the memory overfitting risk is related to an inherent property of the CL problem concerned: the ratio λ between the current task data size and the memory data size.While previous works extensively report empirical results on the influence of memory size, to our knowledge, we are the first to point out that the relative data size of an incoming task also plays an important role. Empirical evidence is provided in Section 6.3 to support this claim. With a 2k memory, performing rehearsal on the CORE50 dataset with a larger task data size ($\lambda = 6$) faces a high level of memory overfitting while the CLRS dataset with a smaller task data size ($\lambda = 1.12$) enjoys a lower risk of memory overfitting. **3) Dynamic**: ER in online CL optimizes towards a *dynamic* objective, which varies with each incoming batch t, as the weight on the memory sample depends on N_{cur}^t, and $N_{past}^{\mathcal{T}}$. This analysis shows that memory overfitting may be relatively slight for the first few tasks when $N_{past}^{\mathcal{T}}$ is small. As more tasks arrive, memory overfitting worsens as β_t increases with $N_{past}^{\mathcal{T}}$. In the case of an infinite data stream, $\lim_{N_{past}^{\mathcal{T}}\to\infty}\beta_t = 1$, the memory weight is solely determined by λ.

3.2 Repeated Rehearsal: The Decaying Regularization Effect of Incoming Data

We now investigate whether and why performing multiple iterations is indeed beneficial to rehearsal. We refer to applying multiple iterations in ER as "repeated experience replay" (Repeated ER) or "repeated rehearsal" and formalize it as follows: for each incoming data batch $\mathcal{B}_t \sim \mathcal{P}_t$ from the data stream, we perform multiple gradient updates (K in total) using stochastic gradient descent (SGD) or variants thereof. At each gradient update $k = 1, ...K$, a data batch $\mathcal{B}_{t,k}^{\mathcal{M}}$ is chosen from memory $\mathcal{M}$ and concatenated with the current incoming batch B_t to perform a replay iteration as follows.

$$\theta_{t,k+1} = \theta_{t,k} - \frac{\eta}{|\mathcal{B}_t|}\Sigma_{\mathbf{x},y\in\mathcal{B}_t}\nabla\mathcal{L}\left(f_{\theta_{t,k}}(\mathbf{x}), y\right) - \frac{\eta}{|\mathcal{B}_{t,k}^{\mathcal{M}}|}\Sigma_{\mathbf{x},y\in\mathcal{B}_{t,k}^{\mathcal{M}}}\nabla\mathcal{L}\left(f_{\theta_{t,k}}(\mathbf{x}), y\right) \tag{4}$$

Note that when $k = 1$, this update rule is an unbiased stochastic gradient for Eq3; in the case of $k > 1$, it is a biased gradient estimate as the same incoming batch is used for consecutive gradient updates. To study the influence of this biased gradient update in repeated rehearsal, we examine the internal dynamics of repeated rehearsal by studying the loss landscape.

Memory Overfitting: We compare the loss surfaces regarding the memory samples and the test data of past tasks during repeated rehearsal. Following the visualization method used in Verwimp et al. [2021], Mirzadeh et al. [2020], we examine the learning process on the first two tasks in the Split

Mini-ImageNet dataset and plot the loss landscape in the 2D plane defined by three model parameter vectors[1]: the model w_1 obtained by training on the first task until convergence, the model w_2 obtained on the second task using experience replay, and the model $w_{2,ft}$ obtained after training on the second task using finetuning without replay. Verwimp et al. (2021) use this method to demonstrate the memory overfitting in ER for *offline* continual learning setting with 10 epochs. Our results show that applying ER in the *online* setting yields severe memory overfitting for 10 iterations (see Fig 1 (a)). In other words, increasing the number of iterations means the loss landscape of the rehearsal memory provides a poorer approximation of the loss landscape of previous tasks, as shown by the differences in the positions and the shapes between the left and right red contours in Fig 1 (a). As a result, the learning trajectory of R-ER avoids the high-loss ridge region for the memory data but goes right into the high-loss ridge region for the past tasks' test data.

Regularization Effect of Incoming Data: To investigate why repeated rehearsal may suffer from even more memory overfitting than vanilla rehearsal, we analyze the regularization effects of incoming data during repeated ER. To this end, we examine the training process given an incoming batch and compare the training loss on the memory batch and incoming batch with respect to memory iteration k (see Fig 2). An interesting observation is that during the training session of a given incoming data batch, the decrease in the training loss on the incoming batch is much faster than the decrease in loss on the memory batches. One intuitive explanation is that the former is computed over a fixed batch during multiple iterations and the latter is computed over different memory batch samples. As a result, even though the incoming loss is larger than the memory loss at the start of a training session ($k = 1$), at later iterations (i.e., $k > 5$) the training loss of the incoming batch becomes $10 - 10^2$ times lower than that of the memory batch. This means that at this stage the regularization effect of the incoming data batch is greatly undermined: the joint training on the memory batch and the incoming batch becomes similar to training on memory only.

In summary, our findings imply that the performance of online rehearsal is constrained by the dilemma between overfitting locally and underfitting globally. Specifically, online rehearsal faces the challenge of underfitting of the large data stream but overfitting of a small memorized data subset. Applying repeated rehearsal ameliorates the former problem but aggravate the latter problem. Therefore, the performance gain of repeated rehearsal is quite limited.

4 Repeated Augmented Rehearsal

To deal with the overfitting-underfitting dilemma, we explore a simple strategy, "repeated augmented rehearsal" (RAR), which combines repeated rehearsal with data augmentation. Consider a group of transforms G that acts on the input space $\mathcal{X}$ and is invariant under function f, i.e., $f(g\mathbf{x}) = f(\mathbf{x}), g \in G, \mathbf{x} \in \mathcal{X}$. Given an incoming batch from the data stream, $\mathcal{B}_t$, multiple replay iterations are conducted using this batch. At each replay iteration n, a random memory batch $\mathcal{B}_{t,k}^{\mathcal{M}}$ is sampled and concatenated with the incoming batch. Then, a random transform $g_{k,t} \in G$ is sampled and applied to each data point $\mathbf{x}_i$ in the concatenated minibatch. The model parameters are updated as : $g_{rar} = \frac{1}{|\mathcal{B}_t|} \sum_{\mathbf{x},y \in \mathcal{B}_t} \nabla \mathcal{L}\left(f_\theta(g_{t,k}\mathbf{x}), y\right) + \frac{1}{|\mathcal{B}_{t,k}^{\mathcal{M}}|} \sum_{\mathbf{x},y \in \mathcal{B}_{t,k}^{\mathcal{M}}} \nabla \mathcal{L}\left(f_\theta(g_{t,n}\mathbf{x}), y\right)$. Intuitively, the augmentation can help alleviate memory overfitting in two ways. First, we observe applying augmentation on the incoming batch helps strengthen the regularization effect. As shown in Fig 2(b), the decaying regularization effect of incoming data is alleviated in RAR, as the loss of the incoming batch stays comparable to the loss of memory batch during multiple iterations. Second, rehearsal on augmented memory batch can help depict a more accurate past tasks' data distribution. With RAR, the loss landscapes of memory data and past tasks' test data become very similar (see Fig 1 (b)), which suggests that the model ends up in a part of the parameter space where the rehearsal memory approximates the past tasks' distribution well. Moreover, the continual learning solution identified with RAR avoids the high-loss ridge not only in the memory data loss landscape but also in the test data loss landscape.

Theoretically, we prove that augmented rehearsal reduces the generalization error in OCL. Specifically, assume the augmentation group G is a compact topological group and follows a probability distribution $\mathbb{Q}$. Similar to **Proposition 1**, it can be easily proven that the augmented rehearsal gradient

[1]When training w_2 and $w_{2,ft}$, the model is initialized from w_1. The memory contains 100 samples/task (see Fig 1 (b) left and right). The 2-d coordinate system is built by orthogonalizing $w_2 - w_1$ and $w_{2,ft} - w_1$.

Algorithm 1: RL-based RAR

$\mathcal{M}$ is a memory with fixed size,
$\mathcal{B}_t$ is the incoming batch data from current task,
θ are the parameters of CL network,
w is the parameter of RL agent,
K is the number of memory iterations,
P, Q are the augmentation hyperparameters

1: **procedure** RAR($\mathcal{M}_t$, $\mathcal{B}_t$, θ_t, w_t)
2: $K_t, P_t, Q_t = SampleAction(w_t)$
3: **for** $k = 1, ..., K_t$ **do**
4: $\mathcal{B}^{\mathcal{M}}_{t,k} \sim MemRetrieval(\mathcal{M}_t)$
5: $\mathcal{B}_{aug} \leftarrow aug(\mathcal{B}^{\mathcal{M}}_{t,k} \cup \mathcal{B}_t, P_t, Q_t)$
6: $r_t \leftarrow ComputeReward(\mathcal{B}_{aug}, \theta_{t,k})$
7: $\theta_{t,k+1} \leftarrow SGD(\mathcal{B}_{aug}, \theta_{t,k})$
8: **end for**
9: $\mathcal{M}_{t+1} \leftarrow MemUpdate(\mathcal{M}_t, \mathcal{B}_t)$
10: $w_{t+1} \leftarrow UpdateRL(r_t)$
11: **end procedure**

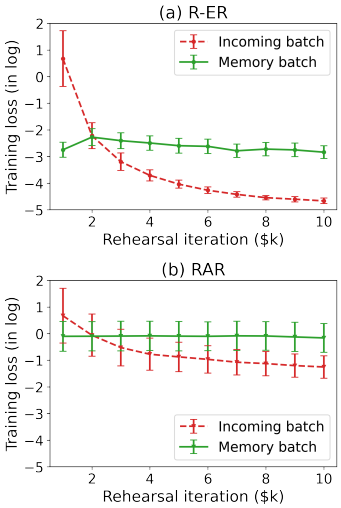


Figure 2: Memory loss vs. incoming loss.

corresponds to unbiased SGD on an augmented empirical risk (see **Proposition 3** in Appendix A):

$$\bar{\mathcal{R}}_t(\theta) = \Sigma_{\mathbf{x},y\in\mathcal{D}_\mathcal{T}} \int_G \mathcal{L}(f_\theta(g\mathbf{x}), y) d\mathbb{Q}(g) + \beta_t \lambda \Sigma_{\mathbf{x},y\in\mathcal{D}_\mathcal{M}} \int_G \mathcal{L}(f_\theta(g\mathbf{x}), y) d\mathbb{Q}(g) \quad (5)$$

This result shows that applying augmented rehearsal is equivalent to performing an averaging operation of the loss of rehearsal in Eq 3 over the orbits of a certain group that keeps the data distribution approximately invariant. In the standard i.i.d. learning setting, Chen et al. [2020] found that such an orbit-averaging operation can reduce both the variance and generalization error. Based on Eq 3 and Eq 5, we show that this theoretical benefit of using augmentation to boost model invariance is also applicable to rehearsal in continual learning. In fact, as discussed in Section 3.1, rehearsal in online CL has a biased empirical risk Eq 3, which leads to inherent memory overfitting and poor generalization ability. Thus, this benefit of augmentation in reducing generalization error is particularly important when applying rehearsal in online CL.

The modifications required for the RAR procedure are summarized in Lines 3 and 5 of Algorithm 2. It uses a general framework for ER-based continual learning that consists of three key components: sampling from memory, joint training of memory data and incoming data, and updating of the memory. As mentioned in the related work section, different ER variants have been proposed to improve these components. RAR can flexibly be combined with any of these ER variants, and we investigate the effectiveness of RAR on these different ER variants in the experiment section.

5 Reinforcement Learning-based Adaptive Repeated Augmented Rehearsal

There are two key components in RAR: repeated rehearsal and augmented rehearsal. The interplay of the two is determined by the number of memory iterations and the strength of augmentation. A key question is how to choose these hyperparameters. In general, hyperparameter tuning (HPT) still remains an unsolved challenge for online CL due to the single-pass assumption [Chaudhry et al., 2018]. Finding suitable RAR hyperparameters needs to account for the severity of memory overfitting and poses extra challenges. In particular, as shown in the ERM analysis, the extent of memory overfitting is related to the CL problem features (e.g., task data size and memory size) and also varies at different training stages of the continual learning process. To automatically select suitable RAR hyperparameters for the different CL problems and different training stages, we propose to use reinforcement learning to adaptively adjust the hyperparameters (see Algorithm 2).

In particular, we design the hyperparameter of RAR as the action space and use the training statistics as the reward (see line 2, 6, 10 in Algorithm 1). A major challenge of applying RL in online HPT is sample efficiency. The exploration horizon (i.e training steps) in the OCL environment is quite limited due to the constraint of single-pass of data and poor action choices (undesirable hyperparameters) may lead to bad gradient update step and hurt the OCL training process. To address the sample efficiency

Table 1: Accuracy on four OCL benchmarks with 2k and 5k memory.

	Seq-CIFAR100		Seq-MINI-IMAGENET		CORE50-NC		CLRS25-NC	
	2K	5K	2K	5K	2K	5K	2K	5K
Finetune	3.2 ± 0.1		4.3 ± 0.8		7.7 ± 0.2		6.5 ± 0.9	
LWF	8.7 ± 0.5		10.9 ± 0.5		9.6 ± 0.3		12.4 ± 2.2	
AGEM	8.5 ± 0.4	9.2 ± 0.2	11.6 ± 0.1	13.1 ± 0.3	18.6 ± 0.4	19.4 ± 1.8	14.6 ± 1.4	14.4 ± 0.3
ER	19.0 ± 0.6	26.2 ± 0.2	20.0 ± 0.8	23.0 ± 0.6	24.0 ± 2.0	27.8 ± 0.2	18.7 ± 1.6	19.2 ± 0.3
ER-RAR	27.8 ± 0.5	36.2 ± 0.7	30.0 ± 0.9	36.5 ± 0.4	39.3 ± 1.4	45.0 ± 2.7	28.6 ± 2.7	28.9 ± 1.5
MIR	18.4 ± 0.8	25.7 ± 1.8	19.4 ± 0.6	22.3 ± 0.2	25.2 ± 1.3	26.9 ± 0.9	14.3 ± 3.6	15.2 ± 3.0
MIR-RAR	27.5 ± 0.2	36.1 ± 0.3	29.5 ± 0.6	34.9 ± 0.7	39.1 ± 1.0	44.6 ± 1.7	27.8 ± 1.6	29.2 ± 2.6
ASER	20.9 ± 0.3	24.3 ± 2.0	15.7 ± 0.1	17.5 ± 0.7	16.4 ± 1.4	16.7 ± 2.3	19.4 ± 1.3	19.7 ± 1.4
ASER-RAR	28.1 ± 0.3	35.8 ± 1.0	27.0 ± 0.3	32.2 ± 0.6	24.2 ± 0.4	30.0 ± 1.6	28.7 ± 0.2	29.5 ± 0.2
SCR	32.0 ± 1.1	37.4 ± 0.2	29.7 ± 1.0	33.1 ± 1.9	45.1 ± 0.1	50.3 ± 1.9	23.5 ± 2.2	23.6 ± 3.0
SCR-RAR	37.1 ± 0.7	45.8 ± 0.2	35.4 ± 0.7	43.7 ± 0.4	53.4 ± 0.9	61.1 ± 1.1	37.4 ± 1.0	41.5 ± 0.9
ER_{RW}	21.0 ± 1.0	26.2 ± 0.2	20.1 ± 0.8	23.0 ± 0.6	24.6 ± 0.6	27.8 ± 0.8	19.2 ± 0.6	19.2 ± 0.3
ER_{RW}-RAR	30.8 ± 0.1	36.5 ± 0.4	30.4 ± 1.3	36.5 ± 0.4	45.3 ± 2.2	50.8 ± 0.9	28.6 ± 2.7	28.9 ± 1.5
DER	8.4 ± 0.6	9.1 ± 0.3	11.8 ± 0.5	12.3 ± 1.7	23.8 ± 0.6	23.4 ± 2.5	11.8 ± 2.6	19.7 ± 0.4
DER-RAR	30.0 ± 1.2	41.9 ± 0.5	26.2 ± 0.4	35.5 ± 1.5	37.7 ± 1.4	42.0 ± 3.7	28.4 ± 3.2	28.9 ± 1.5

Table 2: Accuracy of variants of RAR and different hyperparameter tuning methods.

	Seq-CIFAR100	Seq-Mini-ImageNet	CORE50-NC	CLRS25-NC
ER	19.0 ± 0.6	20.0 ± 0.8	26.5 ± 0.8	18.5 ± 1.8
RAR-MEM	25.4 ± 0.7	27.4 ± 0.8	38.6 ± 0.7	28.8 ± 1.0
RAR-INC	21.6 ± 0.2	24.5 ± 0.1	35.7 ± 1.1	29.1 ± 1.1
RAR-BOTH	27.8 ± 0.5	30.0 ± 0.9	39.3 ± 1.4	28.8 ± 1.5
RAR-HTOCL	23.4 ± 0.2	26.0 ± 0.2	40.8 ± 0.7	26.9 ± 0.5
RAR-RL	$\mathbf{29.6 \pm 0.4}$	$\mathbf{32.1 \pm 1.0}$	$\mathbf{44.4 \pm 0.8}$	$\mathbf{35.0 \pm 0.7}$

issue, we employ multi-arm bandit framework and apply bootstrapped policy gradient (BPG) [Zhang and Goh, 2019]. The key idea of BPG is to incorporate prior knowledge to bootstrap the policy gradient to achieve stable and fast convergence with limited samples. To obtain prior knowledge in OCL problem, we use the training accuracy on the memory batch as the stability-plasticity feedback, as a higher memory accuracy suggests a high level of stability. Reward is defined as the distance between the current memory accuracy and target memory accuracy. Compared against a target memory accuracy (e.g. 0.9), the current memory accuracy is used to indicate whether the current choice of rehearsal iteration or augmentation causes too much stability (and less plasticity) and then the action selection probability is adjusted according following BPG (see Appendix C.2 for more RL design and implementation details).

6 Experiments

6.1 Experiment Setup

Baseline: We apply RAR to four ER-based continual learning algorithms: ER [Chaudhry et al., 2019], MIR [Aljundi et al., 2019], ASER [Shim et al., 2021], and SCR [Mai et al., 2021]. We also compare it with other continual learning methods, including the regularization-based method LWF [Li and Hoiem, 2017] and the constrained optimization-based method A-GEM [Chaudhry et al., 2018].

Dataset: Four CL benchmarks are used in the experiments: Seq-CIFAR100 (20 tasks), Seq-MiniImageNet (10 tasks) Vinyals et al. [2016], CORE50-NC (9 tasks) Lomonaco and Maltoni [2017] and CLRS25-NC (5 tasks) Li et al. [2020](see Appendix B for more details). Additionally, we also investigate ER and RAR in a large-scale ImageNet-1k dataset in Appendix!D.4.

Implementation: All datasets are trained on a reduced ResNet-18 following Mai et al. [2021], Aljundi et al. [2019]. Single-head evaluation is employed with a shared final layer trained for all the tasks. RandAugmentation Cubuk et al. [2020] is used for auto augmentation. Given a set of augmentation operations, it randomly selects P augmentation operations and exerts an augmentation magnitude of Q for all the selected augmentation operations on each image. All the experimental results we present are the average of three runs. We summarize all hyperparameter details in Appendix C. Supplementary results using ResNet-18 can be found in Appendix D.5. The running time of different algorithms is shown in Appendix D.8.

6.2 Main Results

We first analyze RAR's performance with a pre-defined hyperparameter set ($K = 10, P = 1, Q = 14$).As shown in Table 1, RAR greatly improves the ER method on the four datasets, by 8.7% - 18.8%. Moreover, RAR also leads to substantial gains for the other algorithmic variants of ER for all datasets

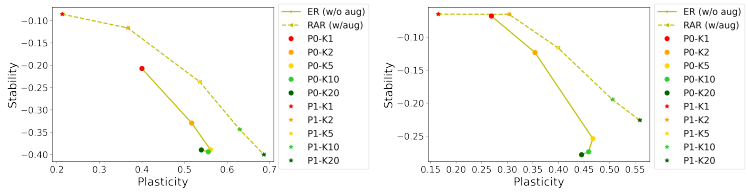


Figure 3: Stability and plasticity trade-off in RAR: CIFAR100 (left) and Mini-imagenet(right)

(MIR: 9.0% - 17.7%, ASER: 7.2% - 14.7%, SCR: 5.1% - 17.9%). These results suggest that even with advanced memory management, OCL still benefits a lot from repeated augmented rehearsal.

Stability and Plasticity Trade-off Based on the definition of accuracy A_T, forgetting F_T and backward transfer B_T in section 2, we find that these three metrics have the following relationship:

$$A_T = \underbrace{\frac{1}{T}\Sigma_{i=1}^{T} a_{i,i}}_{\text{plasticity}} + \underbrace{\frac{T-1}{T} B_T}_{\text{stability}} \geq \tfrac{1}{T}\Sigma_{i=1}^{T} a_{i,i} - \tfrac{T-1}{T} F_T.$$

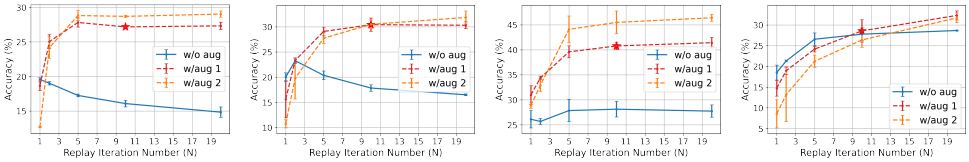
Interestingly, this finding shows that accuracy is related to the ability of learning the new task $\frac{1}{T}\sum_{i=1}^{T} a_{i,i}$ and the forgetting of past tasks, which draws a connection to the more general problem of the stability-plasticity trade-off in neural networks and continual learning [Grossberg, 2012, Delange et al., 2021]. Plasticity refers to the ability to integrate new knowledge and stability refers to the ability to retain old knowledge. Fig 3 presents the stability and plasticity trade-off in RAR. Generally, we observe increasing the repeated rehearsal iterations (K) leads to a higher level of plasticity. However, this may also cause a decrease of stability, i.e., introducing forgetting. On the other hand, the use of augmentation generally improves stability. More importantly, the use of augmentation in repeated rehearsal shifts the stability-plasticity trade-off curve towards the upper right thus creating a better stability-plasticity trade-off frontier.

Hyperparameter Tuning for RAR We compare the RL based hyperparameter tuning method with the hyperparameter tuning framework for continual learning (HTOCL) method used in Chaudhry et al. [2018], Mai et al. [2022] (see Section 2). The results in Table 9 show that RL-based RAR significantly outperforms using HTOCL to select hyperparameters for RAR. One reason is that the extent of memory overfitting varies at the different training stages of CL. HTOCL only uses the first few tasks to select hyperparameters for RAR, which may not optimal for later stages of CL. In fact, we observe HTOCL tends to select a large number of repeats and small augmentation strength. This selection strategy may be desirable for problems with short task sequences but problematic for long task sequences with increased memory overfitting risk. In contrast, RL-based method can take into account the latest feedback (e.g. train memory accuracy) to adjust hyperparameter choices. The selected iteration numbers and augmentation are shown in Appendix D.7.

RAR with Modified Rehearsal Loss Besides using the vanilla online rehearsal loss in Eq 3, we investigate the effectiveness of RAR with two advanced rehearsal loss designs: 1) **Reweighted memory loss (ER-rw):** one straightforward way to deal with the biased ER loss is to balance the weight of the memory loss and incoming loss by introducing a reweighting hyperparameter α in the gradient of Eq 2. 2) **Distillation-based memory loss**: DER [Buzzega et al., 2020] employs the logits-based distillation loss for memory samples, instead of the cross entropy loss. The results in Table 1 show RAR leads to large performance gains for ER-rw (for the best α choice; further results can be found in Appendix D.3)) and DER for all four datasets. This suggests that even with advanced rehearsal loss design, repeated augmented rehearsal is important for online rehearsal.

6.3 Ablation Studies

Interplay between Repeated and Augmented Rehearsal We investigate the interaction of the number of replay iterations N with the augmentation strength in Fig 4. For three out of four datasets, using augmentation with rehearsal leads to even worse performances than rehearsal without augmentation (see $K = 1$ in Fig 4). One explanation is the underfitting challenge of online CL. Training on augmented samples can make model underfitting even worse. The only exception is the CORE50 dataset, which inherently has a high memory overfitting risk with $\lambda = 6$ making it benefit



(a) CIFAR100 (b) MINI-IMAGENET (c) CORE50 (d) CLRS25

Figure 4: Effects of augmentation and rehearsal iterations (red stars: accuracy in Table 1).

more from augmentation. Similarly, employing repeated rehearsal alone also harms performance in three out of four datasets, with the CLRS dataset as the only exception, which enjoys a low memory overfitting risk with $\lambda = 1.12$. An important takeaway message is that repeated rehearsal or augmented rehearsal is not always helpful in OCL settings and whether they will benefit or harm the performance is dependent on the structure of the OCL problem at hand (e.g., the task data size and memory data size).

RAR's Robustness to Large Numbers of Repeats Although the performance curve flattens out around 10 iterations, there is no evident drop in performance even with 20 iterations. This result reinforces how RAR can help with the underfitting-overfitting dilemma: it can support the use of more training in OCL without having to worry about the performance drop introduced by memory overfitting. In comparison, for repeated rehearsal without augmentation (solid lines), the accuracy starts to drop quickly when using more than two iterations in CIFAR100.

Augmenting the Memory vs. Augmenting the Incoming Data RAR applies augmentation to both the memory batch and the incoming batch. We examine the effectiveness of the two separately. Table 2 presents the performance of RAR applied (a) solely with memory augmentation, RAR-mem and (b) solely with incoming data augmentation, RAR-inc. We find that RAR achieves the best performance compared to RAR-mem and RAR-inc. This shows that it is beneficial to apply augmentation to both the memory batch and the incoming batch. Interestingly, RAR-inc itself also achieves consistent performance improvements over R-ER, e.g. a gain of 7.7% in CORE50. As discussed in Section 3, adding augmentation on the incoming batch can strengthen the regularization effect of the incoming task and indirectly alleviate memory overfitting.

RAR with MIR, ASER, SCR We also perform ablation studies to investigate RAR's strong performance with the ER variants (see Appendix D.2). Similar to ER, the performance gains for RAR-MIR and RAR-ASER come from the combination of repeated rehearsal and augmented rehearsal. However, for SCR, the repeated rehearsal itself also leads to a consistent performance boost. One reason is that SCR already includes a strong augmentation procedure with four augmentation operations to construct the supervised contrastive loss used in this method. It also works poorly without augmentation.

Augmentation for Offline Rehearsal Although this paper focuses on online CL, the (augmented) empirical risk analysis is also relevant to offline CL (see Proposition 2 and 3 in Appendix A). This suggests the memory overfitting risk in offline rehearsal is related to the task-to-memory size ratio λ and can be alleviated by augmented rehearsal (see more empirical results in Table 9 of Appendix D.9).

7 Discussion and Conclusion

Rehearsal-based methods play a central role in fighting catastrophic forgetting when learning from non-stationary data streams. Compared to offline rehearsal, online rehearsal has faced particular challenges in tackling complex CL datasets due to the single-pass-through data constraint. This work tries to analyze the internal workings of online rehearsal from theoretical and conceptual perspective and identifies the fundamental challenge that it faces as the dilemma between overfitting locally and underfitting globally. To deal with this challenge, we propose a simple baseline: repeated and augmented rehearsal (RAR). Surprisingly, despite its simplicity, RAR achieves a large performance boost for a set of different rehearsal-based methods. Additionally, we propose a RL-based method to tune the hyperparameters of RAR to balance the stability-plasticity trade-off online. It achieves promising results compared to the validation data based hyperparameter tuning. A remaining question is how to apply RL to tune different types of OCL hyperparameters of jointly. Also, this work is focused on the continual learning for classification problems with image data. An interesting future research direction is to look at other CL domains, e.g., text/audio inputs or RL problems.

References

R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems*, 32:11849–11860, 2019.

J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8227, 2021.

P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

H. Cha, J. Lee, and J. Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525, 2021.

A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

S. Chen, E. Dobriban, and J. H. Lee. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71, 2020.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

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.

S. T. Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media, 2012.

H. Li, H. Jiang, X. Gu, J. Peng, W. Li, L. Hong, and C. Tao. Clrs: Continual learning benchmark for remote sensing image scene classification. *Sensors*, 20(4):1226, 2020.

Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Z. Mai, R. Li, H. Kim, and S. Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3589–3599, 2021.

Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

S. I. Mirzadeh, M. Farajtabar, D. Gorur, R. Pascanu, and H. Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2020.

S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021.

E. Verwimp, M. De Lange, and T. Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9385–9394, 2021.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

Y. Zhang and W.-B. Goh. Bootstrapped policy gradient for difficulty adaptation in intelligent tutoring systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 711–719, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Theoretical Analysis

This section contains the theoretical analysis of the loss function of offline experience replay (**Proposition 2**), augmented experience replay (**Proposition 3**) and online experience replay with reservoir sampling (**Proposition 1**).

Proposition 2 (Empirical risk minimization for experience replay): We assume a memory set $\mathcal{D}_{\mathcal{M}}$ and an incoming task stream $\mathcal{D}_{\mathcal{T}}$ with different data distribution $\mathbb{P}(\mathcal{D}_{\mathcal{T}}) \neq \mathbb{P}(\mathcal{D}_{\mathcal{M}})$. At each iteration t, a batch of data is sampled from memory $\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}$ and $\mathcal{B}_t^{\mathcal{M}} = \{\mathbf{x}_i, y_i\}_{i=1,\ldots|\mathcal{B}_t^{\mathcal{M}}|}$, and a batch of data from incoming task $\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}$ and $\mathcal{B}_t = \{\mathbf{x}_i, y_i\}_{i=1,\ldots|\mathcal{B}_t|}$. To update the parameter θ of a function mapping f_θ: $f_\theta \times \mathcal{X} \rightarrow \mathcal{Y}$ based on loss function $\mathcal{L}$, consider a parameter update rule defined by $\theta = \theta - \frac{\eta}{|\mathcal{B}_t|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t} \nabla\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) - \frac{\eta}{|\mathcal{B}_t^{\mathcal{M}}|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t^{\mathcal{M}}} \nabla\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)$, then it is an unbiased stochastic gradient descent for the following empirical risk:

$$\mathcal{R}(\theta) = \sum_{x_i, y_i \in \mathcal{D}_{\mathcal{T}}} \mathcal{L}(f_\theta(\mathbf{x}_i), y_i) + \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}|}\sum_{x_i, y_i \in \mathcal{D}_{\mathcal{M}}} \mathcal{L}(f_\theta(\mathbf{x}_i), y_i).$$

Proof: Given the empirical gradient $\hat{g}_{ER} = \frac{1}{|\mathcal{B}_t|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t} \nabla\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) + \frac{1}{|\mathcal{B}_t^{\mathcal{M}}|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t^{\mathcal{M}}} \nabla\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)$ during stochastic optimization, we can derive the gradient expectation as follows:

$$\begin{aligned}
\mathbb{E}_{\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}, \mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}}[\hat{g}_{ER}] &= \nabla(\mathbb{E}_{\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}}[\frac{1}{|\mathcal{B}_t^{\mathcal{M}}|}\sum_{\mathcal{B}_t^{\mathcal{M}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] + \mathbb{E}_{\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}}[\frac{1}{|\mathcal{B}_t|}\sum_{\mathcal{B}_t} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)]) \\
&= \nabla(\mathbb{E}_{x_i, y_i \sim \mathcal{D}_{\mathcal{M}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] + \mathbb{E}_{x_i, y_i \sim \mathcal{D}_{\mathcal{T}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)]) \\
&= \nabla(\frac{1}{|\mathcal{D}^{\mathcal{M}}|}\sum_{\mathcal{D}_{\mathcal{M}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) + \frac{1}{|\mathcal{D}_{\mathcal{T}}|}\sum_{\mathcal{D}_{\mathcal{T}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)) \\
&= \frac{1}{|\mathcal{D}_{\mathcal{T}}|}\nabla\left(\frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}|}\sum_{\mathcal{D}_{\mathcal{M}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) + \sum_{\mathcal{D}_{\mathcal{T}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)\right)
\end{aligned}$$

Proposition 3 (Augmented empirical risk minimization for experience replay): Assume a memory set $\mathcal{D}_{\mathcal{M}} = \{\mathbf{x}_i, y_i\}_{i=1,..,|\mathcal{D}_{\mathcal{M}}|}$ and an incoming task stream $\mathcal{D}_{\mathcal{T}} = \{\mathbf{x}_i, y_i\}_{i=1,..,|\mathcal{D}_{\mathcal{T}}|}$ with different data distribution $\mathbb{P}(\mathcal{D}_{\mathcal{T}}) \neq \mathbb{P}(\mathcal{D}_{\mathcal{M}})$, and a compact topological group of transform G with a probability distribution of $\mathbb{Q}$ that acts on the input space $\mathcal{X}$ and is invariant under function f, i.e. $f(gx) = f(x), g \in G, x \in \mathcal{X}$. At each iteration t, a batch data is sampled from memory $\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}$ and $\mathcal{B}_t^{\mathcal{M}} = \{\mathbf{x}_i, y_i\}_{i=1,\ldots|\mathcal{B}_t^{\mathcal{M}}|}$, a batch of data from incoming task $\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}$ and $\mathcal{B}_t = \{\mathbf{x}_i, y_i\}_{i=1,\ldots|\mathcal{B}_t|}$, and a randomly selected group operation $g_t \sim G$. To update the parameter θ of the function f_θ based on loss function $\mathcal{L}$, consider a parameter update rule defined by $\theta = \theta - \frac{\eta}{|\mathcal{B}_t|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t} \nabla\mathcal{L}\left(f_\theta(g_t\mathbf{x}_i), y_i\right) - \frac{\eta}{|\mathcal{B}_t^{\mathcal{M}}|}\sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t^{\mathcal{M}}} \nabla\mathcal{L}\left(f_\theta(g_t\mathbf{x}_i), y_i\right)$, then it is an unbiased stochastic gradient descent for the following loss function

$$\bar{\mathcal{R}}(\theta) = \sum_{x_i, y_i \in \mathcal{D}_{\mathcal{T}}} \int_G \mathcal{L}(f_\theta(gx_i), y_i) d\mathbb{Q}(g) + \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}|}\sum_{x_i, y_i \in \mathcal{D}_{\mathcal{M}}} \int_G \mathcal{L}(f_\theta(gx_i), y_i) d\mathbb{Q}(g).$$

Proof:

Given the augmented empirical gradient

during stochastic optimization, we can derive the gradient expectation as follows:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}, \mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}, g \sim \mathbb{Q}}[\hat{g}] \\
&= \nabla(\mathbb{E}_{\mathcal{B}_t^{\mathcal{M}} \sim \mathcal{D}_{\mathcal{M}}}[\frac{1}{|\mathcal{B}_t^{\mathcal{M}}|}\sum_{\mathcal{B}_t^{\mathcal{M}}} \int_g \mathcal{L}\left(f_\theta(g\mathbf{x}_i), y_i\right) d\mathbb{Q}(g)] + \mathbb{E}_{\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}}[\frac{1}{|\mathcal{B}_t|}\sum_{\mathcal{B}_t} \int_g \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) d\mathbb{Q}(g)]) \\
&= \frac{1}{|\mathcal{D}_{\mathcal{T}}|}\nabla\left(\sum_{x_i, y_i \in \mathcal{D}_{\mathcal{T}}} \int_G \mathcal{L}(f_\theta(gx_i), y_i) d\mathbb{Q}(g) + \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}|}\sum_{x_i, y_i \in \mathcal{D}_{\mathcal{M}}} \int_G \mathcal{L}(f_\theta(gx_i), y_i) d\mathbb{Q}(g)\right)
\end{aligned}$$

The second equality is based on the results of proposition 1.

Proposition 1 (Empirical risk minimization for online experience replay): Assume an initial memory set $\mathcal{D}^0_{\mathcal{M}}$ and an incoming task stream $\mathcal{D}_{\mathcal{T}}$ with different data distribution $\mathbb{P}(\mathcal{D}_{\mathcal{T}}) \neq \mathbb{P}(\mathcal{D}_{\mathcal{M}})$. At each iteration t,$t = 1, ..T$, a batch of data from incoming task $\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}$ and $\mathcal{B}_t = \{\mathbf{x}_i, y_i\}_{i=1,...|\mathcal{B}_t|}$ and a batch data is sampled from memory $\mathcal{B}^{\mathcal{M}}_t \sim \mathcal{D}^t_{\mathcal{M}}$ and $\mathcal{B}^{\mathcal{M}}_t = \{\mathbf{x}_i, y_i\}_{i=1,...|\mathcal{B}^{\mathcal{M}}_t|}$. To update the parameter θ of a function mapping f_θ: $f_\theta \times \mathcal{X} \rightarrow \mathcal{Y}$ based on loss function $\mathcal{L}$, consider a parameter update rule defined by $\theta = \theta - \frac{\eta}{|\mathcal{B}_t|} \sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t} \nabla \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) - \frac{\eta}{|\mathcal{B}^{\mathcal{M}}_t|} \sum_{\mathbf{x}_i, y_i \in \mathcal{B}^{\mathcal{M}}_t} \nabla \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)$. Memory is updated at the end of each iteration using reservoir sampling Vitter [1985], Chaudhry et al. [2019] $\mathcal{D}^{t+1}_{\mathcal{M}} \leftarrow RS(\mathcal{D}^t_{\mathcal{M}}, \mathcal{B}_t)$. Then the gradient is an unbiased stochastic gradient descent for the following loss function

$$\mathcal{R}_t(\theta) = \sum_{x_i, y_i \in \mathcal{D}_{\mathcal{T}}} \mathcal{L}(f_\theta(x_i), y_i) + \beta_t \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}^0_{\mathcal{M}}|} \sum_{x_i, y_i \in \mathcal{D}^0_{\mathcal{M}}} \mathcal{L}(f_\theta(x_i), y_i)$$

where $\beta_t = \frac{1}{1+2*\frac{N^t_{cur}}{N_{past}}}$, $N^t_{cur} = \sum_{i=1}^{i=t} |\mathcal{B}_i|$ denotes the number of samples of current task that have been seen so far and $N_{past} = \sum_{j=1}^{j=\mathcal{T}} |\mathcal{D}_j|$ denotes the number of samples of past tasks.

Note 1: The objective function $\mathcal{R}_t(\theta)$ changes with respect to batch number t due to the changes in β_t.

Note 2: $\frac{1}{1+2*\frac{|\mathcal{D}_{\mathcal{T}}|}{N_{past}}} \leq \beta_t \leq 1$ and β_t is decreasing with the batch number t. At the start of a task, $\beta_{t=0} = \beta_{max} = 1$. At the end of a task, $\beta_{t=T} = \beta_{min} = \frac{1}{1+2*\frac{|\mathcal{D}_{\mathcal{T}}|}{N_{past}}}$.

Note 3: Consider a balanced continual learning dataset (e.g. Split-CIFAR100, Split-Mini-ImageNet), i.e. $|\mathcal{D}_j| = |\mathcal{D}_{\mathcal{T}}|, j = 1, ..\mathcal{T}$ we have $\beta_{min} = \frac{\mathcal{T}-1}{\mathcal{T}+1} = (1 - \frac{2}{\mathcal{T}+1})$ and $\lim_{\mathcal{T}\rightarrow\infty} \beta_{min} = 1$.

Note 4: Consider general continual learning datasets. As CL learns more tasks, N_{past} increases, $\lim_{N_{past}\rightarrow\infty} \beta_t = \lim_{N_{past}\rightarrow\infty} \frac{1}{1+2*\frac{N^t_{cur}}{N_{past}}} = 1$.

Proof: Given the empirical gradient $\hat{g}_{ER} = \frac{1}{|\mathcal{B}_t|} \sum_{\mathbf{x}_i, y_i \in \mathcal{B}_t} \nabla \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) + \frac{1}{|\mathcal{B}^{\mathcal{M}}_t|} \sum_{\mathbf{x}_i, y_i \in \mathcal{B}^{\mathcal{M}}_t} \nabla \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)$ during stochastic optimization, we can derive the gradient expectation as follows:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}^t_{\mathcal{M}}}[\mathbb{E}_{\mathcal{B}^{\mathcal{M}}_t \sim \mathcal{D}^t_{\mathcal{M}}, \mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}}[\hat{g}_{ER}]] \\
&= \nabla \mathbb{E}_{\mathcal{D}^t_{\mathcal{M}}} \left[\mathbb{E}_{\mathcal{B}^{\mathcal{M}}_t \sim \mathcal{D}^t_{\mathcal{M}}}[\frac{1}{|\mathcal{B}^{\mathcal{M}}_t|} \sum_{\mathcal{B}^{\mathcal{M}}_t} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] + \mathbb{E}_{\mathcal{B}_t \sim \mathcal{D}_{\mathcal{T}}}[\frac{1}{|\mathcal{B}_t|} \sum_{\mathcal{B}_t} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] \right] \\
&= \nabla(\mathbb{E}_{\mathcal{D}^t_{\mathcal{M}}} \left[\mathbb{E}_{x_i, y_i \sim \mathcal{D}^t_{\mathcal{M}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] \right] + \mathbb{E}_{x_i, y_i \sim \mathcal{D}_{\mathcal{T}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)]) \\
&= \nabla(\frac{N_{past}}{N_{past} + N^t_{cur}} \mathbb{E}_{x_i, y_i \sim \mathcal{D}^0_{\mathcal{M}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)] + (\frac{N^t_{cur}}{N_{past} + N^t_{cur}} + 1) \mathbb{E}_{x_i, y_i \sim \mathcal{D}_{\mathcal{T}}}[\mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right)]) \\
&= \frac{N_{past} + 2 * N^t_{cur}}{(N_{past} + N^t_{cur}) * |\mathcal{D}_{\mathcal{T}}|} \nabla \left(\frac{N_{past}}{N_{past} + 2 * N^t_{cur}} \frac{|\mathcal{D}_{\mathcal{T}}|}{|\mathcal{D}_{\mathcal{M}}|} \sum_{\mathcal{D}_{\mathcal{M}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) + \sum_{\mathcal{D}_{\mathcal{T}}} \mathcal{L}\left(f_\theta(\mathbf{x}_i), y_i\right) \right)
\end{aligned}$$

The third equality is based on the condition that $\mathcal{D}^t_{\mathcal{M}}$ is updated using reservoir sampling so that all the data seen so far have a equal probability of being stored in the memory. In other words, at a given time t, the memory contains a fraction of $\frac{N_{past}}{N_{past}+N^t_{cur}}$ samples coming for past tasks and a fraction of $\frac{N^t_{curr}}{N_{past}+N^t_{cur}}$ samples coming for the current task.

B Dataset details

Table 3 lists the image size, the number of classes and the number of tasks and data size per task of the four CL benchmarks.

Table 3: Dataset information of four CL benchmarks.

	IMAGE SIZE	#TASK	# CLASS	TRAIN PER TASK	TEST PER TASK
SEQ-CIFAR100	3x32x32	20	100	2,500	500
SEQ-MINI-IMAGENET	3x84x84	10	100	5,000	1,000
CORE50-NC	3x128x128	9	50	12,000-24,000	4,500-9,000
CLRS-NC	3x256x256	5	25	2,250	750

C Implementation Details

C.1 Continual Learning Implementation

The hyperparameter settings are summarized in Table 4. All models are optimized using vanilla SGD. For all experiments, we use the learning rate of 0.1 following the same setting as in Aljundi et al. [2019], Shim et al. [2021], and the Nearest-Class-Mean (NCM) classifier is used for evaluation, as Mai et al. reported (2021) considerable and consistent performance gains when replacing the Softmax classifier with the NCM classifier.Each mini batch during training consists of 10 new and 10 memory samples, except for the SCR methods, which employs 100 memory samples and 10 new incoming samples Mai et al. [2021]. By default, the repeated rehearsal parameters for all the results is $N = 10$ and the augmented rehearsal parameters are $P = 1, Q = 14$.

This paper uses Randaugment [Cubuk et al., 2020], which is an auto augmentation method. It randomly selects P number of augmentation operators from a set of 14 operators and applies them to the images. The augmentation operator sets include: 'Identity', 'AutoContrast', 'Equalize', 'Rotate', 'Solarize', 'Color','Posterize', 'Contrast', 'Brightness', 'Sharpness','ShearX', 'ShearY','TranslateX', 'TranslateY'.

Table 4: Hyperparameter setting.

	HYPERPARAMETER
LWF	LR=0.1
AGEM	LR=0.1
ER	BATCHSIZE=10,LR=0.1
MIR	BATCHSIZE=10,C=50,LR=0.1
ASER	K=3, N_SMP_CLS=1.5,BATCHSIZE=10,LR=0.1
SCR	TEMP =0.07, BATCH SIZE = 100,LR=0.1
DER	$\alpha = 0.3$, AUGMENTATION: FLIP AND CROP, $K = 50$

C.2 RL-based hyperparameter tuning implementation

The memory iteration choices are from 1 to 20 and augmentation choices are $(1, 5), (1, 14), (2, 14), (3, 14), (4, 14)$. Action selection probabilities are modeled with softmax weight $\pi_w(a_i) = \frac{e^{w_i}}{\sum_k e^{w_k}}$. Bootstrapped policy gradient are used to adjust action weight.

$$g^{BPG} = \mathbb{E}_{a_i \sim \pi_w} \left[|r_{a_i}| \left(\nabla_w \log \widehat{\pi}_w^+ (a_i) - \nabla_w \log \widehat{\pi}_w^- (a_i) \right) \right]$$

where $\hat{\pi}_w^+ (a_i) := \Sigma_{a_k \in \mathcal{X}_{a_i}^+} \pi_w (a_k)$ and $\hat{\pi}_w^- (a_i) := \Sigma_{a_k \in \mathcal{X}_{a_i}^-} \pi_w (a_k)$

Better/Worse action set The key of BPG idea is to incorporate prior information into the construction of better and worse action set. To apply BPG into OCL environment, we propose to determine the better/worse action set based on the feedback of current memory batch accuracy $A_{\mathcal{M}}$, which reflects the stability level of CL agent. We want the stability level is neither too high or too low. The desirble stability level is defined by $A^*_{\mathcal{M}}$. Based on the stability-plasticity analysis (see Fig 3), we find a higher repeated replay iteration leads to higher stability. Therefore, the better/worse action set is

defined as follows:

$$\mathcal{X}_a^+ := \begin{cases} \forall a_k \mid Iter\,(a_k) < Iter(a), & A_{\mathcal{M}}(a) > A^*_{\mathcal{M}} \\ \forall a_k \mid Iter\,(a_k) > Iter(a), & A_{\mathcal{M}}(a) < A^*_{\mathcal{M}} \\ \emptyset, & A_{\mathcal{M}}(a) = A^*_{\mathcal{M}} \end{cases} \mathcal{X}_a^- := \begin{cases} \forall a_k \mid Iter\,(a_k) > Iter(a), & A_{\mathcal{M}}(a) > A^*_{\mathcal{M}} \\ \forall a_k \mid Iter\,(a_k) < Iter(a), & A_{\mathcal{M}}(a) < A^*_{\mathcal{M}} \\ \forall a_k \mid Iter\,(a_k) \neq Iter(a), & A_{\mathcal{M}}(a) = A^*_{\mathcal{M}} \end{cases}$$

$$\mathcal{X}_a^+ := \begin{cases} \forall a_k \mid Aug\,(a_k) > Aug(a), & A_{\mathcal{M}}(a) > A^*_{\mathcal{M}} \\ \forall a_k \mid Aug\,(a_k) < Aug(a), & A_{\mathcal{M}}(a) < A^*_{\mathcal{M}} \\ \emptyset, & A_{\mathcal{M}}(a) = A^*_{\mathcal{M}} \end{cases} \mathcal{X}_a^- := \begin{cases} \forall a_k \mid Aug\,(a_k) < Aug(a), & A_{\mathcal{M}}(a) > A^*_{\mathcal{M}} \\ \forall a_k \mid Aug\,(a_k) > Aug(a), & A_{\mathcal{M}}(a) < A^*_{\mathcal{M}} \\ \forall a_k \mid Aug\,(a_k) \neq Aug(a), & A_{\mathcal{M}}(a) = A^*_{\mathcal{M}} \end{cases}$$

Reward A challenge in the hyperparameter tuning for OCL setting is that the CL agent may face new tasks with unseen data distribution. Therefore, it is often infeasible to assume the existence of an external validation data containing all the tasks in advance for hyperparameter tuning. To achieve online hyperparameter tuning without external validation data, we propose to define the reward based on the accuracy on the memory. Given a target memory accuracy $A^*_{\mathcal{M}}$, reward is defined $r = |A_{\mathcal{M}} - A^*_{\mathcal{M}}|$

Non-stationary To address the non-stationary nature in the CL envrionment, we reset the weight of BPG to uniform weight at the start of each task.

The analysis of the selected action can be found in Figure 12 and 13 in section D.7.

Algorithm 2: BPG-based RAR

$\mathcal{M}$ is a memory with fixed size,
$\mathcal{B}_t$ is the incoming batch data from current task,
θ are the parameters of CL network,
w is the parameter of RL agent,
K is the number of memory iterations,
P, Q are the augmentation hyperparameters
$A^*_{\mathcal{M}}$ target memory accuracy

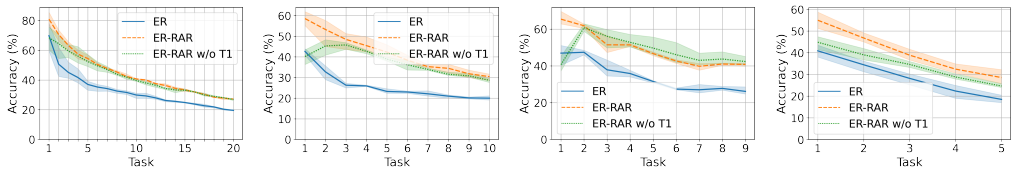
procedure RAR($\mathcal{M}_t$, $\mathcal{B}_t$, θ_t, w_t)
  $K_t \sim \pi_{w_t^{iter}}$
  $P_t, Q_t \sim \pi_{w_t^{aug}}$
  for $k = 1, ..., K_t$ **do**
    $\mathcal{B}^{\mathcal{M}}_{t,k} \sim MemRetrieval(\mathcal{M}_t)$
    $\mathcal{B}_{aug} \leftarrow aug(\mathcal{B}^{\mathcal{M}}_{t,k} \cup \mathcal{B}_t, P_t, Q_t)$
    $A_{\mathcal{M}} \leftarrow MemAcc(\mathcal{M}, \theta_{t,k})$
    $\mathcal{X}^+, \mathcal{X}^- \leftarrow ActionSet(A_{\mathcal{M}}, A^*_{\mathcal{M}})$
    $\theta_{t,k+1} \leftarrow SGD(\mathcal{B}_{aug}, \theta_{t,k})$
  end for
  $\mathcal{M}_{t+1} \leftarrow MemUpdate(\mathcal{M}_t, \mathcal{B}_t)$
  $w^{iter}_{t+1}, w^{aug}_{t+1} \leftarrow UpdateRL(A_{\mathcal{M}}, \mathcal{X}^+, \mathcal{X}^-)$
end procedure

D Supplementary Experimental Results

D.1 Temporal effectiveness of RAR

The previous experiments have analyzed RAR based on the final accuracy it achieves at the *end* of continual learning. This section is focused on the performance of RAR *during* the continual learning process. As shown in Fig 5, ER-RAR (the orange lines) performs better than ER (the blue lines) during the whole training process of continual learning.

However, it should be noted that for the first task, ER-RAR already has some advantages over ER. The training scenario for the first task is actually an i.i.d. setting since the data distributions from the memory and incoming batch are the same—as they are both sampled from the distribution of task 1. Since we are more interested in RAR's effectiveness in non-i.i.d settings, we design a new experiment model, RAR w/o T1, which does not employ RAR for the first task and only starts to use it from the second task onward. The result in Fig 5 shows that without applying RAR for the first task (the green lines), the subsequent performance is a bit lower for the first few tasks but quickly



(a) CIFAR100 (b) MINI-IMAGENET (c) CORE50 (d) CLRS

Figure 5: The temporal performance: ER-RAR outperforms ER during the whole continual learning process. ER-RAR without the first task (ER-RAR w/o T1) achieves similar final accuracy with ER-RAR.

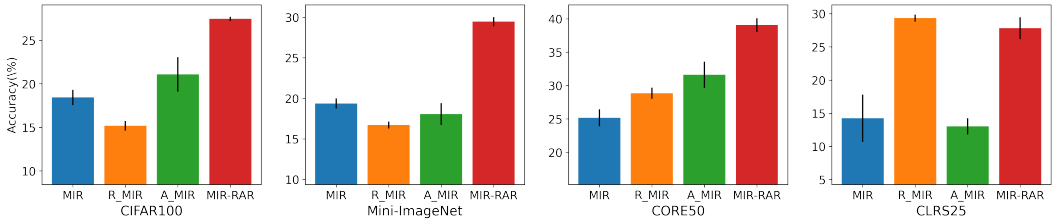


Figure 6: Ablation study MIR-RAR in four datasets with a 2k memory

catches up. More importantly, in terms of the final accuracy, there is not much difference between ER-RAR and ER-RAR w/o T1. This result highlights RAR's effectiveness in a non-i.i.d. learning setting with distribution mismatch in experience replay.

D.2 Ablation studies for MIR-RAR, ASER-RAR and SCR-RAR

As shown in Table 1, RAR improves ER and ER variants (MIR, ASER and SCR). The detailed results of the ablation studies of RAR with ER variants are presented in this section. In particular, Fig 6 and 7 shows the comparison of only using repeated rehearsal or augmented rehearsal for MIR and ASER respectively. Neither of them alone consistently improves the performance of the baseline. This result suggests the influence of RAR for MIR/ASER is similar to ER. The performance gain comes from the combination of the repeated rehearsal and augmented rehearsal. Fig 8 shows the results for SCR. Repeated rehearsal leads to consistent performance gain because SCR already includes augmentation.

In terms of temporal performance during the CL learning process, RAR-MIR, RAR-ASER and RAR-SCR outperform the baselines during the whole training process, and the improvement is still evident without employing the RAR technique for the first task. Fig 9 shows the results of RAR-MIR, RAR-ASER and RAR-SCR on CIFAR100 datasets with a 2k memory.

D.3 Reweighted ER

To deal with the biased ER loss, one straightforward way is to balance the weight of the memory loss and incoming loss by introducing a reweighting hyperparameter α in the gradient of Eq 2.Specifically,

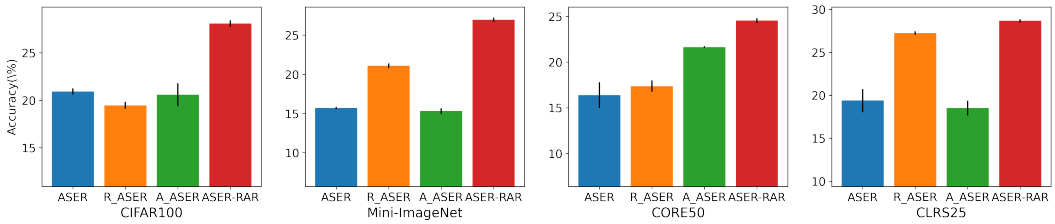


Figure 7: Ablation study ASER-RAR in four datasets with a 2k memory

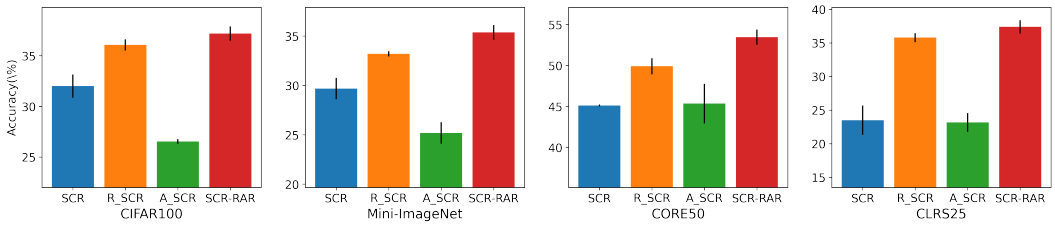
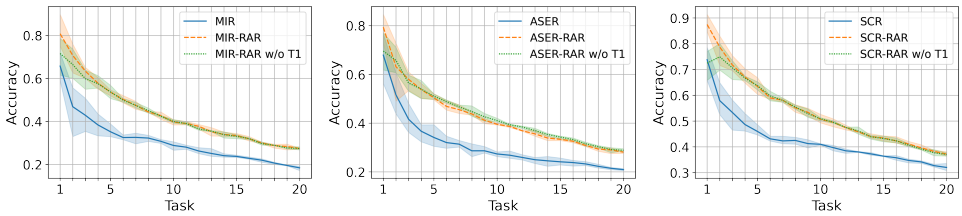


Figure 8: Ablation study SCR-RAR in four datasets with a 2k memory



(a) MIR-RAR (b) ASER-RAR (c) SCR-RAR

Figure 9: Temporal performance of MIR-RAR, ASER-RAR and SCR-RAR for CIFAR100 with a 2k memory.

the gradient for reweighted ER is implemented as below:

$$g_t^{ER-rw} = (1-\alpha)\frac{1}{|B_t|}\sum_{x,y\in B_t}\nabla L(f_\theta(x),y) + \alpha\frac{1}{|B_t^M|}\sum_{x,y\in B_t^M}\nabla L(f_\theta(x),y), \alpha\in(0,1)$$

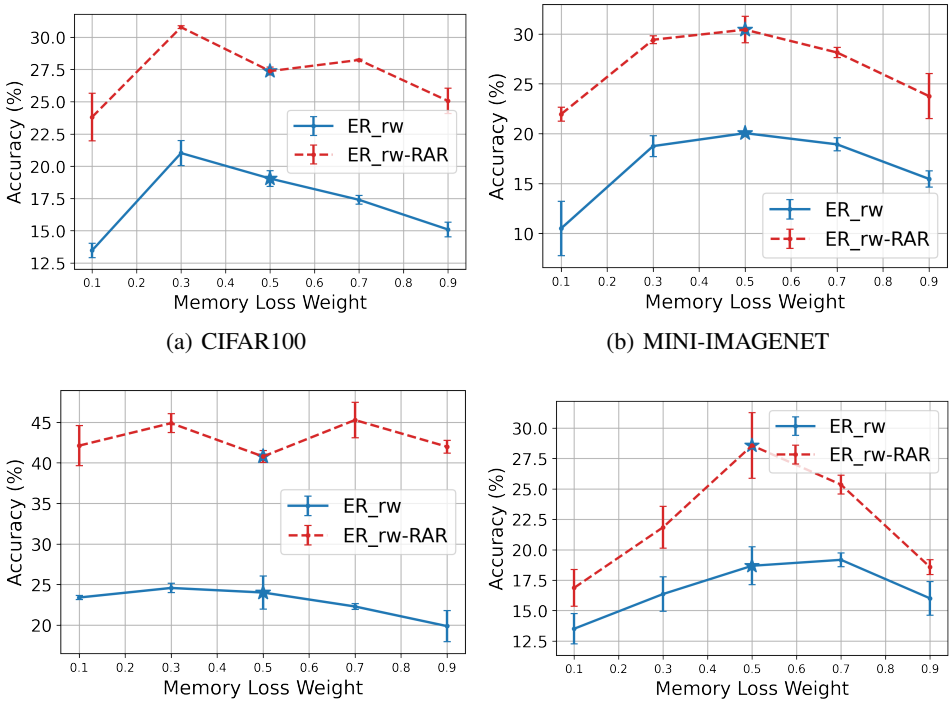
The performance of ER and RAR with respect to the different reweighting hyperparameter $\alpha \in [0.1, 0.3, 0.5, 0.7, 0.9]$ is shown in Fig 10, where $\alpha = 0.5$ denotes vanilla ER loss with the equal weighting of memory loss and incoming loss. To keep the learning rate comparable to vanilla ER, the learning rate of ER-rw is twice as that of ER. A key observation is that similar to vanilla ER, reweighted ER (ER-rw) also significantly benefits from repeated and augmented rehearsal, as ER-rw-rar greatly improves over ER-rw (see the red line and blue line in Fig 10) for all four datasets.

D.4 Large-scale online CL

To examine the effectiveness of RAR in large-scale continual learning problem, we apply RAR in ImageNet-1k. ImageNet-1k is split into 10 tasks and each task contains 100 classes. The training dataset of 10 tasks contains 1,281,167 images in total. Considering the task size, we evaluate with a memory size of M= 20k (with the task to memory size ratio $\lambda \approx 6.4$) and a memory size of M=100k (with the task to memory size ratio $\lambda \approx 1.28$). These choices are similar to Seq-CIFAR100 ($\lambda = 1.25$) and CORE50 ($\lambda = 6$) with a 2k memory. We randomly crop the images to size 224x224, and use the ResNet-18 architecture for training on ImageNet-1k with a single epoch. An incoming batch size of 32 and a memory batch size of 32 are used. We employ a learning rate of 0.1 with 0.001 of weight decay. For RAR method, we use 10 memory iterations with RandAugment with P=1 and Q=14. The results are shown in the Table 5.

We observe that repeated augmented rehearsal (RAR) is effective in this large-scale online continual learning problem and improves the vanilla rehearsal from 2.1% to 15.1% with a 20k memory and 8.7% to 34.7% with a 100k memory.

The discussion in Buzzega et al. [2020] suggests that to deal with complex continual learning datasets, it is necessary to employ multiple epochs of training with offline CL to avoid the underfitting problem present in online learning. Our results show that RAR can greatly improve online rehearsal for large-scale CL problems. Although several offline CL algorithms have been evaluated on ImageNet-1k, to our knowledge, our result is the first attempt to apply online CL to this problem. We aim to investigate other online rehearsal-based methods on ImageNet-1k in future work.



(a) CIFAR100 (b) MINI-IMAGENET

(c) CORE50 (d) CLRS

Figure 10: The performance of reweighting the memory loss of ER (ER-rw) and its effectiveness with RAR in four datasets. (Star symbols denote the accuracy of ER and ER-RAR with reweigting value of 0.5).

Table 5: Accuracy of ER and ER-RAR for ImageNet-1k with a 20k and 100k memory.

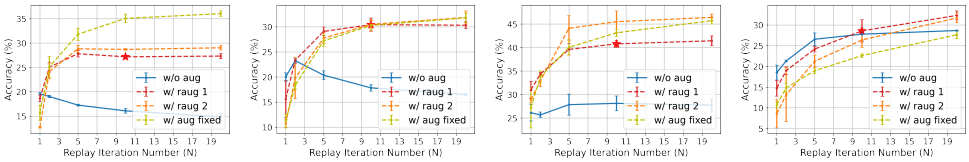
IMAGENET-1K	M=20K	M=100K
ER	2.1 ± 0.3	8.7 ± 0.5
ER-RAR	15.1 ± 0.4	34.7 ± 0.1
GAINS	13.0 ↑	26.0 ↑

D.5 The influence of model size

We investigate the influence of neural network model size by comparing the performance of reduced ResNet-18 and ResNet-18. There is not much difference for CIFAR100, Mini-ImageNet and CLRS. For CORE50, the accuracy of ER decreases with a larger model but the accuracy of ER-RAR increases (see Table 6). A larger model usually introduces higher overfitting risk, which may explain the greater performance gain of RAR in ResNet-18.

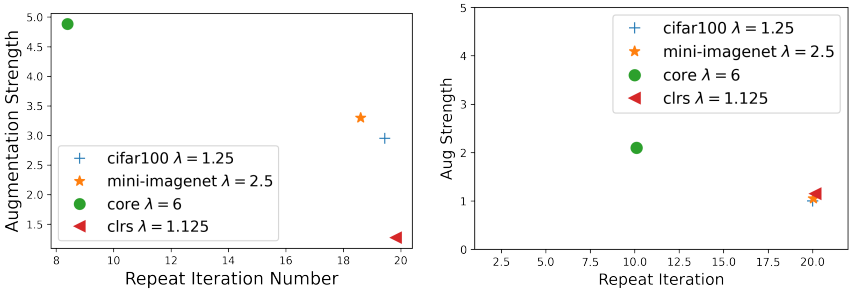
Table 6: Accuracy of ER and ER-RAR using different models for CORE50 with a 2k memory.

	REDUCED-RESNET-18	RESNET-18
ER	26.5 ± 0.8	21.6 ± 1.4
ER-RAR	39.3 ± 1.4	41.3 ± 1.4
GAINS	12.8 ↑	19.7 ↑



(a) CIFAR100 (b) MINI-IMAGENET (c) CORE50 (d) CLRS

Figure 11: The influence of augmentation strength and the number of replay iterations. The red stars represent the results reported in Table 1. The experiment was conducted with ER as the baseline method and a memory of size 2k.



(a) RAR-RL (b) RAR-HTOCL

Figure 12: The average of selected hyperparameters of RAR (iteration and augmentation values) for four datasets.

D.6 Other types of augmentation

Besides using the auto augmentation method of RandAugment, we investigate the effectiveness of using a pre-defined fixed augmentation strategy for RAR. Fig 11 shows the results of employing augmentation with four operators (crop, flip, color jitter and gray). The main findings on the interplay of repetitive replay and augmentation are similar to the case with RandAugment. Nevertheless, compared to auto augmentation, a fixed augmentation may be particularly suitable for one dataset, e.g., CIFAR100, and not suitable for others, like CLRS.

D.7 Action Selection in Reinforcement Learning

The selected hyperparameters for four datasets are shown in Figure 12. Interestingly, the RL-based method assigns a stronger augmentation and lower iteration for a dataset with a higher λ attribute. As discussed in the ERM analysis, a dataset with a higher λ suggests a higher risk of overfitting. RL-based method successfully takes this into account to tune the hyperparameters. In contrast, the OCL-HT method selects the lowest augmentation and highest iteration for three datasets.

Figure 13 presents the selected hyperparameters at different CL training stages. Generally, as the continual learning continues, RL-based method selects stronger augmentation strength and lower iterations, to balance off the increasing risk of memory overfitting.

D.8 Running time

D.8.1 Running time of RAR

The running time of ER and ER-RAR is shown in Table 7. Experients are conducted using an Nvidia GeForce RTX 2080 TI Graphics Card on Mini-ImageNet. The running time of RAR grows linearly with respect to the number of replay iterations. Hence, compared to vanilla experience replay, RAR requires more running time due to the multiple iteration design. However, RAR is still much more computationally efficient compared with offline CL with multiple epochs. An interesting future research direction is to study how to dynamically adjust the iteration number and augmentation strength of RAR to balance the trade-off of accuracy and running time.

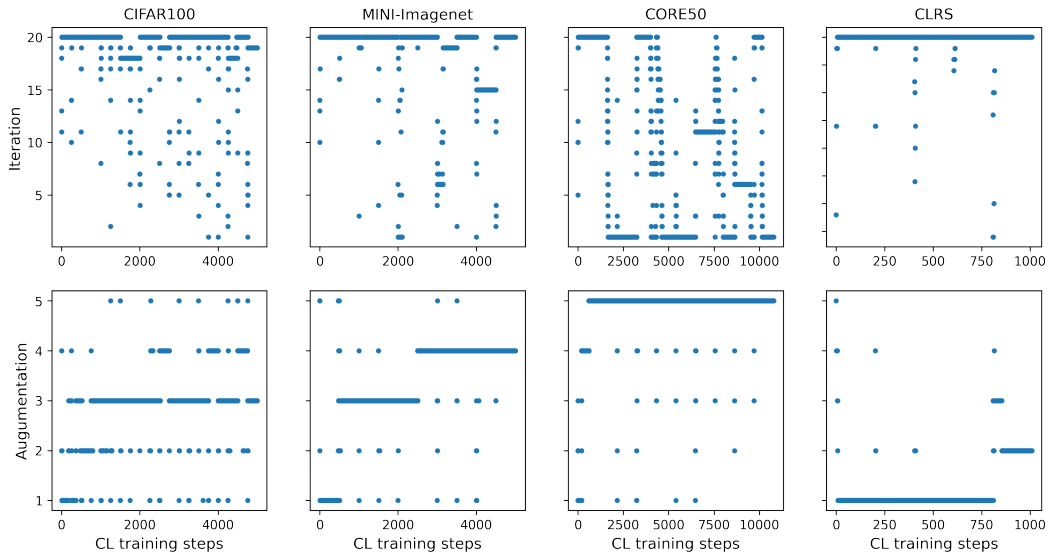


Figure 13: The selected hyperparameters of RAR (iteration and augmentation values) of RL-based method.

Table 7: Running time of ER and ER-RAR using an Nvidia GeForce RTX 2080 TI Graphics Card on Mini-ImageNet . The offline setting employs 50 epochs of training.

	RUNNING TIME (S)	ACCURACY (%)
ER	277 ± 19	20.0 ± 0.8
RAR ($N = 5$)	1383 ± 4	29.1 ± 0.9
RAR ($N = 10$)	$2{,}345 \pm 4$	30.4 ± 1.3
R-ER ($N = 10$)	$2{,}236 \pm 1$	17.8 ± 0.6
ER-OFFLINE ($E = 50$)	$10{,}878 \pm 31$	20.4 ± 0.6
RAR-RL ($N = 19.6$)	$18{,}037 \pm 71$	32.1 ± 1.0

D.8.2 Running time of RL

RL-based hyperparameter optimization is an online method that does not require repeated running over different hyperparameter choices. Therefore, RL-based HPO is much more computationally efficient than offline hyperparameter selection methods. More specifically, the hyperparameter search space in our problem is 100 with 5 augmentation strength levels and 20 memory iteration numbers. Grid search would need to run the OCL algorithm 100 times while RL only needs to run it once.

Nevertheless, the training of RL agents indeed introduces extra computation. In practice, we observe the running time of RL-RAR is about two times slower than that of RAR, as shown in the Table 8.

Table 8: Running time with and without RL-based hyperparameter optimization using an Nvidia GeForce RTX 2080 TI Graphics Card on CIFAR100.

	RUNNING TIME (S)	ACCURACY (%)
RAR ($N = 10$)	1294 ± 255	27.3 ± 0.3
RAR ($N = 20$)	3499 ± 406	27.3 ± 0.5
RL-RAR ($N = 19.8$)	6137 ± 34	29.2 ± 0.3

D.9 Offline Continual Learning

Table 9: Performance of offline rehearsal with and without augmentation.

	SEQ-CIFAR100	SEQ-MINI-IMAGENET	CORE50-NC	CLRS25-NC
OFFLINE ER W/O AUG	17.0 ± 0.6	20.4 ± 0.6	30.2 ± 1.4	33.5 ± 1.6
OFFLINE ER W/ AUG	28.3 ± 0.6	32.6 ± 0.1	44.5 ± 1.3	35.7 ± 0.8
GAINS	$11.3 \uparrow$	$12.2 \uparrow$	$14.3 \uparrow$	$2.2 \uparrow$

Although this paper is mostly focused on online continual learning, some analysis and discussion are also of independent interest to offline continual learning. Specifically, from theoretical perspective, the empirical risk minimization of offline rehearsal is shown in Proposition2 in Section A and its augmented risk is shown in Proposition3. Two conclusions can be drawn from this analysis. First, the risk of memory overfitting in offline rehearsal is also related to the problem characteristic, the ratio λ between task size and memory size. Second, augmentation can help with offline rehearsal since orbit-averaging operation in the augmented empirical risk can reduce both the model variance and generalization error. From empirical perspective, Table 9 shows the performance of offline ER with and without augmentation in four datasets. We use a epoch number of 50 and a memory size of 2000. For all four datasets, offline ER with augmentation achieves significant performance gain over offline ER without augmentation. More interestingly, compared to datasets with a small λ (e.g. CLRS with $\lambda = 1.125$) we observe the datasets with a higher task-to-memory size ratio (e.g. CORE50 with $\lambda = 6$) tend to benefit more from augmentation, due to the increased risk of memory overfitting.