CLASS-INCREMENTAL CONTINUAL LEARNING WITH MANY TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an in-depth experimental analysis of replay-based, class-incremental continual learning (CL), emphasizing a large number of tasks and the particular choice of implementing replay (single-loss or weighted sum of two losses). We exclusively focus on class-incremental learning as the most difficult of current CL paradigms, since we believe that recent results on task-incremental CL cannot be generalized to CL in general. Datasets used for the investigation are various splits based on MNIST, EMNIST-Letters, and CIFAR-10, analyzed for an exhaustive set of relevant hyper-parameters. Based on a mathematical analysis, we show that the way of implementing the replay loss has little impact, justifying the use of the more implementation-friendly single-loss strategy. More importantly, a mathematical analysis and experimental findings show that, for a large number of tasks, the size of the replay buffer must be scaled according to the total number of tasks. This renders naive replay-based CL with a fixed replay budget, as favored in recent literature, problematic in practice, since the compute requirements of CL will thus scale with the total number of tasks. We discuss the implications of this finding for CL and outline potential solutions.

1 INTRODUCTION

This article is in the context of continual learning (CL), a subfield of machine learning that focuses on systems intended for long-term or even life-long learning. CL describes the process required for a general-purpose embodied system to operate: the ability to incrementally acquire, retain, and adapt knowledge over an extended lifespan, analogous to human learning.

According to van de Ven et al. (2022), CL may be conceptually simplified from arbitrary, non-stationary data distributions to commonly used CL scenarios such as, e.g., task-incremental learning (TIL) and class-incremental learning (CIL), as shown in fig. 1. Common to both scenarios is that distribution changes only occur at specific known times, which mark the transitions between successively presented *tasks* of stationary data statistics. TIL may be considered simpler, since the classes to be distinguished are identical for all tasks, and task identity is explicitly provided to the model at training and test time. For CIL, in contrast, new classes are trained successively over time, and the model must learn to distinguish between all classes observed so far, without being informed of the task identity. Both CL scenarios are commonly used in the literature to develop proposals for solving CL.

The major issue in CL with deep neural networks (DNNs) when acquiring additional ("new") knowledge is known as catastrophic forgetting (CF), first described by McCloskey & Cohen (1989); French (1999): an abrupt and severe degradation of performance on previously learned ("old") tasks.

To tackle CF, so-called replay (or rehearsal)-based CL methods attempt to learn from new samples, while retaining knowledge from previous tasks by additionally presenting old samples. Old samples are either produced by a generative model in deep generative replay (DGR, Shin et al., 2017) or retrieved from a memory buffer (e.g., experience replay (ER), Rolnick et al., 2019). Alternative prominent approaches include parameter regularization, such as EWC (Kirkpatrick et al., 2017) and SI (Zenke et al., 2017), and knowledge distillation, such as LwF (Li & Hoiem, 2017). They are not discussed here since they are not well-suited to large-scale CL with many tasks and resource constraints, and since their performance has been found inferior to that of ER or DGR methods in numerous studies (e.g., De Lange et al., 2021).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

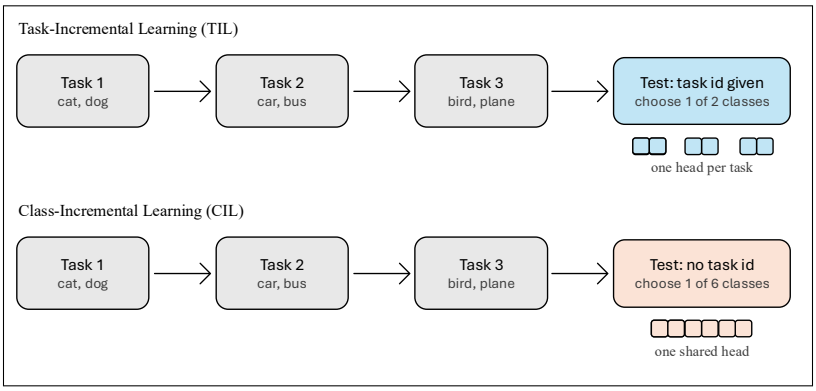


Figure 1: Comparison of TIL and CIL. Both types train on the same stream of two-class tasks and differ only in what is available at test time. In TIL the task identity is given, so a separate output head can be trained for each task and the model only has to separate classes within a task. In CIL no task identity is given, and a single shared head must distinguish among all classes seen so far.

1.1 Motivation

The motivation for this article is twofold: First, replay always implies a parameter β which governs the relative weight of new and old samples. This parameter is commonly applied in two ways: in **single-loss replay** (Chaudhry et al., 2019b; Mai et al., 2020), the buffer is over-sampled such that the ratio between old and new samples used to compute the loss is in agreement with β, which allows replay to be performed for arbitrary models without modifying them. This is not the case for **two-loss replay** (van de Ven et al., 2020), where separate losses are computed for old and new samples and explicitly weighted by $(1-\beta)$ and β, respectively, often using a small, fixed buffer size. Since both methods are used in the literature, we aim at determining whether this choice affects performance.

Second, recent work has tended to focus on the use of small memory buffers using a modest ($\sim$5) number of tasks for CIL problems (e.g., Buzzega et al., 2020; Caccia et al., 2022). CL problems with more tasks are generally tackled in the context of TIL (Farquhar & Gal, 2019), but conclusions are often drawn based on TIL results, while implying that they hold for CIL as well. In particular, several works, e.g., van de Ven et al. (2022), argue for the general applicability of replay-based learning with a small, constant-size buffer (fixed replay budget).

In this article, we investigate the feasibility of replay-based CIL using small replay buffers and a large number of tasks. This should determine whether CIL is indeed mostly solved with a fixed replay budget or whether the size of the replay buffer needs to be scaled with the number of tasks.

1.2 Contributions

- A mathematical analysis and derivation of optimality constraints for class-incremental, replay-based CL
- Experimental comparison between single-loss and two-loss replay using ER and DGR for an exhaustive set of hyper-parameters
- An analysis of class-incremental CL performance for problems with a larger number of tasks (10 and 26) than the commonly used 5-fold MNIST/CIFAR-10 split
- Experimental analysis of whether buffer size should be proportional to the number of tasks or whether a small fixed replay budget is sufficient to prevent forgetting

2 METHODS

2.1 REPLAY METHODS FOR CIL

Experience replay (ER) is a replay-based method described by Rolnick et al. (2019), randomly populating a buffer $\mathcal{O}$ with new sample/target pairs drawn from the current task, its capacity O being a free parameter. In this way, the capacity available for each distinct task shrinks as the task sequence progresses: after observing t tasks, each task retains approximately O/t samples. Since this strategy stores real data in the buffer $\mathcal{O}$, replay targets are ground-truth labels, and the buffer distribution is an unbiased but increasingly sparse sample of the true distribution of previous tasks.

Deep generative replay (DGR), first described in Shin et al. (2017), populates $\mathcal{O}$ at the start of each task with samples produced by a generator (typically a generative adversarial network (GAN) or a variational autoencoder (VAE)), which is trained along with the solver (the model that performs classification) on buffer (old) samples and new samples for each task. The memory cost (excluding the generative model itself) is identical to that of ER. In our experiments, we use a conditional VAE (CVAE, Kingma & Welling, 2022; Sohn et al., 2015) as a generator. Before each task, the generator produces an equal number of samples from every class observed so far, so that the replay stream cannot collapse onto whichever classes happen to be reproduced most easily. An unconditional generator is not able to offer such control, since its samples follow the generator's own marginal (Lesort et al., 2019). We deliberately do not treat the requested class as the replay target, but use the solver to infer it, since generator conditioning on a class does not guarantee that the sample also belongs to the requested class. The generator itself is also trained continually: after each task, it is updated on the current task's data and its own replayed samples.

2.2 MATHEMATICAL ANALYSIS OF REPLAY-BASED CIL

We assume that there is a basic (supervised) loss, defined for a single sample/target pair: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y})$. We may assume a cross-entropy loss for $\mathcal{L}$, but nothing in the following derivations depends on this. For any given task t of a class-incremental CL problem, we assume that there exists a replay buffer $\mathcal{O} = \{\boldsymbol{x}_o, \boldsymbol{y}_o\}$ (either filled during previous tasks for experience replay, or else generated prior to the current task for generative replay) of size $O = |\mathcal{O}|$, which is a fundamental free parameter of all replay methods. Another key parameter is the mixing coefficient $\beta \in [0, 1]$ which governs the relative importance of old and new samples, the latter being described by a set $\mathcal{C} = \{\boldsymbol{x}_c, \boldsymbol{y}_c\}$ of training samples for the current task of size $C = |\mathcal{C}|$.

For example, $\beta = 1$ signifies that only the new task is relevant, while $\beta = 0$ implies the reverse. An ideal choice may depend on the current task t, the number of samples per task, the difficulty of each task seen so far, and so on. We assume for simplicity:

- All tasks have approximately the same number of samples C
- Buffer size is relatively small: $O < C$
- Batch size is small: $B < O, C$
- Task size is a multiple of the buffer size: $C = kO$, $k \in \mathbb{N}$. This is a technical definition that simplifies later derivation, without loss of generality.

We now define two variants of performing replay in general, with the goal of determining under which conditions these are equivalent, or at least approximately so, assuming a mini-batch size of B for stochastic gradient descent (SGD):

2.3 TWO-LOSS FORMULATION

Here, the total loss is separated into two contributions, coming from old and new samples, which are weighted by $(1-\beta)$ and β, respectively. For implementing SGD, we draw B samples each from $\mathcal{O}$ and $\mathcal{C}$ for each loss term, with repetitions for $\mathcal{O}$ since we assume $O < C$. Since C is a multiple of O by assumption, we can rewrite the average loss over C samples, over-sampled exactly k times from the buffer, as the average loss over the buffer itself. An epoch is thus defined by C samples

and never processes more than $2C$ samples in total. The loss $\mathcal{L}^2$ reads:

$$\mathcal{L}^2 = (1-\beta)\frac{1}{O}\sum_{o=1}^{O}\mathcal{L}(\boldsymbol{x}_o, \boldsymbol{y}_o) + \beta\frac{1}{C}\sum_{c=1}^{C}\mathcal{L}(\boldsymbol{x}_c, \boldsymbol{y}_c) \tag{1}$$

The two-loss formulation seems to produce promising results at a constant, bounded computational cost. On the downside, knowledge about β must be incorporated into each of the models used through two weighted losses, which can be complex and error-prone, and the validity of approximating old knowledge using a fixed-size replay buffer may be challenged.

2.4 SINGLE-LOSS FORMULATION

Here, all models involved are trained using a single loss, based on a single set of samples $\mathcal{M}$ whose composition is governed by the chosen mixing coefficient β and buffer size O. Prior to SGD training, we (over-)sample from $\mathcal{O}$, or generate, $O'=\frac{(1-\beta)}{\beta}C$ samples, and merge this enlarged set $\mathcal{O}'$ with $\mathcal{C}$ to form the set $\mathcal{M}$ containing $M=O' + C=\frac{C}{\beta}$ samples. This set contains old and new samples in proportion of $\frac{1-\beta}{\beta}$. The loss then simply reads as follows:

$$\mathcal{L}^1 = \frac{1}{M}\sum_{n=1}^{M}\mathcal{L}(\boldsymbol{x}_n, \boldsymbol{y}_n) \tag{2}$$

The advantage of the single-loss formulation is that the individual models involved in replay (e.g., the solver and generator DNNs for DGR) do not need to know about β, O and replay in general and therefore do not need to be adapted at all when performing replay-based CL. On the other hand, over-sampling the buffer $\mathcal{O}$ leads to training set sizes of $\frac{C}{\beta}$, which can be significant for incorrectly chosen values of β.

2.5 OPTIMALITY REQUIREMENTS FOR SINGLE-LOSS AND TWO-LOSS FORMULATIONS

The basic loss to be approximated by eqs. (1), (2) is the *joint-training loss* at the last task T of a CL problem, defined as the training on all current and previous-task samples. Assuming we have C samples $\{(\boldsymbol{x}_n^t, \boldsymbol{y}_n^t), n = 1 \dots C\}$ for each task $t \leq T$, the loss reads:

$$\begin{aligned}
\mathcal{L}^j &= \frac{1}{TC}\sum_{t=1}^{T}\sum_{n}\mathcal{L}(\boldsymbol{x}_n^t, \boldsymbol{y}_n^t) \\
&= \frac{(T-1)}{T}\frac{1}{(T-1)C}\sum_{t=1}^{(T-1)}\sum_{n=1}^{C}\mathcal{L}(\boldsymbol{x}_n^t, \boldsymbol{y}_n^t) + \frac{1}{T}\frac{1}{C}\sum_{c=1}^{C}\mathcal{L}(\boldsymbol{x}_c^T, \boldsymbol{y}_c^T) \\
&\equiv (1-\beta^*)\frac{1}{O^*}\sum_{t=1}^{(T-1)}\sum_{n=1}^{C}\mathcal{L}(\boldsymbol{x}_n^t, \boldsymbol{y}_n^t) + \beta^*\frac{1}{C}\sum_{c=1}^{C}\mathcal{L}(\boldsymbol{x}_c^T, \boldsymbol{y}_c^T) \\
&\equiv (1-\beta^*)\frac{1}{O^*}\sum_{o=1}^{O^*}\mathcal{L}(\boldsymbol{x}_o, \boldsymbol{y}_o) + \beta^*\frac{1}{C}\sum_{c=1}^{C}\mathcal{L}(\boldsymbol{x}_c, \boldsymbol{y}_c)
\end{aligned} \tag{3}$$

For the single-loss formulation, knowledge about the learning history (how many previous tasks $(T-1)$, samples C per task) induces optimal values $\beta^* = \frac{1}{T}$ (in agreement with van de Ven et al., 2020) and $O^* = (T-1)C$ to match exactly $\mathcal{L}^j$ by replay, i.e., if previous-task samples are taken from a replay buffer of size O^*. The optimal buffer size is defined by the size required to solve the last task and therefore depends on T. For previous tasks $t < T$, a smaller buffer would be sufficient. We may furthermore assume that the learning problem has some redundancy, i.e., that any task of C samples can in fact be learned just as well from νC as from C samples, $\nu \leq 1$. Thus, we can reduce the optimal buffer size to $O^* = \nu(T-1)C$, whereas β is unaffected. We clearly perceive that the optimal buffer size, however reduced by redundancy, has a linear dependency on T, implying that the storage and compute requirements grow linearly with the number of tasks. Concerning the two-loss formulation, the second terms in eqs.3 and 1 are obviously equal. For the first terms, we wish to

know under which circumstances we can approximate the exact left-hand term by the corresponding terms in the single-loss and the two-loss formulations.

$$(1-\beta^*)\frac{1}{O^*}\sum_{o=1}^{O^*}\mathcal{L}(\boldsymbol{x}_o,\boldsymbol{y}_o) \stackrel{?}{\approx} (1-\beta)\frac{1}{O}\sum_{o=1}^{O}\mathcal{L}(\boldsymbol{x}_o,\boldsymbol{y}_o) \tag{4}$$

Choosing an optimal $\beta = \beta^* = 1/T$ for the last task (and $1/t$ for $t < T$) is always possible, and equality holds if $O \geq O^* = \nu(T-1)C$, which can only be valid up to a certain number of tasks $T^{\max} = 1 + \frac{O}{\nu C}$.

For example, if we take a 5-task split of the MNIST dataset ($T = 5$, $C \approx 12\,000$), and assume a redundancy coefficient of $\nu = \frac{1}{40}$, we obtain an optimal buffer size of $O^* = 600$, which explains why two-loss replay works well with typically proposed buffer sizes of $O = 1\,000$. For a 10-task split, we obtain $O^* = 1\,350$, which may explain inferior results when using $O = 1\,000$. For establishing a recommended buffer size, it is fortunate that ν can actually be measured experimentally, see section 3.3.

3 Experiments

3.1 Datasets and CL problems

MNIST dataset (Lecun et al., 1998): $60\,000$ training and $10\,000$ testing 28×28 grayscale images of handwritten digits (0–9).
EMNIST-Letters dataset (Cohen et al., 2017): 26 balanced classes, containing $4\,800$ training and 800 testing 28×28 grayscale images of handwritten letters (lowercase and uppercase) each.
CIFAR-10 dataset (Krizhevsky, 2009): consisting of $50\,000$ training and $10\,000$ testing 32×32 color images in 10 balanced classes.
Under these premises, our CL problems, on which our experiments are conducted, consist of four benchmarks varying in the number of tasks, number of classes, and data, all following the class-incremental protocol, with no task identity available during testing:

CL benchmark	MNIST-5	MNIST-10	EMNIST-26	CIFAR-10
details	2 classes/task	1 class/task	1 class/task	1 class/task

3.2 Evaluation metrics

To evaluate CL performance, we followed standard practice in the literature (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018) using two metrics:

Average accuracy: Let $a_{k,B_k,j} \in [0,1]$ denote the accuracy evaluated on the test set of task j, immediately after the model has been continually trained on all mini-batches up to and including mini-batch B_k of task k. The average accuracy $A_k \in [0,1]$ after training up to task k is defined as

$$A_k = \frac{1}{k}\sum_{j=1}^{k} a_{k,B_k,j}. \tag{5}$$

Therefore, A_T denotes the average accuracy across all T tasks after the last task has been learned, which is the most commonly reported metric in life-long learning (Chaudhry et al., 2019a).

Forgetting measure: The forgetting measure $F_k \in [-1,1]$ (Chaudhry et al., 2018) quantifies forgetting as the average degradation in performance on previously learned tasks after continual training up to task k:

$$F_k = \frac{1}{k-1}\sum_{j=1}^{k-1} f_j^k, \quad \text{with} \quad f_j^k = \max_{l\in\{1,\ldots,k-1\}} a_{l,B_l,j} - a_{k,B_k,j}. \tag{6}$$

In this context, f_j^k is the forgetting on task j after model training on all mini-batches up to task k. Intuitively, f_j^k measures the gap between the highest accuracy ever achieved on task j at any earlier

checkpoint and its accuracy after training on task k. Consequently, it quantifies the accuracy drop on previous tasks and gives an indirect notion of how efficiently the model transfers knowledge to new tasks (Chaudhry et al., 2019a). Thus, forgetting is bounded by how well a task was learned in the first place: a task that was never learned well cannot contribute much forgetting. Forgetting values must therefore be interpreted in relation to the accuracy attained on each task.

3.3 Measuring redundancy

We measure the redundancy coefficient ν for a given dataset and solver model simply by subsampling the dataset by various factors $\nu^{-1} \in \{1, 2, 4, 8, 16, 32, 64\}$ and verifying whether the resulting accuracy on the full test set is comparable to the $\nu = 1$ case. We obtain values of $\nu^{-1} = 32$ for MNIST, $\nu^{-1} = 4$ for EMNIST-Letters, and $\nu^{-1} = 1$ for CIFAR-10.

3.4 Experimental setup

We perform a single, large experiment involving a large number of configurations defined by the CL benchmark and hyper-parameters. A full overview of the DNNs used can be seen in table 2. The complete hyper-parameter grid is given in table 3. We run exhaustive variations over buffer size $O \in \{128, 256, 512, 1024, 4096\}$, solver learning rate $\eta_s \in \{10^{-3}, 10^{-4}\}$, mixing coefficient $\beta \in \{\frac{1}{\text{task}}, 0.1, 0.3, 0.5\}$, number of epochs per task $\{5, 10, 20\}$, batch size $B \in \{128, 256\}$ and the two replay variants (single-loss and two-loss). Every configuration is trained three times with different seeds, and we report averages and standard deviations. Where we report the best configuration, it is the configuration with the highest mean accuracy.

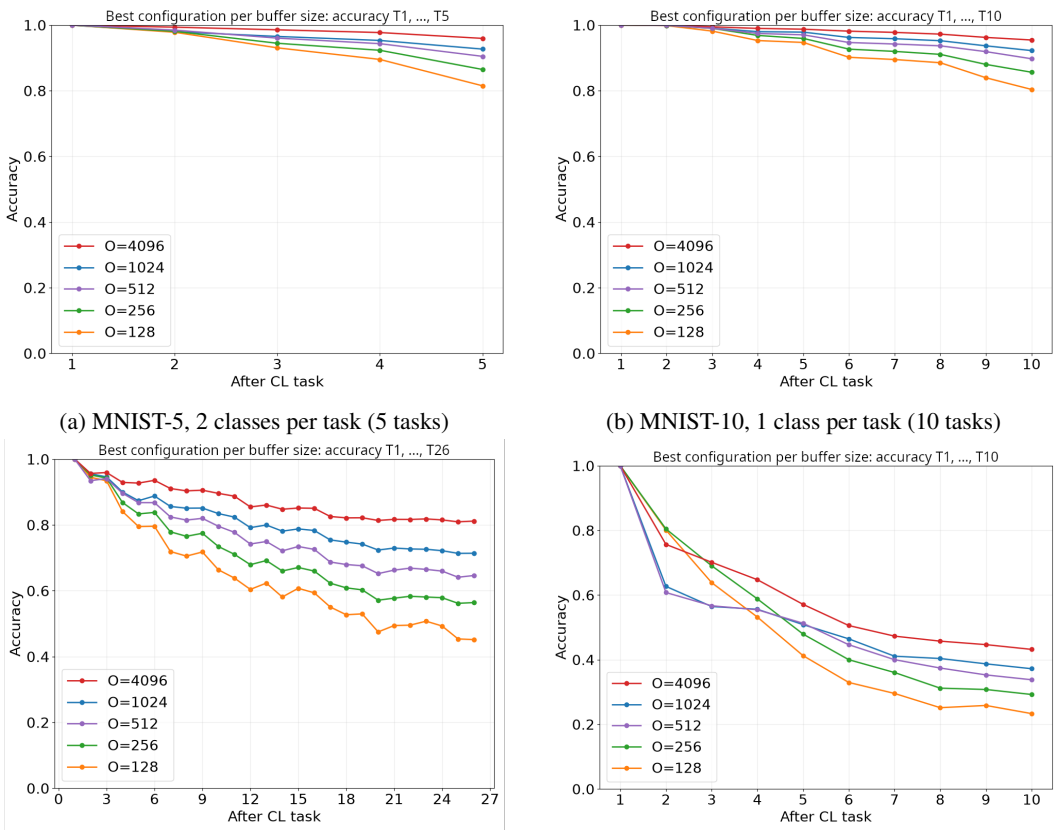
While ER is run on all four benchmarks, DGR is restricted to MNIST-5 and MNIST-10. Training a generative replay model scales with the length of the task sequence, which makes EMNIST-26 prohibitively expensive at the grid size used here. Standard generative replay has also been reported to degrade sharply beyond roughly 15 tasks (van de Ven et al., 2020). For natural images, the same work finds that standard generative replay breaks down on split CIFAR-100, making comparable results on CIFAR-10 unlikely at this scale. DGR therefore serves a different purpose in this work: rather than providing a like-for-like comparison across all benchmarks, it tests whether the effects observed for ER also appear for a method that reconstructs past data instead of storing it. Full results for ER and DGR are reported in section A.3.1 and section A.3.2, respectively. Evaluation is performed with two goals in mind:

E1: assess effects of buffer size/number of tasks E1 focuses on CL performance over long task sequences, building on the common protocol described above with the main focus on buffer size O and the length of the task sequence.

E2: compare single-loss and two-loss replay E2 examines single-loss and two-loss replay with the goal of determining whether these approaches vary in performance and, if so, under which circumstances both are approximately equal, as seen in eq. (4). Both variants are implemented as described in sections 2.4 and 2.3, with all configurations evaluated such that every single-loss configuration has a two-loss counterpart that is identical in all other hyper-parameters. Since ranking configurations by either variant selects on the target metric and thereby favors the variant it was ranked by, we report the comparison in both directions: once on the five best two-loss configurations per benchmark, and once on the five best single-loss configurations. Each row of the resulting tables is one configuration evaluated with both variants, so the two values in a row differ only in the loss formulation. Configurations ranking among the best five under both selection criteria can appear twice.

3.5 Results E1

Figure 2 depicts, for each CL benchmark and buffer size O, the best resulting ER accuracy curves, and table 1 lists the corresponding values together with the configurations that produced them. Across all benchmarks, we observe that larger buffers retain accuracy longer, and the gap between buffer sizes O widens with the number of tasks. On MNIST-5, the spread between the smallest and the largest buffer is roughly 0.15, while on EMNIST-26 it grows to 0.36 and on CIFAR-10 to 0.2. The ordering of the buffer sizes is strictly monotone in final accuracy for every benchmark, so the



(a) MNIST-5, 2 classes per task (5 tasks)

(b) MNIST-10, 1 class per task (10 tasks)

(c) EMNIST-26, 1 class per task (26 tasks)

(d) CIFAR-10, 1 class per task (10 tasks)

Figure 2: Accuracy on the tasks seen so far, for the best ER configuration per buffer size (averaged over three runs). Each panel shows one benchmark. Larger buffers retain accuracy longer, and the gap between buffer sizes widens with the number of tasks.

effect is not an artifact of a single favorable configuration. The best configurations are obtained at a buffer size of 4096 throughout, reaching 0.9592 on MNIST-5, 0.9543 on MNIST-10, 0.8115 on EMNIST-26, and 0.4317 on CIFAR-10. The same buffer size also yields the lowest forgetting, with 0.0391, 0.0434, 0.1228, and 0.2154, respectively. As with accuracy, the ordering follows buffer size without exception. It should also be noted that the buffer size required to retain a given accuracy level increases with the difficulty of the benchmark rather than being an absolute quantity: a buffer of 1024 retains 0.9266 on MNIST-5 and 0.9221 on MNIST-10, but on EMNIST-26, the same buffer reaches only 0.7138, and even 4096 drops to 0.8115. EMNIST-26 is drawn from the same data domain as the MNIST benchmarks and differs from them mainly in the number of tasks, so the decline cannot be attributed to more complex inputs, as it can on CIFAR-10. A buffer that is sufficient for ten tasks is therefore not sufficient for 26: what has to be retained grows with the sequence, while the buffer does not. Since final accuracy does not reveal how retention is distributed over individual tasks, fig. 3 resolves the best configuration on EMNIST-26 into per-task accuracies.

Instead of showing best accuracy curves, fig. 4 and table 4 report these curves averaged over all configurations for a given buffer size. The buffer-size ordering is preserved, indicating that it is a property of the method and not of a particular configuration. The standard deviations are an order of magnitude larger than those in table 1, since they capture the spread across different hyperparameters.

The same buffer-size effect appears for DGR, as shown in fig. 5. On MNIST-5 the best configuration drops from 0.9076 at a buffer size of 4096 to 0.7644 at 128, and on MNIST-10 from 0.8692 to 0.7133, with ordering monotone at every buffer size on both benchmarks (table 7). The dependence

on buffer size is thus not specific to storing raw samples: it also holds when old samples are reconstructed by a generative model. At equal buffer size DGR stays below ER throughout, by roughly 0.05 on MNIST-5 and 0.08 on MNIST-10 at 4096, which is consistent with the generator being an additional source of error.

3.6 RESULTS E2

Table 5 reports both selection directions. For each benchmark, the upper five rows are the best two-loss configurations with the corresponding single-loss result on the same settings, and the lower five rows the reverse.

In terms of final accuracy, the two variants are very close on the simpler benchmarks. Averaged over both selection directions, the absolute accuracy difference is 0.0029 on MNIST-5 and 0.0043 on MNIST-10, which is of the same order as the seed-to-seed variation, depicted in table 6. Individual rows differ by as little as 0.0002. Whatever ordering the tables show on these two benchmarks, it is not resolvable with three seeds, and we consider the variants equivalent in accuracy there. The gap widens slightly with benchmark difficulty: 0.0126 on EMNIST-26 and 0.0268 on CIFAR-10, the latter clearly exceeding seed noise. The averaged accuracy difference on CIFAR-10 remains below 0.03 on a benchmark where absolute accuracy is around 0.41.

The variants separate more clearly in forgetting. The mean absolute gap increases with the length and difficulty of the sequence: 0.0104 on MNIST-5, 0.0114 on MNIST-10, 0.0448 on EMNIST-26, and 0.1306 on CIFAR-10. On CIFAR-10 the gap alone exceeds the entire forgetting level measured on either MNIST benchmark at this buffer size. Unlike the accuracy ordering, which reverses with the selection direction as expected, the forgetting ordering favors two-loss in both directions on average, so it is not an artifact of how the configurations were chosen.

The comparison of both loss variants was repeated for DGR on MNIST-5 and MNIST-10 (tabs. 9 and 10). In terms of final accuracy, the two variants are further apart than for ER. The mean absolute accuracy difference is 0.0088 on MNIST-5 and 0.0164 on MNIST-10. The forgetting ordering again favors two-loss in both selection directions on average. The accuracy picture is less clean than for ER: when configurations are selected by two-loss, all ten rows favor two-loss as expected, but when they are selected by one-loss, four of ten rows still favor two-loss, so the reversal is only partial. The mean absolute forgetting gap is 0.0168 on MNIST-5 and 0.0238 on MNIST-10.

4 DISCUSSION AND CONCLUSION

We present one of the first studies exclusively targeting **long-term class-incremental learning** with > 5 tasks. A **linear buffer scaling requirement** in replay approaches is demonstrated empirically and mathematically as a function of the number of tasks. Another useful result is the **feasibility of the single-loss strategy**, which allows existing model implementations to be repurposed for replay without any adaptation. The **automatic mixing coefficient adaptation** proposed in van de Ven et al. (2020) as a value in the favor of replay arises naturally from our mathematical treatment of the problem and is generally the best way to set this parameter. For the field of CL in general, there are wider implications:

Fixed-buffer strategies are infeasible: there are several works (van de Ven et al., 2020; 2022) that use a moderate, fixed buffer size together with a two-loss replay strategy. This is feasible as long as the buffer size satisfies the optimality requirements laid out in section 2.5 for the given model and number of tasks. However, a fixed-size replay buffer should not be taken as a general solution to class-incremental CL.

Replay has linear scaling behavior: even if the buffer size is suited to the model and CL problem, it will scale linearly with the number of tasks according to section 2.5. This leads to linearly increasing computational demands, making long-term CL with many tasks effectively infeasible. Strategies such as those of Krawczyk & Gepperth (2024) and Aljundi et al. (2019), which replay only samples that interfere with the current task, should be further investigated to counter this problem.

Concerning generative replay: while it is an elegant idea, we found that DGR has two principal problems. First of all, it is very sensitive to parameters (especially learning rate and training duration) which cannot be obtained by cross-validation, since the core idea of CL is not to access future

tasks (see the training methodology proposed in Chaudhry et al. (2019a)). In addition to the high computational demand of training, this renders DGR problematic in practice. What CL requires are fast-to-train generators, which would ideally share resources with the solver (see, e.g., van de Ven et al., 2020; Krawczyk & Gepperth, 2024) to minimize computational load.

Table 1: Best configuration per buffer size O for each benchmark with ER. The learning rate is 10^{-4} for all configurations.

BUFFER O	AVERAGE ACCURACY	FORGETTING	β	EPOCHS	BATCH	VARIANT
MNIST-5						
4096	0.9592 ± 0.0017	0.0391 ± 0.0030	0.1	5	128	$\mathcal{L}^1$
1024	0.9266 ± 0.0017	0.0518 ± 0.0028	0.1	5	256	$\mathcal{L}^2$
512	0.9045 ± 0.0034	0.0940 ± 0.0036	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
256	0.8646 ± 0.0033	0.1478 ± 0.0041	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
128	0.8148 ± 0.0065	0.2097 ± 0.0080	0.1	5	256	$\mathcal{L}^2$
MNIST-10						
4096	0.9543 ± 0.0013	0.0434 ± 0.0015	$\frac{1}{\text{task}}$	20	128	$\mathcal{L}^2$
1024	0.9221 ± 0.0008	0.0707 ± 0.0012	$\frac{1}{\text{task}}$	5	128	$\mathcal{L}^2$
512	0.8971 ± 0.0040	0.0882 ± 0.0063	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
256	0.8561 ± 0.0107	0.1421 ± 0.0121	0.1	5	256	$\mathcal{L}^2$
128	0.8037 ± 0.0078	0.2039 ± 0.0094	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
EMNIST-26						
4096	0.8115 ± 0.0015	0.1228 ± 0.0045	0.1	5	128	$\mathcal{L}^2$
1024	0.7138 ± 0.0039	0.1761 ± 0.0036	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
512	0.6462 ± 0.0021	0.2982 ± 0.0028	0.1	5	256	$\mathcal{L}^2$
256	0.5639 ± 0.0081	0.3855 ± 0.0068	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
128	0.4513 ± 0.0156	0.5204 ± 0.0191	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
CIFAR-10						
4096	0.4317 ± 0.0030	0.2154 ± 0.0049	0.1	10	128	$\mathcal{L}^2$
1024	0.3719 ± 0.0094	0.2346 ± 0.0209	0.1	5	256	$\mathcal{L}^2$
512	0.3378 ± 0.0018	0.3090 ± 0.0083	0.1	5	256	$\mathcal{L}^2$
256	0.2922 ± 0.0040	0.4516 ± 0.0167	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$
128	0.2330 ± 0.0163	0.6024 ± 0.0037	$\frac{1}{\text{task}}$	5	256	$\mathcal{L}^2$

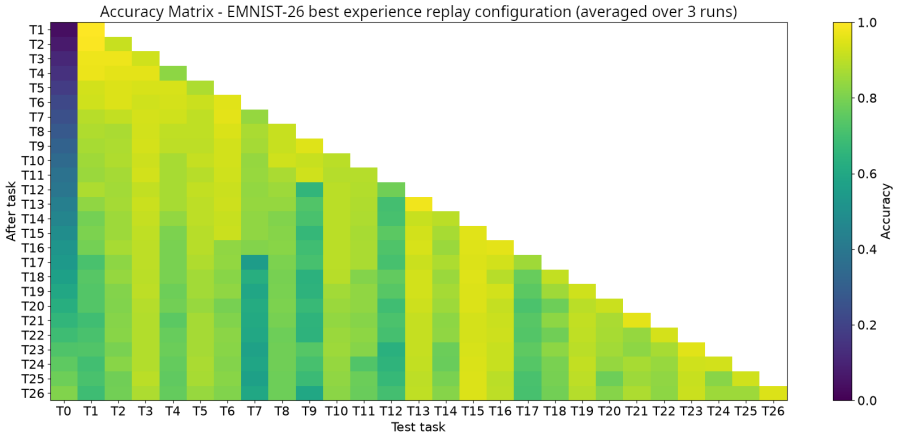


Figure 3: Accuracy matrix of the best ER configuration on EMNIST-26. Entry (i, j) is the accuracy on test task T_j after training through task T_i, whereas column T_0 reports the global accuracy over all classes. Cells above the diagonal are undefined, as those tasks have not yet been observed. Descending a column shows how accuracy degrades for that task as training continues: e.g., for T_1, accuracy steadily decreases over the remaining sequence, indicating catastrophic forgetting.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Opus 5 by Anthropic) for language editing, including improving grammar, sentence structure, and clarity of text and notes originally drafted by the authors, as well as for formatting tables in LaTeX. We reviewed all AI-assisted text and manually checked it against the original intended meaning to ensure technical accuracy and that no unintended claims were introduced. Furthermore, all reported numbers were verified against the underlying experimental data. In addition, we used generative AI to assist parts of software code production (bug fixes, plotting support) and to research and analyze the existing literature.

We have **not** used generative AI tools to:

- generate research ideas and experimental design
- clean, reformat, or generate synthetic datasets
- develop theoretical models or conceptual frameworks
- formulate and prove mathematical claims
- propose or refine hypotheses
- design or provide feedback on research methodology or experiments
- implement methods
- support qualitative and thematic data analysis and interpret results

We take responsibility for the final content of this work, including text and code produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, the following efforts have been made:

- specification of the model architecture in section A.1
- overview of the full hyper-parameter grid in table 3
- implementation details, cluster details, and software stack in section A.2
- complete methods and experimental setup described in section 2 and in section 3.4
- description of datasets and definition of benchmarks in section 3.1
- definition of evaluation metrics in section 3.2
- full evaluation for the best configurations reported in section A.3.1 and in section A.3.2
- source code for the replay implementations is available at: `https://github.com/anon-scientist/big-study-on-replay`

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. TensorFlow: A system for large-scale machine learning. *CoRR*, abs/1605.08695, 2016. URL `http://arxiv.org/abs/1605.08695`.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems*, 32, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/b704ea2c39778f07c617f6b7ce480e9e-Abstract.html`.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning, 2022. URL https://arxiv.org/abs/2104.05025.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, pp. 556–572. Springer International Publishing, 2018. ISBN 9783030012526. doi: 10.1007/978-3-030-01252-6_33. URL http://dx.doi.org/10.1007/978-3-030-01252-6_33.

Arslan Chaudhry, Marc' Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM, 01 2019a. URL http://arxiv.org/abs/1812.00420. arXiv:1812.00420.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019b. URL http://arxiv.org/abs/1902.10486.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL http://arxiv.org/abs/1702.05373.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning, 06 2019. URL http://arxiv.org/abs/1805.09733. arXiv:1805.09733 [stat.ML].

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 04 1999. ISSN 1364-6613. doi: 10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/pii/S1364661399012942.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL https://www.nature.com/articles/s41586-020-2649-2.

John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes, 2022. URL https://arxiv.org/abs/1312.6114.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 03 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114. URL http://arxiv.org/abs/1612.00796. arXiv:1612.00796.

Alexander Krawczyk and Alexander Gepperth. Adiabatic replay for continual learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 04 2009.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Timothée Lesort, Alexander Gepperth, Andrei Stoian, and David Filliat. Marginal replay vs conditional replay for continual learning, 2019. URL https://arxiv.org/abs/1810.12069.

Zhizhong Li and Derek Hoiem. Learning without Forgetting, 02 2017. URL http://arxiv.org/abs/1606.09282. arXiv:1606.09282 [cs.CV].

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Zheda Mai, Hyunwoo Kim, Jihwan Jeong, and Scott Sanner. Batch-level experience replay with review for continual learning. *CoRR*, abs/2007.05683, 2020. URL https://arxiv.org/abs/2007.05683.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Wes Mckinney. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*, 01 2011.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML 2010*, pp. 807–814, 2010.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85): 2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay, 2017.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):4069, 08 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17866-2. URL https://www.nature.com/articles/s41467-020-17866-2.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 12 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL https://www.nature.com/articles/s42256-022-00568-3.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore,

Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. URL https://www.nature.com/articles/s41592-019-0686-2.

Andy B. Yoo, Morris A. Jette, and Mark Grondona. SLURM: Simple Linux utility for resource management. In Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn (eds.), *Job Scheduling Strategies for Parallel Processing*, pp. 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39727-4.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence, 06 2017. URL http://arxiv.org/abs/1703.04200. arXiv:1703.04200.

A Appendix

A.1 Network

All experiments use the same solver so that differences in performance can be attributed to the investigated replay mechanisms and hyper-parameters rather than to the architecture. The network architectures are depicted in table 2 and the parameter setups of experiments in table 3.

Table 2: Network architectures of the solver and generator. All layers are fully connected, and all hidden layers use ReLU activations (Nair & Hinton, 2010). D is the flattened input size ($D = 784$ for MNIST and EMNIST-Letters, $D = 3072$ for CIFAR-10), K the number of classes ($K = 10$ for MNIST and CIFAR-10, $K = 26$ for EMNIST-Letters), $H = 400$ the hidden width and $L = 100$ the latent dimension. The solver is identical for ER and DGR and is trained with softmax cross-entropy on its logits, using Adam (Kingma & Ba, 2017) with learning rate η_s. The conditional generator (DGR-CVAE, MNIST only) receives the one-hot label y through a separate embedding that is concatenated with the data path. It is trained by minimizing the negative ELBO, $-\big(\log p_\theta(x \mid z, y) + \log p(z) - \log q_\phi(z \mid x, y)\big)$, estimated with a single latent sample, a Bernoulli likelihood and prior $p(z) = \mathcal{N}(0, I)$, using Adam with learning rate η_g. When generating replay samples, a sigmoid is applied to the decoder logits to obtain pixel intensities in $[0, 1]$.

LAYER	OUTPUT SIZE	ACTIVATION
Solver (classifier)		
input x	$28 \times 28 \times 1$ / $32 \times 32 \times 3$	–
flatten	D	–
dense	$H = 400$	ReLU
dense	$H = 400$	ReLU
dense	$H = 400$	ReLU
dense (logits)	K	linear
Generator, encoder $q_\phi(z \mid x, y)$		
input x, flatten	$D = 784$	–
label embedding (one-hot y)	$H = 400$	ReLU
concatenate	$D + H = 1184$	–
dense	$H = 400$	ReLU
dense	$H = 400$	ReLU
dense	$2L = 200$	linear
split into μ, $\log \sigma^2$	$L = 100$ each	–
Generator, decoder $p_\theta(x \mid z, y)$		
input z	$L = 100$	–
label embedding (one-hot y)	$L = 100$	ReLU
concatenate	$2L = 200$	–
dense	100	ReLU
dense	500	ReLU
dense	1000	ReLU
dense (logits)	$D = 784$	linear
reshape	$28 \times 28 \times 1$	–

Table 3: Hyper-parameters of the experiments. Braces denote the variations of parameters used. All remaining values are held fixed. Every configuration is repeated for three independent runs.

PARAMETER	VALUE(S)
Data and task sequence	
datasets	MNIST, EMNIST-Letters, CIFAR-10
input shape	$28 \times 28 \times 1$, $32 \times 32 \times 3$
task instances	MNIST-5, MNIST-10, EMNIST-26, CIFAR-10
number of classes K	5×2, 10×1, 26×1, 10×1
runs per configuration	3
Optimization	
optimizer	Adam
solver learning rate η_s	$\{10^{-3},\ 10^{-4}\}$
solver epochs per task	$\{5,\ 10,\ 20\}$
batch size B	$\{128,\ 256\}$
Replay and loss	
replay type	ER, DGR-CVAE
loss calculation type	$\mathcal{L}^1$, $\mathcal{L}^2$
mixing/weighting coefficient β	$\{0.1,\ 0.3,\ 0.5,\ \text{dynamic:}\frac{1}{\text{task}}\}$
buffer size (ER)/replay budget (DGR) O	$\{128,\ 256,\ 512,\ 1024,\ 4096\}$
Generator (DGR only)	
latent dimension L	100
encoder hidden width H	400
generator learning rate η_g	10^{-3}
generator epochs per task	20

A.2 IMPLEMENTATION

- **Container:** All experiments were run inside an Apptainer container built from nvidia/cuda:12.8.0-cudnn-devel-ubuntu24.04, using CUDA 12.8 / cuDNN stack on Ubuntu 24.04.

- **Deep learning framework:** TensorFlow/Keras 2.19.0 (Abadi et al., 2016) on the Ubuntu 24.04 system Python 3. Protobuf was constrained to the 5.x series ($\geq 5.28.0, < 6$) to satisfy the dependency requirements of TensorFlow 2.19.0.

- **Scientific stack:** NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), scikit-learn (Pedregosa et al., 2011), pandas (Mckinney, 2011) and TensorFlow Datasets for data handling and evaluation. Matplotlib (Hunter, 2007) and imageio for figure generation.

- **Rendering:** Headless rendering through EGL using NVIDIA driver libraries (libnvidia-gl-550) provided inside the container.

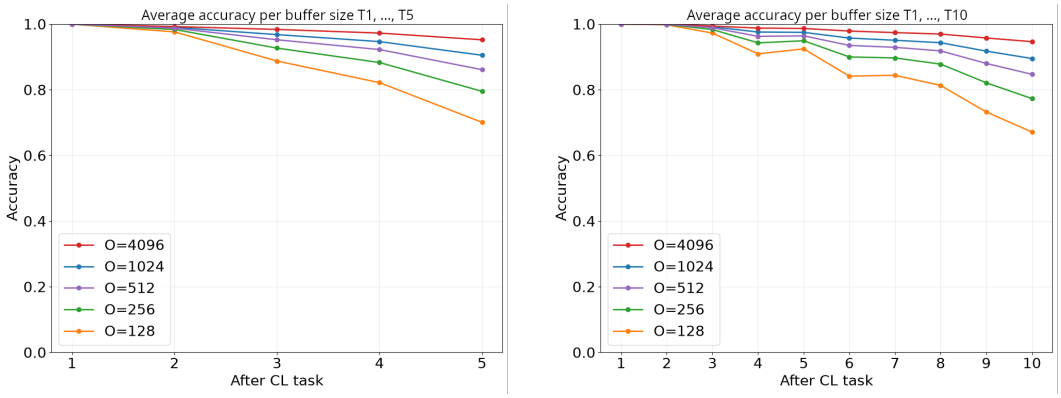
- **Compute infrastructure:** A heterogeneous GPU cluster of 22 compute nodes managed by Slurm (slurm-wlm 23.11.4, Yoo et al., 2003). Since node hardware differs, only hardware-independent metrics are reported and wall-clock runtimes are not compared across nodes.

A.3 EVALUATIONS

A.3.1 EXPERIENCE REPLAY (ER)

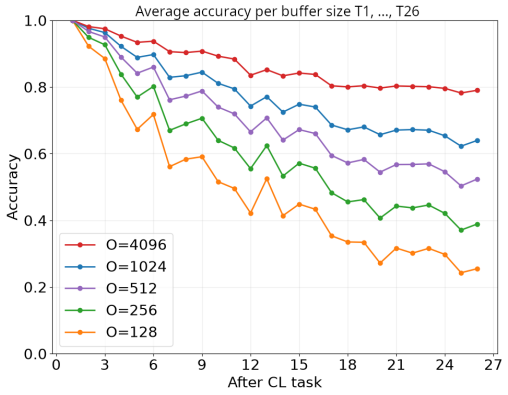
Table 4: Average final performance per buffer size for each benchmark with ER. Standard deviations combine seed noise with the variation between hyper-parameter settings.

BUFFER SIZE	AVERAGE ACCURACY	FORGETTING
MNIST-5		
4096	0.9518 ± 0.0049	0.0472 ± 0.0086
1024	0.9048 ± 0.0114	0.1097 ± 0.0181
512	0.8609 ± 0.0197	0.1662 ± 0.0276
256	0.7949 ± 0.0321	0.2497 ± 0.0425
128	0.7005 ± 0.0466	0.3687 ± 0.0603
MNIST-10		
4096	0.9462 ± 0.0074	0.0502 ± 0.0093
1024	0.8947 ± 0.0112	0.1112 ± 0.0157
512	0.8468 ± 0.0197	0.1662 ± 0.0249
256	0.7728 ± 0.0330	0.2496 ± 0.0388
128	0.6711 ± 0.0510	0.3636 ± 0.0581
EMNIST-26		
4096	0.7900 ± 0.0185	0.1879 ± 0.0311
1024	0.6397 ± 0.0419	0.3567 ± 0.0541
512	0.5238 ± 0.0599	0.4827 ± 0.0709
256	0.3889 ± 0.0709	0.6271 ± 0.0802
128	0.2549 ± 0.0678	0.7694 ± 0.0756
CIFAR-10		
4096	0.3066 ± 0.1097	0.4818 ± 0.2351
1024	0.2427 ± 0.0815	0.5940 ± 0.2068
512	0.2067 ± 0.0647	0.6750 ± 0.1883
256	0.1719 ± 0.0450	0.7541 ± 0.1708
128	0.1423 ± 0.0304	0.8254 ± 0.1481

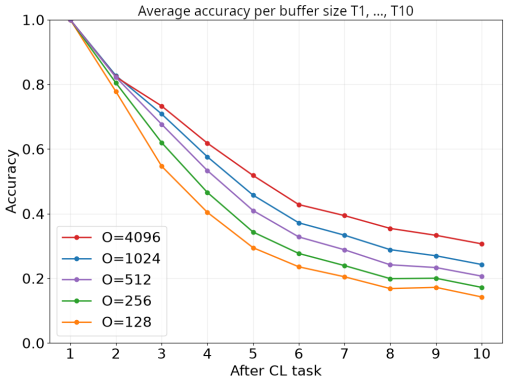


(a) MNIST-5, 2 classes per task (5 tasks)

(b) MNIST-10, 1 class per task (10 tasks)



(c) EMNIST-26, 1 class per task (26 tasks)



(d) CIFAR-10, 1 class per task (10 tasks)

Figure 4: Average accuracy over the tasks seen so far, over all configurations (which are averaged over three runs each) for ER, per buffer size. Each panel shows one benchmark. Not only for best configurations (fig. 2), but also on average, larger buffers retain accuracy longer, and the gap between buffer sizes widens with the number of tasks.

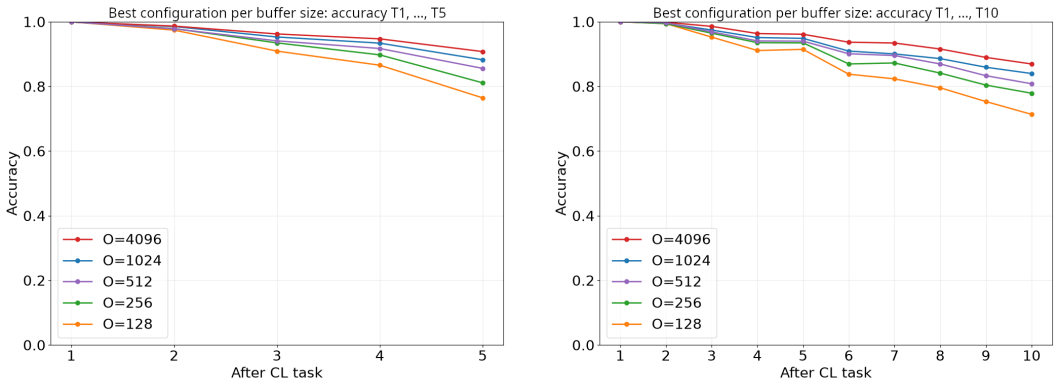
Table 5: Paired comparison of the single-loss ($\mathcal{L}^1$) and two-loss ($\mathcal{L}^2$) variants for ER. Each row is one configuration, evaluated with both variants and averaged over three runs each with all other hyper-parameters identical. Per benchmark, the upper five rows are the best configurations selected by $\mathcal{L}^2$ and the lower five those selected by $\mathcal{L}^1$. Three configurations rank among the best five under both selection criteria and therefore appear twice. All rows use a buffer size of 4096 and a learning rate of 10^{-4}, except the row marked †, which uses 10^{-3}. Δ is signed so that negative values favor $\mathcal{L}^2$ in both groups: $\mathcal{L}^1 - \mathcal{L}^2$ for accuracy, $\mathcal{L}^2 - \mathcal{L}^1$ for forgetting.

CONFIGURATION			AVERAGE ACCURACY			FORGETTING		
β	EPOCHS	BATCH	$\mathcal{L}^1$	$\mathcal{L}^2$	Δ	$\mathcal{L}^1$	$\mathcal{L}^2$	Δ
MNIST-5, selected by $\mathcal{L}^2$								
0.1	20	256	0.9574	0.9588	−0.0014	0.0444	0.0372	−0.0072
0.1	20	128	0.9556	0.9582	−0.0026	0.0468	0.0402	−0.0066
$\frac{1}{\text{task}}$	20	128	0.9564	0.9573	−0.0009	0.0458	0.0434	−0.0024
0.1	10	128	0.9576	0.9568	+0.0008	0.0435	0.0369	−0.0066
0.1	5	128	0.9592	0.9566	+0.0026	0.0391	0.0217	−0.0174
MNIST-5, selected by $\mathcal{L}^1$								
0.1	5	128	0.9592	0.9566	+0.0026	0.0391	0.0217	−0.0174
$\frac{1}{\text{task}}$	10	128	0.9589	0.9546	+0.0043	0.0427	0.0436	+0.0009
0.1	5	256	0.9580	0.9478	+0.0102	0.0412	0.0158	−0.0254
0.1	10	128	0.9576	0.9568	+0.0008	0.0435	0.0369	−0.0066
0.1	10	256	0.9576	0.9548	+0.0028	0.0428	0.0289	−0.0139
MNIST-10, selected by $\mathcal{L}^2$								
$\frac{1}{\text{task}}$	20	128	0.9515	0.9543	−0.0028	0.0488	0.0434	−0.0054
$\frac{1}{\text{task}}$	10	128	0.9515	0.9542	−0.0027	0.0485	0.0388	−0.0097
0.1	10	128	0.9505	0.9537	−0.0032	0.0494	0.0385	−0.0109
0.1	10	256	0.9519	0.9521	−0.0002	0.0476	0.0300	−0.0176
0.1	20	256	0.9517	0.9521	−0.0004	0.0483	0.0414	−0.0069
MNIST-10, selected by $\mathcal{L}^1$								
$\frac{1}{\text{task}}$	5	256†	0.9540	0.9476	+0.0064	0.0456	0.0436	−0.0020
0.1	5	256	0.9540	0.9439	+0.0101	0.0437	0.0240	−0.0197
0.1	5	128	0.9539	0.9512	+0.0027	0.0453	0.0290	−0.0163
$\frac{1}{\text{task}}$	20	256	0.9525	0.9512	+0.0013	0.0471	0.0427	−0.0044
$\frac{1}{\text{task}}$	5	256	0.9520	0.9393	+0.0127	0.0443	0.0237	−0.0206
EMNIST-26, selected by $\mathcal{L}^2$								
0.1	5	128	0.8017	0.8115	−0.0098	0.1816	0.1228	−0.0588
0.3	10	256	0.8052	0.8113	−0.0061	0.1811	0.1597	−0.0214
0.1	10	256	0.7972	0.8101	−0.0129	0.1884	0.1305	−0.0579
0.1	10	128	0.7924	0.8100	−0.0176	0.1946	0.1518	−0.0428
$\frac{1}{\text{task}}$	10	256	0.7996	0.8088	−0.0092	0.1852	0.1124	−0.0728
EMNIST-26, selected by $\mathcal{L}^1$								
$\frac{1}{\text{task}}$	5	256	0.8112	0.7999	+0.0113	0.1650	0.0758	−0.0892
0.5	5	128	0.8111	0.7897	+0.0214	0.1758	0.1930	+0.0172
0.3	5	256	0.8096	0.7794	+0.0302	0.1696	0.1741	+0.0045
0.3	5	128	0.8076	0.8027	+0.0049	0.1778	0.1663	−0.0115
$\frac{1}{\text{task}}$	5	128	0.8056	0.8077	−0.0021	0.1781	0.1066	−0.0715
CIFAR-10, selected by $\mathcal{L}^2$								
0.1	10	128	0.3873	0.4317	−0.0444	0.4169	0.2154	−0.2015
$\frac{1}{\text{task}}$	10	256	0.4033	0.4251	−0.0218	0.3660	0.2006	−0.1654
$\frac{1}{\text{task}}$	20	128	0.3788	0.4245	−0.0457	0.4965	0.2780	−0.2185
$\frac{1}{\text{task}}$	5	128	0.4134	0.4227	−0.0093	0.3026	0.2024	−0.1002
0.1	10	256	0.3878	0.4225	−0.0347	0.3974	0.2155	−0.1819
CIFAR-10, selected by $\mathcal{L}^1$								
$\frac{1}{\text{task}}$	5	256	0.4177	0.4017	+0.0160	0.2902	0.1866	−0.1036
$\frac{1}{\text{task}}$	5	128	0.4134	0.4227	−0.0093	0.3026	0.2024	−0.1002
0.1	5	256	0.4096	0.4019	+0.0077	0.3294	0.1830	−0.1464
0.3	5	128	0.4095	0.3912	+0.0183	0.3952	0.4247	+0.0295
0.3	5	256	0.4078	0.3471	+0.0607	0.4219	0.4811	+0.0592

Table 6: Standard deviations for the paired comparison in table 5 for ER. Values are means over three runs with the standard deviation across those runs. Per benchmark, the upper five rows are the best configurations selected by $\mathcal{L}^2$ and the lower five those selected by $\mathcal{L}^1$. Three configurations rank among the best five under both selection criteria and therefore appear twice. All rows use a buffer size of 4096 and a learning rate of 10^{-4}, except the row marked †, which uses 10^{-3}.

CONFIGURATION			AVERAGE ACCURACY		FORGETTING	
β	EPOCHS	BATCH	$\mathcal{L}^1$	$\mathcal{L}^2$	$\mathcal{L}^1$	$\mathcal{L}^2$
MNIST-5, selected by $\mathcal{L}^2$						
0.1	20	256	0.9574 ± 0.0008	0.9588 ± 0.0016	0.0444 ± 0.0006	0.0372 ± 0.0020
0.1	20	128	0.9556 ± 0.0023	0.9582 ± 0.0003	0.0468 ± 0.0035	0.0402 ± 0.0011
$\frac{1}{\text{task}}$	20	128	0.9564 ± 0.0006	0.9573 ± 0.0007	0.0458 ± 0.0010	0.0434 ± 0.0008
0.1	10	128	0.9576 ± 0.0016	0.9568 ± 0.0010	0.0435 ± 0.0020	0.0369 ± 0.0014
0.1	5	128	0.9592 ± 0.0017	0.9566 ± 0.0010	0.0391 ± 0.0030	0.0217 ± 0.0016
MNIST-5, selected by $\mathcal{L}^1$						
0.1	5	128	0.9592 ± 0.0017	0.9566 ± 0.0010	0.0391 ± 0.0030	0.0217 ± 0.0016
$\frac{1}{\text{task}}$	10	128	0.9589 ± 0.0018	0.9546 ± 0.0009	0.0427 ± 0.0030	0.0436 ± 0.0001
0.1	5	256	0.9580 ± 0.0012	0.9478 ± 0.0016	0.0412 ± 0.0012	0.0158 ± 0.0023
0.1	10	128	0.9576 ± 0.0016	0.9568 ± 0.0010	0.0435 ± 0.0020	0.0369 ± 0.0014
0.1	10	256	0.9576 ± 0.0019	0.9548 ± 0.0018	0.0428 ± 0.0024	0.0289 ± 0.0026
MNIST-10, selected by $\mathcal{L}^2$						
$\frac{1}{\text{task}}$	20	128	0.9515 ± 0.0017	0.9543 ± 0.0013	0.0488 ± 0.0017	0.0434 ± 0.0015
$\frac{1}{\text{task}}$	10	128	0.9515 ± 0.0015	0.9542 ± 0.0003	0.0485 ± 0.0017	0.0388 ± 0.0008
0.1	10	128	0.9505 ± 0.0041	0.9537 ± 0.0010	0.0494 ± 0.0038	0.0385 ± 0.0015
0.1	10	256	0.9519 ± 0.0032	0.9521 ± 0.0013	0.0476 ± 0.0028	0.0300 ± 0.0005
0.1	20	256	0.9517 ± 0.0023	0.9521 ± 0.0013	0.0483 ± 0.0025	0.0414 ± 0.0005
MNIST-10, selected by $\mathcal{L}^1$						
$\frac{1}{\text{task}}$	5	256†	0.9540 ± 0.0020	0.9476 ± 0.0029	0.0456 ± 0.0024	0.0436 ± 0.0041
0.1	5	256	0.9540 ± 0.0024	0.9439 ± 0.0017	0.0437 ± 0.0019	0.0240 ± 0.0011
0.1	5	128	0.9539 ± 0.0017	0.9512 ± 0.0014	0.0453 ± 0.0017	0.0290 ± 0.0028
$\frac{1}{\text{task}}$	20	256	0.9525 ± 0.0032	0.9512 ± 0.0029	0.0471 ± 0.0041	0.0427 ± 0.0026
$\frac{1}{\text{task}}$	5	256	0.9520 ± 0.0023	0.9393 ± 0.0034	0.0443 ± 0.0026	0.0237 ± 0.0048
EMNIST-26, selected by $\mathcal{L}^2$						
0.1	5	128	0.8017 ± 0.0011	0.8115 ± 0.0015	0.1816 ± 0.0012	0.1228 ± 0.0045
0.3	10	256	0.8052 ± 0.0048	0.8113 ± 0.0034	0.1811 ± 0.0050	0.1597 ± 0.0034
0.1	10	256	0.7972 ± 0.0052	0.8101 ± 0.0019	0.1884 ± 0.0053	0.1305 ± 0.0040
0.1	10	128	0.7924 ± 0.0038	0.8100 ± 0.0018	0.1946 ± 0.0036	0.1518 ± 0.0019
$\frac{1}{\text{task}}$	10	256	0.7996 ± 0.0023	0.8088 ± 0.0009	0.1852 ± 0.0026	0.1124 ± 0.0019
EMNIST-26, selected by $\mathcal{L}^1$						
$\frac{1}{\text{task}}$	5	256	0.8112 ± 0.0002	0.7999 ± 0.0012	0.1650 ± 0.0012	0.0758 ± 0.0028
0.5	5	128	0.8111 ± 0.0023	0.7897 ± 0.0008	0.1758 ± 0.0024	0.1930 ± 0.0004
0.3	5	256	0.8096 ± 0.0055	0.7794 ± 0.0031	0.1696 ± 0.0057	0.1741 ± 0.0035
0.3	5	128	0.8076 ± 0.0023	0.8027 ± 0.0014	0.1778 ± 0.0011	0.1663 ± 0.0022
$\frac{1}{\text{task}}$	5	128	0.8056 ± 0.0015	0.8077 ± 0.0014	0.1781 ± 0.0012	0.1066 ± 0.0028
CIFAR-10, selected by $\mathcal{L}^2$						
0.1	10	128	0.3873 ± 0.0060	0.4317 ± 0.0030	0.4169 ± 0.0222	0.2154 ± 0.0049
$\frac{1}{\text{task}}$	10	256	0.4033 ± 0.0078	0.4251 ± 0.0017	0.3660 ± 0.0151	0.2006 ± 0.0051
$\frac{1}{\text{task}}$	20	128	0.3788 ± 0.0032	0.4245 ± 0.0089	0.4965 ± 0.0215	0.2780 ± 0.0135
$\frac{1}{\text{task}}$	5	128	0.4134 ± 0.0045	0.4227 ± 0.0051	0.3026 ± 0.0089	0.2024 ± 0.0095
0.1	10	256	0.3878 ± 0.0065	0.4225 ± 0.0033	0.3974 ± 0.0089	0.2155 ± 0.0129
CIFAR-10, selected by $\mathcal{L}^1$						
$\frac{1}{\text{task}}$	5	256	0.4177 ± 0.0041	0.4017 ± 0.0034	0.2902 ± 0.0111	0.1866 ± 0.0150
$\frac{1}{\text{task}}$	5	128	0.4134 ± 0.0045	0.4227 ± 0.0051	0.3026 ± 0.0089	0.2024 ± 0.0095
0.1	5	256	0.4096 ± 0.0045	0.4019 ± 0.0008	0.3294 ± 0.0204	0.1830 ± 0.0195
0.3	5	128	0.4095 ± 0.0093	0.3912 ± 0.0088	0.3952 ± 0.0194	0.4247 ± 0.0224
0.3	5	256	0.4078 ± 0.0046	0.3471 ± 0.0077	0.4219 ± 0.0348	0.4811 ± 0.0211

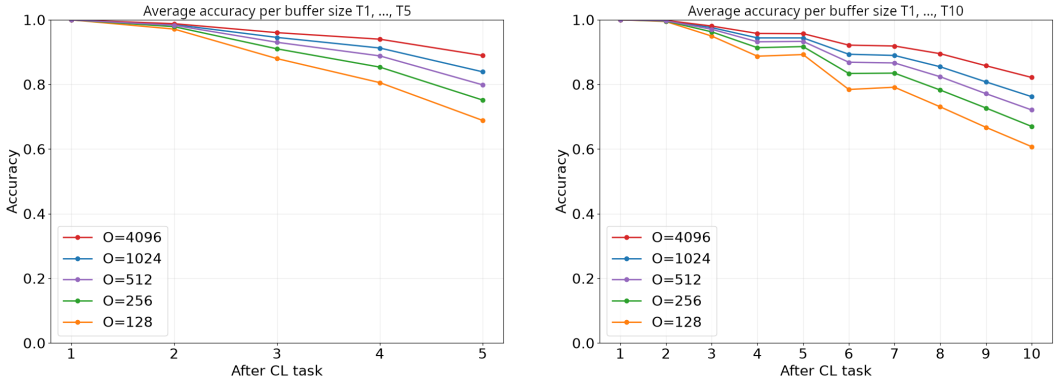
A.3.2 DEEP GENERATIVE REPLAY (DGR)



(a) MNIST-5, 2 classes per task (5 tasks)

(b) MNIST-10, 1 class per task (10 tasks)

Figure 5: Accuracy over the tasks seen so far, for the best DGR configuration per buffer size (averaged over three runs). Each panel shows one benchmark. Larger buffers retain accuracy longer, and the gap between buffer sizes widens with the number of tasks.



(a) MNIST-5, 2 classes per task (5 tasks)

(b) MNIST-10, 1 class per task (10 tasks)

Figure 6: Average accuracy over the tasks seen so far, over all configurations (which are averaged over three runs each) for DGR, per buffer size. Each panel shows one benchmark. Not only for best configurations (fig. 5), but also on average, larger buffers retain accuracy longer, and the gap between buffer sizes widens with the number of tasks.

Table 7: Best configuration per buffer size for MNIST-5 and MNIST-10 with DGR. All configurations use the two-loss variant, 5 epochs and a learning rate of 10^{-4}.

BUFFER	AVERAGE ACCURACY	FORGETTING	β	BATCH
MNIST-5				
4096	0.9076 ± 0.0009	0.0968 ± 0.0013	$\frac{1}{\text{task}}$	128
1024	0.8823 ± 0.0041	0.1311 ± 0.0051	$\frac{1}{\text{task}}$	128
512	0.8556 ± 0.0038	0.1565 ± 0.0056	$\frac{1}{\text{task}}$	256
256	0.8109 ± 0.0067	0.2111 ± 0.0118	0.1	256
128	0.7644 ± 0.0183	0.2749 ± 0.0219	0.1	256
MNIST-10				
4096	0.8692 ± 0.0093	0.1284 ± 0.0110	$\frac{1}{\text{task}}$	128
1024	0.8396 ± 0.0050	0.1524 ± 0.0065	$\frac{1}{\text{task}}$	256
512	0.8080 ± 0.0034	0.1930 ± 0.0027	$\frac{1}{\text{task}}$	256
256	0.7788 ± 0.0141	0.2306 ± 0.0153	$\frac{1}{\text{task}}$	256
128	0.7133 ± 0.0037	0.3067 ± 0.0054	$\frac{1}{\text{task}}$	256

Table 8: Average final performance per buffer size for MNIST-5 and MNIST-10 with DGR. Standard deviations combine seed noise with the variation between hyper-parameter settings.

BUFFER SIZE	AVERAGE ACCURACY	FORGETTING
MNIST-5		
4096	0.8892 ± 0.0106	0.1274 ± 0.0155
1024	0.8387 ± 0.0194	0.1927 ± 0.0268
512	0.7984 ± 0.0262	0.2442 ± 0.0351
256	0.7513 ± 0.0316	0.3039 ± 0.0415
128	0.6884 ± 0.0395	0.3834 ± 0.0508
MNIST-10		
4096	0.8213 ± 0.0229	0.1917 ± 0.0276
1024	0.7619 ± 0.0369	0.2596 ± 0.0433
512	0.7208 ± 0.0419	0.3063 ± 0.0485
256	0.6699 ± 0.0442	0.3640 ± 0.0505
128	0.6073 ± 0.0500	0.4343 ± 0.0565

Table 9: Paired comparison of the single-loss ($\mathcal{L}^1$) and two-loss ($\mathcal{L}^2$) variants for DGR. Each row is one configuration, evaluated with both variants and averaged over three runs each with all other hyper-parameters identical. Per benchmark, the upper five rows are the best configurations selected by $\mathcal{L}^2$ and the lower five those selected by $\mathcal{L}^1$. Three configurations rank among the best five under both selection criteria and therefore appear twice. All rows use a buffer size of 4096 and a learning rate of 10^{-4}, except the rows marked †, which use 10^{-3}. Δ is signed so that negative values favor $\mathcal{L}^2$ in both groups: $\mathcal{L}^1 - \mathcal{L}^2$ for accuracy, $\mathcal{L}^2 - \mathcal{L}^1$ for forgetting.

CONFIGURATION			**AVERAGE ACCURACY**			**FORGETTING**		
β	EPOCHS	BATCH	$\mathcal{L}^1$	$\mathcal{L}^2$	Δ	$\mathcal{L}^1$	$\mathcal{L}^2$	Δ
MNIST-5, selected by $\mathcal{L}^2$								
$\frac{1}{\text{task}}$	5	128	0.8819	0.9076	-0.0257	0.1387	0.0968	-0.0419
0.1	5	128	0.8871	0.9062	-0.0191	0.1310	0.0928	-0.0382
$\frac{1}{\text{task}}$	10	256	0.8999	0.9047	-0.0048	0.1152	0.1013	-0.0139
$\frac{1}{\text{task}}$	5	256	0.8958	0.9036	-0.0078	0.1155	0.0862	-0.0293
0.5	10	128	0.8984	0.9022	-0.0038	0.1193	0.1135	-0.0058
MNIST-5, selected by $\mathcal{L}^1$								
$\frac{1}{\text{task}}$	10	256	0.8999	0.9047	-0.0048	0.1152	0.1013	-0.0139
$\frac{1}{\text{task}}$	20	256	0.8997	0.8988	$+0.0009$	0.1171	0.1163	-0.0008
0.5	5	256†	0.8988	0.8867	$+0.0121$	0.1153	0.1285	$+0.0132$
0.5	10	128	0.8984	0.9022	-0.0038	0.1193	0.1135	-0.0058
0.5	20	256	0.8981	0.8933	$+0.0048$	0.1195	0.1247	$+0.0052$
MNIST-10, selected by $\mathcal{L}^2$								
$\frac{1}{\text{task}}$	5	128	0.8176	0.8692	-0.0516	0.1991	0.1284	-0.0707
$\frac{1}{\text{task}}$	5	256	0.8180	0.8572	-0.0392	0.1973	0.1291	-0.0682
0.3	5	256†	0.8313	0.8517	-0.0204	0.1822	0.1551	-0.0271
0.5	5	128†	0.8314	0.8510	-0.0196	0.1810	0.1575	-0.0235
0.5	10	128†	0.8391	0.8487	-0.0096	0.1726	0.1613	-0.0113
MNIST-10, selected by $\mathcal{L}^1$								
0.5	5	128	0.8400	0.8392	$+0.0008$	0.1719	0.1693	-0.0026
0.5	10	128†	0.8391	0.8487	-0.0096	0.1726	0.1613	-0.0113
0.3	5	256	0.8360	0.8346	$+0.0014$	0.1757	0.1667	-0.0090
0.5	5	256†	0.8359	0.8448	-0.0089	0.1748	0.1629	-0.0119
0.5	20	256	0.8332	0.8301	$+0.0031$	0.1805	0.1824	$+0.0019$

Table 10: Standard deviations for the paired comparison in table 9 for DGR. Values are means over three runs with the standard deviation across those runs. Per benchmark, the upper five rows are the best configurations selected by $\mathcal{L}^2$ and the lower five those selected by $\mathcal{L}^1$. Three configurations rank among the best five under both selection criteria and therefore appear twice. All rows use a buffer size of 4096 and a learning rate of 10^{-4}, except the rows marked †, which use 10^{-3}.

CONFIGURATION			AVERAGE ACCURACY		FORGETTING	
β	EPOCHS	BATCH	$\mathcal{L}^1$	$\mathcal{L}^2$	$\mathcal{L}^1$	$\mathcal{L}^2$
MNIST-5, selected by $\mathcal{L}^2$						
$\frac{1}{\text{task}}$	5	128	0.8819 ± 0.0062	0.9076 ± 0.0009	0.1387 ± 0.0076	0.0968 ± 0.0013
0.1	5	128	0.8871 ± 0.0034	0.9062 ± 0.0064	0.1310 ± 0.0028	0.0928 ± 0.0088
$\frac{1}{\text{task}}$	10	256	0.8999 ± 0.0085	0.9047 ± 0.0033	0.1152 ± 0.0107	0.1013 ± 0.0046
$\frac{1}{\text{task}}$	5	256	0.8958 ± 0.0076	0.9036 ± 0.0059	0.1155 ± 0.0098	0.0862 ± 0.0063
0.5	10	128	0.8984 ± 0.0035	0.9022 ± 0.0039	0.1193 ± 0.0046	0.1135 ± 0.0045
MNIST-5, selected by $\mathcal{L}^1$						
$\frac{1}{\text{task}}$	10	256	0.8999 ± 0.0085	0.9047 ± 0.0033	0.1152 ± 0.0107	0.1013 ± 0.0046
$\frac{1}{\text{task}}$	20	256	0.8997 ± 0.0017	0.8988 ± 0.0015	0.1171 ± 0.0022	0.1163 ± 0.0026
0.5	5	256†	0.8988 ± 0.0040	0.8867 ± 0.0059	0.1153 ± 0.0061	0.1285 ± 0.0072
0.5	10	128	0.8984 ± 0.0035	0.9022 ± 0.0039	0.1193 ± 0.0046	0.1135 ± 0.0045
0.5	20	256	0.8981 ± 0.0025	0.8933 ± 0.0086	0.1195 ± 0.0025	0.1247 ± 0.0121
MNIST-10, selected by $\mathcal{L}^2$						
$\frac{1}{\text{task}}$	5	128	0.8176 ± 0.0122	0.8692 ± 0.0093	0.1991 ± 0.0141	0.1284 ± 0.0110
$\frac{1}{\text{task}}$	5	256	0.8180 ± 0.0075	0.8572 ± 0.0077	0.1973 ± 0.0089	0.1291 ± 0.0109
0.3	5	256†	0.8313 ± 0.0064	0.8517 ± 0.0062	0.1822 ± 0.0084	0.1551 ± 0.0077
0.5	5	128†	0.8314 ± 0.0091	0.8510 ± 0.0014	0.1810 ± 0.0095	0.1575 ± 0.0023
0.5	10	128†	0.8391 ± 0.0227	0.8487 ± 0.0029	0.1726 ± 0.0250	0.1613 ± 0.0033
MNIST-10, selected by $\mathcal{L}^1$						
0.5	5	128	0.8400 ± 0.0137	0.8392 ± 0.0045	0.1719 ± 0.0154	0.1693 ± 0.0053
0.5	10	128†	0.8391 ± 0.0227	0.8487 ± 0.0029	0.1726 ± 0.0250	0.1613 ± 0.0033
0.3	5	256	0.8360 ± 0.0099	0.8346 ± 0.0044	0.1757 ± 0.0103	0.1667 ± 0.0065
0.5	5	256†	0.8359 ± 0.0168	0.8448 ± 0.0061	0.1748 ± 0.0194	0.1629 ± 0.0073
0.5	20	256	0.8332 ± 0.0026	0.8301 ± 0.0019	0.1805 ± 0.0029	0.1824 ± 0.0028