Two Complementary Perspectives to Continual Learning: Ask Not Only *What* to Optimize, But Also *How*

Timm Hess
KU Leuven

Tinne Tuytelaars
KU Leuven

Gido M. van de Ven
KU Leuven

Abstract

Recent years have seen considerable progress in the continual training of deep neural networks, predominantly thanks to approaches that add replay or regularization terms to the loss function to approximate the joint loss over all tasks so far. However, we show that even with a perfect approximation to the joint loss, these approaches still suffer from temporary but substantial forgetting when starting to train on a new task. Motivated by this 'stability gap', we propose that continual learning strategies should focus not only on the optimization objective, but also on the way this objective is optimized. While there is some continual learning work that alters the optimization trajectory (e.g., using gradient projection techniques), this line of research is positioned as alternative to improving the optimization objective, while we argue it should be complementary. In search of empirical support for our proposition, we perform a series of pre-registered experiments combining replay-approximated joint objectives with gradient projection-based optimization routines. However, this first experimental attempt fails to show clear and consistent benefits. Nevertheless, our conceptual arguments, as well as some of our empirical results, demonstrate the distinctive importance of the optimization trajectory in continual learning, thereby opening up a new direction for continual learning research.

1 INTRODUCTION

Learning continually from a stream of non-stationary data is challenging for deep neural networks. When these networks are trained on something new, their default behaviour is to quickly forget most of what was learned before (McCloskey and Cohen, 1989; Ratcliff, 1990). Considerable progress has been made in recent years towards overcoming such 'catastrophic forgetting', for a large part thanks to methods using replay (Robins, 1995; Rolnick et al., 2019) and regularization (Kirkpatrick et al., 2017; Li and Hoiem, 2017). These methods work by adding extra terms to the loss function, and they can be interpreted as attempts to approximate the joint loss over all tasks so far.

Recently a peculiar property of replay and regularization methods was pointed out. These approaches tend to suffer from substantial forgetting when starting to learn a new task, although this forgetting is often temporary and followed by a phase of performance recovery (De Lange et al., 2023). We postulate that avoiding this 'stability gap' is important, both because the transient drops in performance themselves can be problematic (e.g., for safety-critical applications) and because doing so might lead to more efficient and better performing algorithms, as constantly having to re-learn past tasks seems wasteful. Importantly, however, we demonstrate that the stability gap cannot be overcome by merely improving replay or regularization. This motivates us to propose that, instead, continual learning needs an additional perspective: rather than focusing only on *what* to optimize (i.e., the optimization objective), the field should also think about *how* to optimize (i.e., the optimization trajectory).

There are existing works that explore modifying the optimization trajectory as a mechanism for continual learning, but so far this line of research has been positioned as alternative to improving the optimization objective, rather than as complementary. A prime example is Gradient Episodic Memory (GEM; Lopez-Paz and Ranzato, 2017), which alters the optimization process by projecting gradients to encourage parameter updates that do not strongly interfere with old tasks. Crucially, GEM applies this optimization routine to optimizing the loss on the new task, while according to our proposal such modified optimization routines should be used to optimize approximations of the joint loss. As a first evaluation of the merits of our proposition, in this work we use GEM's gradient projection-based optimization routine to instead optimize replay-approximated

Proceedings of the 1st ContinualAI Unconference, 2023, Virtual. PMLR Volume 249. Copyright 2023 by the author(s).

versions of the joint loss. In a series of pre-registered experiments, using both domain- and class-incremental learning benchmarks, we test whether this combined approach reduces the stability gap, and whether this in turn leads to higher learning efficiency and better final performance.

The remainder of the paper is organized as follows. In section 2 we develop our main proposition. In section 3 we review existing optimization-based methods for continual learning. In section 4 we propose experiments of which we hope that they can provide proof-of-concept demonstrations for our main proposition, and in section 5 we describe the detailed pre-registered experimental protocol. In section 6 we present the results, and we end with a discussion and outlook in section 7.

2 TWO PERSPECTIVES TO CONTINUAL LEARNING

In this section we use conceptual arguments and preliminary data to develop the proposition that continual learning should focus not only on *what* to optimize, but also on *how*. We first describe the current dominant approach to continual learning (subsection 2.1), we then point out a fundamental issue with this approach (subsection 2.2), and we finally propose a complementary approach and explain why it could address this issue (subsection 2.3).

To help us reason about the different approaches that continual learning methods could take, in this section we consider the following continual learning problem. Assume a model f_w, parameterized by w, that has learned a set of weights $\widehat{w}_{\text{old}}$ for an initial task[1], or a set of tasks, by optimizing a loss function ℓ_{old} on training data $D_{\text{old}} \sim \mathcal{D}_{\text{old}}$. We then wish to continue training the same model on a new task, by optimizing a loss function ℓ_{new} on training data $D_{\text{new}} \sim \mathcal{D}_{\text{new}}$, in such a way that the model maintains (or possibly improves) its performance on the previously learned task(s). As has been thoroughly described in the continual learning literature, if the model is trained on the new task in the standard way (i.e., optimize the new loss ℓ_{new} with stochastic gradient descent), the typical result is catastrophic forgetting and a solution $\widehat{w}_{\text{new}}$ that is good for the new task but no longer for the old one(s).

2.1 The Standard Approach to Continual Learning: Improving the Loss Function

To mitigate catastrophic forgetting, continual learning research from the past few years has typically focused on making changes to the loss function that is optimized. In particular, rather than optimizing the loss on the new task, many continual learning methods can be interpreted as optimizing an approximate version of the joint loss:

$$\widetilde{\ell}_{\text{joint}} = \ell_{\text{new}} + \widetilde{\ell}_{\text{old}}, \quad (1)$$

with $\widetilde{\ell}_{\text{old}}$ the method's proxy for the loss on the old tasks.

A straight-forward example of this approach is 'experience replay', which approximates ℓ_{old} by revisiting a subset of previously observed examples that are stored in an auxiliary memory buffer. In continual learning experiments, typically limits are imposed on the buffer's storage capacity and/or on the computational budget for training the model (Lesort et al., 2020; Wang et al., 2022; Prabhu et al., 2023). Both constraints prevent full replay of all previously observed data, meaning that ℓ_{old} can only be approximated. A wide range of studies aims to improve the quality of this approximation, for example by modifying the way samples are selected to be stored in the buffer (Rebuffi et al., 2017; Chaudhry et al., 2019b; Aljundi et al., 2019b; Lin et al., 2021; Mundt et al., 2023), or by adaptively selecting which samples from the buffer to replay (Riemer et al., 2018; Aljundi et al., 2019a). As an alternative to storing past samples explicitly, generative models can be learned to approximate the input distributions of previous tasks (Robins, 1995; Shin et al., 2017; van de Ven et al., 2020).

Another popular class of methods for continual learning is based on regularization. As proxy for the loss on the old tasks, these methods add regularizing terms to the loss that impose penalties either for changes to the network's weights ('parameter regularization'; Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018) or for changes to the network's input-output mapping ('functional regularization'; Li and Hoiem, 2017; Dhar et al., 2019; Lee et al., 2019; Titsias et al., 2020). That these methods can be interpreted as attempts to approximate the joint loss can be shown by taking either a Bayesian perspective (Nguyen et al., 2018; Farquhar and Gal, 2019; Kao et al., 2021; Rudner et al., 2022) or a geometric perspective (Kolouri et al., 2020).

In summary, both replay- and regularization-based methods for continual learning operate by changing the loss function that is optimized, often with the aim of creating an approximate version of the joint loss. When developing new continual learning methods of this kind, the challenge is to design better objective functions.

2.2 The Stability Gap: A Challenge for the Standard Approach to Continual Learning

It was recently pointed out that even when replay- or regularization-based methods are considered to perform well (in the sense that they obtain good performance on both old and new tasks after finishing training on the new task), these methods still suffer from substantial, albeit often temporary, forgetting during the initial phase of training on a new task (De Lange et al., 2023). Until recently, this

[1]The term 'task' is used here in a rather general way; it loosely refers to a combination of a data distribution and a loss function.

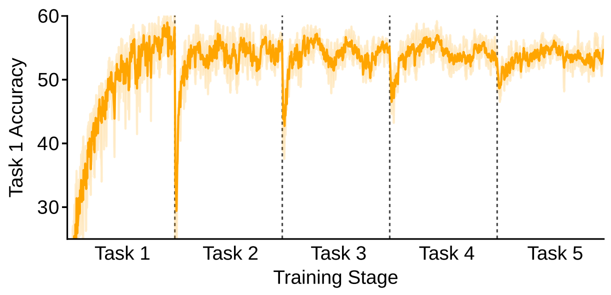


Figure 1: **The stability gap occurs even with incremental joint training (or 'full replay').** Shown is the test accuracy on the first task while the network is incrementally trained on all five tasks of Domain CIFAR. During the n-th task, the network is trained jointly on all training data from the first n tasks. Even with this ideal approximation to ℓ_{joint}, performance severely drops upon encountering a new task. Displayed are the means over five repetitions, shaded areas are ± 1 standard error of the mean. Vertical dashed lines indicate task switches.

phenomenon – referred to as the *stability gap* – had not been observed, or been paid little attention to, due to the common evaluation setup in continual learning that only measures performance after training on the new data has converged. Nevertheless, the stability gap is undesirable, especially for safety-critical applications in which sudden drops in performance can be highly problematic. But also from the perspective of computational efficiency, or even if only the final learning outcome is of interest, avoiding the stability gap might be beneficial, as preventing forgetting seems easier and more efficient than having to re-learn later on (see Figure 4 of Van de Ven et al. (2020) for empirical support for this intuition).

Why does the stability gap happen? One possibility is that the stability gap is due to imprecision in the approximations of the joint loss made by replay and regularization. If this were the case, it could be interpreted as good news for the standard approach to continual learning, as it would imply that by continuing to improve the quality of replay or regularization the stability gap could be overcome. However, this is not the case, as in preliminary experiments we find that the stability gap is consistently observed even with incremental joint training (Figure 1). This indicates that with better approximations to the joint loss alone, the stability gap cannot be solved.

2.3 Proposed Complementary Approach: Improving the Optimization Trajectory

The above observations relating to the stability gap suggest that the standard approach to continual learning of focusing on the loss function is not sufficient. We believe that continual learning would benefit from an additional perspective: rather than concentrating only on improving the optimization objective (i.e., what loss function to optimize), we argue that continual learning should also focus on improving the optimization trajectory (i.e., how to optimize that loss function).

To help explain why we believe that focusing on the optimization trajectory can yield benefits, we revisit the continual learning problem discussed at the start of section 2, which is schematically illustrated in Figure 2. Starting point is a model f_w that has already learned a solution $\widehat{w}_{\text{old}}$ by optimizing loss ℓ_{old}. Continuing to train this model by optimizing loss ℓ_{new} would result in catastrophic forgetting. Instead, as discussed, the standard approach in continual learning is to optimize $\widetilde{\ell}_{\text{joint}} = \ell_{\text{new}} + \widetilde{\ell}_{\text{old}}$, an approximation to the joint loss, rather than ℓ_{new}. Optimizing a suitably approximated version of the joint loss results in a solution $\widehat{w}_{\text{joint}}$ that is good for both the old and the new tasks. However, if this loss is optimized with standard stochastic gradient descent, the trajectory that is taken from $\widehat{w}_{\text{old}}$ to $\widehat{w}_{\text{joint}}$ goes through a region in parameter space where the loss on the old tasks is high. The corresponding transient drop in performance on the old tasks is the stability gap.

A first possibility that must be dealt with is that the stability gap is unavoidable, in the sense that there simply is no path from $\widehat{w}_{\text{old}}$ to $\widehat{w}_{\text{joint}}$ that does not traverse a region where the performance on old tasks is poor. Although this is theoretically possible, this option seems unlikely given recent work on mode connectivity in deep neural networks (Draxler et al., 2018; Garipov et al., 2018) showing that

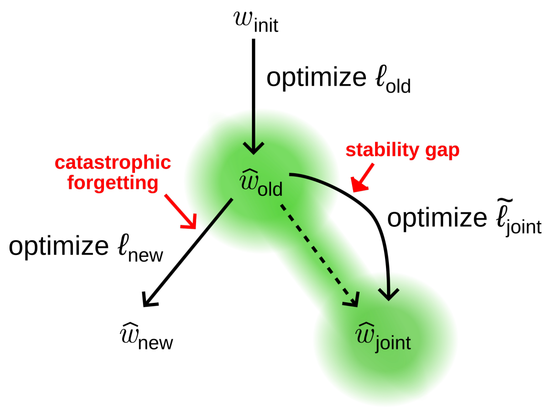

w_{init}
optimize ℓ_{old}
catastrophic forgetting
stability gap
$\widehat{w}_{\text{old}}$
optimize $\widetilde{\ell}_{\text{joint}}$
optimize ℓ_{new}
$\widehat{w}_{\text{new}}$
$\widehat{w}_{\text{joint}}$

Figure 2: **Schematic of the stability gap, and how adjusting the optimization trajectory could avoid it.** When, starting from a solution for the old tasks ($\widehat{w}_{\text{old}}$), a proxy of the joint loss ($\widetilde{\ell}_{\text{joint}}$) is optimized with standard stochastic gradient descent, the optimization trajectory first passes through a region in parameter space with high loss on the old tasks before converging to a solution that is good for all tasks ($\widehat{w}_{\text{joint}}$). Work on mode connectivity suggests that a low-loss path between $\widehat{w}_{\text{old}}$ and $\widehat{w}_{\text{joint}}$ exists as well (dashed arrow), indicating that it should be possible to overcome the stability gap with a different optimization routine. Green shading indicates areas of low loss on the old tasks.

different local optima found by stochastic gradient descent are often connected by simple paths of non-increasing loss. In particular, Mirzadeh et al. (2021) showed that when optimizing a neural network using stochastic gradient descent on the joint loss while starting from a single task solution, the resulting joint solution is connected to the single task solution by a linear manifold of low loss on the single task. The same holds when, starting from a single task solution, the replay-approximated joint loss is optimized rather than the joint loss itself (Verwimp et al., 2021).

The above work on mode connectivity thus suggests that by changing the optimization routine it should be possible to avoid the stability gap. Besides that reducing the stability gap is important for safety-critical applications, we believe that it could bring other benefits for continual learning as well. For example, getting rid of the repeated re-learning cycles that characterize the stability gap could increase the learning efficiency. Moreover, if the loss function is non-convex, as is the case with deep neural networks, changing the way the loss is optimized could also lead to different, and hopefully better, final learning outcomes. The possibility of improved final performance is supported by the observation from Caccia et al. (2022) that the abrupt forgetting after task switches is not always recovered later on. This leads us to the following three hypotheses:

Main hypothesis
Better optimization routines for continual learning can:
(H1) reduce the stability gap.

Secondary hypotheses
Reducing the stability gap can:
(H2) increase learning efficiency;
(H3) improve the final learning outcome.

How to Improve Optimization for Continual Learning? In the above we have made an argument that continual learning should focus on improving its optimization routines, but we have not yet discussed *how* this could be done. To avoid the stability gap, an optimization routine is needed that is less greedy than standard stochastic gradient descent and that favors parameter updates that do not substantially increase the loss on old tasks. Interestingly, as we review in section 3, optimization routines based on gradient projection have been explored in the continual learning literature that already have these properties. Importantly, however, currently these gradient projection-based optimization routines are not used in the way envisioned by us, as they are used to optimize the loss on the new task rather than an approximated version of the joint loss.

2.4 Another Way to Avoid the Stability Gap?

An alternative approach that can circumvent the stability gap in continual learning is to use certain parts of the network only for specific tasks. This approach is employed by network expansion methods (Rusu et al., 2016; Yoon et al., 2018; Yan et al., 2021) and parameter isolation methods (Serra et al., 2018; Masse et al., 2018). To see why this approach can help to avoid the stability gap, consider the extreme case of using a separate sub-network for each task. In this case there is no forgetting at all, and thus also no stability gap. However, the use of task-specific components has some important disadvantages. Firstly, if task identity is not always provided, as is the case with domain- and class-incremental learning (van de Ven et al., 2022), it might not be clear which parts of the network should be used. This issue could be addressed by inferring task identity, for example using generative models or other out-of-distribution detection techniques (van de Ven et al., 2021; Henning et al., 2021; Kim et al., 2022; Zając et al., 2024), but such task inference can be challenging. Secondly, having separate parts of the network per task limits the potential of positive transfer between tasks, which is an important desideratum for continual learning (Hadsell et al., 2020).

3 GRADIENT PROJECTION-BASED OPTIMIZATION

A tool that has been explored in the continual learning literature for modifying the way a given loss function $\ell(w)$ is optimized is 'gradient projection'. With gradient projection, rather than basing the parameter updates on the original gradient $g = \nabla_w \ell(w)$, they are based on a projected version $\bar{g}$ of that gradient. Important from the perspective of our paper, gradient projection does not alter the loss function that is optimized (e.g., the loss landscape and its local minima remain unchanged), it only changes the way the loss function is optimized (Kao et al., 2021). In this section we review two current lines of continual learning studies that make use of gradient projection.

3.1 Orthogonal Gradient Projection

Orthogonal gradient projection methods aim to avoid interference between tasks by confining the training of each new task to previously unused subspaces. To restrict parameter updates to directions that do not interfere with the performance on old tasks, the gradient of the loss on the new task is projected to the orthogonal complement of the 'gradient subspaces' of old tasks. Various ways to construct these gradient subspaces have been proposed. Several studies use the subspaces spanned by all layer-wise inputs of old tasks, which they characterize using conceptors (He and Jaeger, 2018) or by iteratively accumulating projector matrices using a recursive least squares algorithm (Zeng et al., 2019; Guo et al., 2022). To reduce the memory and computational costs, Saha et al. (2021) approximate the input subspaces of each task using singular value decomposition and its k-rank approximation. Farajtabar et al. (2020) instead use the span of a set of stored gradient directions of old tasks as the gradient subspace to protect.

Constraining sequential optimization to orthogonal subspaces can mitigate forgetting effectively, but it restricts the learning of new tasks to successively smaller subspaces and it eliminates the potential to further improve the model with respect to old tasks. Orthogonal gradient projection methods especially struggle or break down when the input spaces of different tasks substantially overlap with each other (He, 2018).

Recent advancements in this line of work therefore focus on relaxing the constraints of the orthogonal projection framework to enable knowledge transfer between tasks. Deng et al. (2021) dynamically scale gradients and search for flatter minima, and Lin et al. (2022) use the idea of 'trust regions' (Schulman et al., 2015) to selectively relax constraints for protected subspaces of old tasks most related to the new task. As an alternative, Kao et al. (2021) take a Bayesian perspective and transform the gradients using the inverse of a Kronecker-factored approximation to the Fisher information matrix, and additionally protect previous knowledge by parameter-based regularization.

3.2 Gradient Episodic Memories

Another class of gradient projection-based optimization methods for continual learning, which also does not enforce strict orthogonality of future updates, is based on Gradient Episodic Memory (GEM; Lopez-Paz and Ranzato, 2017). The projection mechanism of this approach is motivated by a constrained optimization problem, where the goal is to optimize ℓ_{new} without increasing ℓ_{old}:

$$\min_w \ell_{\text{new}}(w), \text{ such that } \ell_{\text{old}}(w) \leq \ell_{\text{old}}(\widehat{w}_{\text{old}}). \quad (2)$$

To determine whether a parameter update based on $g = \nabla_w \ell_{\text{new}}(w)$ might increase ℓ_{old}, the gradient(s) for the old task(s) are estimated using examples from a replay buffer: $g_{\text{old}} = \nabla_w \widetilde{\ell}_{\text{old}}(w)$. If the directions of g and g_{old} align (in the sense that their angle does not exceed 90°), it is conjectured that a parameter update based on g is unlikely to increase ℓ_{old}, and g is left unchanged. If the angle between g and g_{old} exceeds 90°, g is projected to $\bar{g} = g - \frac{g^{\mathrm{T}} g_{\text{old}}}{g_{\text{old}}^{\mathrm{T}} g_{\text{old}}} g_{\text{old}}$, which is the closest gradient to g (in l_2-norm) with a 90° angle to g_{old}. Because in our continual learning example it is the case that ℓ_{old} encompasses all past tasks, this description actually corresponds to Averaged GEM (A-GEM; Chaudhry et al., 2019a), a computationally more efficient version of GEM. The original formulation of GEM enforces $\bar{g}$ to align with the gradient of each individual past task (Lopez-Paz and Ranzato, 2017), see Appendix D for details.

Given that GEM and A-GEM explicitly aim to prevent increases of the loss on old tasks, these methods might be able to avoid the stability gap. Empirically, however, this is not the case, as GEM suffers from considerably larger stability gaps than experience replay (De Lange et al., 2023). Moreover, also in terms of final performance, experience replay consistently outperforms both GEM and A-GEM (De Lange et al., 2022; van de Ven et al., 2022).

We expect that the disappointing performance of GEM is due to its choice of objective function: GEM optimizes the loss on the new task (i.e., ℓ_{new}) rather than an approximation to the joint loss (i.e., $\widetilde{\ell}_{\text{joint}}$). In other words, we believe that GEM under-utilizes its replay buffer by solely delineating gradient constraints but not actively optimizing the replay-approximated joint loss. When GEM was proposed, it was assumed that directly optimizing a joint loss approximated with a relatively small replay buffer could not work well due to overfitting (Lopez-Paz and Ranzato, 2017), but recent work indicates such overfitting is not as detrimental as thought (Chaudhry et al., 2019b; Verwimp et al., 2021). Nevertheless, as far as we are aware, changing GEM's objective function has not been explored.

4 PROOF-OF-CONCEPT EXPERIMENTS

As discussed in the last section, the gradient projection-based optimization routine of GEM encourages parameter updates that do not strongly interfere with old tasks without imposing overly strict constraints that would fully segregate tasks. Yet, so far this optimization routine has not been used to optimize proxies of the joint loss. This makes GEM a convenient tool for a first set of proof-of-concept experiments to evaluate the merits of our proposition that continual learning should consider both *what* and *how* to optimize. We plan to combine GEM's optimization routine both with a basic version of experience replay that explicitly approximates the joint loss (subsection 4.1) and with state-of-the-art replay-based methods (subsection 4.2).

4.1 Experience Replay with Gradient Projection-based Optimization

In a first set of experiments we test whether, when the optimization objective is a standard replay-approximated version of the joint loss, using GEM's gradient projection-based optimization routine provides the benefits hypothesized in subsection 2.3.

Approximating the Joint Loss To approximate the joint loss we use a basic version of Experience Replay (ER). In our implementation of ER, at the end of each task new examples are added to the memory buffer using class-balanced sampling from the training set, and in each training iteration uniform sampling from the buffer is used to choose which samples to replay. To approximate the joint loss as closely as possible, when training on the n-th task, we balance the loss on the current data and the loss on the replayed data using $\widetilde{\ell}_{\text{joint}} = \frac{1}{n}\ell_{\text{new}} + (1 - \frac{1}{n})\widetilde{\ell}_{\text{old}}$. In each iteration, the total number of replayed samples from all past tasks combined is always equal to b, which is the size of the mini-batch from the current task.

In addition to approximating the joint loss with ER, we also run experiments using the joint loss itself. For this, all training data from past tasks are stored, and in each iteration we use b samples from each past task to compute ℓ_{old}. This can be thought of as 'full replay'.

Optimization Trajectory To try to improve the sub-optimal optimization trajectory that is taken by standard ER, we use the gradient projection-based optimization routines of GEM and A-GEM. Importantly, we only use the optimization routines of GEM and A-GEM, not their optimization objectives. As optimization objective we instead use the replay-approximated joint loss $\widetilde{\ell}_{\text{joint}}$. To achieve this, in the description of GEM in the first paragraph of subsection 3.2, we only need to replace all mentions of ℓ_{new} with $\widetilde{\ell}_{\text{joint}}$. To further illustrate our proposed combination approach, pseudocode for ER + A-GEM is provided in Algorithm 1.

Algorithm 1 ER + A-GEM

```
Require: parameters $w$, loss function $\ell$, learning rate $\lambda$,
         data stream $\{D_1, ..., D_T\}$
  $M \leftarrow \{\}$
  for $t = 1, ..., T$ do
      for $(x, y) \in D_t$ do
          $g \leftarrow \nabla_w \ell(f_w(x), y)$
          $(\tilde{x}, \tilde{y}) \leftarrow$ SAMPLE($M$)
          $g_{\text{old}} \leftarrow \nabla_w \ell(f_w(\tilde{x}), \tilde{y})$
          $g_{\text{joint}} \leftarrow \frac{1}{t} g + (1 - \frac{1}{t}) g_{\text{old}}$
          $\bar{g} \leftarrow$ PROJECT_AGEM($g_{\text{joint}}, g_{\text{old}}$)
          $w \leftarrow$ OPTIMIZER_STEP($w, \lambda, \bar{g}$)
      end for
      $M \leftarrow$ UPDATE_BUFFER($M, D_t$)
  end for

  function PROJECT_AGEM($g, g_{\text{ref}}$)
      if $g^{\mathrm{T}} g_{\text{ref}} \geq 0$ then
          return $g$
      else
          return $g - \frac{g^{\mathrm{T}} g_{\text{ref}}}{g_{\text{ref}}^{\mathrm{T}} g_{\text{ref}}} g_{\text{ref}}$
      end if
  end function
```

Approaches to Compare The main experimental comparison of interest is between standard ER and our proposed combination approach ER + GEM, as this allows testing whether, when doing continual learning by optimizing a proxy of the joint loss, benefits can be gained by changing the way this objective is optimized. Additionally, to probe the individual contributions of the optimization objective and the optimization routine, we also include GEM itself and continual finetuning in our experimental comparison. See Table 1 for an overview of the approaches we compare. In this table ER can be replaced by 'full replay', and GEM can be replaced by A-GEM. We run experiments with all combinations of these base methods.

Table 1: Overview of the approaches to compare in our proof-of-concept experiment, illustrated with ER and GEM as base methods. GP: gradient projection.

Method	Approximate joint loss	GP-based optimization
Finetuning	✗	✗
ER	✓	✗
GEM	✗	✓
ER + GEM	✓	✓

4.2 Improving State-of-the-art

Next we ask whether the use of gradient projection-based optimization could improve the performance of state-of-the-art replay-based methods. To test this, we run the methods Dark Experience Replay (DER; Buzzega et al., 2020) and Bias Correction (BiC; Wu et al., 2019) both with and without using the optimization routine of A-GEM. For these experiments we only consider A-GEM, as it is not straight-forward to combine DER with the original version of GEM. We implement DER and BiC according to their original papers. For completeness, details for both methods are provided in Appendix B. As BiC is a method that is specialized for class-incremental learning, it is included only with the class-incremental learning benchmarks.

5 EXPERIMENTAL PROTOCOL

5.1 Setup

We consider a task-aware supervised continual learning setting, with a task sequence $\mathcal{T} = \{T_1, ..., T_T\}$ of T disjoint classification tasks T_t. A fixed capacity neural network model f_w is incrementally trained on these tasks with a cross-entropy classification loss. When training on task T_t, the model has only access to the training data $D_t = \{X_t, Y_t\}$ of that task and the data in the memory buffer (see below), and the goal is to learn a model with strong performance on all tasks $T_{\leq t}$ encountered so far. The model may be evaluated after any parameter update.

Benchmarks We conduct our study on four benchmarks, covering the domain- and class-incremental learning scenarios (van de Ven et al., 2022). As class-incremental learning benchmarks we use Split CIFAR-100, which is based on the CIFAR-100 dataset (Krizhevsky et al., 2009), and Split Mini-Imagenet, which is based on Mini-Imagenet (Vinyals et al., 2016). Both original datasets contain 50,000 RGB images of 100 classes; CIFAR-100 in resolution 32x32, Mini-Imagenet in resolution 84x84. For Split CIFAR-100 the classes are divided into ten tasks with ten classes each, for Split Mini-Imagenet the classes are divided into twenty tasks with five classes each. In both cases, the classes are divided over the tasks randomly, and for each random seed a different division is used. We also use the CIFAR-100 dataset to construct Domain CIFAR, a domain-incremental learning benchmark. For this benchmark, each of the twenty super-classes of CIFAR-100 is split across five tasks, such that every task contains one member of each super-class (i.e., there are twenty classes per task, one from each super-class). The goal in each task is to predict to which super-class a sample belongs. The other domain-incremental learning benchmark is Rotated MNIST. Each task consists of the entire MNIST dataset (LeCun et al., 1998) with a certain static rotation applied. We construct three tasks with rotations $\{0°, 80°, 160°\}$, as De Lange et al. (2023) found these to provoke the largest stability gaps without inducing ambiguity between digits 6 and 9 (which would happen with rotations close to 180°).

Architectures For the Rotated MNIST benchmark, we use a fully-connected neural network with two hidden layers of 400 ReLUs each, followed by a softmax output layer. For the other benchmarks, following Lopez-Paz and Ranzato (2017), we use a reduced ResNet-18 architecture. Compared to a standard ResNet-18 (He et al., 2016), this architecture has three times less channels in each layer and replaces the 7×7 kernel with stride of 2 in the initial convolutional layer by a 3×3 kernel with stride of 1. The latter prevents an early stark information reduction for images with small resolution. All benchmarks are trained with a single-headed final layer that is shared between all tasks.

Memory Buffer The memory buffer can store up to 100 samples of each class. For the domain-incremental learning benchmarks this means 100 samples of each class per task (e.g., with Rotated MNIST, for each digit the buffer can store 100 examples with rotation 0°, 100 examples with rotation 80° and 100 examples with rotation 160°). Exceptions to this are the experiments with 'full replay', in which all training data are stored.

Offline & Online All experiments are run in both an 'offline version' and an 'online version'. In the offline version, multiple passes over the data are allowed, and the number of training iterations is set relatively high to encourage near convergence for each task. In the online version only a single epoch per task is allowed (i.e., each sample is seen just once, with the exception if it is replayed from memory).

Training Hyperparameters All models are trained using an SGD optimizer with momentum 0.9 and no weight-decay. When gradient projection is used, this optimizer acts on the projected gradients. Except for whitening (with mean and standard deviation of the respective full training sets), no data augmentations are used. Exceptions to this are the experiments with DER and BiC, for which we use the data augmentations described in the original papers that proposed these methods (Buzzega et al., 2020; Wu et al., 2019). In the offline experiments, we train with mini-batch size 128 for around five epochs (Rotated MNIST) or ten epochs (Domain CIFAR, Split CIFAR-100 and Split Mini-Imagenet) per task. To be exact, we use 2000 iterations per task for Rotated MNIST, 800 for Domain CIFAR, 400 for Split CIFAR-100, and 200 for Split Mini-Imagenet. For each experiment in the offline setting, we sweep a set of static learning rates $\{0.1, 0.01, 0.001\}$. For each experiment in the online setting, we sweep both a set of mini-batch sizes $\{10, 64, 128\}$ and a set of static learning rates $\{0.1, 0.01, 0.001\}$. In the online setting, the number of iterations per task is determined by the selected mini-batch size and the number of training samples.

5.2 Evaluation

We track the performance of all methods throughout training using 'continual evaluation' (De Lange et al., 2023). In particular, after every training iteration we evaluate for each task the accuracy of the model on a hold-out test set.

Testing the Hypotheses To quantitatively compare the stability gap of different approaches (i.e., to evaluate **H1**), we use the 'average minimum accuracy' metric defined by De Lange et al. (2023). For completeness, details of this metric are provided in Appendix A. To qualitatively compare the stability gaps, we plot per-task accuracy curves with per-iteration resolution (e.g., as in Figure 1). To compare the learning efficiency of different approaches (i.e., to evaluate **H2**), we use the final average accuracy of the online experiments. To compare the final learning outcomes of different approaches (i.e., to evaluate **H3**), we use the final average accuracy of the offline experiments.

Computational Complexity To provide insight into the computational complexity of the considered methods, we report for each method its empirical training time on the online version of Split CIFAR-100. For this evaluation all methods are run on identical hardware.

Standard Errors Each experiment is run five times, with a different random seed and different division of the classes over tasks for each run. For each metric, both the mean over these runs and the standard error of the mean are reported.

6 RESULTS

In Table 2 we report the quantitative results for our main experimental comparisons across all benchmarks. The average minimum accuracy (MIN) indicates the worst-case accuracy throughout training, which we use as proxy for the stability gap. The final average accuracy (ACC) reflects the performance of the continual learning model at the end of training. The results indicate that combining standard ER, or incremental joint training, with the optimization mechanism of A-GEM does not significantly change either the stability gap or the final performance on any of the tested benchmarks. On the other hand, combining ER or incremental joint training with GEM's optimization mechanism induces clear effects, although these are not always beneficial. For Rotated MNIST, using the optimization mechanism of GEM on top of ER reduces the stability gap, as reflected by an increase of the average minimum accuracy, and improves final performance. These positive effects of GEM's optimization mechanism persist even when used on top of incremental joint training (see also Table E.1 in the Appendix). However, for the other benchmarks, using the optimization mechanism of GEM does not yield these benefits and instead frequently impairs performance.

In the following, we first examine the stability gap for a selection of benchmarks in more detail (subsection 6.1-6.3). Then, we look at the results for extending recent state-of-the-art replay-based methods with A-GEM's optimization mechanism (subsection 6.4). Finally, we evaluate the additional computational overhead induced by the GEM and A-GEM gradient projection mechanisms (subsection 6.5).

6.1 Rotated MNIST

Our results in Table 2 hint that the domain-incremental learning problem of Rotated MNIST provides a case where replay can be improved by altering the optimization trajectory with GEM's gradient projection mechanism. In Figure 3, we depict the per-iteration accuracy on the first task while incrementally training on all the tasks, with detailed views at the task switches to highlight the stability gap. We observe that when training on the third task, ER + GEM modestly reduces the stability gap compared to ER. Regarding the final performance, ER + GEM shows a clear improvement relative to ER. For incremental joint training, the effects of using GEM's optimization routine are more modest, which might be related to the relatively low number of gradients being projected, as indicated in the bottom

Table 2: **Main quantitative results.** For all benchmarks we report the final average accuracy (AVG) and average minimum accuracy (MIN) for standard ER and incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. For each benchmark, this table reports the results obtained with the learning rate (LR) and mini-batch size (BS) that resulted in the highest final accuracy for standard ER and incremental joint training. Bold values mark the highest performance within each 'comparison group' (ER, ER + GEM and ER + A-GEM are one such group; Joint, Joint + GEM and Joint + A-GEM another). Reported is the mean ± standard error over five random seeds. The full quantitative results for all pre-registered experiments are provided in Tables E.1-E.4 in Appendix E.

			ER	ER + GEM	ER + A-GEM	Joint	Joint + GEM	Joint + A-GEM
Rotated MNIST	Offline LR 0.1	MIN	83.1 ± 0.5	**84.1 ± 0.4**	82.5 ± 0.7	**86.7 ± 0.8**	**87.5 ± 0.9**	**86.6 ± 0.6**
		AVG	91.9 ± 0.1	**93.7 ± 0.1**	91.8 ± 0.2	97.5 ± 0.0	**97.8 ± 0.0**	97.5 ± 0.0
	Online LR 0.01; BS 10	MIN	86.8 ± 0.3	**89.1 ± 0.4**	87.1 ± 0.4	92.3 ± 0.4	**92.8 ± 0.3**	92.4 ± 0.1
		AVG	92.7 ± 0.1	**94.2 ± 0.1**	92.8 ± 0.2	**96.8 ± 0.0**	**96.8 ± 0.1**	**96.8 ± 0.1**
Domain CIFAR-100	Offline LR 0.1	MIN	**33.2 ± 1.3**	7.2 ± 2.6	**34.0 ± 0.9**	**42.9 ± 0.7**	**42.2 ± 1.1**	**42.7 ± 0.8**
		AVG	**48.6 ± 0.5**	23.9 ± 1.8	**48.6 ± 0.5**	**52.4 ± 0.8**	**52.6 ± 0.6**	**52.0 ± 0.7**
	Online LR 0.01; BS 10	MIN	**29.2 ± 0.7**	4.3 ± 0.1	**29.2 ± 0.7**	**35.5 ± 1.1**	**35.6 ± 1.3**	**35.1 ± 0.9**
		AVG	**38.3 ± 0.8**	19.8 ± 1.4	**38.3 ± 0.8**	**49.8 ± 1.0**	**49.4 ± 1.5**	**49.7 ± 1.2**
Split CIFAR-100	Offline LR 0.1	MIN	**12.4 ± 0.3**	0.0 ± 0.0	**12.1 ± 0.3**	22.5 ± 0.2	2.8 ± 1.9	**23.1 ± 0.4**
		AVG	**22.8 ± 0.4**	7.1 ± 1.9	**22.4 ± 0.6**	**32.2 ± 0.5**	16.5 ± 6.7	**32.5 ± 0.4**
	Online LR 0.01; BS 10	MIN	**10.6 ± 0.3**	0.0 ± 0.0	**10.7 ± 0.4**	**27.0 ± 0.5**	**27.2 ± 0.5**	**27.2 ± 0.6**
		AVG	**20.7 ± 0.3**	8.5 ± 0.9	**20.5 ± 0.4**	**41.2 ± 0.6**	**41.1 ± 0.4**	**41.4 ± 0.2**
Split Mini-ImageNet	Offline LR 0.1	MIN	**8.6 ± 0.5**	0.0 ± 0.0	**8.3 ± 0.4**	**16.5 ± 0.5**	0.0 ± 0.0	**16.4 ± 0.4**
		AVG	**16.9 ± 0.9**	4.7 ± 1.6	**16.9 ± 0.5**	**28.1 ± 0.6**	3.3 ± 0.4	**28.5 ± 0.7**
	Online LR 0.01; BS 64	MIN	**3.3 ± 0.2**	0.0 ± 0.0	**3.2 ± 0.2**	**9.1 ± 0.2**	0.0 ± 0.0	**8.9 ± 0.3**
		AVG	**13.9 ± 0.5**	5.8 ± 0.3	**14.6 ± 0.3**	**27.9 ± 0.8**	5.5 ± 0.5	**28.3 ± 1.3**

part of the figure. The stability gap is almost unaltered, but we still observe a statistically significant improvement of the average final performance. In contrast, using A-GEM's optimization routine does not yield any improvements on top of ER or incremental joint training. In Appendix D, we investigate a hyperparameter of GEM that can explain part of this difference between ER + GEM and ER + A-GEM .

6.2 Offline Natural Image Benchmarks

In the offline versions of the continual learning benchmarks based on CIFAR-100 and Mini-Imagenet, the benefits we observed with ER + GEM on Rotated MNIST do not recreate. Instead, ER + GEM substantially deteriorates the performance on these benchmarks. As before, using A-GEM's optimization routine does not seem to have any real impact.

Detailed views for Domain CIFAR-100 are depicted in Figure 4. The performance of ER + GEM decreases incrementally due to a collapse originating from the combination of ER with GEM's gradient projection. For Joint + GEM, a collapse does not arise but there is also no benefit over Joint, which may be related to the low number of gradient projections. For Split CIFAR-100, detailed in Figure 5, we find slightly varied behavior. With ER + GEM, there is a small reduction of the stability gap after the first task switch, however, seemingly at the expense of model's ability to recover lost performance. For Joint + GEM, collapse is visible from task 6 onwards, but already during earlier task switches, such as when starting training on task 5, there is an increased instability that is enlarging the stability gap instead of closing it. The results for Split Mini-ImageNet, provided in Figure E.1 in Appendix E, are comparable to those of Split CIFAR-100.

6.3 Online Benchmarks

After running the experiments that we had proposed for the online versions, we found them to be less insightful than we had anticipated. On the one hand, for Rotated MNIST, due to its relatively large training set and rapid convergence of the model, the results for the online version are largely similar to those for the offline version, and therefore do not provide additional insights. On the other hand, for the natural image benchmarks, we found that for the settings that we had chosen for the online versions, there are no clear stability gaps with standard ER or with incremental joint training. This is illustrated for Domain CIFAR-100 in Figure 6. It can further be seen that there are very few gradient projections, which might be related to the absence of a stability gap in these training conditions. For ER + A-GEM there is not a single projection, meaning it fully reduces to standard ER, while for ER + GEM there are projections during the training of later tasks, which seem mostly disruptive to the learning as indicated by the increased noise and subsequent performance collapse. We find similar results for the other natural image benchmarks. For Split CIFAR-100, shown in Figure E.2 in the Appendix, there

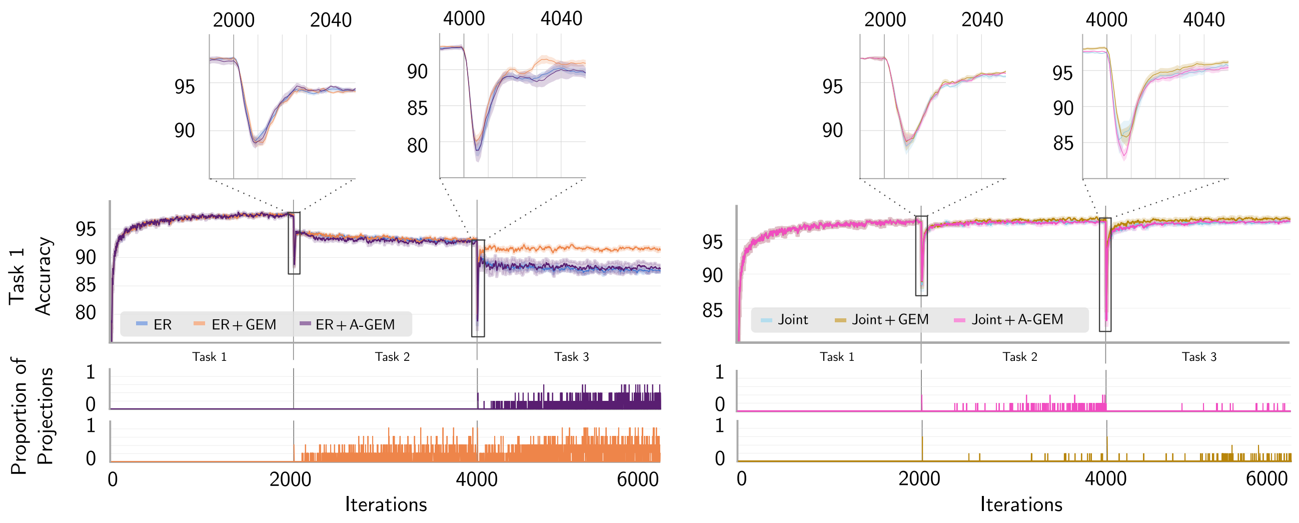


Figure 3: **Stability gaps for the first task of offline Rotated MNIST.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

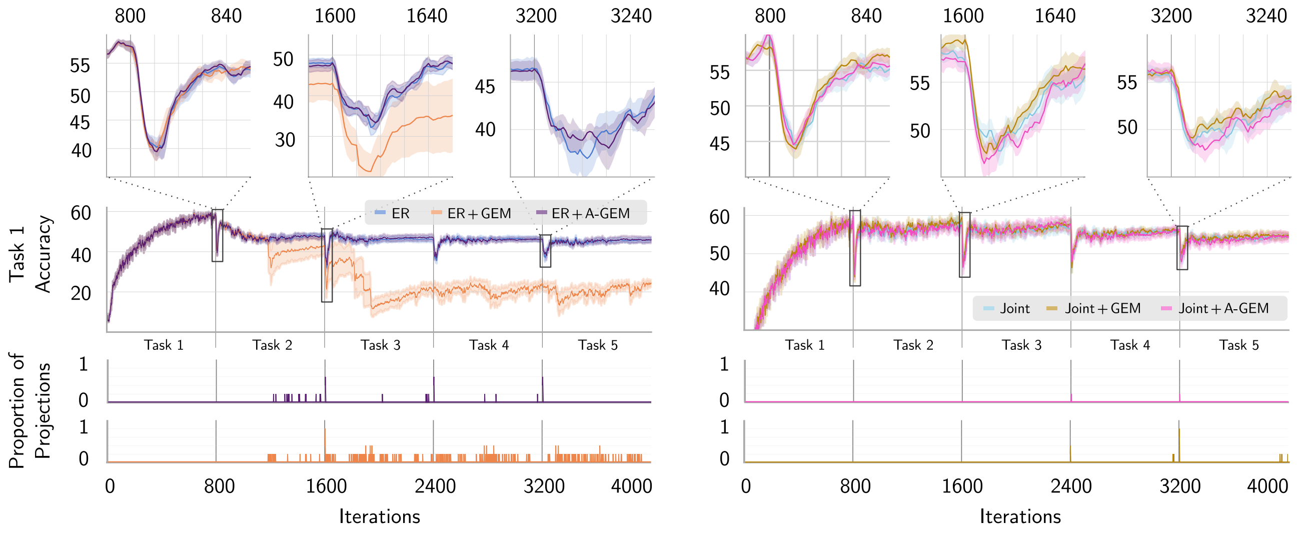


Figure 4: **Stability gaps for the first task of offline Domain CIFAR-100.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

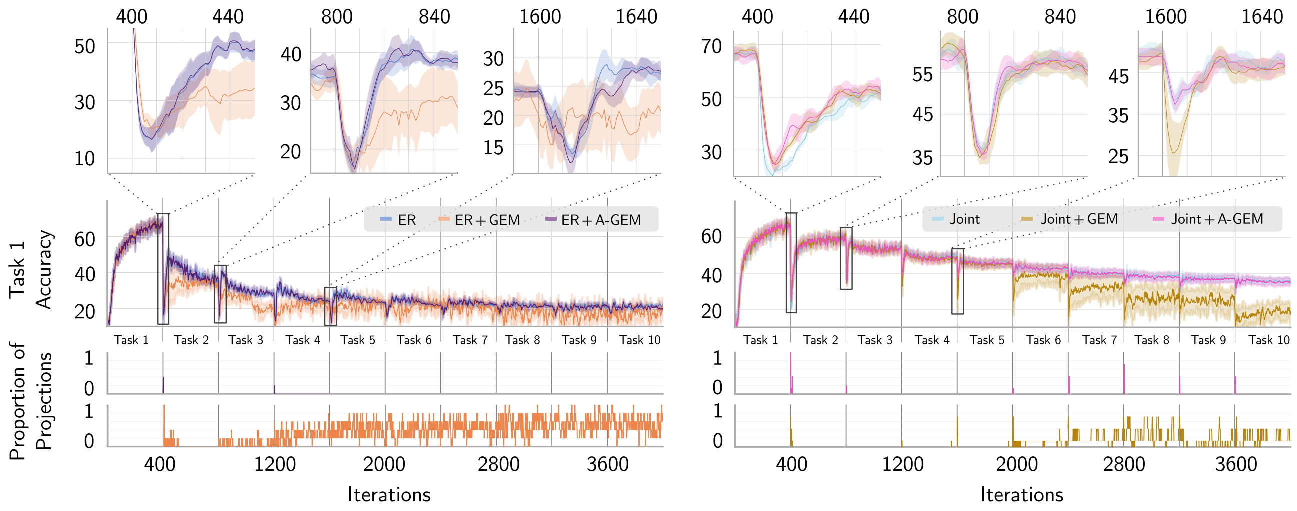


Figure 5: **Stability gaps for the first task of offline Split CIFAR-100.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

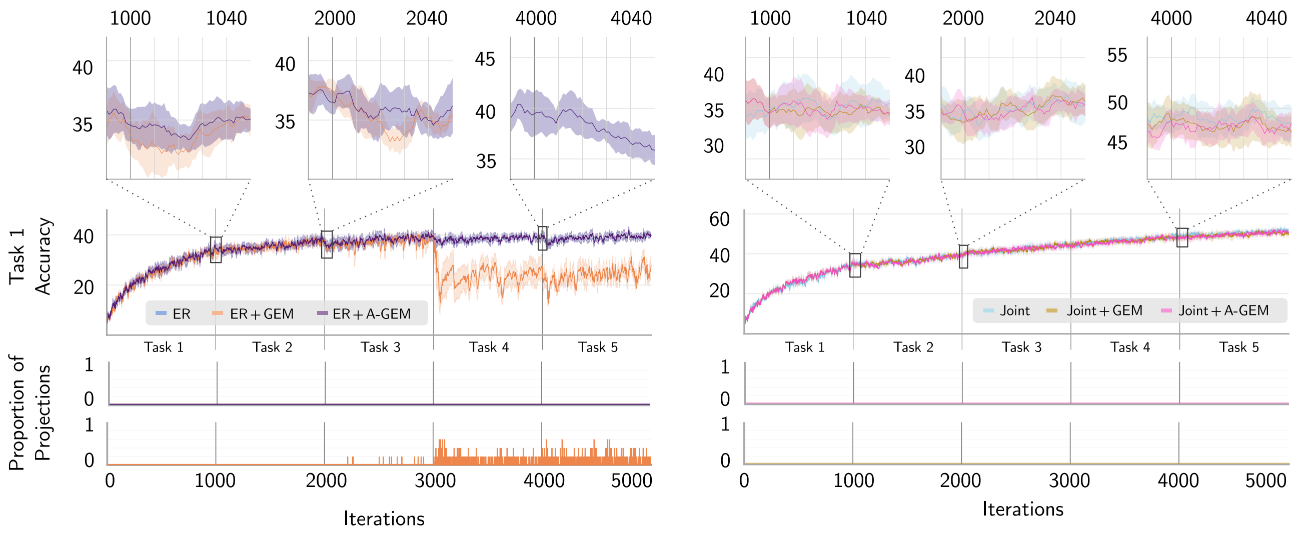


Figure 6: **Stability gaps for first task of online Domain CIFAR-100.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

is only a small stability gap and a few A-GEM projections after the first task switch, but after that not anymore.

6.4 Combining DER and BiC with A-GEM

In Table 3, we report the results for combining the recent state-of-the-art replay-based methods DER and BiC with the gradient projection-based optimization mechanism of A-GEM. Similar as in our main experiments in Table 2, we find no consistent improvements by using A-GEM's optimization mechanism on top of either DER or BiC.

6.5 Computational Complexity

In Table 4, we evaluate the computational complexity of ER + GEM and ER + A-GEM relative to standard ER, by comparing their empirical training time on the online version of Split CIFAR-100. For a fair comparison, all methods are run on the same hardware and we attempted to use comparable implementations. We observe that ER + GEM takes the longest time to train. This can be explained because ER + GEM needs to compute a reference gradient for each previous task and solve the quadratic program

Table 3: **Using A-GEM on top of state-of-the-art methods.** For the benchmarks in the offline setting, we report the final average accuracy (AVG) and average minimum accuracy (MIN) for DER and BiC – both by themselves and in combination with A-GEM's optimization mechanism. This table shows the results for the learning rate (LR) with the best final average accuracy for the standard versions of DER and BiC. Reported is the mean ± standard error over five random seeds.

			DER	DER + A-GEM	BiC	BiC + A-GEM
Rotated MNIST	LR 0.01	MIN	82.6 ± 0.5	83.0 ± 0.4	-	-
		AVG	87.3 ± 0.2	87.1 ± 0.2	-	-
Domain CIFAR-100	LR 0.01	MIN	44.4 ± 0.7	44.8 ± 1.0	-	-
		AVG	59.7 ± 0.5	60.7 ± 0.4	-	-
Split CIFAR-100	LR 0.1	MIN	1.4 ± 0.1	1.6 ± 0.2	22.3 ± 0.9	23.1 ± 1.1
		AVG	26.0 ± 0.9	25.6 ± 1.3	38.3 ± 1.0	38.4 ± 0.9
Split Mini-ImageNet	LR 0.01	MIN	0.1 ± 0.0	0.2 ± 0.1	13.3 ± 0.3	13.5 ± 0.2
		AVG	7.4 ± 0.5	7.6 ± 0.7	25.9 ± 0.7	26.2 ± 0.5

Table 4: **Computational complexity.** The empirical training time on online Split CIFAR-100 is shown. Reported is the mean ± standard error over five runs with learning rate 0.01 and mini-batch size 10.

	Time [s]
ER	59.24 ± 0.66
ER + A-GEM	61.52 ± 0.60
ER + GEM	252.25 ± 1.21

integrated in its gradient projection mechanism. The difference in training time between ER + A-GEM and standard ER is relatively small. The overhead introduced by ER + A-GEM compared to standard ER is governed by the line $\bar{g} \leftarrow \texttt{PROJECT\_AGEM}(g_{\text{joint}}, g_{\text{old}})$ in Algorithm 1: the only additional computations are the dot-product $g^{\mathrm{T}} g_{\text{ref}}$ and the gradient projection $g - \frac{g^{\mathrm{T}} g_{\text{ref}}}{g_{\text{ref}}^{\mathrm{T}} g_{\text{ref}}}$. Extra computations to calculate the reference gradients, which are required for A-GEM's projection, are not needed because those are already available from the ER mechanism.

7 DISCUSSION

Most of the recent progress in continual learning has been achieved by the addition of replay or regularization terms to the loss function, which is done typically with the aim of approximating the loss that one would like to optimize (e.g., the joint loss over all tasks so far). However, in this work we have shown that even if current replay- or regularization-based methods would manage to perfectly approximate the desired loss, they would still suffer from the stability gap. This insight highlights that the current approach to continual learning is not sufficient. This has motivated us to argue that rather than concentrating only on improving the optimization objective (i.e., what loss function to optimize), continual learning should also focus on improving the optimization trajectory (i.e., how to optimize that loss function). Our proposition thus delineates two complementary perspectives to continual learning, which we believe to be a useful framework for developing new continual learning methods.

In search of a proof-of-concept demonstration for our proposition, we proposed a series of pre-registered experiments combining established continual learning methods that approximate the joint loss by experience replay, with existing continual learning methods that change the optimization trajectory. These existing optimization-based continual learning methods, namely GEM and A-GEM, had previously only been used to optimize the loss on the new task, and not – as we propose – to optimize an approximated version of the joint loss. Unfortunately, these pre-registered experiments did not show clear and consistent benefits of this combined approach. In particular, altering the optimization trajectory with A-GEM did not show any substantial impact on the results compared to standard replay. On the other hand, using GEM's optimization on top of standard replay had positive effects on the Rotated MNIST benchmark, but also a disruptive effect on the performance on other benchmarks composed of natural images.

While our pre-registered experiments were not able to confirm our hypotheses or provide convincing proof-of-concept demonstrations, we emphasize that the contributions of this paper extend beyond those empirical results. Indeed, taking inspiration from the stability gap (De Lange et al., 2023), the main contribution of this paper has been to develop the conceptual proposition that continual learning should ask not only *what* to optimize, but also *how*. Following the release of the pre-registered proposal of this paper, our proposition has started to inspire follow-up work. Building on our work, Kamath et al. (2024) demonstrate that the stability gap can occur even with incremental learning of the same task (i.e., when there is no distribution shift). In other recent work, Yoo et al. (2024) provide supporting evidence for our proposition by showing that in an online continual learning setup, the performance of various established replay methods can be improved by using a proximal point method-inspired optimization routine.

Although we have provided compelling arguments that the way in which optimization is done in continual learning can be improved, how to do this largely remains an open question that demands further exploration. Promising directions of future work include the development of more principled methods for gradient projection to bridge the gap between theoretical guarantees and practical performance. Beyond gradient projection, other constrained optimization mechanisms could be explored, such as using Lagrange multipliers (e.g., building upon Elenter et al., 2023) or trust region methods (e.g., building upon Kao et al., 2021). Additionally, such mechanisms may also incorporate second order optimization, intermediate objectives and dynamic learning rates to allow for a smooth learning process.

Acknowledgements

We thank Matthias De Lange, Eli Verwimp and anonymous reviewers for useful comments. This project has been supported by funding from the European Union under the Horizon 2020 research and innovation program (ERC project KeepOnLearning, grant agreement No. 101021347) and under Horizon Europe (Marie Skłodowska-Curie fellowship, grant agreement No. 101067759).

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., and Tuytelaars, T. (2019a). Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems*, 32.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019b). Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems*, 33.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. (2022). New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations (ICLR)*.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019a). Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations (ICLR)*.

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

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2022). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385.

De Lange, M., van de Ven, G. M., and Tuytelaars, T. (2023). Continual evaluation for lifelong learning: Identifying the stability gap. In *International Conference on Learning Representations (ICLR)*.

Deng, D., Chen, G., Hao, J., Wang, Q., and Heng, P.-A. (2021). Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34.

Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. (2019). Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5138–5146.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. (2018). Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, pages 1309–1318.

Elenter, J., NaderiAlizadeh, N., Javidi, T., and Ribeiro, A. (2023). Primal-dual continual learning: Stability and plasticity through lagrange multipliers. *arXiv preprint arXiv:2310.00154*.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. (2020). Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3762–3773.

Farquhar, S. and Gal, Y. (2019). A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in Neural Information Processing Systems*, 31.

Guo, Y., Hu, W., Zhao, D., and Liu, B. (2022). Adaptive orthogonal projection for batch and online continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6783–6791.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, X. (2018). Continual learning by conceptor regularization. In *Continual Learning Workshop (NeurIPS 2018)*.

He, X. and Jaeger, H. (2018). Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations (ICLR)*.

Henning, C., Cervera, M., D'Angelo, F., Von Oswald, J., Traber, R., Ehret, B., Kobayashi, S., Grewe, B. F., and Sacramento, J. (2021). Posterior meta-replay for continual learning. *Advances in Neural Information Processing Systems*, 34.

Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456.

Kamath, S., Soutif-Cormerais, A., Van De Weijer, J., and Raducanu, B. (2024). The expanding scope of the stability gap: Unveiling its presence in joint incremental learning of homogeneous tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4182–4186.

Kao, T.-C., Jensen, K., van de Ven, G., Bernacchia, A., and Hennequin, G. (2021). Natural continual learning: success is a journey, not (just) a destination. *Advances in Neural Information Processing Systems*, 34.

Kim, G., Xiao, C., Konishi, T., Ke, Z., and Liu, B. (2022). A theoretical study on solving continual learning. *Advances in Neural Information Processing Systems*, 35.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kolouri, S., Ketz, N. A., Soltoggio, A., and Pilly, P. K. (2020). Sliced cramer synaptic consolidation for preserving deeply learned representations. In *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Techinal Report*, University of Toronto, Canada.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K., Lee, K., Shin, J., and Lee, H. (2019). Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 312–321.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68.

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

Lin, S., Yang, L., Fan, D., and Zhang, J. (2022). TRGP: Trust region gradient projection for continual learning. In *International Conference on Learning Representations (ICLR)*.

Lin, Z., Shi, J., Pathak, D., and Ramanan, D. (2021). The CLEAR benchmark: Continual LEArning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30.

Masse, N. Y., Grant, G. D., and Freedman, D. J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier.

Mirzadeh, S. I., Farajtabar, M., Gorur, D., Pascanu, R., and Ghasemzadeh, H. (2021). Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations (ICLR)*.

Mundt, M., Hong, Y., Pliushch, I., and Ramesh, V. (2023). A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations (ICLR)*.

Prabhu, A., Al Kader Hammoud, H. A., Dokania, P. K., Torr, P. H., Lim, S.-N., Ghanem, B., and Bibi, A. (2023). Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3698–3707.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285.

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

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations (ICLR)*.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

Rudner, T. G., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. (2022). Continual learning via sequential function-space variational inference. In *International Conference on Machine Learning*, pages 18871–18887.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Saha, G., Garg, I., and Roy, K. (2021). Gradient projection memory for continual learning. In *International Conference on Learning Representations (ICLR)*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, pages 4548–4557.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *Advances in Neural Information Processing Systems*, 30.

Titsias, M. K., Schwarz, J., de G. Matthews, A. G., Pascanu, R., and Teh, Y. W. (2020). Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations (ICLR)*.

van de Ven, G. M., Li, Z., and Tolias, A. S. (2021). Class-incremental learning with generative classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR-W)*, pages 3611–3620.

van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11:4069.

van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197.

Verwimp, E., De Lange, M., and Tuytelaars, T. (2021). 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.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 29.

Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Hong, L., Zhang, S., Li, Z., Zhong, Y., and Zhu, J. (2022). Memory replay with data compression for continual learning. In *International Conference on Learning Representations (ICLR)*.

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

Yan, S., Xie, J., and He, X. (2021). DER: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3014–3023.

Yoo, J., Liu, Y., Wood, F., and Pleiss, G. (2024). Layerwise proximal replay: A proximal point method for online continual learning. *arXiv preprint arXiv:2402.09542*.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations (ICLR)*.

Zając, M., Tuytelaars, T., and van de Ven, G. M. (2024). Prediction error-based classification for class-incremental learning. In *International Conference on Learning Representations (ICLR)*.

Zeng, G., Chen, Y., Cui, B., and Yu, S. (2019). Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, pages 3987–3995.

APPENDIX

The Appendix contains additional information for the content presented in the main body. Appendices A and B expand upon the average minimum accuracy metric utilized in the main text to quantify the stability gap, and provide the details on the BiC and DER continual learning mechanisms. Appendix C details the nuances of our ER + GEM combination approach implementation. Appendix D contains additional experiments beyond our original experimental protocol that analyze the influence of GEM's hyperparameter γ. Finally, in Appendix E, we disclose the results for all experiments including those not already featured in the main text.

Documented code to reproduce all experiments is publicly available at `https://github.com/TimmHess/TwoComplementaryPerspectivesCL`.

A ACCURACY METRICS

Classification accuracy is the base metric we use throughout this work. Given a data set D and model f, the classification accuracy $\mathbf{A}(D, f)$ is the percentage of samples in D that is correctly classified by f.

Final average accuracy is the classification accuracy averaged over all tasks at the end of training. Formally:

$$\textbf{avg-ACC} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{A}(\hat{D}_t, f_{w_{\text{final}}}), \tag{3}$$

where $\hat{D}_t, \forall t \in [1, ..., T]$ is the evaluation set of each task, and $f_{w_{\text{final}}}$ is the final model after concluding training on the all T tasks. This is a common metric to express the quality of the continually learned model.

Average minimum accuracy denotes the average of the lowest classification accuracies observed for each task, as measured from directly after finishing training on that task until the end of all learning in the task sequence:

$$\textbf{min-ACC}(T_t) = \min_{|T_t|<n\leq|T_T|}\mathbf{A}(\hat{D}_t, f_{w_n}), \tag{4}$$

$$\textbf{avg-min-ACC} = \frac{1}{T-1}\sum_{t}^{T-1}\textbf{min-ACC}(T_t), \tag{5}$$

where f_{w_n} indicates the model after the n^{th} training iteration. By slight abuse of notation, $n = |T_t|$ refers to the last training iteration of task T_t, and $n = |T_T|$ marks the final training iteration of the entire task sequence $\mathcal{T}$. This metric is a worst case measure of how well the model's classification accuracy is maintained at any point throughout continual training. To render per-iteration calculation of $\mathbf{A}(\hat{D}_t, f_{w_n})$ computationally feasible, we resort to a reduced evaluation set of size 1000 per task. The reduced set is sampled uniformly from each evaluation set respectively, once for every run. This is similar to De Lange et al. (2023) and has empirically shown to closely approximate using the full evaluation set.

B METHOD DETAILS

BiC (**Bi**as-**C**orrection) was proposed by Wu et al. (2019), who were motivated by the observation that class-incremental continual training causes the continually trained classifier to become biased towards the most recently observed set of classes. The approach is specifically designed for class-incremental learning settings where each observed task T_t introduces a set of non-overlapping classes m, such that the corresponding data $X_t^m = \{(x_i, y_i), \forall y_i \in [n+1, ..., n+m]\}$. Here, x_i, y_i denote the example and label pair, and n is the number of already observed classes. To correct the bias, a (small) set of validation data for each task is stored. These data are taken from the training set and excluded from training the model. They are only used for correcting the classifier's bias in a separate training stage. The bias-correction itself is realized by a linear layer that consists of two parameters, α and β, and is called the bias-correction layer. The logits produced by the model for previously observed classes are kept unaltered, but the bias in the logits produced for the m newly observed classes ($n + 1, ..., n + m$) is corrected by the bias-correction layer:

$$q_k = \begin{cases} o_k & 1 \leq k \leq n \\ \alpha o_k + \beta & n+1 \leq k \leq n+m, \end{cases} \tag{6}$$

where o_k denotes the output logits for the k^{th} class. The bias-correction parameters (α, β) are shared for all new classes and optimized via a cross-entropy classification loss:

$$L_b = -\sum_{k=1}^{n+m}\log[p_k(q_k)], \tag{7}$$

with $p_k(.)$ indicating the output probability, i.e. softmax of the logits.

Next to the bias-correction, BiC uses data augmentation, a replay mechanism, and a distillation mechanism to continually train the model. The data-augmentation comprises random cropping with scales ranging 0.2 to 1.0 and random horizontal flipping with chance of $p = 0.5$. These augmentations are also applied to buffered samples during replay. The general replay mechanism is discussed in Section 4 of the main body of this paper. Here, we simplify it to allocating an auxiliary buffer $\hat{X}$ of size M that allows to interleave training with exemplars from previously observed n classes. A notable addition is that this buffer holds both, the replay exemplars for training and for validation, with the latter already including samples from the current task. Wu et al. (2019) found a allocation ratio of $9:1$ for training/validation to be sufficient. The replay interleaved cross-entropy training loss is formulated as:

$$L_c = \sum_{(x,y)\in \hat{X}^n \cup X_t^m} \sum_{k=1}^{n+m} -\delta_{y=k} \log(p_k(x)). \tag{8}$$

The additional regularizing distillation loss is formulated as:

$$L_d = \sum_{x\in \hat{X}^n \cup X^m} \sum_{k=1}^{n} -\hat{\pi}_k(x) \log[\pi_k(x)], \tag{9}$$

$$\hat{\pi}_k = \frac{e^{\hat{o}_k^n(x)/T}}{\sum_{j=1}^{n} e^{\hat{o}_j^n(x)/T}}, \quad \pi_k(x) = \frac{e^{o_k^{n+m}(x)/T}}{\sum_{j=1}^{n} e^{o_j^{n+1}(x)/T}},$$

with $\hat{o}^n$ denoting the logits from the previous, old, model and T the temperature scalar. Note that the previous bias-correction is applied in $\hat{o}^n$. Ultimately, both losses are combined to the total training loss:

$$L = \lambda L_d + (1-\lambda) L_c, \tag{10}$$

with balancing scalar $\lambda = \frac{n}{n+m}$, with n and m being the number of old and new classes respectively.

DER (**D**ark **E**xperience **R**eplay) is an experience replay approach that extents the standard replay formulation (*c.f.* Eq. 8) by a regularization term based on distillation (Hinton et al., 2014):

$$L_{\text{DER}} = L_c + \alpha \mathbb{E}_{(x,o_k)\sim \hat{X}} [\mathrm{D}_{\mathrm{KL}}(p_k(o_k) || f(x))], \tag{11}$$

with loss discounting hyperparameter α. The auxiliary replay buffer is defined as:

$$\hat{X} = \{(x_i, o_i), 0 \leq i < M\},$$

containing M pairs (x_i, o_i) of previous tasks exemplars x_i along with the models output logits o_i (at the time of adding them to the buffer), instead of targets y_i.
Further, to avoid information loss in the softmax-function when comparing model output $f(x)$ to o_i, the authors chose to approximate the KL divergence (D_{KL} by the Euclidean distance. With that, the final loss becomes:

$$L_{\text{DER}} = L_c + \alpha \mathbb{E}_{(x,z)\sim \hat{X}} [\, ||z - h(x)||_2^2 \,]. \tag{12}$$

As for BiC, data augmentation by random cropping with scales ranging 0.2 to 1.0 and random horizontal flipping with chance of $p = 0.5$ are applied to all forwarded data.

C IMPLEMENTATION DETAILS

The practical implementation of combining experience replay (ER) and the gradient projection mechanism of GEM or A-GEM includes multiple aspects that benefit from additional clarification. In this section we discuss the calculation of the reference gradients and the handling of batch normalization. To accompany this discussion, we provide detailed pseudocodes for our implementations of ER + A-GEM (Algorithm 2) and ER + GEM (Algorithm 3), complementing the higher-level pseudocode for ER + A-GEM in the main text.

Algorithm 2 ER + AGEM (detailed)

```
Require: parameters $w$, loss function $\ell$, learning rate $\lambda$,
         data stream $\{D_1, ..., D_T\}$
 1: $M \leftarrow \{\}$
 2: for $t = 1, ..., T$ do
 3:     for $(x, y) \in D_t$ do
 4:         #1. Sample from memory buffer
 5:         $(\tilde{x}_k, \tilde{y}_k) \leftarrow \texttt{SAMPLE}(M)$
 6:
 7:         #2. Compute gradients of (approx.) joint loss
 8:         $[z_x, z_{\tilde{x}}] \leftarrow f_w([x, \tilde{x}])$
 9:         $g \leftarrow \nabla_w \ell(z_x, y)$
10:         $g_{\text{old}} \leftarrow \nabla_w \ell(z_{\tilde{x}}, \tilde{y})$
11:         $g_{\text{joint}} \leftarrow \frac{1}{t} g + (1 - \frac{1}{t}) g_{\text{old}}$
12:
13:         #3. Compute reference gradients
14:         $g_{\text{ref}} \leftarrow g_{\text{old}}$
15:
16:         #4. Gradient projection
17:         $\bar{g} \leftarrow \texttt{PROJECT\_AGEM}(g_{\text{joint}}, g_{\text{ref}})$
18:
19:         #5. Update model parameters
20:         $w \leftarrow \texttt{OPTIMIZER\_STEP}(w, \lambda, \bar{g})$
21:     end for
22:     $M \leftarrow \texttt{UPDATE\_BUFFER}(M, D_t)$
23: end for
```

Calculation of reference gradients: In the GEM and A-GEM mechanism, reference gradients inform the gradient projection by indicating the gradient direction that decreases the loss on previously learned tasks. When combining ER with A-GEM, we take the reference gradients to be the same as the gradients that are used in optimizing the approximate joint loss (i.e., $g_{\text{ref}} = g_{\text{old}}$). Computing the reference gradients on a separately sampled mini-batch from the memory buffer may have beneficial effects for training, but would come at an increased computational cost of effectively doubling the replay mini-batch size. When combining ER with GEM, obtaining the reference gradients is more complex because a separate reference gradient is required for each previously learned tasks. In the original formulation of GEM by Lopez-Paz and Ranzato (2017), the reference gradients are calculated with respect to the entire memory buffer. This quickly becomes computationally very costly if large amounts of data are stored. As a mitigation, rather than using each task's entire buffer to compute the reference gradient, we sample one mini-batch

Algorithm 3 ER + GEM (detailed)

```
Require: parameters $w$, loss function $\ell$, learning rate $\lambda$,
         hyperparameter $\gamma$, data stream $\{D_1, ..., D_T\}$
 1: $M_t \leftarrow \{\}, \forall\, t = 1, ..., T$
 2: for $t = 1, ..., T$ do
 3:     for $(x, y) \in D_t$ do
 4:         #1. Sample from memory buffer
 5:         $(\tilde{x}_k, \tilde{y}_k) \leftarrow$ SAMPLE$(M_k)$ for all $k < t$
 6:         $\overline{M} \leftarrow \{(\tilde{x}_k, \tilde{y}_k)\}$ for all $k < t$
 7:         $(\tilde{x}_{k<t}, \tilde{y}_{k<t}) \leftarrow$ SAMPLE$(\overline{M})$
 8:
 9:         #2. Compute gradients of (approx.) joint loss
10:         $[z_x, z_{\tilde{x}_{k<t}}] \leftarrow f_w([x, \tilde{x}_{k<t}])$
11:         $g \leftarrow \nabla_w \ell(z_x, y)$
12:         $g_{\text{old}} \leftarrow \nabla_w \ell(z_{\tilde{x}_{k<t}}, \tilde{y}_{k<t})$
13:         $g_{\text{joint}} \leftarrow \frac{1}{t} g + (1 - \frac{1}{t}) g_{\text{old}}$
14:
15:         #3. Compute reference gradients
16:         FREEZE_BATCH_NORM$(f_w)$
17:         $g_{\text{ref}_k} \leftarrow \nabla_w \ell(f_w(\tilde{x}_k), \tilde{y}_k)$ for all $k < t$
18:         UNFREEZE_BATCH_NORM$(f_w)$
19:
20:         #4. Gradient projection
21:         $\bar{g} \leftarrow$ PROJECT_GEM$(g_{\text{joint}}, [g_{\text{ref}_1}, ..., g_{\text{ref}_k}], \gamma)$
22:
23:         #5. Update model parameters
24:         $w \leftarrow$ OPTIMIZER_STEP$(w, \lambda, \bar{g})$
25:     end for
26:     $M_t \leftarrow$ FILL_BUFFER$(D_t)$
27: end for
```

per previously observed task. In particular, when training on task t, we sample $k = t-1$ mini-batches, one from each task-specific memory buffer M_k, see Algorithm 3 line 5. All sampled mini-batches are of the same size as the mini-batch currently observed by the model. In order to closely approximate the 'same-mini-batch' relation between the replay gradient and the reference gradients, we then obtain the replay mini-batch by uniformly sampling from the k mini-batches of the reference gradients, see Algorithm 3 lines 6-7.

Batch norm: When implementing ER + A-GEM or ER + GEM, another aspect requiring careful consideration is the use of batch normalization (Ioffe and Szegedy, 2015), which is included in the reduced ResNet-18 architecture that we use for all benchmarks except Rotated MNIST. When training with batch norm, the normalization statistics are computed relative to each individual mini-batch that is forwarded through the model. This means that when the current data and the replay data are forwarded through the model in different mini-batches, they use different normalization statistics, which might induce instability in the training. For standard replay, as well as for ER + A-GEM, we can mitigate this potential instability by forwarding the current and replayed data together, see Algorithm 2 line 8. However, when using replay in combination with GEM, it becomes more complex, because more data from the memory buffer is forwarded through the model than required for approximating the joint loss. Forwarding all data together seems undesirable as it would bias the normalization statistics too much toward the data from previous tasks (and doing so might also be impractical as the mini-batch size might become too large for forwarding all data together). Instead, common implementations of GEM typically forward the data of each past task separately, which means that each reference gradient is computed with a custom, task-specific normalization. Such a different normalization for each reference gradient might induce instability in the training. To try to mitigate this, when computing the reference gradients for GEM, we freeze the batch-norm layers, see Algorithm 3 line 16-18.

D ADDITIONAL STUDY OF THE OPTIMIZATION ROUTINE OF GEM

Here, we take a detailed look at the optimization routine of GEM; in particular, at its gradient projection step (i.e., line 21 in Algorithm 3). While performing the experiments for this paper, we realized that the optimization routine of GEM is not unambiguously defined and that it has an influential hyperparameter. The effect of this hyperparameter, which is not present in the optimization routine of A-GEM, can explain for a large part the difference in performances that we observed between using the optimization routine of GEM versus that of A-GEM (see section 6 in the main text).

The motivation behind the optimization mechanism of GEM is to allow only such gradient updates that do not increase the loss on any previous task. Mathematically, Lopez-Paz and Ranzato (2017) formulated GEM's optimization mechanism as follows. When training on task t, the gradient $\bar{g}$ based upon which the optimization step is taken is given by the solution to:

$$\text{minimize}_{\bar{g}} \quad \frac{1}{2}||g - \bar{g}||_2^2 \tag{13}$$

$$\text{subject to} \quad \langle \bar{g}, g_k \rangle \geq 0, \forall k < t, \tag{14}$$

where g is the gradient of the loss being optimized and g_k is the reference gradient computed on stored data for the k^{th} task. Equations (13) and (14) define a quadratic program (QP) in p variables, with p the number of trainable parameters of the neural network. To solve this QP, GEM uses the dual problem, which is given by:

$$\text{minimize}_v \quad \frac{1}{2} v^{\mathrm{T}} G G^{\mathrm{T}} + g^{\mathrm{T}} G^{\mathrm{T}} v \tag{15}$$

$$\text{subject to} \quad v \geq 0, \tag{16}$$

with $G = (g_1, ..., g_{t-1})$.[2] This dual problem is a QP in only $t-1$ variables, and can therefore be solved more efficiently. After solving this dual problem, $\bar{g}$ can be recovered as:

$$\bar{g} = G^{\mathrm{T}} v^* + g, \tag{17}$$

where v^* is the solution to the dual problem.

This is however not the full story. Lopez-Paz and Ranzato (2017) further introduced a hyperparameter γ, because in practice they found that *'adding a small constant $\gamma \geq 0$ to v^* biased the gradient projection to updates that favored beneficial backward transfer'* (p. 4). Based on this description, the reader might expect Equation (17) to change to $\tilde{g} = G^{\mathrm{T}}(v^* + \gamma) + g$, but in the official code implementation of GEM,[3] γ is instead added to the right-hand side of the inequality constraint of the dual problem (i.e., the inequality in Equation (16) changes to $v \geq \gamma$).

Furthermore, setting $\gamma > 0$ introduces another subtlety, because it makes that the solution $\bar{g}$ to the dual problem is always different from g, even if the constraint $\langle g, g_k \rangle \geq 0$ is satisfied for each past task k. Nevertheless, in the official code implementation of GEM, $\bar{g}$ is still set to g if $\langle g, g_k \rangle \geq 0$ for all $k < t$. In other words, the hyperparameter γ is used only if $\langle g, g_k \rangle < 0$ for at least one past task k. This thus introduces a discontinuity (see Figure D.1 for an empirical evaluation of the effect of this).

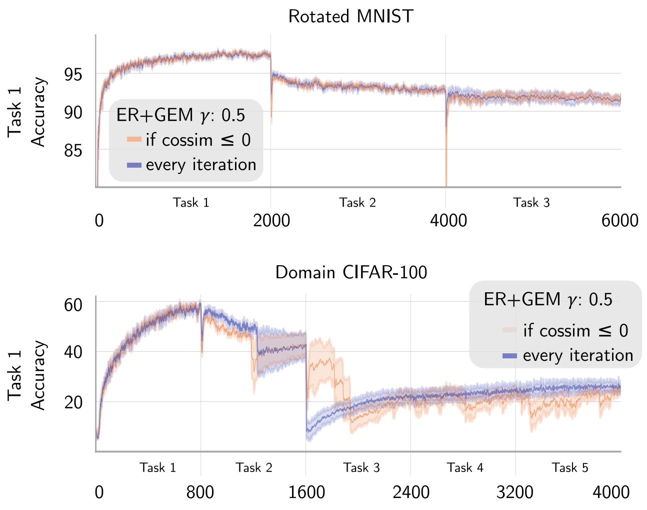


Figure D.1: **When to perform the GEM projection.** Illustrated is the difference for ER + GEM between performing the GEM projection operation at every iteration *versus* only when the cosine-similarity of the current gradient with at least one reference gradient is ≤ 0.

[2]In the published version of Lopez-Paz and Ranzato (2017), G is erroneously defined as $-(g_1, ..., g_{t-1})$. In September 2022, the authors corrected this in the ArXiv version.

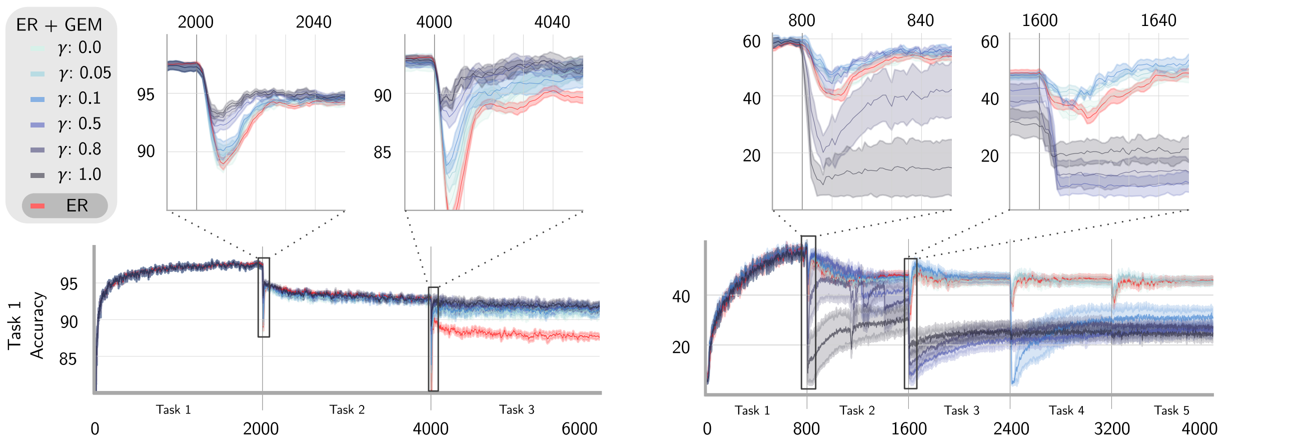
[3]`https://github.com/facebookresearch/GradientEpisodicMemory`

The way that hyperparameter γ is treated in GEM's official code implementation is typically taken over by other publicly available implementations of GEM. For the pre-registered experiments reported in the main text, we followed the official code implementation of GEM as well, and we used $\gamma = 0.5$, as this is the value that Lopez-Paz and Ranzato (2017) used for all their main experiments.

In this Appendix we report additional experiments that explore the impact of hyperparameter γ. First, we note that the effect of γ can be interpreted as enlarging the influence of the reference gradients g_k on $\bar{g}$. The reason for this is that γ tends to increase v^*, and $\bar{g}$ is related to v^* through $\bar{g} = G^{\mathrm{T}} v^* + g$. Figure D.2 empirically evaluates the impact of varying γ on the performance of ER + GEM on the offline versions of Rotated MNIST and Domain CIFAR-100. For these experiments, the discontinuity regarding hyperparameter γ is removed (i.e., the dual problem is solved at every iteration, we do not first check whether $\langle g, g_k \rangle < 0$ for at least one $k < t$). For Rotated MNIST, we find that increasing γ leads to both a reduction in the stability gap and an increase in final performance. For Domain CIFAR-100, we find that the lower γ, the later the collapse appears, and with $\gamma \leq 0.05$ we no longer observe a collapse.

E ADDITIONAL RESULTS

In this section we provide the remaining results we obtained from our experimental protocol but did not include in the main text to avoid clutter. An extensive overview of all data can be found in the Table E.1 for Rotated MNIST, Table E.2 for Domain CIFAR-100, Table E.3 for Split CIFAR-100, and Table E.4 for Mini-ImageNet. Also, we accompany the tabular view by additional plots for qualitative assessment (Figures E.1 and E.2).



	ER + GEM						ER
γ	0.0	0.05	0.1	0.5	0.8	1.0	–
MIN	84.4 ± 0.7	85.6 ± 0.5	86.8 ± 0.5	89.1 ± 0.2	89.5 ± 0.4	**90.0 ± 0.2**	83.1 ± 0.5
AVG	93.6 ± 0.1	93.7 ± 0.2	93.8 ± 0.2	93.9 ± 0.1	94.1 ± 0.1	**94.1 ± 0.2**	91.9 ± 0.1

	ER + GEM						ER
γ	0.0	0.05	0.1	0.5	0.8	1.0	–
MIN	**33.0 ± 0.6**	**33.6 ± 1.5**	8.4 ± 1.5	9.0 ± 1.0	10.0 ± 1.1	10.7 ± 1.8	**33.2 ± 1.3**
AVG	**48.1 ± 0.7**	47.2 ± 0.7	29.0 ± 4.9	21.3 ± 1.2	22.9 ± 0.9	21.3 ± 2.3	**48.6 ± 0.5**

Figure D.2: **Influence of GEM's hyperparameter γ on Rotated MNIST (left) and Domain CIFAR-100 (right).** The middle panels show the test accuracy on the first task while the model is incrementally trained on all tasks. The top panels show zoomed in views of the first 50 iterations after a task switch, allowing a more detailed qualitative comparison of the stability gaps. The bottom panels display tables that quantitatively compare average minimum accuracy (MIN) and final average accuracy (AVG).

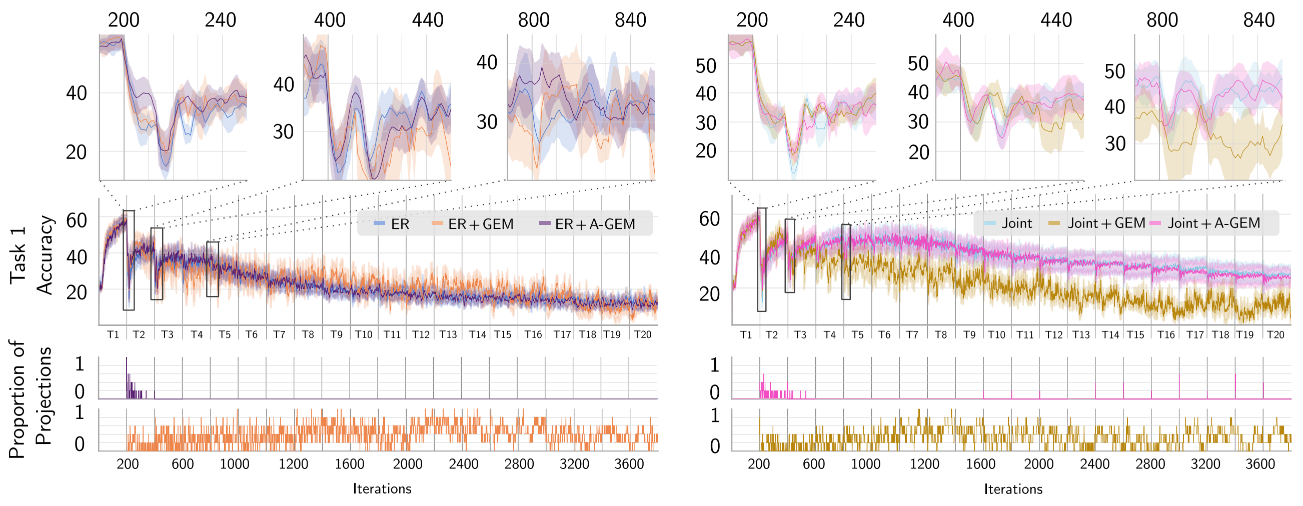


Figure E.1: **Stability gaps for first task of offline Split Mini-ImageNet.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

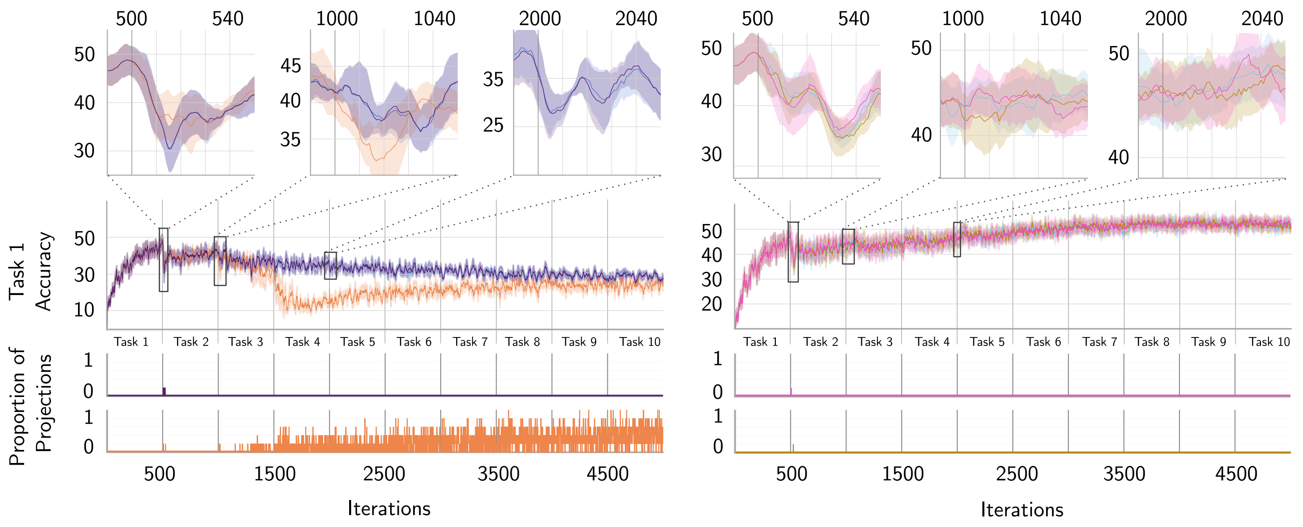


Figure E.2: **Stability gaps for the first task of online Split CIFAR-100.** The left side shows standard ER, the right side incremental joint training (or 'full replay') – both by themselves and in combination with the optimization mechanism of GEM and A-GEM. The middle panels show the test accuracy on the first task while the model is incrementally trained for all tasks of the benchmark. The top panels show zoomed-in views of the first 50 training iterations after a task switch, allowing a more detailed qualitative comparison of the stability gap. These plots show the mean ± standard error (shaded area) over five runs with different random seeds. The bottom panel shows for every iteration the proportion of runs where the gradient was projected, with 0 indicating that at this iteration there was no run in which a gradient was projected and 1 indicating that there was a gradient projection in every run.

Table E.1: **Final average accuracy (AVG) and average minimum accuracy (MIN) for all hyperparameter settings of online and offline Rotated MNIST.** Runs marked with gray background are used for comparisons in the main body. Results reported as mean $\pm$ standard error over 5 runs with different random seeds.

Offline

LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	Joint	Joint + GEM	Joint + AGEM
0.1	MIN	20.9 ± 0.8	43.6 ± 3.7	33.0 ± 1.3	83.1 ± 0.5	84.1 ± 0.4	82.5 ± 0.7	42.3 ± 3.9	30.3 ± 8.1	86.7 ± 0.8	87.5 ± 0.9	86.6 ± 0.6
	AVG	52.8 ± 0.6	91.8 ± 0.2	65.2 ± 0.8	91.9 ± 0.1	93.7 ± 0.1	91.8 ± 0.2	54.8 ± 4.5	44.8 ± 9.9	97.5 ± 0.0	97.8 ± 0.0	97.5 ± 0.0
0.01	MIN	30.2 ± 0.4	55.9 ± 3.0	39.5 ± 1.2	82.8 ± 0.6	84.4 ± 0.2	82.8 ± 0.7	82.6 ± 0.5	83.0 ± 0.4	86.2 ± 0.5	87.5 ± 0.5	86.9 ± 0.5
	AVG	56.2 ± 0.2	92.4 ± 0.1	71.1 ± 0.5	91.7 ± 0.1	93.4 ± 0.1	91.8 ± 0.1	87.3 ± 0.2	87.1 ± 0.2	96.6 ± 0.0	97.0 ± 0.0	96.5 ± 0.0
0.001	MIN	31.2 ± 0.5	84.6 ± 0.4	47.7 ± 0.7	84.9 ± 0.3	86.5 ± 0.4	84.9 ± 0.3	60.2 ± 1.1	60.2 ± 1.1	88.0 ± 0.3	88.1 ± 0.3	88.0 ± 0.3
	AVG	53.2 ± 0.3	90.8 ± 0.1	65.4 ± 0.4	87.6 ± 0.2	89.4 ± 0.1	87.6 ± 0.2	68.6 ± 0.4	68.6 ± 0.4	92.0 ± 0.1	92.1 ± 0.0	92.0 ± 0.0

Online

BS	LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	Joint	Joint + GEM	Joint + AGEM
10	0.1	MIN	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	7.6 ± 0.4	8.0 ± 0.2	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0
		AVG	10.1 ± 0.4	10.1 ± 0.4	10.1 ± 0.4	10.1 ± 0.4	10.1 ± 0.4	10.1 ± 0.4	10.3 ± 0.3	10.3 ± 0.3	10.1 ± 0.4	10.1 ± 0.4	10.1 ± 0.4
	0.01	MIN	23.7 ± 0.7	79.5 ± 2.6	36.3 ± 1.2	86.8 ± 0.3	89.1 ± 0.4	87.1 ± 0.4	43.8 ± 4.5	46.7 ± 3.5	92.3 ± 0.4	92.8 ± 0.3	92.4 ± 0.1
		AVG	53.8 ± 0.6	92.9 ± 0.1	64.6 ± 1.1	92.7 ± 0.1	94.2 ± 0.1	92.8 ± 0.2	57.9 ± 6.1	60.9 ± 2.7	96.8 ± 0.0	96.8 ± 0.1	96.8 ± 0.1
	0.001	MIN	31.4 ± 0.3	82.5 ± 0.5	47.2 ± 0.7	86.3 ± 0.3	88.0 ± 0.2	86.3 ± 0.3	69.8 ± 0.7	69.1 ± 0.9	90.9 ± 0.4	91.5 ± 0.3	91.0 ± 0.3
		AVG	55.8 ± 0.3	92.4 ± 0.1	69.3 ± 0.2	90.8 ± 0.3	92.3 ± 0.1	90.8 ± 0.3	78.3 ± 0.3	78.3 ± 0.3	95.1 ± 0.1	95.4 ± 0.0	95.1 ± 0.1
64	0.1	MIN	16.1 ± 1.2	54.3 ± 2.6	23.4 ± 1.1	84.2 ± 0.7	84.2 ± 1.0	84.3 ± 0.6	7.3 ± 0.3	7.9 ± 0.3	87.5 ± 0.7	88.4 ± 0.8	86.1 ± 0.5
		AVG	48.0 ± 0.6	86.9 ± 0.8	56.3 ± 0.7	91.2 ± 0.2	93.0 ± 0.3	91.0 ± 0.1	9.9 ± 0.1	9.9 ± 0.1	96.2 ± 0.1	96.2 ± 0.1	96.1 ± 0.0
	0.01	MIN	28.9 ± 0.3	70.0 ± 2.0	40.6 ± 0.8	83.9 ± 0.4	84.6 ± 0.5	83.9 ± 0.3	72.5 ± 0.4	72.1 ± 0.5	87.6 ± 0.6	88.8 ± 0.6	87.3 ± 0.5
		AVG	54.3 ± 0.3	91.9 ± 0.2	66.5 ± 0.7	91.2 ± 0.2	92.7 ± 0.1	91.3 ± 0.1	80.5 ± 0.3	80.4 ± 0.3	95.6 ± 0.1	95.8 ± 0.1	95.5 ± 0.1
	0.001	MIN	30.0 ± 0.3	79.0 ± 0.3	39.6 ± 0.6	81.6 ± 0.3	82.4 ± 0.3	81.6 ± 0.3	42.6 ± 0.6	42.8 ± 0.6	83.9 ± 0.3	83.9 ± 0.4	84.1 ± 0.3
		AVG	51.4 ± 0.2	87.9 ± 0.2	58.6 ± 0.5	83.7 ± 0.3	84.9 ± 0.2	83.8 ± 0.3	52.1 ± 0.4	52.2 ± 0.4	86.5 ± 0.1	86.6 ± 0.1	86.5 ± 0.1
128	0.1	MIN	20.4 ± 0.7	39.0 ± 3.9	27.4 ± 0.8	83.2 ± 0.6	83.6 ± 0.6	83.5 ± 0.6	59.8 ± 1.2	60.2 ± 0.6	87.1 ± 0.5	88.8 ± 0.4	87.2 ± 0.8
		AVG	51.6 ± 0.3	91.4 ± 0.1	57.9 ± 0.5	91.6 ± 0.2	93.4 ± 0.1	91.5 ± 0.1	71.3 ± 0.4	72.2 ± 0.5	96.5 ± 0.0	96.8 ± 0.1	96.5 ± 0.1
	0.01	MIN	28.7 ± 0.5	61.8 ± 2.3	38.2 ± 0.9	83.4 ± 0.6	83.5 ± 0.5	83.3 ± 0.6	66.7 ± 0.6	66.7 ± 0.6	86.9 ± 0.4	87.2 ± 0.5	87.0 ± 0.4
		AVG	53.7 ± 0.4	90.7 ± 0.3	62.8 ± 0.4	90.0 ± 0.1	91.6 ± 0.1	90.0 ± 0.1	76.1 ± 0.3	75.9 ± 0.2	94.4 ± 0.1	94.7 ± 0.0	94.4 ± 0.0
	0.001	MIN	33.7 ± 0.4	76.1 ± 0.4	41.3 ± 0.6	77.6 ± 0.2	78.1 ± 0.2	77.6 ± 0.2	29.6 ± 0.7	30.0 ± 0.7	79.1 ± 0.2	79.2 ± 0.2	79.1 ± 0.2
		AVG	52.8 ± 0.2	83.5 ± 0.2	58.0 ± 0.2	77.1 ± 0.1	77.9 ± 0.1	77.1 ± 0.1	37.9 ± 0.4	38.1 ± 0.4	78.8 ± 0.1	78.8 ± 0.1	78.7 ± 0.1

Table E.2: **Final average accuracy (AVG) and average minimum accuracy (MIN) for all hyperparameter settings of online and offline Domain CIFAR-100**. Runs marked with gray background are used for comparisons in the main body. Results reported as mean $\pm$ standard error over 5 runs with different random seeds.

Offline

LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	Joint	Joint + GEM	Joint + AGEM
0.1	MIN	12.2 ± 1.1	3.5 ± 0.3	12.2 ± 0.5	33.2 ± 1.3	7.2 ± 2.6	34.0 ± 0.9	46.0 ± 1.0	46.3 ± 1.0	42.9 ± 0.7	42.2 ± 1.1	42.7 ± 0.8
	AVG	35.9 ± 0.7	17.6 ± 3.1	36.1 ± 0.9	48.6 ± 0.5	23.9 ± 1.8	48.6 ± 0.5	59.6 ± 1.0	60.3 ± 0.6	52.4 ± 0.8	52.6 ± 0.6	52.0 ± 0.7
0.01	MIN	11.6 ± 0.8	5.1 ± 0.5	12.9 ± 1.2	32.6 ± 0.6	3.8 ± 0.4	30.8 ± 0.6	44.4 ± 0.7	44.8 ± 1.0	41.0 ± 0.8	19.8 ± 6.9	40.2 ± 0.6
	AVG	36.9 ± 0.6	18.0 ± 1.2	37.2 ± 0.5	46.2 ± 0.3	16.4 ± 1.5	46.2 ± 0.2	59.7 ± 0.5	60.7 ± 0.4	50.2 ± 0.2	42.2 ± 6.9	50.1 ± 0.3
0.001	MIN	16.9 ± 0.6	4.4 ± 0.5	16.1 ± 0.4	29.0 ± 0.4	3.5 ± 0.4	29.4 ± 0.4	34.4 ± 0.4	34.4 ± 0.3	36.8 ± 0.2	36.5 ± 0.4	36.6 ± 0.4
	AVG	32.2 ± 0.4	20.4 ± 1.2	32.5 ± 0.3	35.0 ± 0.2	13.7 ± 2.5	35.6 ± 0.3	46.2 ± 0.3	46.0 ± 0.3	39.6 ± 0.1	39.4 ± 0.2	39.3 ± 0.4

Online

BS	LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	Joint	Joint + GEM	Joint + AGEM
10	0.1	MIN	11.9 ± 0.6	6.6 ± 1.3	12.1 ± 0.4	25.6 ± 0.9	5.6 ± 1.6	25.6 ± 0.9	15.4 ± 0.7	14.8 ± 0.6	29.1 ± 0.7	30.7 ± 1.0	30.6 ± 1.0
		AVG	23.8 ± 0.8	20.9 ± 1.8	25.1 ± 0.6	35.1 ± 0.8	20.7 ± 4.7	35.1 ± 0.8	26.0 ± 0.8	25.5 ± 0.8	43.2 ± 0.5	45.3 ± 0.9	45.0 ± 1.0
	0.01	MIN	14.6 ± 0.4	14.0 ± 0.6	15.1 ± 0.5	29.2 ± 0.7	4.3 ± 0.1	29.2 ± 0.7	20.3 ± 0.3	20.1 ± 0.4	35.5 ± 1.1	35.6 ± 1.3	35.1 ± 0.9
		AVG	27.6 ± 0.6	21.1 ± 0.8	28.0 ± 0.7	38.3 ± 0.8	19.8 ± 1.4	38.3 ± 0.8	30.7 ± 0.3	30.6 ± 0.7	49.8 ± 1.0	49.4 ± 1.5	49.7 ± 1.2
	0.001	MIN	16.3 ± 0.6	15.3 ± 0.8	17.1 ± 0.6	30.7 ± 0.7	4.0 ± 0.1	30.7 ± 0.7	23.7 ± 0.2	23.8 ± 0.3	38.1 ± 0.8	37.9 ± 0.8	37.9 ± 0.7
		AVG	28.9 ± 0.6	23.8 ± 1.2	29.9 ± 0.4	38.0 ± 0.5	26.7 ± 3.3	38.0 ± 0.5	34.0 ± 0.5	33.5 ± 0.5	48.3 ± 0.4	48.1 ± 0.6	49.0 ± 0.4
64	0.1	MIN	13.3 ± 0.4	3.9 ± 0.2	13.7 ± 0.5	25.7 ± 0.6	21.2 ± 4.3	25.7 ± 0.6	17.3 ± 0.5	17.3 ± 0.5	31.0 ± 0.8	31.6 ± 1.1	30.9 ± 1.2
		AVG	24.2 ± 0.4	12.1 ± 2.0	24.9 ± 0.4	34.7 ± 1.2	29.3 ± 4.5	34.7 ± 1.2	26.3 ± 0.7	26.4 ± 0.5	45.4 ± 1.1	44.2 ± 1.3	44.9 ± 1.2
	0.01	MIN	14.6 ± 0.3	5.5 ± 0.4	15.4 ± 0.5	28.0 ± 0.3	27.7 ± 1.0	28.0 ± 0.3	21.5 ± 0.5	21.7 ± 0.4	37.1 ± 1.0	37.0 ± 0.8	37.0 ± 0.9
		AVG	27.2 ± 0.3	19.5 ± 1.9	28.2 ± 0.7	35.9 ± 0.4	35.1 ± 0.5	35.9 ± 0.4	31.5 ± 0.6	32.2 ± 0.8	48.9 ± 0.2	49.5 ± 0.6	48.9 ± 0.1
	0.001	MIN	16.1 ± 0.4	8.6 ± 0.9	16.2 ± 0.3	25.3 ± 0.3	6.1 ± 2.7	25.3 ± 0.3	19.6 ± 0.3	20.0 ± 0.2	27.8 ± 0.3	22.6 ± 3.1	27.8 ± 0.4
		AVG	24.4 ± 0.5	23.4 ± 2.5	24.8 ± 0.6	31.5 ± 0.2	13.9 ± 4.4	31.5 ± 0.2	25.9 ± 0.5	26.1 ± 0.6	34.4 ± 0.2	34.5 ± 0.2	34.5 ± 0.2
128	0.1	MIN	11.8 ± 0.5	3.5 ± 0.2	11.8 ± 0.7	24.7 ± 0.6	20.5 ± 4.2	24.7 ± 0.6	17.6 ± 0.4	17.6 ± 0.5	29.3 ± 1.0	29.8 ± 0.8	29.9 ± 0.8
		AVG	23.4 ± 1.0	12.0 ± 0.5	23.5 ± 0.7	34.0 ± 0.7	28.6 ± 4.8	34.0 ± 0.7	25.1 ± 0.5	25.7 ± 0.8	42.0 ± 0.8	43.4 ± 1.0	44.3 ± 1.0
	0.01	MIN	13.8 ± 0.6	4.1 ± 0.2	13.1 ± 0.8	27.2 ± 0.7	26.7 ± 0.5	27.2 ± 0.7	20.2 ± 0.3	20.3 ± 0.4	34.1 ± 0.5	34.1 ± 0.5	34.2 ± 0.5
		AVG	25.9 ± 0.6	12.5 ± 1.8	26.0 ± 0.9	34.7 ± 0.4	35.3 ± 0.8	34.7 ± 0.4	30.2 ± 0.4	30.1 ± 0.3	45.1 ± 0.5	45.0 ± 0.6	45.0 ± 0.7
	0.001	MIN	15.1 ± 0.7	6.6 ± 1.0	15.5 ± 0.5	22.5 ± 0.5	9.5 ± 3.3	22.5 ± 0.5	18.3 ± 0.5	18.4 ± 0.5	24.0 ± 0.5	22.5 ± 0.8	23.4 ± 0.5
		AVG	22.3 ± 0.4	10.3 ± 2.0	22.7 ± 0.5	26.9 ± 0.2	18.1 ± 3.7	26.9 ± 0.2	23.7 ± 0.2	23.7 ± 0.3	28.1 ± 0.2	26.6 ± 1.5	28.2 ± 0.2

Table E.3: **Final average accuracy (AVG) and average minimum accuracy (MIN) for all hyperparameter settings of online and offline Split-CIFAR100.** Runs marked with gray background are used for comparisons in the main body. Results reported as mean ± standard error over 5 runs with different random seeds.

Offline

LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	BiC*	BiC* + AGEM	Joint	Joint + GEM	Joint + AGEM
0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	12.4 ± 0.3	0.0 ± 0.0	12.1 ± 0.3	1.4 ± 0.1	1.6 ± 0.2	22.3 ± 0.9	23.1 ± 1.1	22.5 ± 0.2	2.8 ± 1.9	23.1 ± 0.4
	AVG	6.5 ± 0.4	1.5 ± 0.2	6.4 ± 0.3	22.8 ± 0.4	7.1 ± 1.9	22.4 ± 0.6	26.0 ± 0.9	25.6 ± 1.3	38.3 ± 1.0	38.4 ± 0.9	32.2 ± 0.5	16.5 ± 6.7	32.5 ± 0.4
0.01	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	10.8 ± 0.6	0.0 ± 0.0	10.9 ± 0.7	1.0 ± 0.3	1.0 ± 0.2	27.4 ± 0.4	27.8 ± 0.5	21.6 ± 0.7	3.8 ± 0.7	21.3 ± 0.8
	AVG	6.6 ± 0.4	3.4 ± 0.5	6.4 ± 0.4	18.5 ± 0.6	7.6 ± 1.6	19.6 ± 0.2	22.1 ± 1.0	22.6 ± 0.7	34.3 ± 0.5	34.4 ± 0.8	25.8 ± 0.4	22.6 ± 3.6	25.8 ± 0.4
0.001	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8.3 ± 0.5	2.0 ± 1.9	8.3 ± 0.4	0.1 ± 0.0	0.1 ± 0.0	10.5 ± 0.4	10.7 ± 0.6	16.7 ± 0.6	13.8 ± 3.5	16.5 ± 0.6
	AVG	6.6 ± 0.3	3.7 ± 0.2	6.7 ± 0.3	12.5 ± 0.3	10.2 ± 1.7	11.7 ± 0.3	9.1 ± 0.2	9.2 ± 0.2	16.7 ± 0.8	16.9 ± 0.7	20.6 ± 0.4	17.2 ± 3.9	20.3 ± 0.5

Online

BS	LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	BiC*	BiC* + AGEM	Joint	Joint + GEM	Joint + AGEM
10	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	9.0 ± 0.4	0.0 ± 0.0	9.1 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	7.5 ± 0.4	8.6 ± 0.2	22.1 ± 0.5	22.9 ± 0.6	22.8 ± 0.4
		AVG	5.0 ± 0.4	1.3 ± 0.2	5.0 ± 0.4	19.3 ± 0.4	6.9 ± 0.9	19.2 ± 0.3	5.5 ± 0.3	5.4 ± 0.3	15.9 ± 0.5	17.0 ± 0.6	38.4 ± 1.2	38.6 ± 0.6	38.7 ± 0.4
	0.01	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	10.6 ± 0.3	0.0 ± 0.0	10.7 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	11.1 ± 0.4	11.1 ± 0.4	27.0 ± 0.5	27.2 ± 0.5	27.2 ± 0.6
		AVG	5.1 ± 0.3	2.1 ± 0.1	5.0 ± 0.3	20.7 ± 0.3	8.5 ± 0.9	20.5 ± 0.4	5.8 ± 0.3	5.9 ± 0.2	18.4 ± 0.4	18.4 ± 0.4	41.2 ± 0.6	41.1 ± 0.4	41.4 ± 0.2
	0.001	MIN	0.0 ± 0.0	0.1 ± 0.1	0.0 ± 0.0	10.0 ± 0.2	0.0 ± 0.0	10.0 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	7.9 ± 0.3	7.9 ± 0.3	21.2 ± 0.2	10.5 ± 4.6	21.4 ± 0.3
		AVG	6.0 ± 0.3	3.0 ± 0.3	5.8 ± 0.3	18.8 ± 0.2	12.2 ± 1.3	18.8 ± 0.2	6.4 ± 0.4	6.4 ± 0.4	12.6 ± 0.4	12.6 ± 0.3	32.6 ± 0.3	26.5 ± 6.2	32.8 ± 0.3
64	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	5.7 ± 0.5	0.0 ± 0.0	5.6 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	4.5 ± 0.6	4.5 ± 0.6	12.2 ± 0.7	11.1 ± 0.9	12.1 ± 0.4
		AVG	2.2 ± 0.3	2.4 ± 0.8	2.4 ± 0.5	15.2 ± 0.5	5.5 ± 0.3	15.4 ± 0.4	3.8 ± 0.3	4.0 ± 0.3	15.2 ± 0.4	15.2 ± 0.4	29.1 ± 1.0	29.2 ± 0.9	28.8 ± 1.0
	0.01	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8.0 ± 0.4	3.3 ± 2.1	8.3 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	7.9 ± 0.3	7.9 ± 0.3	17.6 ± 0.4	18.5 ± 0.2	18.3 ± 0.4
		AVG	1.9 ± 0.2	5.3 ± 1.4	1.9 ± 0.2	17.3 ± 0.7	8.7 ± 4.0	17.0 ± 0.8	4.0 ± 0.3	3.6 ± 0.3	15.1 ± 0.4	15.1 ± 0.4	33.8 ± 0.2	33.7 ± 0.4	34.3 ± 0.3
	0.001	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	6.3 ± 0.2	0.0 ± 0.0	6.3 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	1.8 ± 0.3	1.8 ± 0.3	8.2 ± 0.3	1.3 ± 0.4	8.3 ± 0.3
		AVG	4.1 ± 0.2	6.9 ± 1.5	4.2 ± 0.3	13.0 ± 0.5	5.9 ± 1.4	13.0 ± 0.5	4.1 ± 0.3	4.1 ± 0.4	7.0 ± 0.4	7.0 ± 0.4	16.0 ± 0.4	8.4 ± 2.4	15.7 ± 0.4
128	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1.5 ± 0.2	0.0 ± 0.0	1.6 ± 0.3	0.3 ± 0.2	0.0 ± 0.0	2.8 ± 0.2	2.8 ± 0.2	3.6 ± 0.6	0.9 ± 0.8	4.5 ± 0.4
		AVG	1.1 ± 0.1	1.5 ± 0.2	1.7 ± 0.3	9.1 ± 0.6	4.1 ± 0.7	9.1 ± 0.2	1.4 ± 0.2	1.6 ± 0.4	12.0 ± 0.4	12.0 ± 0.4	14.4 ± 0.9	7.4 ± 2.1	15.0 ± 0.8
	0.01	MIN	0.3 ± 0.2	0.0 ± 0.0	0.4 ± 0.2	6.7 ± 0.5	1.4 ± 1.4	6.6 ± 0.4	0.0 ± 0.0	0.3 ± 0.2	5.0 ± 0.3	5.0 ± 0.3	13.2 ± 0.5	5.3 ± 3.3	13.4 ± 0.6
		AVG	1.3 ± 0.1	5.3 ± 0.5	1.6 ± 0.2	15.2 ± 0.2	6.6 ± 2.8	15.3 ± 0.4	2.1 ± 0.1	2.6 ± 0.2	12.1 ± 0.4	12.1 ± 0.4	25.7 ± 1.0	11.8 ± 5.7	25.4 ± 0.8
	0.001	MIN	1.0 ± 0.3	0.0 ± 0.0	1.0 ± 0.2	3.0 ± 0.2	0.0 ± 0.0	3.0 ± 0.2	1.1 ± 0.3	1.0 ± 0.2	0.9 ± 0.1	0.9 ± 0.1	3.2 ± 0.4	1.1 ± 0.2	3.3 ± 0.3
		AVG	3.4 ± 0.2	4.8 ± 1.4	3.4 ± 0.3	10.2 ± 0.3	3.6 ± 1.3	10.2 ± 0.3	3.2 ± 0.3	3.3 ± 0.3	4.9 ± 0.3	4.9 ± 0.3	11.6 ± 0.4	4.5 ± 0.3	11.6 ± 0.4

Table E.4: **Final average accuracy (AVG) and minimum average accuracy (MIN) for all hyperparameter settings of online and offline Split Mini-ImageNet.** Runs marked with gray background are used for comparisons in the main body. Results reported as mean ± standard error over 5 runs with different random seeds.

Offline

LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	BiC*	BiC* + AGEM	Joint	Joint + GEM	Joint + AGEM
0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8.6 ± 0.5	0.0 ± 0.0	8.3 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	7.8 ± 0.9	8.1 ± 0.8	16.5 ± 0.5	0.0 ± 0.0	16.4 ± 0.4
	AVG	3.2 ± 0.1	1.0 ± 0.0	3.2 ± 0.1	16.9 ± 0.9	4.7 ± 1.6	16.9 ± 0.5	4.3 ± 0.2	4.2 ± 0.8	20.3 ± 0.6	20.4 ± 1.2	28.1 ± 0.6	3.3 ± 0.4	28.5 ± 0.7
0.01	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4.1 ± 0.5	0.0 ± 0.0	4.0 ± 0.7	0.1 ± 0.0	0.2 ± 0.1	13.3 ± 0.3	13.5 ± 0.2	8.9 ± 1.1	0.0 ± 0.0	8.5 ± 0.7
	AVG	3.1 ± 0.1	1.9 ± 0.1	3.1 ± 0.1	13.5 ± 0.6	6.4 ± 1.9	14.1 ± 0.5	7.4 ± 0.5	7.6 ± 0.7	25.9 ± 0.7	26.2 ± 0.5	19.1 ± 1.2	4.6 ± 0.5	18.5 ± 0.9
0.001	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4.0 ± 0.3	0.0 ± 0.0	3.8 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	6.8 ± 0.6	6.7 ± 0.4	9.6 ± 0.8	0.0 ± 0.0	9.6 ± 0.7
	AVG	3.2 ± 0.1	3.6 ± 0.6	3.4 ± 0.1	10.1 ± 0.4	5.2 ± 0.8	9.9 ± 0.2	4.2 ± 0.2	4.5 ± 0.2	14.4 ± 0.4	14.2 ± 0.3	16.7 ± 0.7	6.4 ± 1.1	16.9 ± 0.7

Online

BS	LR	Final ACC	Finetune	GEM	AGEM	ER	ER + GEM	ER + AGEM	DER*	DER* + AGEM	BiC*	BiC* + AGEM	Joint	Joint + GEM	Joint + AGEM
10	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.1	0.0 ± 0.0	0.3 ± 0.2	0.8 ± 0.2	1.1 ± 0.0	0.6 ± 0.2	0.1 ± 0.1	0.2 ± 0.1	0.1 ± 0.0	1.0 ± 0.3
		AVG	1.0 ± 0.0	1.6 ± 0.5	1.1 ± 0.1	6.6 ± 0.5	1.8 ± 0.5	6.4 ± 0.6	1.0 ± 0.0	1.0 ± 0.0	8.3 ± 0.8	3.7 ± 0.3	3.9 ± 0.4	3.4 ± 0.4	8.0 ± 1.1
	0.01	MIN	0.1 ± 0.1	0.0 ± 0.0	0.3 ± 0.2	1.0 ± 0.2	0.0 ± 0.0	0.9 ± 0.3	0.4 ± 0.2	0.2 ± 0.2	1.6 ± 0.2	0.3 ± 0.1	0.5 ± 0.2	0.0 ± 0.0	2.0 ± 0.2
		AVG	1.1 ± 0.1	3.0 ± 0.5	1.3 ± 0.1	10.8 ± 0.4	5.5 ± 0.6	10.9 ± 0.7	1.5 ± 0.2	1.7 ± 0.3	16.2 ± 0.6	7.8 ± 0.5	8.0 ± 0.6	4.8 ± 0.6	16.7 ± 0.9
	0.001	MIN	0.2 ± 0.1	0.0 ± 0.0	0.2 ± 0.1	1.4 ± 0.3	0.0 ± 0.0	1.5 ± 0.3	0.4 ± 0.1	0.6 ± 0.2	1.3 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1.5 ± 0.3
		AVG	1.2 ± 0.2	4.0 ± 0.8	1.2 ± 0.1	11.0 ± 0.3	5.2 ± 0.9	11.1 ± 0.4	1.8 ± 0.2	2.5 ± 0.2	12.3 ± 0.3	4.5 ± 0.3	4.5 ± 0.3	5.4 ± 0.9	12.3 ± 0.3
64	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.7 ± 0.2	0.0 ± 0.0	1.0 ± 0.2	0.4 ± 0.3	0.4 ± 0.3	0.2 ± 0.1	0.2 ± 0.1	3.1 ± 0.4	0.0 ± 0.0	3.1 ± 0.6
		AVG	1.0 ± 0.0	1.0 ± 0.1	1.3 ± 0.1	8.3 ± 0.3	1.7 ± 0.2	8.6 ± 0.6	1.1 ± 0.1	1.1 ± 0.1	5.0 ± 0.5	5.3 ± 0.4	16.9 ± 0.8	3.5 ± 0.5	16.8 ± 1.4
	0.01	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3.3 ± 0.2	0.0 ± 0.0	3.2 ± 0.2	0.1 ± 0.1	0.1 ± 0.0	2.0 ± 0.3	1.9 ± 0.4	9.1 ± 0.2	0.0 ± 0.0	8.9 ± 0.3
		AVG	1.2 ± 0.1	2.7 ± 0.9	1.4 ± 0.2	13.9 ± 0.5	5.8 ± 0.3	14.6 ± 0.3	1.6 ± 0.3	1.8 ± 0.2	11.1 ± 0.5	11.0 ± 0.7	27.9 ± 0.8	5.5 ± 0.5	28.3 ± 1.3
	0.001	MIN	0.2 ± 0.1	0.0 ± 0.0	0.3 ± 0.3	3.7 ± 0.5	0.0 ± 0.0	3.8 ± 0.5	0.9 ± 0.1	0.9 ± 0.3	0.8 ± 0.1	0.8 ± 0.2	4.1 ± 1.5	0.0 ± 0.0	5.9 ± 0.5
		AVG	1.4 ± 0.1	8.3 ± 0.8	1.7 ± 0.2	13.6 ± 0.2	6.6 ± 0.9	13.7 ± 0.3	2.3 ± 0.2	1.8 ± 0.1	6.3 ± 0.3	6.4 ± 0.3	12.4 ± 3.6	3.6 ± 1.1	17.9 ± 0.5
128	0.1	MIN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.1	0.0 ± 0.0	0.3 ± 0.2	0.8 ± 0.2	1.1 ± 0.0	0.1 ± 0.1	0.2 ± 0.1	0.6 ± 0.2	0.1 ± 0.0	1.0 ± 0.3
		AVG	1.0 ± 0.0	1.6 ± 0.5	1.1 ± 0.1	6.6 ± 0.5	1.8 ± 0.5	6.4 ± 0.6	1.0 ± 0.0	1.0 ± 0.0	3.7 ± 0.3	3.9 ± 0.4	8.3 ± 0.8	3.4 ± 0.4	8.0 ± 1.1
	0.01	MIN	0.1 ± 0.1	0.0 ± 0.0	0.3 ± 0.2	1.0 ± 0.2	0.0 ± 0.0	0.9 ± 0.3	0.4 ± 0.2	0.2 ± 0.2	0.3 ± 0.1	0.5 ± 0.2	1.6 ± 0.2	0.0 ± 0.0	2.0 ± 0.2
		AVG	1.1 ± 0.1	3.0 ± 0.5	1.3 ± 0.1	10.8 ± 0.4	5.5 ± 0.6	10.9 ± 0.7	1.5 ± 0.2	1.7 ± 0.3	7.8 ± 0.5	8.0 ± 0.6	16.2 ± 0.6	4.8 ± 0.6	16.7 ± 0.9
	0.001	MIN	0.2 ± 0.1	0.0 ± 0.0	0.2 ± 0.1	1.4 ± 0.3	0.0 ± 0.0	1.5 ± 0.3	0.4 ± 0.1	0.6 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	1.3 ± 0.2	0.0 ± 0.0	1.5 ± 0.3
		AVG	1.2 ± 0.2	4.0 ± 0.8	1.2 ± 0.1	11.0 ± 0.3	5.2 ± 0.9	11.1 ± 0.4	1.8 ± 0.2	2.5 ± 0.2	4.5 ± 0.3	4.5 ± 0.3	12.3 ± 0.3	5.4 ± 0.9	12.3 ± 0.3