Gradient based Memory Editing for Task-Free Continual Learning

Xisen Jin [1] **Arka Sadhu** [1] **Junyi Du** [1] **Xiang Ren** [1]

Abstract

We explore task-free continual learning (CL), in which a model is trained to avoid catastrophic forgetting, but without being provided any explicit task boundaries or identities. Among many efforts on task-free CL, a notable family of approaches are memory-based, which look to select a subset of seen training examples to store in memory for future replay. However, since CL models are continually updated, the utility of stored seen examples may diminish over time. Here, we propose Gradient based Memory EDiting (GMED), a framework for editing stored examples in continuous input space via gradient updates, in order to create a wide range of more "challenging" examples for replay. GMED-edited examples remain similar to their unedited forms, but can yield increased loss in the upcoming model updates, thereby making the future replays more effective in overcoming catastrophic forgetting. By construction, GMED can be seamlessly applied in conjunction with other memory-based CL algorithms to bring further improvement. Experiments on six datasets validate that GMED is effective, and our single best method significantly outperforms existing approaches on three datasets. Code and data can be found at https://github.com/INK-USC/GMED.

1. Introduction

Learning from a continuous stream of data – referred to as *continual learning (CL)* or *lifelong learning* – has recently seen a surge in interest, and many works have proposed ways to mitigate CL models' catastrophic forgetting of previously learned knowledge (Robins, 1995; Lange et al., 2019; Parisi et al., 2019). Here, we study *task-free* CL (Aljundi et al., 2019b), where task identifiers and boundaries are absent from the data stream. This setting reflects many real-world data streams (Liu, 2020; Caccia et al., 2020) and offers a challenging testbed for online CL research.

[1]University of Southern California, Los Angeles, United States. Correspondence to: Xiang Ren <xiangren@usc.edu>.

Preprint. Under Review.

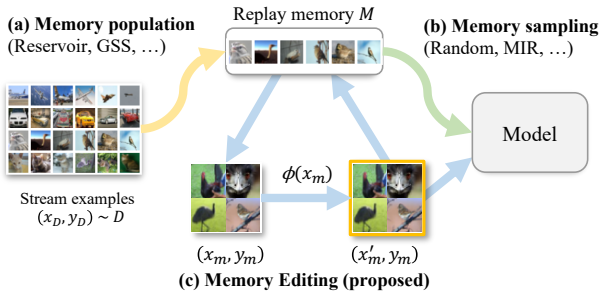


Figure 1: **Overview of key components in memory-based continual learning approaches.** Prior works study (a) memory population and (b) memory sampling strategies, improving how to store streaming examples into the memory or sampling them from memory for replay. In contrast, our work focuses on "editing" memory examples which can subsequently be stored, edited again, and be replayed.

Memory-based methods, a common approach in task-free continual learning, store a small number of training examples (from the data stream) in a memory and replay them at later training iterations (Robins, 1995; Rolnick et al., 2019). Existing research in memory-based CL methods has primarily focused on two key aspects: (a) *memory population*, *i.e.*, which examples from the data stream should be stored in the memory (Aljundi et al., 2019c; Chrysakis & Moens, 2020); and (b) *memory sampling*, *i.e.*, which examples from memory should be replayed (Aljundi et al., 2019a). In these prior works, all optimization processes (*e.g.*, selecting examples to store or to replay) operate on the original examples obtained from the data stream. However, for continually updating models, using stored-seen examples, in their original form, may lead to diminishing utility over time—*e.g.*, updated model may gradually overfit to the stored examples after runs of replay. As an important extension, in this work we look to "edit" (*i.e.*, via a small gradient update) stored examples such that they can yield increased loss in the upcoming model updates. These "edited" examples are stored (to replace their unedited counterparts), replayed, and further edited, thereby making the future replays more effective in overcoming catastrophic forgetting. We illustrate this key idea in Figure 1.

The main consideration in allowing "editing" is the choice of the optimization objective. In light of previous works (Aljundi et al., 2019a; Toneva et al., 2019; Chaudhry et al., 2020), we hypothesize that "forgotten" examples (*i.e.*, past examples that suffer from increased loss) should be prior-

itized for replay. Given this hypothesis, the challenge is to compute an estimate of "forgetting" of the examples in an online setup. Moreover, the edited example should remain close to the original example to keep it in-distribution. This presents an additional challenge since even a one-step update removes access to the original example without forgoing additional space.

To address the above issues, we propose a modular framework dubbed Gradient based Memory EDiting (**GMED**). For a particular stored example, GMED finds an edit (a small update over the example) such that the resulting edited example yields the most increase in loss (when replayed) while remaining close to the original example. As a result, replaying these edited examples instead of the original examples is more effective in overcoming catastrophic forgetting. To estimate "forgetting" in the online setup, we iteratively edit the drawn memory samples and accumulate the "forgetting" computed for each upcoming time-step. To enforce proximity to the original sample, we penalize the loss increase in the edited example so that it stays in-distribution. Since GMED focuses only on editing the stored examples, by construction, GMED is modular, *i.e.*, it can be seamlessly integrated with other state-of-the-art memory-based replay methods (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019a; Buzzega et al., 2020).

We demonstrate the effectiveness of GMED with a comprehensive set of experiments over six benchmark datasets. In most cases, we find that combining GMED with existing memory-based approaches results in consistent and statistically significant improvements. Moreover, our single best method sets a new state-of-art performance on three datasets. We further show that the objective used in GMED outperforms competing baselines.

To summarize, our contributions are two-fold: (i) we introduce GMED, a modular framework for task-free online continual learning, and test its performance under various datasets and parameter (*e.g.*, memory size) setups; and (ii) we show that the gains realized by GMED are significantly larger than those obtained from regularization effects in random perturbation, and can be accumulated upon data augmentation to further improve performance.

2. Related Works

Continual Learning (LifeLong Learning) studies the problem of learning from a data stream with changing data distributions over time (Lesort et al., 2020; Lange et al., 2019). A major bottleneck towards this goal is the phenomenon of catastrophic forgetting (Robins, 1995) where the model "forgets" knowledge learned from past examples when exposed to new ones. To mitigate this effect, a wide variety of approaches have been investigated such as adding regularization (Kirkpatrick et al., 2017; Zenke et al., 2017; Nguyen et al., 2018; Adel et al., 2020), separating parameters for previous and new data (Rusu et al., 2016; Serrà et al., 2018; Li et al., 2019), replaying examples from memory or a generative model (Robins, 1995; Shin et al., 2017; Lopez-Paz & Ranzato, 2017), meta-learning (Javed & White, 2019). In this work, we build on memory-based approaches which have been more successful in the online task-free continual learning setting that we study.

Online Task-free Continual Learning (Aljundi et al., 2019b) is a specific formulation of the continual learning where the task boundaries and identities are not available to the model. Due to its wider applicability to real-world data-streams, a number of algorithms have been adapted to the task-free setup (Zeno et al., 2018; Aljundi et al., 2019a; Harrison et al., 2020; He et al., 2019; Lee et al., 2020). In particular, memory-based CL algorithms which store a subset of examples and later replay them during training, have seen elevated success in the task-free setting. Improvements in this space have focused on: *storing diverse examples* as in Gradient-based Sample Selection (GSS) (Aljundi et al., 2019c), and *replaying examples with larger estimated "forgetting"* as in Maximally Interfering Retrieval (MIR) (Aljundi et al., 2019a). MIR is also applied to generative replay methods (GEN-MIR) to retrieve examples interfering to the main model for replay, but the algorithm does not learn to generate interferring examples for the main model. Different from these, GMED optimizes towards searching examples around orignal examples that suffer most from forgetting, and also does not require a generative model to be trained in pararrel.

Adversarial Example Construction and our GMED framework both propose training on edited examples to achieve either adversarial robustness (Song et al., 2018; Papernot et al., 2017) or to mitigate catastrophic forgetting respectively. However, the key difference lies in the optimization objective: while adversarial construction focuses on finding mis-classified examples (Goodfellow et al., 2015; Madry et al., 2018), GMED aims at constructing examples that would suffer from loss increase in future training steps.

3. Method

We first introduce the formulation of task-free continual learning (Sec. 3.1) and discuss memory-based continual learning algorithms and their limitations (Sec. 3.2). We then present our gradient-based memory editing method GMED (Sec. 3.3) and describe how GMED can be integrated into existing memory-based CL algorithms (Sec. 3.4).

3.1. Problem Formulation

In task-free continual learning (CL), we consider a (potentially infinite) stream D of labeled examples $\{(x, y)\}$,

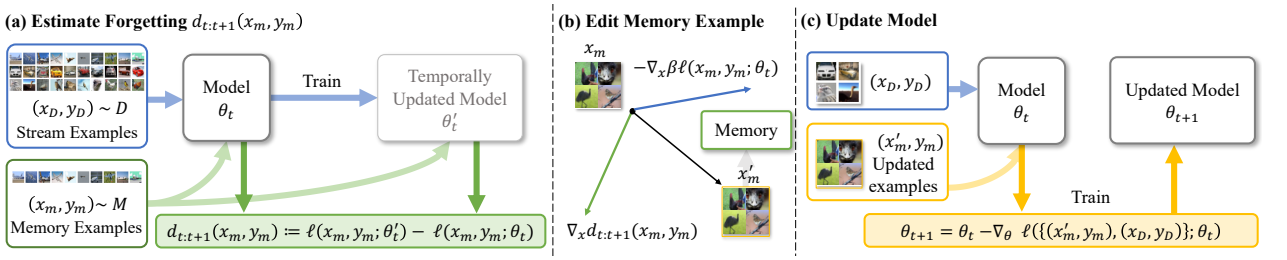


Figure 2: **A schematic of our proposed GMED framework**. (a) Given an example from the data-stream (x_D, y_D) at time t, the model randomly draws an example from the memory (x_m, y_m) and estimates "forgetting" Eq. 1 after one-step roll-out. (b) The example drawn from the memory is then updated using our proposed editing objective (Eq. 6) via gradient ascent, resulting in $(\hat{x}_m, y_m)$, and written back into the memory. (c) Finally, the model is updated using the edited example $(\hat{x}_m, y_m)$ and the example from the data-stream (x_D, y_D).

having a non-stationary data distribution, *i.e.*, the data distribution $p(x, y)$ evolves over time. Let (x_t, y_t) denote the labeled example (or mini-batch of examples) received by the model at time-step t from the data-stream D. We assume, for simplicity, that (x_t, y_t) is generated by first sampling a *latent* "task" $z \sim p(z; t)$, followed by sampling a data example from a joint data distribution $p(x, y|z)$ that is conditioned on task z, *i.e.*, $(x_t, y_t) \sim p(x, y|z)$. Here $p(z; t)$ is non-i.i.d and time-dependent. In task-free CL setting, the latent task z is *not* revealed to the model. Finally, we emphasize that while models in task-aware setup proceed to the next task only after convergence on the current task (Kirkpatrick et al., 2017; Zenke et al., 2017), models in online learning setup perform a single pass over the entire data stream (Chaudhry et al., 2019; Chrysakis & Moens, 2020).

Following the above definitions, our goal is to learn a classification model $f(x; \theta)$. The model is trained to minimize a predefined loss $\ell(x, y; \theta)$ on a new example without increasing loss on previously visited examples. If the model is evaluated at time-step T, and if $p_c(x, y; T)$ denotes the distribution of the data visited until the time step T, then our goal is to minimize the expected loss $\mathbb{E}_{p_c(x,y;T)}\ell(x, y; \theta)$. Essentially, the objective implies retaining performance on previous tasks is crucial.

3.2. Memory-based CL Methods

In the online task-free setting, previously visited examples cannot be accessed and thus computing the loss over all visited examples is not possible. To address this issue, a popular class of methods called memory-based continual learning algorithms have been studied. These algorithms store a subset of examples in a fixed-size memory and replay or regularize future training with them (Robins, 1995; Lopez-Paz & Ranzato, 2017). Figure 1 (a), (b) summarize the two key components in these methods: *memory population* step and *memory sampling* step respectively.

In the memory population step, the algorithm chooses to store new examples from the stream (denoted as (x_D, y_D)) or replace stored examples in the memory. Existing works rely on reservoir sampling (Vitter, 1985), or formulate the problem as an optimization problem to store a label-balanced set of examples (Chrysakis & Moens, 2020) (when the training stream is imbalanced) or encourage the diversity of stored examples (Aljundi et al., 2019c).

In the memory sampling step, the algorithm chooses one or a mini-batch of memory examples (denoted by (x_m, y_m)) from the memory M for replay. This step can be performed by either random sampling, or retrieving examples that are maximally *forgotten* (Aljundi et al., 2019a). Taking MIR algorithm as an example, suppose we train the model on D until a time step T. Then the "*forgetting*" measurement of an example (x, y) for the model at time t is defined as the "loss increase" (denoted by $d_{t:T}(x, y)$) at time T compared to that at time t:

$$d_{t:T}(x, y) = \ell(x, y; \theta_T) - \ell(x, y; \theta_t), \tag{1}$$

where θ_t is the model parameters at the current time step t. A larger $d_{t:T}(x, y)$ indicates that the particular example is important and contains "forgotten" knowledge making it necessary to be stored in the memory. Such samples would be utilized for replaying by MIR algorithm.

Limitations. While recent works improve memory based CL algorithms with more fine-grained strategies to select examples to store or replay, these methods are inherently limited to selecting *original* examples. Moving away from replaying real examples, a line of works on deep generative replay (Shin et al., 2017; Hu et al., 2019; Rostami et al., 2019) generates synthetic examples to replay with a generative model trained online, but the approach is rather an alternative to draw past examples in a restricted scenario and does not search optimal examples to be replayed. Buzzega et al. (2020) suggest replaying augmented examples (*e.g.*, random crop, horizontal flip) reduce over-fitting to the small replay memory and significantly improves performance. However, these data-augmentations are *pre-defined* and cannot adapt to the forgetting pattern of the model.

3.3. Gradient based Memory Editing (GMED)

Overview. We address the limitations in prior works in our proposed approach Gradient based Memory Editing

(GMED) by allowing examples stored in the memory to be edited in the continuous input space. In particular, the editing step is guided by an optimization objective instead of being pre-defined. Figure 1(c) illustrates the additional memory editing step in GMED. This step involves drawing examples from the memory, editing the drawn examples, replaying the edited examples and at the same time writing the edited examples back to the memory. We now state our optimization objectives of example editing followed by algorithmic details of GMED.

Memory Editing Objectives. Clearly, the most crucial step involved in GMED is identifying "*how*" should the stored examples in the memory be edited. If (x_m, y_m) denotes an original example obtained from the data-stream D, the goal is to design a suitable editing function ϕ to generate the edited example $\hat{x}_m$ where $\hat{x}_m{=}\phi(x_m)$. In general, the editing function ϕ is a design choice. For instance, the editing function could be an additive random noise, or a function that produces adversarial examples (Song et al., 2018; Goodfellow et al., 2015; Papernot et al., 2017).

In the context of online continual learning, we hypothesize that the resulting edited examples in memory should capture the knowledge forgotten (not retained) by the model. Thus, replaying these edited examples would be effective in alleviating catastrophic forgetting. Re-using the notations in Eq. 1 and denoting $d_{0:T}$ as d_T, we require

$$\hat{x}_m = \phi(x_m) = \arg\max_x \quad d_T(x, y_m). \tag{2}$$

We note that a similar hypothesis is used in prior art, MIR (Aljundi et al., 2019a). However, a key distinction is that in MIR the notion of using most effective example for replaying is used to sample a particular stored example for replay; whereas in GMED it is used only for updating the stored example and can complement both random sampling (ER) or maximally inferred retrieval (MIR).

We notice that using the editing function in Eq. 2 could lead to $\hat{x}_m$ to have no resemblance to x_m which makes it significantly out-of-distribution. To avoid this, we constrain the resulting editing $\hat{x}_m$ to be in close proximity to original example x_m. The resulting objective is:

$$\hat{x}_m = \phi(x_m) = \arg\max_x \quad d_T(x, y_m); \tag{3}$$

$$\text{subject to} \quad \|x - x_m\|_p \leq \epsilon, \tag{4}$$

where $\|x - x_m\|_p$ denotes the p-normed distance between x and x_m. Since the distance is constrained, we assume that the ground truth label y_m does not change. This assumption is consistent with the literature on adversarial example construction (Goodfellow et al., 2015; Madry et al., 2018).

Computing the Edited Example. If the constraint in Eq. 3 is replaced with a differentiable regularization term such as $\|x - x_m\|_2$ (for L2-norm), one could apply gradient ascent to obtain $\hat{x}_m$. However, directly penalizing $\|x - x_m\|_2^2$ requires storing the original example x_m and costs extra memory. To address this bottleneck, we instead propose a heuristic regularization term $\ell(x, y; \theta_t)$, *i.e.* the loss evaluated on the example (x, y) at current time step t. This regularization term discourages any drastic changes from the original example as a smaller value of $\ell(x, y_m; \theta_t)$ suggests x is close to x_m. Using the regularization term in Eq. 3 and applying gradient ascent results in:

$$x_m \leftarrow x_m + \nabla_x[d_T(x_m, y_m) - \beta\ell(x_m, y_m; \theta_t)]. \tag{5}$$

In practice, we perform only one gradient step for efficiency and restricting the amount of editing.

We observe that the "forgetting" term $d_T(x_m, y_m)$ required for optimization can only be computed at a future time step T and not online. Furthermore, approximations such as rolling out one future time-step Aljundi et al. (2019a) can be too coarse and subject to noise. To resolve this issue, we iteratively update the examples alongside training and accumulate one-step roll-out of the "forgetting term". More specifically, at the time step t the memory example (x_m^k, y_m^k) has already been drawn for replay for $k{-}1$ times. The iterative update is written as,

$$x_m^k \leftarrow x_m^{k-1} + \gamma^k \alpha \nabla_x[d_{t:t+1}(x_m^{k-1}, y_m) - \beta\ell(x_m^{k-1}, y_m; \theta_t)], \tag{6}$$

where γ is a decay factor of edit performed on the model. Throughout the experiments, we fix γ as 0.99 or 1.0 (*i.e.*, no decay). A decay factor less than 1.0 could effectively prevent x_m^k from drastically deviating from their original state. The hyper-parameter α controls the overall stride of the edit and is tuned with first three tasks together with the regularization strength β.

3.4. The GMED Algorithm

With the objectives of editing in place, we detail the process of incorporating GMED with Experience Replay (ER). Algorithm 1 summarizes the process and Figure 2 provides a schematic of the steps involved in GMED.

At time step t, the model receives an example (x_D, y_D) from the training stream D, and randomly draws a memory example (x_m, y_m) from the memory M. We first compute the forgetting (*i.e.,* loss increase) on the memory example (x_m, y_m). Here, (x_m, y_m) may have already been edited in previous time steps. We evaluate the forgetting of the example (x_m, y_m) when the model performs one gradient update on parameters with the stream example (x_D, y_D).

$$\theta_t' = \theta_t - \nabla_\theta \ell(x_D, y_D; \theta_t); \tag{7}$$

$$d_{t:t+1}(x_m, y_m) = \ell(x_m, y_m; \theta_t') - \ell(x_m, y_m; \theta_t), \tag{8}$$

Algorithm 1 Memory Editing with ER (ER+GMED)

Input: learning rate τ, edit stride α, regularization strength β, decay rate γ, model parameters θ
Receives: stream example (x_D, y_D)
Initialize: replay memory M
for $t = 1$ **to** T **do**
  $(x_m, y_m) \sim M$; $k \leftarrow$ replayed_time(x_m, y_m)
  $\ell_{\text{before}} \leftarrow \text{loss}(x_m, y_m, \theta_t)$; $\ell_{\text{stream}} \leftarrow \text{loss}(x_D, y_D, \theta_t)$
  `//update model with stream examples`
  $\theta'_t \leftarrow \text{SGD}(\ell_{\text{stream}}, \theta_t, \tau)$
  `//evaluate forgetting of memory examples`
  $\ell_{\text{after}} \leftarrow \text{loss}(x_m, y_m, \theta'_t)$
  $d \leftarrow \ell_{\text{after}} - \ell_{\text{before}}$
  `//edit memory examples`
  $x'_m \leftarrow x_m + \gamma^k \alpha \nabla_x (d - \beta \ell_{\text{before}})$
  $\ell = \text{loss}(\{(x'_m, y_m), (x_D, y_D)\}, \theta_t)$
  $\theta_{t+1} \leftarrow \text{SGD}(\ell, \theta_t, \tau)$
  replace (x_m, y_m) with (x'_m, y_m) in M
  reservoir_update(x_D, y_D, M)
end for

where θ_t and θ'_t are model parameters before and after the gradient update respectively. Following Eq. 6, we perform a gradient update on x to increase its "forgetting". The algorithm then discards θ'_t, and updates model parameters θ_t using the edited memory example (x'_m, y_m) and the stream example (x_D, y_D), in a similar way to ER.

$$\theta_{t+1} = \theta_t - \nabla_\theta \ell(\{(x'_m, y_m), (x_D, y_D)\}; \theta_t). \quad (9)$$

We replace the original examples in the memory with the edited example. In this way, we continuously edit examples stored in the memory alongside training.

Applying GMED to ER$_{aug}$, MIR and GEM. Since the process to edit the original examples in GMED is modular, we can integrate GMED with a range of existing memory-based CL algorithms. In addition to ER, we also explore ER with data augmentation (ER$_{aug}$) (Buzzega et al., 2020), MIR and GEM in our experiments.

ER$_{aug}$ applies standard data augmentations (*e.g.*, random cropping, horizontal flipping, denoted as $\mathcal{T}$) to examples (x_m, y_m) drawn from the memory which are replayed at each time step. In ER$_{aug}$+GMED, we edit original example x_m and replay both the augmented example $\mathcal{T}(x_m)$ and the edited example $\hat{x}_m$. To keep the number of replayed examples the same, for ER$_{aug}$ method, we also replay both the edited example $\mathcal{T}(x_m)$ and the original example x_m. Finally, we write the edited example $\hat{x}_m$ to the memory.

For integration with MIR, recall that MIR retrieves and then replays the most "interfered" examples (as defined in Eq. 1) in the memory at each time-step. As a result, simply making GMED edits on these MIR-retrieved examples may induce a loop of further increasing "forgetting" of the examples that are already the most interferring ones. Instead, in our MIR+GMED implementation, we edit a mini-batch of examples randomly sampled from the memory — a process that is *independent* of the MIR replay operation[1]. Moreover, such design (*i.e.*, random sampling) may also help prevent GMED editing from intensifying potential biases created from the MIR retrieval process (*e.g.*, retrieved examples are edited and thus become more interferred).

For GEM+GMED, while all memory examples are used in forming the GEM regularization term, we apply GMED to edit a mini-batch of randomly sampled examples, for fair comparison. Details of the integrated algorithms (*i.e.*, ER$_{aug}$+GMED, MIR+GMED, and GEM+GMED) can be found in Algorithms 2, 3 and 4 in Appendix B. There can be more sophisticated designs to integrate existing CL algorithm with GMED, *e.g.*, by optimizing the retrieval and editing operations jointly — we leave this as future work.

4. Experiments

Our experiments look to address two questions: (1) performance of using GMED with other state-of-art CL methods (2) effectiveness of the proposed editing objective compared to alternative editing objectives (*e.g.*, adversarial edit) and other data-augmentation approaches. In the rest of this section, we first briefly describe the datasets (Sec. 4.1) and the compared baselines (Sec. 4.2). We then detail our main results and validate the advantage of GMED over random editing and standard data augmentation followed by a brief qualitative analysis (Sec. 4.3).

4.1. Datasets

We use six public CL datasets in our experiments.

Split / Permuted / Rotated MNIST are constructed from the MNIST (LeCun et al., 1998) dataset which contains images of handwritten digits. **Split MNIST** (Goodfellow et al., 2013) creates 5 disjoint subsets based on the class labels and considers each subset as a separate task. The goal then is to classify over all 10 digits when the training ends. **Permuted MNIST** (Goodfellow et al., 2013) consists of 10 tasks, where for a particular task a random permutation in the pixel space is chosen and applied to all images within that task. The model then has to classify over the 10 digits without knowing which random permutation was applied. **Rotated MNIST** (Lopez-Paz & Ranzato, 2017) rotates every sample in MNIST by a fixed angle between 0 to 180. Similar to the previous datasets, the goal is to classify over 10 digits without any knowledge of the angle of rotation. For all MNIST experiments, each task consists of 1,000 training examples following Aljundi et al. (2019a).

[1]This also ensures the integrated approach will replay the same number of examples as the baselines, yielding a fair comparison.

Table 1: **Mean and standard deviation of final accuracy (%)** in 10 runs for non-model-expansion-based approaches. See Table 3 for the results of expansion based approaches. For Split mini-ImageNet and Split CIFAR-100 datasets, we set the memory size to 10,000 examples; we use 500 for other datasets. * and ** over GMED methods indicate significant improvement over the counterparts without GMED with p-values less than 0.1 and 0.05 respectively in single-tailed paired t-tests. * and ** over Previous SoTA results indicate significant improvement of the best performing GMED method over best performing baselines.

Methods / Datasets	Task-free	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split CIFAR-100	Split mini-ImageNet
Fine tuning	✓	18.80 ± 0.6	66.34 ± 2.6	41.24 ± 1.5	18.49 ± 0.2	3.06 ± 0.2	2.84 ± 0.4
AGEM (Chaudhry et al., 2019)	✓	29.02 ± 5.3	72.17 ± 1.5	50.77 ± 1.9	18.49 ± 0.6	2.40 ± 0.2	2.92 ± 0.3
GSS-Greedy (Aljundi et al., 2019c)	✓	84.16 ± 2.6	77.43 ± 1.4	73.66 ± 1.1	28.02 ± 1.3	19.53 ± 1.3	16.19 ± 0.7
BGD (Zeno et al., 2018)	✓	13.54 ± 5.1	19.38 ± 3.0	77.94 ± 0.9	18.23 ± 0.5	3.11 ± 0.2	24.71 ± 0.8
HAL (Chaudhry et al., 2020)	✗	77.92 ± 4.2	77.55 ± 4.2	78.48 ± 1.5	32.06 ± 1.5	21.11 ± 1.4	21.18 ± 2.1
GEM (Lopez-Paz & Ranzato, 2017)	✗	87.18 ± 1.3	78.59 ± 1.0	78.40 ± 0.8	20.05 ± 1.4	8.75 ± 0.4	11.27 ± 3.4
GEM + GMED	✗	87.64 ± 1.3	78.67 ± 1.1	78.62 ± 0.4	20.49 ± 1.5	8.72 ± 0.4	11.35 ± 0.3
ER (Robins, 1995)	✓	80.96 ± 2.3	79.69 ± 1.0	76.95 ± 1.7	33.34 ± 1.5	20.65 ± 1.3	26.00 ± 1.0
ER + GMED	✓	$82.68^{**} \pm 2.1$	79.70 ± 1.1	$77.89^{**} \pm 0.9$	$36.21^{**} \pm 3.0$	20.87 ± 1.4	$28.19^{**} \pm 0.8$
MIR (Aljundi et al., 2019a)	✓	84.88 ± 1.7	79.96 ± 1.3	78.30 ± 1.0	34.47 ± 2.0	20.18 ± 1.7	25.01 ± 1.3
MIR + GMED	✓	$\mathbf{87.86}^{**} \pm \mathbf{1.1}$	$\mathbf{80.11}^{**} \pm \mathbf{1.2}$	$79.16^{**} \pm 0.9$	$36.44^{**} \pm 1.1$	$\mathbf{21.49}^{**} \pm \mathbf{0.6}$	$26.29^{**} \pm 1.2$
ER_{aug} (Buzzega et al., 2020)	✓	79.91 ± 1.7	79.00 ± 0.8	80.60 ± 1.3	47.34 ± 3.1	18.66 ± 2.0	31.36 ± 2.1
ER_{aug} + GMED	✓	$82.39^{**} \pm 2.1$	79.08 ± 0.8	$\mathbf{80.66} \pm \mathbf{1.2}$	$\mathbf{48.34}^{*} \pm \mathbf{2.6}$	19.56 ± 1.3	$\mathbf{32.45}^{*} \pm \mathbf{1.2}$
Previous SoTA	-	87.18 ± 1.3	$79.96^{**} \pm 1.3$	80.60 ± 1.3	$47.34^{**} \pm 3.1$	21.11 ± 1.4	$31.36^{*} \pm 1.6$
iid online	-	85.99 ± 0.3	73.58 ± 1.5	81.30 ± 1.3	62.23 ± 1.5	18.13 ± 0.8	17.53 ± 1.6
iid offline (upper bound)	-	93.87 ± 0.5	87.40 ± 1.1	91.38 ± 0.7	76.36 ± 0.9	42.00 ± 0.9	37.46 ± 1.3

Split CIFAR-10 and Split CIFAR-100 comprise of 5 and 20 disjoint subsets respectively based on their class labels. The model then classifies over the space of all class labels. Similarly, **Split mini-ImageNet** (Aljundi et al., 2019a) splits the mini-ImageNet (Deng et al., 2009; Vinyals et al., 2016) dataset into 20 disjoint subsets based on their labels. The models classify over all 100 classes.

Finally, note that our models are trained under Task-free continual learning setting *i.e.*task identities or boundaries are not provided to the model at both train and test time. Following the taxonomy proposed in van de Ven & Tolias (2019), the Split MNIST, Split CIFAR-10, Split CIFAR-100, and Split mini-ImageNet experiments are categorized under class-incremental setup, while Permuted and Rotated MNIST experiments belong to domain-incremental setup.

4.2. Compared Methods

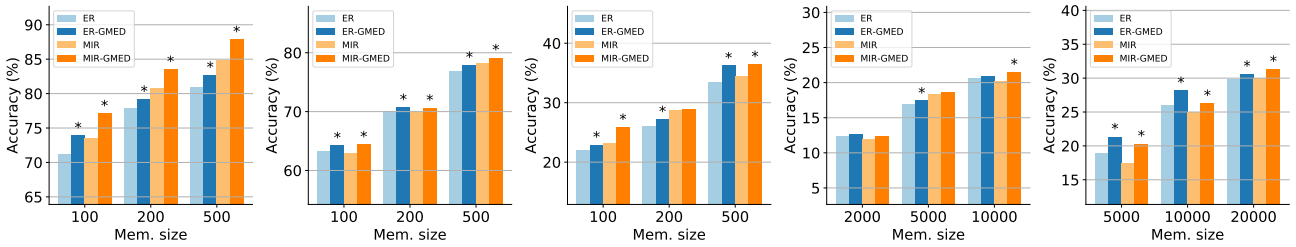
We compare against several task-free memory based continual learning methods namely, Experience Replay (ER), Averaged Gradient Episodic Memory (AGEM), Gradient based Sample Selection (GSS), and Maximally Interfering Retrieval (MIR). We omit the generative replay method GEN-MIR proposed together in (Aljundi et al., 2019a) as it underperforms their memory-based counterparts even on simple datasets such as Split MNIST.

We also compare with data augmentation (Buzzega et al., 2020) such as random rotations, scaling, and horizontal flipping applied to memory examples drawn for replay in ER, noted as ER_{aug} (except for MNIST datasets). We also include regularization-based, model expansion-based and task-aware approaches, namely Bayesian Graident Descent (BGD), Neural Dirichlet Process Mixture Model (CN-DPM) (Lee et al., 2020), Progressive Networks (Prog.NN) (Rusu et al., 2016), Compositional Lifelong Learning (Mendez & EATON, 2021) and Graident Episodic Memory (GEM) and Hindsight Anchor Learning (HAL) (Chaudhry et al., 2020) respectively.

Finally, we report three baseline models: (i) Fine Tuning, where no continual learning algorithms are used for online updates to model parameters, (ii) iid Online, where we randomly shuffle the data stream, so that the model visits an i.i.d. stream of examples, and (iii) iid Offline, where multiple passes over the dataset is allowed. See Appendix A for more details on compared methods and their implementation details. We build our proposed GMED approach upon four baselines, noted as ER+GMED, MIR+GMED, GEM+GMED, and ER_{aug}+GMED.

Alternative Editing Objectives. We consider two alternatives to the proposed editing objective (Eq. 6): (i) Random Edit: memory examples are updated in a random direction with a fixed stride; (ii) Adversarial Edit: following the literature of adversarial example construction (Goodfellow et al., 2015), the edit increases the loss of memory by following the gradient sgn $\nabla_x \ell(x_m, y_m; \theta)$.

Implementation Details. We set the size of replay memory as $10K$ for split CIFAR-100 and split mini-ImageNet, and 500 for all remaining datasets. Following Chaudhry et al. (2019), we tune the hyper-parameters α (editing stride) and β (regularization strength) with only the first three tasks. While γ (decay rate of the editing stride) is a hyper-



(a) Split MNIST *(b)* Rotated MNIST *(c)* Split CIFAR-10 *(d)* Split CIFAR-100 *(e)* Split mini-ImgNet

Figure 3: **Performance of ER, GMED+ER, MIR, and GMED+MIR across different memory sizes**. For mini-ImageNet dataset, we use memory sizes of $1K$, $5K$, $10K$, and $20K$ examples; for Split CIFAR-100 dataset, we use $1K$, $2K$, $5K$ and $10K$; for other datasets, we use 100, 200, 500, and 1000. * indicates whether the improvement of ER+GMED or MIR+GMED is significant ($p < 0.05$).

Table 2: **Performance of methods over data streams with fuzzy task boundaries**. In this setup, examples from the next tasks are introduced and gradually dominate the stream when half of the examples from the current task is visited. * indicates whether the improvement is significant ($p < 0.05$)

Methods / Datasets	Split MNIST	Rotated MNIST	Split CIFAR-10	Split mini-ImageNet
Vanilla	21.53 ± 0.1	42.58 ± 0.4	20.69 ± 2.4	3.05 ± 0.6
AGEM	25.30 ± 0.9	52.21 ± 0.3	21.01 ± 0.2	3.88 ± 0.0
ER	79.74 ± 4.0	76.45 ± 1.2	37.15 ± 1.6	26.47 ± 2.3
MIR	85.80 ± 1.9	77.04 ± 1.2	38.70 ± 1.7	25.83 ± 1.5
ER$_{aug}$	81.30 ± 2.0	$\mathbf{80.31 \pm 0.9}$	47.97 ± 3.5	31.75 ± 1.0
ER + GMED	$82.73^* \pm 2.6$	76.55 ± 1.0	$40.57^* \pm 1.7$	$28.20^* \pm 0.6$
MIR + GMED	$\mathbf{86.17 \pm 1.7}$	$77.56^* \pm 1.1$	$41.22^* \pm 1.1$	$26.86^* \pm 0.7$
ER$_{aug}$ + GMED	$82.39^* \pm 3.7$	80.30 ± 1.0	$\mathbf{51.38^* \pm 2.2}$	$\mathbf{31.83 \pm 0.8}$

parameter that may flexibly control the deviation of edited examples from their original states, we find γ=1.0 (*i.e.*, no decay) leads to better performance in our experiments. Results under different γ setups are provided in Appendix C, and in the remaining sections we assume no decay is applied. For model architectures, we mostly follow the setup of Aljundi et al. (2019a): for the three MNIST datasets, we use a MLP classifier with 2 hidden layers with 400 hidden units each. For Split CIFAR-10, Split CIFAR-100 and Split mini-ImageNet datasets, we use a ResNet-18 classifier. See Appendix C for more details.

4.3. Results and Performance Analysis

We report the final accuracy and the standard deviation over 10 runs achieved by different methods in Table 1 and make the following key observations.

Overall Performance. We find MIR+GMED achieves the best performance on Permuted MNIST dataset; while on Split CIFAR-10 and Split mini-ImageNet dataset, ER$_{aug}$+GMED achieves the best performance. Performance of the best performing GMED method on other datasets is also on par with best performing baseline.

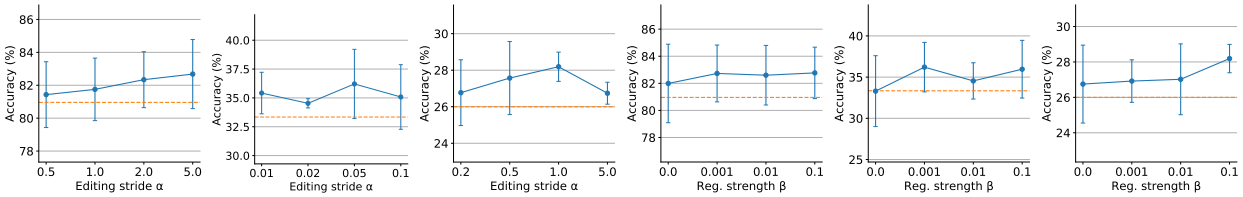
Effect of GMED. GMED significantly improves performance on 4 and 6 datasets when built upon ER and MIR. The improvement of MIR+GMED corroborates that the optimization in the continuous input spaces of GMED is complementary to sample selection over real examples as in MIR. We also notice significant improvement of ER$_{aug}$ over GMED on 3 of datasets. It implies the benefit of GMED is beyond the simple regularization effects that overcome overfitting. However, we don't observe any significant improvements on integrating GMED with GEM. This is likely because GEM performs regularization but doesn't replay examples and cannot capture the subtle changes in the examples. This further implies integrating gradient updates to regularization based approaches requires more investigation and we leave this to future work.

Table 3: **Comparison with model-expansion-based approaches** under the same memory overhead as CN-DPM. The overhead is the size of the replay memory plus the extra model components (e.g. a generator or modules to solve individual tasks), shown in the equivalent number of memory examples (#. Mem). † indicates quoted numbers from original works.

Method	Split MNIST		Split CIFAR-10		Split CIFAR-100	
	Acc.	#. Mem	Acc.	#. Mem	Acc.	#. Mem
ER	92.67	2,581	62.96	6,024	21.79	21,295
ER+GMED	94.16	2,581	63.28	6,024	22.12	21,295
CN-DPM†	93.23	2,581	45.21	6,024	20.10	21,295
Prog. NN	89.46	9,755	49.68	6,604	19.17	31,766
CompCL	91.27	9,755	45.62	6,604	20.51	31,766

Performance under Various Memory Sizes. Figure 3 shows the performance of ER, ER+GMED, MIR, and MIR+GMED under various memory sizes. The improvement on Split MNIST, Rotated MNIST, and Split mini-ImageNet are significant over all memory size setups. On Split CIFAR-10 the improvements are also mostly significant. The improvement on Split CIFAR-100 is less competitive, probably because the dataset is overly difficult for class-incremental learning, from the accuracy around or less than 20% in all memory setups.

Comparison with Model Expansion Approach (CN-DPM). Non-memory based continual learning approaches introduce extra overhead in storing model parameters. For example, CN-DPM employs a generative model, a dynamically expanding classifier, and further utilizes a short-term



(a) Split MNIST *(b)* Split CIFAR-10 *(c)* Split mini-ImgNet, *(d)* Split MNIST *(e)* Split CIFAR-10 *(f)* Split mini-ImgNet

Figure 4: **Parameter sensitivity analysis** of the editing stride α (a,b,c) and the regularization strength β (d,e,f) on GMED-ER. The dashed horizontal line indicates the performance of ER.

Table 4: **Alternate Memory Editing Objectives.** Comparison of Random Edit, Adversarial Edit (Adv. Edit) to our proposed objective in GMED. * indicates significant improvement ($p<0.05$) compared to adversarial edit.

Methods / Datasets	Split MNIST	Rotated MNIST	Split CIFAR-10	Split mini-ImageNet
ER + Random Edit	81.04 ± 2.1	77.59 ± 1.0	32.26 ± 1.9	26.23 ± 2.1
ER + Adv. Edit	80.66 ± 2.3	77.82 ± 1.7	31.69 ± 0.8	26.09 ± 1.6
ER + GMED	**82.68*** ± **2.1**	**77.89** ± **0.9**	**36.21*** ± **3.0**	**28.19*** ± **0.8**
MIR + Random Edit	84.76 ± 1.4	78.19 ± 1.0	35.39 ± 3.0	24.86 ± 0.7
MIR + Adv. Edit	86.67 ± 0.5	78.06 ± 1.5	35.79 ± 0.4	25.48 ± 1.3
MIR + GMED	**87.86** ± **1.1**	**79.16*** ± **0.9**	**36.44** ± **1.1**	**26.29*** ± **1.2**

memory (STM). Following Hsu et al. (2018), we set the memory size for GMED so that two methods introduces the same amount of the overhead. Table 3 show the results of ER, ER+GMED and the reported results of CN-DPM. Interestingly, ER by itself achieves comparable performance to model expansion approaches. ER+GMED further outperforms CN-DPM without any extra memory overhead compared to ER. Similarly, GMED outperforms Prog. NN and the recently proposed compositional model expansion (CompCL) with a smaller memory overhead.

Performance under Fuzzy Task Boundaries. While Table 1 assumes clear task boundaries, we further show GMED brings improvement over data streams with fuzzy task boundaries. Table 2 summarizes the results on four datasets. We leave the complete results in Appendix. In this setup, the probability density of a new task grows linearly starting from the point where 50% of examples of the current task are visited. In particular, GMED improves performance on two (ER_{aug}) to three (ER, MIR) out of four datasets.

Parameter Sensitivity. GMED introduces two hyperparameters: the stride of the editing α and the regularization strength γ (Eq. 6). Figure 4 shows the sensitivity of the performance to two hyperparameters. Clearly, ER+GMED outperforms ER over a broad range of α and β setups. Furthermore, in Figure 4 (d,e,f), better performance for non-zero β confirms the benefit of the regularization term.

Comparison with Alternative Editing Strategies. Table 4 compares GMED with the two alternative editing baselines: random editing and adversarial editing (introduced in Sec. 4.2). We notice ER+Random Edit outperforms ER on split CIFAR-10 and CIFAR-100, which confirms the regularization benefits of adding random noise to memory examples. However, using GMED is always more preferable to random editing and adversarial editing baselines across datasets and models. This confirms the merits of the proposed editing objective.

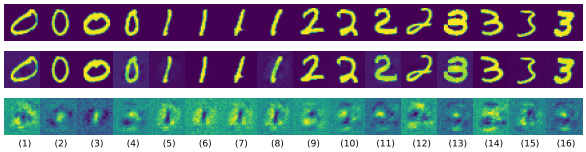


Figure 5: **Visualization of the editing on examples in Split MNIST**. The first two rows show examples before and after editing, and the third row shows the differences.

Case study and Visualization of Edited Examples. In Figure 5, we visualize the editing on memory examples. We show examples from first two task (0/1, 2/3) in the Split MNIST dataset. The first and second rows show the original and edited examples, noted as x_{before} and x_{after}. The third row shows the difference between two $\Delta x = x_{\text{after}} - x_{\text{before}}$. We see no significant visual differences between original and edited examples. However, by looking at the difference Δx, we see there are examples whose contours get exaggerated, *e.g.*, examples 1 and 12, and some get blurred, *e.g.*, examples 2, 3, 5, and 6. Intuitively, to make an ambiguous example more forgettable, the editing should exaggerate its features; while to make a typical example more forgettable, the editing should blur its features. Our visualizations align with the intuition above: examples 1 and 12 are not typically written digits, while examples like 2, 3, 5, and 6 are typical. We also provide a t-SNE (Maaten & Hinton, 2008) plot of the edit vector in Appendix D.

5. Conclusion

In this paper, we propose Gradient based Memory Editing (GMED), a modular framework for task-free continual learning where examples stored in the memory can be edited. Importantly, memory examples edited by GMED remain in-distribution but yield increased loss when replayed, and thus are more effective at alleviating catastrophic forgetting. Finally, we validate that combining GMED with existing memory-based CL approaches leads to consistent improvements across multiple benchmark datasets.

References

Adel, T., Zhao, H., and Turner, R. E. Continual learning with adaptive weights (CLAW). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hklso24Kwr.

Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., and Tuytelaars, T. Online continual learning with maximally interfered retrieval. In *NeurIPS*, 2019a.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 11254–11263. Computer Vision Foundation / IEEE, 2019b. doi: 10.1109/CVPR.2019.01151. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Aljundi_Task-Free_Continual_Learning_CVPR_2019_paper.html.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11816–11825, 2019c. URL https://proceedings.neurips.cc/paper/2019/hash/e562cd9c0768d5464b64cf61da7fc6bb-Abstract.html.

Buzzega, P., Boschini, M., Porrello, A., and Calderara, S. Rethinking experience replay: a bag of tricks for continual learning. *ArXiv*, abs/2010.05595, 2020.

Caccia, M., Rodríguez, P., Ostapenko, O., Normandin, F., Lin, M., Caccia, L., Laradji, I. H., Rish, I., Lacoste, A., Vázquez, D., and Charlin, L. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *NeurIPS*, 2020.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Chaudhry, A., Gordo, A., Dokania, P. K., Torr, P. H. S., and Lopez-Paz, D. Using hindsight to anchor past knowledge in continual learning. *ArXiv*, abs/2002.08165, 2020.

Chrysakis, A. and Moens, M. Online continual learning from imbalanced data. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1952–1961. PMLR, 2020. URL http://proceedings.mlr.press/v119/chrysakis20a.html.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-guided continual learning with bayesian neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HklUCCVKDB.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Harrison, J., Sharma, A., Finn, C., and Pavone, M. Continuous meta-learning without tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/cc3f5463bc4d26bc38eadc8bcffbc654-Abstract.html.

He, X., Sygnowski, J., Galashov, A., Rusu, A. A., Teh, Y. W., and Pascanu, R. Task agnostic continual learning via meta learning. *ArXiv*, abs/1906.05201, 2019.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R. Overcoming catastrophic forgetting for continual learning via model adaptation. In *7th International Conference on Learning Representations, ICLR*

2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGvcoA5YX.

Javed, K. and White, M. Meta-learning representations for continual learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1818–1828, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f4dd765c12f2ef67f98f3558c282a9cd-Abstract.html.

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

Lange, M. D., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *arXiv: Computer Vision and Pattern Recognition*, 2019.

LeCun, Y., Cortes, C., and Burges, C. J. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

Lee, S., Ha, J., Zhang, D., and Kim, G. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxSOJStPr.

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

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3925–3934. PMLR, 2019. URL http://proceedings.mlr.press/v97/li19m.html.

Liu, B. Learning on the job: Online lifelong and continual learning. In *AAAI*, 2020.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6467–6476, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Mendez, J. A. and EATON, E. Lifelong learning of compositional structures. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ADWd4TJO13G.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BkQqq0gRb.

Papernot, N., McDaniel, P., Goodfellow, I. J., Jha, S., Celik, Z. Y., and Swami, A. Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

Parisi, G., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural networks : the official journal of the International Neural Network Society*, 113:54–71, 2019.

Robins, A. V. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 7:123–146, 1995.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. Experience replay for continual learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019,*

December 8-14, 2019, Vancouver, BC, Canada, pp. 348–358, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract.html.

Rostami, M., Kolouri, S., and Pilly, P. K. Complementary learning for overcoming catastrophic forgetting using experience replay. In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 3339–3345. ijcai.org, 2019. doi: 10.24963/ijcai.2019/463. URL https://doi.org/10.24963/ijcai.2019/463.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016.

Serrà, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4555–4564. PMLR, 2018. URL http://proceedings.mlr.press/v80/serra18a.html.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2990–2999, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/0efbe98067c6c73dba1250d2beaa81f9-Abstract.html.

Song, Y., Shu, R., Kushman, N., and Ermon, S. Constructing unrestricted adversarial examples with generative models. In *NeurIPS*, 2018.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3630–3638, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html.

Vitter, J. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11:37–57, 1985.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017. URL http://proceedings.mlr.press/v70/zenke17a.html.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. Task agnostic continual learning using online variational bayes. *ArXiv*, abs/1803.10123, 2018.

Table 5: Hyperparamters of the editing stride and the regularization strength selected for ER+GMED.

Dataset / Hyper-param	Editing stride α	Regularization strength β
ER+GMED, MIR+GMED, GEM+GMED		
Split MNIST	5.0	0.01
Permuted MNIST	0.05	0.001
Rotated MNIST	1.0	0.01
Split CIFAR-10	0.05	0.001
Split CIFAR-100	0.05	0.001
Split mini-ImageNet	1.0	0.1
ER_{aug} + GMED		
Split MNIST	5.0	0.001
Permuted MNIST	0.05	0.001
Rotated MNIST	1.0	0.01
Split CIFAR-10	0.1	0.01
Split CIFAR-100	0.05	0.001
Split mini-ImageNet	0.5	0.0

Algorithm 2 Memory Editing with ER_{aug} (ER_{aug}+GMED)

Input: learning rate τ, edit stride α, regularization strength β, decay rate γ, model parameters θ
Receives: stream example (x_D, y_D)
Initialize: replay memory M
for $t = 1$ **to** T **do**
  `// draw a random mini-batch for edit`
  $(x_e, y_e) \sim M$
  $k \leftarrow$ replayed_time(x_e, y_e)
  $\ell_{\text{before}} \leftarrow \text{loss}(x_e, y_e, \theta_t)$
  $\ell_{\text{stream}} \leftarrow \text{loss}(x_D, y_D, \theta_t)$
  `//update model parameters with stream examples, discarded later`
  $\theta'_t \leftarrow \text{SGD}(\ell_{\text{stream}}, \theta_t, \tau)$
  `//evaluate forgetting of memory examples`
  $\ell_{\text{after}} \leftarrow \text{loss}(x_e, y_e, \theta'_t)$
  $d \leftarrow \ell_{\text{after}} - \ell_{\text{before}}$
  `//edit memory examples`
  $x'_e \leftarrow x_e + \gamma^k \alpha \nabla_x (d - \beta \ell_{\text{before}})$
  replace (x_e, y_e) with (x'_e, y_e) in M
  `//Regularized model updates with GEM using the full memory`
  GEM_regularized_update(x_D, y_D, M, θ)
  `//Update the replay memory following GEM`
  memory_update(x_D, y_D, M)
end for

A. Implementation Details for Compared Baselines

We included detailed descriptions, and implementation details of some selected baselines in this section.

• **Experience Replay (ER)** (Robins, 1995; Rolnick et al., 2019) stores examples in a fix-sized memory for future replay. We use reservoir sampling to decide which examples to store and replace. Following prior works (Aljundi et al., 2019b;a; Chaudhry et al., 2020), at each time step we draw the same number of examples as the batch size from the memory to replay, which are both set to 10. The algorithm applies to the task-free scenario.

• **Gradient Episodic Memory (GEM)** (Lopez-Paz & Ranzato, 2017) also stores examples in a memory. Before each model parameter update, GEM project gradients of model parameters so that the update does not incur loss increase on any previous task. The approach is not task-free.

• **Averaged Gradient Episodic Memory (AGEM)** (Chaudhry et al., 2019) prevents the average loss increase on a randomly drawn subsets of examples from the memory. We draw 256 examples to compute the regularization at each iteration. The approach is task-free.

• **Bayesian Gradient Descent (BGD)** (Zeno et al., 2018) is a regularization-based continual learning algorithm. It adjust learning rate for parameters by estimating their certainty, which notes for their importance to previous data. The approach is task-free.

• **Gradient based Sample Selection (GSS)** (Aljundi et al., 2019c) builds upon ER by encouraging the diversity of stored examples. We use GSS-Greedy, which is the best performing variant in the paper. The approach is task-free.

• **Hindsight Anchor Learning (HAL)** (Chaudhry et al., 2020) learns an pseudo "anchor" example per task per class in addition to the replay memory by maximizing its estimated forgetting, and tries to fix model outputs on the anchors at training. However, unlike GMED, they estimate forgetting with loss increase on examples when the model train for a pass on the replay memory (and thus forgetting is estimated with "hindsight"). The approach is not task-free.

• **Maximally Interfering Retrieval (MIR)** (Aljundi et al., 2019a) improves ER by selecting top forgettable examples from the memory for replay. Following the official implementation, we evaluate forgetting on a candidate set of 25 examples for mini-ImageNet dataset, and 50 examples for others. While the approach is task-free, the official implementation filter out memory examples that belong to the same task as the current data stream, which assumes knowledge about tasks boundaries. We remove this operation to adapt the method to the task-free setup. Therefore, our results are not directly comparable to the official results.

• **Neural Dirichlet Process Model for Continual Learning (CN-DPM)** (Lee et al., 2020) is a task-free model-expansion based continual learning algorithm. We report the official results in the paper. In the comparison study between ER/ER+GMED with CN-DPM, for the base model in ER/ER+GMED, we use the full expanded model in CN-DPM (*i.e.*, the model architecture when the training ends in CN-DPM). We use the same optimizer and the learning rate as CN-DPM in this set of experiments.

Table 6: Sensitivity of the performance of GMED to the decay rate of the editing stride (γ).

Methods / Datasets	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split CIFAR-100	Split mini-ImageNet
ER	80.96 ± 2.3	79.69 ± 1.0	76.95 ± 1.7	33.34 ± 1.5	20.65 ± 1.3	26.00 ± 1.0
ER + GMED$_{\gamma=0.9}$	82.28 ± 1.7	79.07 ± 0.6	77.22 ± 1.2	33.85 ± 1.4	20.06 ± 1.8	26.92 ± 1.9
ER + GMED$_{\gamma=0.99}$	82.60 ± 2.2	79.15 ± 0.6	77.35 ± 1.3	34.10 ± 3.4	19.90 ± 1.5	27.69 ± 0.7
ER + GMED$_{\gamma=1.0}$	**82.68 ± 2.1**	**79.70 ± 1.1**	**77.89 ± 0.9**	**36.21 ± 3.0**	**20.87 ± 1.4**	**28.19 ± 0.8**
MIR	84.88 ± 1.7	79.96 ± 1.3	78.30 ± 1.0	34.47 ± 2.0	20.18 ± 1.7	25.01 ± 1.3
MIR + GMED$_{\gamma=0.9}$	85.67 ± 2.2	79.99 ± 0.7	78.45 ± 1.3	34.98 ± 0.5	19.76 ± 1.7	25.96 ± 1.2
MIR + GMED$_{\gamma=0.99}$	86.76 ± 1.2	79.76 ± 0.7	78.61 ± 0.6	35.78 ± 3.2	20.48 ± 1.7	**27.70 ± 1.3**
MIR + GMED$_{\gamma=1.0}$	**87.86 ± 1.1**	**80.11 ± 1.2**	**79.16 ± 0.9**	**36.44 ± 1.1**	**21.49 ± 0.6**	26.29 ± 1.2

Table 7: Building GMED over ER+T. * indicates significant improvement with p-value less than 0.05.

Methods / Datasets	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split CIFAR-100	Split mini-ImageNet
ER + T	78.35 ± 4.5	77.71 ± 0.7	**80.05 ± 1.3**	**47.55 ± 2.6**	19.40 ± 1.5	31.25 ± 1.5
ER + T + GMED	**83.02*** **± 0.4**	**77.92 ± 0.3**	79.96 ± 0.2	47.39 ± 5.0	**19.75 ± 1.2**	**31.84*** **± 1.3**

B. Algorithmic Details

We show algorithmic details of ER$_{aug}$+GMED, MIR+GMED and GEM+GMED in Algorithms 2, 3 and 4. The main difference of MIR+GMED and GEM+GMED compared to ER+GMED is that we edit a separate mini-batch of memory examples from the mini-batch used for replay or regularization. For ER$_{aug}$+GMED, the main difference is that we additionally replay a mini-batch of edited examples after data augmentation.

Computational Efficiency of GMED. The computational overhead of GMED consists of estimation of forgetting following Eq. 1 and computing the editing direction following 6. Both operations are time efficient: estimation of the forgetting only requires two additional forward pass of the model, while computing the editing direction requires only one backward pass of the model. It offers GMED capability to be applied in online learning scenarios, where models are expected learn over new examples in real-time.

We implemented our models with PyTorch 1.0. We train our models with GTX 1080Ti or 2080Ti GPUs, and we use CUDA toolkit 10.1. We use a mini-batch size of 10 throughout experiments. For ER+GMED, training on Split MNIST, Permuted MNIST, and Rotated MNIST takes 7 seconds, 30 seconds, and 31 seconds respectively (excluding data pre-processing). Training on Split CIFAR-10, Split CIFAR-100, and Split mini-ImageNet takes 11 minutes, 14 minutes, and 46 minutes respectively. In comparison, training with ER takes 6 seconds, 23 seconds, 16 seconds, 7 minutes, 10 minutes, and 32 minutes respectively on six datasets.

C. Hyperparameter Setup

Throughout experiments, we use SGD optimizer with a learning rate of 0.05 for MNIST datasets, 0.1 for Split CIFAR-10 and Split mini-ImageNet datasets, and 0.03 for the Split CIFAR-100 dataset.

Besides, GMED introduces two additional hyperparameters: the stride of the editing α, and the regularization strength β. As we assume no access to the full data stream in the online learning setup, we cannot select hyperparameters according to validation performance after training on the full stream. Therefore, we tune the hyperparameters with only the training and validation set of first three tasks, following (Chaudhry et al., 2019). The tasks used for hyperparameter search are included for reporting final accuracy, following (Ebrahimi et al., 2020). We perform a grid search over all combinations of α and β and select the one with the best validation performance on the first three tasks. We select α from $[0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0]$, and select β from $[0, 10^{-3}, 10^{-2}, 10^{-1}, 1]$. We tune two hyperparameters on ER+GMED and share it across MIR+GMED and GEM+GMED. We tune a separate set of hyperparameters for ER$_{aug}$+GMED. Table 5 show the optimal hyperparameters selected for each dataset.

D. T-SNE Visualization

In Figure 6, we show the t-SNE (Maaten & Hinton, 2008) visualization of the editing vector $\Delta x = x_{\text{after}} - x_{\text{before}}$ for examples from first 2 tasks in Split MNIST. We see the editing vectors cluster by the labels of the examples. It implies the editing performed is correlated with the labels and is clearly not random.

E. Dataset Details

Following (Aljundi et al., 2019a), we limit the number of training examples per task to 1,000 for all MNIST experiments, including Split MNIST, Permuted MNIST, and Rotated MNIST. The datasets consist of 5, 10, and 20 tasks

Table 8: GMED without editing stored examples, *i.e.*, the algorithm replays edited examples, but does not update the original examples in the memory as edited ones. The results are shown after ER + GMED w/o writeback.

Methods / Datasets	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split CIFAR-100	Split mini-ImageNet
ER	80.96 ± 2.3	79.69 ± 1.0	76.95 ± 1.7	33.34 ± 1.5	20.65 ± 1.3	26.00 ± 1.0
ER + GMED	**82.68 ± 2.1**	**79.70 ± 1.1**	77.89 ± 0.9	**36.21 ± 3.0**	**20.87 ± 1.4**	**28.19 ± 0.8**
ER + GMED w/o writeback	82.18 ± 1.7	79.39 ± 0.6	**78.08 ± 1.0**	34.86 ± 2.7	20.75 ± 1.5	27.20 ± 1.8

Table 9: Performance of methods over data streams with fuzzy task boundaries over all six datasets.

Methods / Datasets	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split CIFAR-100	Split mini-ImageNet
ER	79.74 ± 4.0	78.98 ± 0.5	76.45 ± 1.2	37.15 ± 1.6	21.99 ± 1.1	26.47 ± 2.3
ER + GMED	82.73 ± 2.6	78.91 ± 0.5	76.55 ± 1.0	40.57 ± 1.7	21.79 ± 1.9	28.20 ± 0.6
MIR	85.80 ± 1.9	**79.31 ± 0.7**	77.04 ± 1.2	38.70 ± 1.7	21.57 ± 1.4	25.83 ± 1.5
MIR + GMED	86.17 ± 1.7	79.26 ± 0.8	77.56 ± 1.1	41.22 ± 1.1	**22.16 ± 1.1**	26.86 ± 0.7
ER$_{aug}$	81.30 ± 2.0	77.71 ± 0.8	**80.31 ± 0.9**	47.97 ± 3.5	18.47 ± 2.0	31.75 ± 1.0
ER$_{aug}$ + GMED	**82.39 ± 3.7**	77.68 ± 0.8	80.30 ± 1.0	**51.38 ± 2.2**	18.63 ± 1.3	**31.83 ± 0.8**

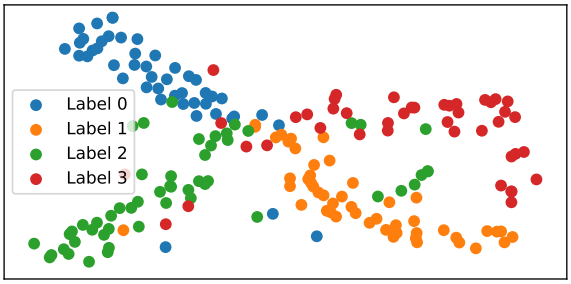


Figure 6: A t-SNE visualization of the editing performed on data examples. We use labels from the first two tasks in Split MNIST.

respectively. In Split CIFAR-10, Split CIFAR-100, and Split mini-ImageNet, each task consists of 10,000, 2,500, and 2,500 training examples. For tuning hyperparameters, we separate out 5% of the training set examples in the first three tasks as the validation set. For Rotated MNIST, we limit the number of testing examples to 1,000 per task; while for other datasets, we use the complete test sets.

F. Sensitivity to γ

Table 6 shows the performance of GMED under various decay rates of the editing stride γ. We find $\gamma = 1.0$ (*i.e.*, no decay) consistently outperforms performance when γ is less than 1. It implies it is not necessary to explicitly discourage the deviation of edited examples from original examples with the hyperparameter γ.

G. Comparison with ER+T

In addition to data augmentation over the replay memory (ER$_{aug}$), Buzzega et al. (2020) also studied a number of other tricks, such as exponential LR decay and balanced reservoir sampling. The integrated method, referred to as ER with Tricks (ER+T), achieved SoTA performance in a task-aware, non-online setup over a number of datasets. We adapt ER+T to an online task-free continual learning setup by discarding the tricks not compatible with the online task-free scenario (namely the Bias Control (BiC) trick) and build GMED upon it. Table 7 summarizes the results. However, find ER+T does not outperform ER$_{aug}$, *i.e.*, the tricks other than the data augmentation is not effective in an single-epoch online training setup, except Split MNIST. We further find ER+T+GMED outperforms or performs comparably with ER+T. The improvement is significant on Split MNIST and Split mini-ImageNet datasets.

H. GMED without Editing Stored Examples

Table 8 summarizes the result of discarding the editing performed after each time step: we replay the edited examples, but do not replace original examples in the memory with edited ones. The results are shown after ER+GMED w/o writeback. We tune a separate set of editing stride and regularization strength for the method. The results show overall inferior performance compared to ER+GMED, confirming the benefit of continuously editing examples stored in the memory by writing edited examples back to the memory.

Algorithm 3 Memory Editing with MIR (MIR+GMED)

Input: learning rate τ, edit stride α, regularization strength β, decay rate γ, model parameters θ
Receives: stream example (x_D, y_D)
Initialize: replay memory M

for $t = 1$ **to** T **do**

`// draw a random mini-batch for edit`
$(x_e, y_e) \sim M$
$k \leftarrow$ replayed_time(x_e, y_e)
$\ell_{\text{before}} \leftarrow \text{loss}(x_e, y_e, \theta_t)$
$\ell_{\text{stream}} \leftarrow \text{loss}(x_D, y_D, \theta_t)$

`// retrieve a separate mini-batch of examples for replay with MIR`
$(x_m, y_m) \leftarrow$ MIR-retrieve(M, θ)

`//update model parameters with stream examples, discarded later`
$\theta'_t \leftarrow \text{SGD}(\ell_{\text{stream}}, \theta_t, \tau)$

`//evaluate forgetting of memory examples`
$\ell_{\text{after}} \leftarrow \text{loss}(x_e, y_e, \theta'_t)$
$d \leftarrow \ell_{\text{after}} - \ell_{\text{before}}$

`//edit memory examples`
$x'_e \leftarrow x_e + \gamma^k \alpha \nabla_x (d - \beta \ell_{\text{before}})$
replace (x_e, y_e) with (x'_e, y_e) in M

`//replay examples retrieved by MIR`
$\ell = \text{loss}((x_m, y_m) \cup (x_D, y_D), \theta_t)$

$\theta_{t+1} \leftarrow \text{SGD}(\ell, \theta_t, \tau)$

reservoir_update(x_D, y_D, M)

end for

Algorithm 4 Memory Editing with GEM (GEM+GMED)

Input: learning rate τ, edit stride α, regularization strength β, decay rate γ, model parameters θ
Receives: stream example (x_D, y_D)
Initialize: replay memory M

for $t = 1$ **to** T **do**

`// draw a random mini-batch for edit`
$(x_e, y_e) \sim M$
$k \leftarrow$ replayed_time(x_e, y_e)
$\ell_{\text{before}} \leftarrow \text{loss}(x_e, y_e, \theta_t)$
$\ell_{\text{stream}} \leftarrow \text{loss}(x_D, y_D, \theta_t)$

`//update model parameters with stream examples, discarded later`
$\theta'_t \leftarrow \text{SGD}(\ell_{\text{stream}}, \theta_t, \tau)$

`//evaluate forgetting of memory examples`
$\ell_{\text{after}} \leftarrow \text{loss}(x_e, y_e, \theta'_t)$
$d \leftarrow \ell_{\text{after}} - \ell_{\text{before}}$

`//edit memory examples`
$x'_e \leftarrow x_e + \gamma^k \alpha \nabla_x (d - \beta \ell_{\text{before}})$
replace (x_e, y_e) with (x'_e, y_e) in M

`//Regularized model updates with GEM using the full memory`
GEM_regularized_update(x_D, y_D, M, θ)
`//Update the replay memory following GEM`
memory_update(x_D, y_D, M)

end for