Gradient Based Memory Editing for Task-Free Continual Learning

Xisen Jin [1] Junyi Du [1] Xiang Ren [1]

Abstract

Prior work on continual learning often operate in a "task-aware" manner, by assuming that the task boundaries and identifies of the data examples are known at all times. While in practice, it is rarely the case that such information are exposed to the methods (*i.e.*, thus called "task-free")–a setting that is relatively underexplored. Recent attempts on task-free continual learning build on previous memory replay methods and focus on developing memory construction and replay strategies such that model performance over previously seen examples can be best retained. In this paper, looking from a complementary angle, we propose a novel approach to "edit" memory examples so that the edited memory can better retain past performance when they are replayed. We use gradient updates to edit memory examples so that they are more likely to be "forgotten" in the future. Experiments on five benchmark datasets show the proposed method can be seamlessly combined with baselines to significantly improve the performance. Code has been released at `https://github.com/INK-USC/GMED`.

1. Introduction

Accumulating past knowledge and adapting to evolving environments are one of the key traits in human intelligence (McClelland et al., 1995). While contemporary deep neural networks have achieved impressive results in a range of machine learning tasks (sometimes at the level of human performance), they haven't yet to manifest the ability of continually learning over evolving data streams (Ratcliff, 1990). These models are observed to suffer from catastrophic forgetting (McCloskey & Cohen, 1989; Robins, 1995) when trained in an online fashion–*i.e.*, performance drop over previously seen examples during the sequential learning process. To this end, continual learning (CL) methods are developed to alleviate catastrophic forgetting issue when models are trained on non-stationary data streams (Goodfellow et al., 2013; Kirkpatrick et al., 2017).

Most existing work on continual learning pose an assumption that, when models are trained on a stream of tasks sequentially, the task specifications such as task boundaries or identities are exposed to the models (*i.e.*, incremental task assumption). These task-aware CL methods make explicit use of task information to avoid catastrophic forgetting issue, including consolidating important parameters on previous tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Nguyen et al., 2018), distilling knowledge from previous tasks (Li & Hoiem, 2017; Rannen et al., 2017), or separating task-specific model parameters (Rusu et al., 2016; Serrà et al., 2018). However, in practice, it is likely the case that the data comes in a sequential, non-stationary fashion without task identity or boundary—a setting that is commonly termed as task-free (online) continual learning (Aljundi et al., 2018).

To tackle this challenging setting, recent attempts on task-free CL methods have been made (Aljundi et al., 2018; Zeno et al., 2018; Lee et al., 2020). Regularization and architecture based approaches rely on inferring task boundaries or identities (Aljundi et al., 2018; Lee et al., 2020), or estimating importance of parameters online (Zeno et al., 2018) to consolidate or separate model parameters. On the other as memory-based CL methods have achieved strong results, which store a small set of previously seen instances in a fix-sized memory, and utilize them for replay (Robins, 1995; Rolnick et al., 2019) or regularization (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a).

The core problem in memory-based CL methods is about how to manage the memory instances and replay them given the limited computation budget (*e.g.*, number of instances to update each time) so that the model performance can be maximally preserved or enhanced. This problem has been studied mainly from two dimensions in previous work: 1) what instances to include in the memory from the data stream (Aljundi et al., 2019b; Rebuffi et al., 2017; Chaudhry et al., 2019b); and 2) which instances in the memory need to be replay at a certain point (Aljundi et al., 2019a).

In this paper, we provide a new angle to solving the memory management problem in task-free continual learning and

[1]Department of Computer Science, University of Southern California, Los Angeles, California, USA. Correspondence to: Xiang Ren <xiangren@usc.edu>.

The *4th* Lifelong Learning (LifelongML) Workshop, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

study how to make gradient updates on the memory examples. We develop a novel memory editing algorithm which are complementary to existing memory replay methods and data sampling strategies for memory update, and can cope with them to form different integrated approaches. It is challenging, as it asks for a principled and sound optimization objective of editing. Our proposed method, named Gradient-based Memory EDiting (**GMED**), edits stored examples with gradient-based updates so that they are more likely to be forgotten. Specifically, we estimate the forgetting of a stored example by its loss increase in the upcoming one online model update, which is differentiable regarding the store examples. We then perform gradient ascent on stored examples so that they are more likely to be forgotten.

Experiments show that our algorithm consistently improves over baselines on four out of five benchmark datasets under various memory sizes. Our ablation studies show the proposed editing mechanism outperforms alternative approaches such as random editing. Notably, our algorithm can be seamlessly combined with best performing baselines to further improve performance. The algorithm is general and can be coped with other models and datasets.

2. Related Works

Task-aware Continual Learning. Most of continual learning algorithms are studied under "task-aware" settings, where the model visits a sequence of clearly separated "tasks". A great portion of algorithms make explicit use of task boundaries (Kirkpatrick et al., 2017; Rusu et al., 2016; Lopez-Paz & Ranzato, 2017), by learning separate parameters for each task, or discourage changes of parameters that are important to old tasks. Existing continual learning algorithms can be summarized into three categories: regularization-based, architecture-based and data-based approaches. Regularization based approaches (Kirkpatrick et al., 2017; Zenke et al., 2017; Nguyen et al., 2018; Adel et al., 2020) discourage the change of parameters that are important to previous data. Architecture-based approaches (Rusu et al., 2016; Serrà et al., 2018; Li et al., 2019) allows expansion of model architecture to separate parameters for previous and current data. Data-based approaches (Robins, 1995; Shin et al., 2017; Lopez-Paz & Ranzato, 2017) replay or constrain model updates with real or synthetic examples.

Task-free Continual Learning. Recently, task-free continual learning (Aljundi et al., 2018) have drawn increasing interest, where we do not assume knowledge about task boundaries. To the best of our knowledge, only a handful number of regularization based (Zeno et al., 2018; Aljundi et al., 2018), architecture based (Lee et al., 2020), generative replay based (Rao et al., 2019) approaches are applicable in the task-free CL setting. Meanwhile, most memory

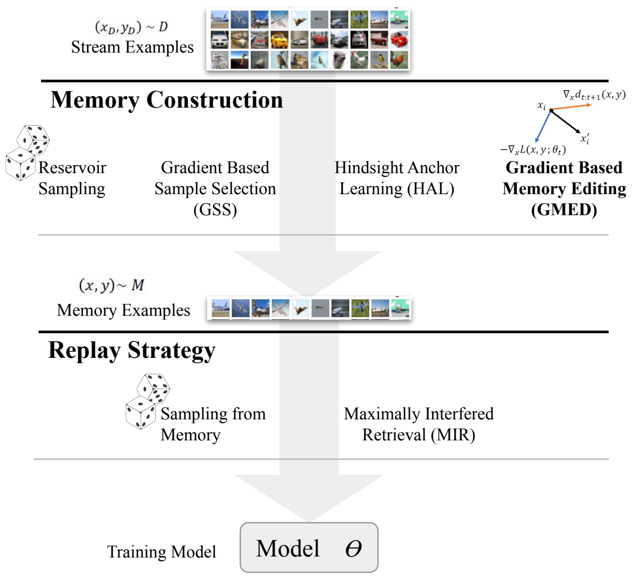

$(x_D, y_D) \sim D$
Stream Examples
Memory Construction
$\nabla_x d_{t:t+1}(x, y)$
x_i
x_i'
$-\nabla_x L(x, y; \theta_t)$
Reservoir Sampling
Gradient Based Sample Selection (GSS)
Hindsight Anchor Learning (HAL)
Gradient Based Memory Editing (GMED)
$(x, y) \sim M$
Memory Examples
Replay Strategy
Sampling from Memory
Maximally Interfered Retrieval (MIR)
Training Model
Model Θ

Figure 1. Categorization of memory-based methods on Task-free Continual Learning. Reservoir sampling and Sampling from Memory are the ways that Experience Replay (ER) uses to construct and replay memory respectively. In the recent line of works, Gradient based Sample Selection (GSS) and Hindsight Anchor Learning (HAL) explored ways to construct memory, while Maximally Interfering Retrieval (MIR) focused on replay strategy. Our method, Gradient-based Memory Editing (GMED) falls under the former category but provides a new angle to memory construction.

based continual learning algorithms are applicable to the task-free setting (Aljundi et al., 2019a;b). Memory-based CL algorithms store a subset of examples in a fix-sized replay memory and utilize them later at training to alleviate forgetting. Experience Replay (ER) (Robins, 1995) is a simple yet surprisingly powerful (Chaudhry et al., 2019b) approach in this category. The algorithm stores a subset of examples in the memory and regularly draw a small batch from the memory to retrain on them later at training. Recent research has studied online strategies to improve the performance gain when examples get replayed from two dimensions: in terms of *which examples to store*, and *which examples to replay*. Figure 1 shows an categorization of memory-based task-free continual learning.

In terms of deciding *which examples to store*, one solution is to randomly select examples to store with reservoir sampling (Vitter, 1985), which ensures each example in the stream has an equal probability of being stored in the memory. In addition to reservoir sampling, recent works also study storing prototypes (Rebuffi et al., 2017) or examples that contribute to the diversity of the stored examples, such as Gradient based Sample Selection (GSS) (Aljundi et al., 2019b). In terms of deciding *which examples to replay*, existing algorithms usually rely on random sampling from the replay memory. Maximally Interfering Retrieval (MIR) (Aljundi et al., 2019a) select examples with the largest estimated forgetting. In particular, a task-aware approach, Hindsight Anchor Learning (HAL) (Chaudhry et al., 2020), shares the same assumption that forgettable examples should be prioritized more. However, HAL only applies to task-aware settings and requires extra memory storage to keep track of the learned anchors.

3. Background

In this section we first present the problem formulation of task-free continual learning and then introduce preliminaries on memory-based continual learning methods.

3.1. Problem formulation

In task-free continual learning, we consider a (potentially infinite) stream of data examples D, which have non-stationary data distribution.At each time step t, the model receives one or a mini batch of examples (x_t, y_t) from the data stream D. For simplicity of discussion, here we assume that example (x_t, y_t) from D is generated by: first sampling a *latent* "task" $z \sim P(z;t)$, and then followed by sampling a data example from a joint data distribution $P(x, y|z;t)$ that is conditioned on task z, *i.e.*, $(x_t, y_t) \sim P(x, y|z;t)$. Here $P(z;t)$ is non-i.i.d and depends on time; and $P(x, y|z;t)$ also changes over time.

The goal of task-free online continual learning is to seek a classification model $f(x;\theta)$, parameterized by θ, over new example(s) (x, y) from the data stream D that minimizes a predefined loss $\ell(x, y;\theta)$ while not increasing the loss on previously seen examples. Such capability is evaluated by testing the model over a held-out set which consists of new examples that follows the task distribution of the stream D.

3.2. Memory-based CL Methods

In a nutshell, memory-based CL algorithms maintain a fix-sized replay memory M which is used for storing (subset of) previously seen examples (x_t, y_t) from the stream D. When the memory is full, the algorithm needs to either identify a memory examples (x, y) to be replaced by new example, or to discard the new example it just received. Following the same setup in previous memory-based CL methods, our experiments use reservoir sampling (Vitter, 1985) to determine how the memory will be updated with new examples received from stream D: every time the model receives a new example, it draws an integer j between 0 and N randomly, where N is the number of examples visited so far. If $j < |M|$ (*i.e.*, the memory size or budget), it replace the example at the j-th position in the memory with the new example; otherwise, this newly received example will be discarded. Reservoir sampling ensures at each time step each visited example is kept with an equal probability $|M|/N$.

At each time step t, the algorithm also needs to determine which of the memory examples will be used for replay. Same as previous methods, we randomly sample one or a mini-batch of examples (x, y) from the memory M. As an alternative replay strategy, MIR (Aljundi et al., 2019a) identifies a subset of memory examples based on a predefined optimization objective (*i.e,* perform one step of training on (x, y).), and then replays the selected examples.

4. Gradient Based Memory Editing

We propose Gradient based Memory Editing (GMED), a novel algorithm for updating the stored memory examples in an online fashion. We state our hypothesis about which examples may help to retain performance in Sec. 4.1. We then formulate an online optimization objective for example editing in Sec. 4.2. In Sec. 4.3 and 4.4, we introduce algorithmic details of GMED and its integration with MIR.

4.1. Hypothesis for Memory Editing

Given that there is no prior knowledge about the forthcoming examples in the data stream D, previous task-free CL methods usually impose (implicit) assumptions regarding what kinds of examples may improve model's test performance after replaying these examples. For example, GSS (Aljundi et al., 2019b) assumes that the diversity of memory examples contributes to the model performance; MIR (Aljundi et al., 2019a) and HAL (Chaudhry et al., 2020) assume that replaying examples that are likely to be "forgotten" can benefit the performance. Empirical study by Toneva et al. (2019) also show that there are constantly forgotten examples, which benefit overall performance when they get replayed compared to other examples.

Our work is based on a similar hypothesis that: *replaying examples that are likely to be forgotten by the current model helps retain its test performance*. However, rather than determining which memory examples will be replayed next, our work studies a complementary aspect on how to edit the memory examples, based on current model's behaviors, to make the examples more "forgettable".

Formally, we use the loss $\ell(x, y;\theta_t)$ to measure the model's performance over an arbitrary example (x, y) at the time step t, where θ_t denotes the model parameters at t. Suppose we train the model on D until the time step T, the "forgetting" measurement of an example (x, y) for the model at t, denoted by $d_{t:T}(x, y)$, is defined as the "loss increase" at time T compared to that at time t, shown as follows.

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

where θ_t is the model parameters at the current time step t. A larger $d_{t:T}(x, y)$ indicates that the example is more likely to be forgotten by the model at the end of training.

Our hypothesis assumes replaying memory examples that

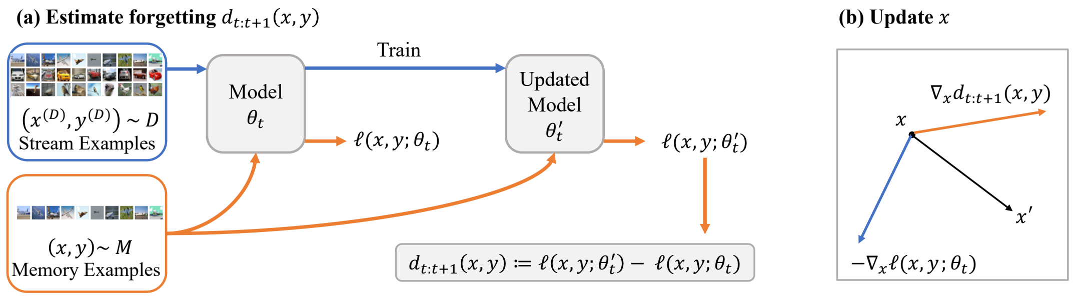

(a) Estimate forgetting $d_{t:t+1}(x,y)$
$(x^{(D)}, y^{(D)}) \sim D$
Stream Examples
Model
θ_t
Train
Updated
Model
θ_t'
$\ell(x,y;\theta_t)$
$\ell(x,y;\theta_t')$
$(x,y) \sim M$
Memory Examples
$d_{t:t+1}(x,y) \coloneqq \ell(x,y;\theta_t') - \ell(x,y;\theta_t)$
(b) Update x
$\nabla_x d_{t:t+1}(x,y)$
x
x'
$-\nabla_x \ell(x,y;\theta_t)$

Figure 2. Overview of the (a) forgetting estimation and (b) example editing in the proposed GMED method. See detailed formulations for forgetting estimation and example editing at Section 4.3.

have larger forgetting measurement helps model to generalize. The hypothesis is formally stated as follows.

Hypothesis 1. *Given a budget of C examples to replay at time t, in order to minimize the loss over new examples from task k (e.g., in test set), an ideal strategy is to replay the most forgettable examples, denoted as S_k, selected from the training examples of task k, denoted as D_k.*

Following Hypothesis 1, the set of examples S_k can be obtained by solving the following optimization problem.

$$S_k = \operatorname*{arg\,max}_{S_k \subseteq D_k, |S_k|=C} \sum_{(x,y)\in S_k} d_{t:T}(x,y), \tag{2}$$

where D_k is the training examples of the latent task k. Unfortunately, the sample selection problem in Eq. 2 cannot be solved in an online setting, even when we know task identities, as one can only evaluate the objective function in Eq. 2 at a future time step T. Therefore, approximations and modifications are necessary to enable online optimization.

4.2. Online Optimization for Memory Editing

We propose an optimization problem that is tractable in an online fashion, which shares the same goal of making memory examples used for replay more likely to be forgotten. We modify Eq. 2 where we (1) edit individual examples stored in memory instead of selecting the most forgettable examples from D_k as shown in Eq. 2, and (2) estimate the forgetting measure in an online fashion, and (3) relax the constraint $S_k \subseteq D_k$ for selecting examples.

Formally, suppose that the model is at the t-th time step and that (x, y) is an example from the a certain task k. In order to retain the performance on test examples from the same task k, we propose the following objective:

$$(x^*, y) = \operatorname*{arg\,max}_{(x,y)} \quad d_{t:t+1}(x,y) - \beta\ell(x,y;\theta_t), \tag{3}$$

where $d(\cdot)$ is the forgetting defined in Eq. 4 and $\ell(x, y; \theta_t)$ is the loss of the example (x, y) at the current time step t. β is a trade-off hyper-parameter deciding the regularization strength. The optimal (x^*, y) has the same label y as the original example (x, y).

Specifically, we discuss three main differences in Eq. 3 compared to Eq. 2 in the Hypothesis 1 as follows.

Editing instead of Selection. Eq. 2 describes a combinatorial optimization problem of selecting k most forgettable examples. In contrast, Eq. 3 describes an optimization problem over an continuous input space with the objective of maximizing forgetting of each individual example.

Estimating Forgetting Online. We maximize the forgetting of the example (x, y) in the upcoming one update at time step $t + 1$, noted as $d_{t:t+1}(x, y)$, instead of the forgetting when the model get evaluated, *i.e.*, $d_{t:T}(x, y)$. The former can be evaluated online efficiently without any overhead on replay memory.

Relaxed Constraints. Eq. 3 do not constrain $(x, y) \in D_k$. It allows (x^*, y) to be an arbitrary example in the input space without being a real example from D_k. In practice, (x, y) is initialized as different input examples, and we perform only one or a few gradient updates each time it is drawn from the memory for replay. We also add a regularization term $\beta\ell(x, y; \theta_t)$ to discourage the loss increase on the example. The editing is made conservative, so that the edited (x', y) is still likely to be an example from its original latent task.

The objective in Eq. 3 is differientiable with respect to x, allowing us to update x with gradient ascent. In the rest of this section, we introduce algorithmic details of GMED.

4.3. The GMED Algorithm

Given the hypothesis and the optimization formulations, we introduce the algorithmic details in GMED. We start by combining GMED with ER. It introduces an additional "editing" step before replaying examples drawn the memory.

For simplicity of discussion, we assume the batch size is 1, but the method also applies with a larger batch size. We assume at time step t the model receives a stream example $(x^{(D)}, y^{(D)})$ from the training stream D, and randomly draws a memory example (x, y) from the memory M. We first compute the forgetting (*i.e.,* loss increase) on the memory example (x, y) when the model performs one gradient update on parameters with the stream example $(x^{(D)}, y^{(D)})$.

$$\theta'_t = \theta_t - \nabla_\theta \ell(x^{(D)}, y^{(D)}; \theta_t); \quad (4)$$

$$d_{t:t+1}(x, y) = \ell(x, y; \theta'_t) - \ell(x, y; \theta_t), \quad (5)$$

where θ_t and θ'_t are model parameters before and after the gradient update respectively. Figure 2(a) visualize the steps to compute forgetting.

Following the optimization objective proposed in Eq. 2, we perform a gradient update on x to increase its forgetting, while using a regularization term to discourage the loss increase on the example at the current time step.

$$x' = x + \alpha \nabla_x [d_{t:t+1}(x, y) - \beta \ell(x, y; \theta_t)], \quad (6)$$

where α is a hyperparameter for the stride of the update. Figure 2(b) visualize the editing step.

The algorithm then discards the updated parameter θ'_t, and updates model parameters θ_t with the updated memory example (x', y) and the stream example $(x^{(D)}, y^{(D)})$, in a similar way to ER.

$$\theta_{t+1} = \theta_t - \nabla_\theta \ell((x', y) \cup (x^{(D)}, y^{(D)}); \theta_t). \quad (7)$$

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. Algorithm 1 summarize the proposed ER+GMED algorithm.

4.4. Hybrid Methods

GMED is studied from a complementary direction compared to most prior approaches. Therefore, we can combine GMED with existing memory-based CL algorithms without much effort. We illustrate the point by proposing a hybrid approach of GMED and MIR.

At each time step, MIR retrieves the most forgettable examples from the memory with the forgetting defined as Eq. 4. We do not edit the selected examples directly; instead, we additionally draw another random mini-batch from the memory to apply editing. The motivation is that examples drawn by MIR are already most forgettable ones; if we directly perform editing on them, we would fall into a loop that letting forgettable examples more forgettable, which is not desired.

5. Experiments

We compare the performance of GMED against state-of-the-art CL algorithms on five benchmark datasets. We introduce our experimental setup and discuss our results on comparisons with baselines and performance analysis.

Algorithm 1 Memory Editing with ER (ER+GMED)

```
Input: learning rate $\tau$, edit stride $\alpha$, regularization strength $\beta$,
model parameters $\theta$
Receives: stream example $(x^{(D)}, y^{(D)})$
Initialize: replay memory $M$
for $t = 1$ to $T$ do
    $(x, y) \sim M$
    $\ell_{\text{before}} \leftarrow \text{loss}(x, y, \theta_t)$
    $\ell_{\text{stream}} \leftarrow \text{loss}(x^{(D)}, y^{(D)}, \theta_t)$
    //update model parameters with stream
    examples, discarded later
    $\theta'_t \leftarrow \text{SGD}(\ell_{\text{stream}}, \theta_t, \tau)$
    //evaluate forgetting of memory examples
    $\ell_{\text{after}} \leftarrow \text{loss}(x, y, \theta'_t)$
    $d \leftarrow \ell_{\text{after}} - \ell_{\text{before}}$
    //edit memory examples
    $x' \leftarrow x + \alpha \nabla_x (d - \beta \ell_{\text{before}})$
    $\ell = \text{loss}((x', y) \cup (x^{(D)}, y^{(D)}), \theta_t)$
    $\theta_{t+1} \leftarrow \text{SGD}(\ell, \theta_t, \tau)$
    replace $(x, y)$ with $(x', y)$ in $M$
    reservoir_update$(x^{(D)}, y^{(D)}, M)$
end for
```

5.1. Datasets

We consider five public CL datasets in our experiments.

Split MNIST (Goodfellow et al., 2013) splits the MNIST (LeCun et al., 1998) dataset of handwritten digit classification into 5 disjoint subsets by their labels as different tasks. The dataset consists of 5 tasks, where each task consists of 1,000 training examples. The goal is to classify over all 10 digits when the training ends.

Permuted MNIST (Goodfellow et al., 2013) applies a fixed random pixel permutation to the MNIST dataset as different tasks. The dataset consists of 10 tasks with 1,000 training examples each. The goal is to classify over 10 digits without knowing the permutation applied.

Rotated MNIST (Lopez-Paz & Ranzato, 2017) applies a fixed image rotation between 0 to 180 degree to the MNIST dataset. At k-th task, the degree is randomly selected from the interval $[\frac{180(k-1)}{K}, \frac{180k}{K}]$, where $k \in \{1, 2, ..., K\}$ and K is the number of tasks. The dataset consists of 20 tasks with 1,000 examples each. The models classify over 10 digits without knowing the rotation when the training ends.

Split CIFAR-10 (Zenke et al., 2017) splits the CIFAR-10 (Krizhevsky, 2009) image classification dataset into 5 disjoint subsets by their labels. The dataset consists of 5 tasks, where each task consists of 10,000 training examples. The models classify over all 10 classes.

Split mini-ImageNet (Aljundi et al., 2019a) splits the mini-

Dataset / Hyper-param	Editing stride α	Regularization strength β
Split MNIST	5.0	0.01
Permuted MNIST	0.05	0.001
Rotated MNIST	1.0	0.01
Split CIFAR	0.05	0.001
Split mini-ImageNet	1.0	0.1

Table 1. Hyperparamters of the editing stride and the regularization strength selected for GMED.

ImageNet (Deng et al., 2009; Vinyals et al., 2016) classification dataset to 20 disjoint subsets by their labels. The dataset consists of 20 tasks, where each task consists of 1,250 examples in total from 5 classes. The models classify over all 100 classes.

We do not provide information about task identities or task boundaries to the model at both training and test time. Following definitions in prior literature (van de Ven & Tolias, 2019), our Split MNIST, Split CIFAR-10, and Split mini-ImageNet experiments are under the class-incremental setup, while Permuted MNIST and Rotated MNIST experiments are under the domain-incremental setup.

5.2. Compared Methods

For our methods, we report the performance of ER + GMED and MIR + GMED, where we build GMED upon ER or MIR as introduced in section 4.3. We compare with several memory based continual learning methods, but do not compare with regularization based approaches, as most of them are known to perform poorly in the class-incremental setup (van de Ven & Tolias, 2019). We include task-aware approaches for comparison, but it should be noted that these approaches make use of extra information compared to task-free approaches.

• **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., 2018; 2019a; 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., 2019a) 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., 2019b) 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.

Besides, we also report the following results for reference.

• **Fine tuning** performs online updates on model parameters without applying continual learning algorithms.

• **iid Online**. We randomly shuffle the data stream, so that the model visits an i.i.d. stream of examples.

• **iid Offline** is similar to iid-Online, but we allow multiple pass over the data.

We also consider two variants of memory editing strategies for ablation study:

• **Random Edit** is an ablation study for GMED. It edit examples to a random direction with a fixed stride. The stride is tuned by validation, similar to GMED.

• **Hindsight Edit** estimates example forgetting in a similar way to HAL. The model obtain temporal updated parameters θ'_t by training on a random mini-batch from the memory instead of stream examples.

5.3. Experiment Setup

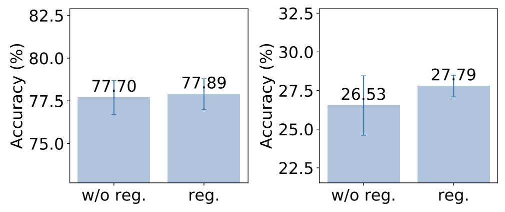
We mostly follow the training setup in (Aljundi et al., 2019a). For three MNIST datasets, we use a MLP classifier with 2

Methods / Datasets	Split MNIST	Permuted MNIST	Rotated MNIST	Split CIFAR-10	Split mini-ImageNet
Fine tuning	18.80 ± 0.6	66.34 ± 2.6	41.24 ± 1.5	18.49 ± 0.2	2.84 ± 0.4
iid online	85.99 ± 0.3	73.58 ± 1.5	81.30 ± 1.3	62.23 ± 1.5	17.53 ± 1.6
AGEM (Chaudhry et al., 2019a)	29.02 ± 5.3	72.17 ± 1.5	50.77 ± 1.9	18.49 ± 0.6	2.92 ± 0.3
GEM (Lopez-Paz & Ranzato, 2017)	87.18 ± 1.3	78.23 ± 1.2	76.49 ± 0.8	20.05 ± 1.4	11.27 ± 3.4
GSS-Greedy (Aljundi et al., 2019b)	84.16 ± 2.6	77.43 ± 1.4	73.66 ± 1.1	28.02 ± 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	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	21.18 ± 2.1
ER (Robins, 1995)	80.96 ± 2.3	79.69 ± 1.0	76.95 ± 1.7	33.34 ± 1.5	26.00 ± 1.0
MIR (Aljundi et al., 2019a)	84.88 ± 1.7	79.96 ± 1.3	78.30 ± 1.0	34.47 ± 2.0	25.01 ± 1.3
ER + GMED	$82.68^{**} \pm 2.1$	79.70 ± 1.1	$77.89^{*} \pm 0.9$	$35.01^{*} \pm 1.5$	$\mathbf{27.79^{**} \pm 0.7}$
MIR + GMED	$\mathbf{87.86^{**} \pm 1.1}$	$\mathbf{80.11^{*} \pm 1.2}$	$\mathbf{79.16^{**} \pm 0.9}$	$\mathbf{35.54 \pm 1.9}$	$26.29^{*} \pm 1.2$
iid offline (upper bound)	93.87 ± 0.5	87.40 ± 1.1	91.38 ± 0.7	75.17 ± 0.7	36.54 ± 1.4

Table 2. Mean and standard deviation of final accuracy(%) in 10 runs. For Split mini-ImageNet dataset, we set the memory size to 10,000 examples; we use 500 for other datasets. * and ** indicate significant improvement over the counterparts without GMED with p-values less than 0.05 and 10^{-3} respectively in single-tailed paired t-tests.

Dataset / Method	GMED	Random Edit	Hindsight Edit
Split MNIST	82.68 ± 2.1	80.13 ± 2.9	70.32 ± 3.9
Permuted MNIST	79.70 ± 1.1	78.85 ± 0.2	79.23 ± 0.6
Rotated MNIST	77.89 ± 0.9	76.40 ± 1.7	76.33 ± 1.1
Split CIFAR-10	35.01 ± 1.5	34.72 ± 1.6	33.08 ± 3.3
Split mini-ImgNet	27.79 ± 0.7	25.73 ± 2.0	27.27 ± 1.7

Table 3. We show the performance where we updates memory examples to a random direction (Random Edit), or estimate forgetting by one step of training on the replay memory following HAL (Hindsight Edit).



(a) Rotated MNIST (b) Split mini-ImgNet

Figure 3. Ablation study of the regularization term. We show the accuracy(%) when we apply GMED without regularization (reg.) and with regularization.

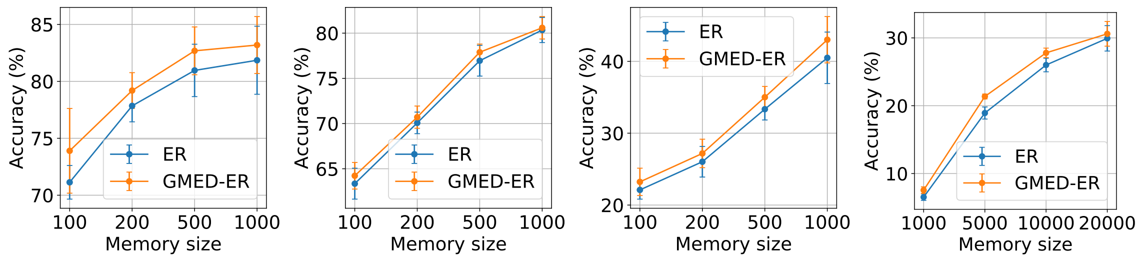
hidden layers with 400 hidden units each. For Split CIFAR-10 and Split mini-ImageNet datasets, we use a ResNet-18 classifier. We use SGD optimizer with a learning rate of 0.05 for MNIST and 0.1 for Split CIFAR-10 and Split mini-ImageNet datasets. We perform three steps of model parameter updates for each example we visit in the Split mini-ImageNet dataset, and one step for others, following (Aljundi et al., 2019a). Similarly, we perform three steps of memory editing in GMED in Split mini-ImageNet dataset, and one for others. We use a batch size of 10 throughout the experiment. By default, we set the size of replay memory as 10,000 for split mini-ImageNet, and 500 for all other datasets. We also report performance under different memory sizes.

Hyperparameter Selection for GMED. GMED introduces two hyperparameters: stride of the editing α, and the regularization strength β. Following (Chaudhry et al., 2019a), we tune the hyperparameters with only the training and validation set of first three tasks. The models are trained until convergence before they proceed to the next task. 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 hyperparameters for ER-GMED and apply the same hyperparameters on MIR+GMED. Table 1 show the optimal hyperparameters selected for each dataset.

5.4. Results and Performance Analysis

We report the final accuracy achieved by different methods in Table 2. We summarize the following findings.

Overall Performance. From the results, we see MIR + GMED achieves best performance among memory based continual learning algorithms. The improvement of GMED differs by datasets. ER+GMED clearly improves over ER by an absolute margin of 1.72%, 1.67%, and 1.79% accuracy respectively in Split MNIST, Split CIFAR-10, and Split mini-ImageNet datasets. On Rotated MNIST the improvement is less significant, with an absolute accuracy improvement of 0.94%, while we do not see a meaningful improvement on Permuted MNIST dataset. MIR+GMED improves performance on all the datasets. Similarly, the improvement is clear on Split MNIST, Split CIFAR-10, and Split mini-ImageNet with an absolute accuracy improvement of 2.98%, 1.07%, and 1.28%, and less signicifant on



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

Figure 4. Performance of ER and GMED-ER in different memory sizes. For mini-ImageNet dataset, we use memory sizes of 1,000, 5,000, 10,000, and 20,000 examples; for other datasets, we use 100, 200, 500, and 1,000.

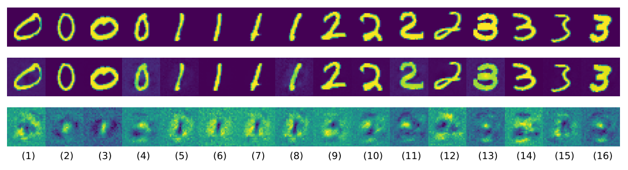


Figure 5. Visualization of the editing on examples from first two tasks in the Split MNIST dataset. The first and the second row shows original and the edited examples, while the third row shows the difference between the edited examples and the original ones.

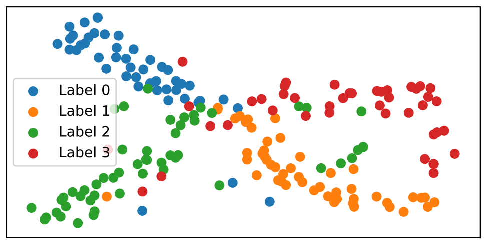


Figure 6. A t-SNE visualization of the editing performed on data examples. We use labels from the first two tasks in Split MNIST.

Rotated MNIST and Permuted MNIST.

Comparison with Random and Hindsight Editing. Table 3 compares between GMED, random editing and hindsight editing. We see ER+Random Edit performs worse than ER + GMED in all cases. However, we see ER+Random Edit outperforms ER on split CIFAR-10. We cojecture the reason is that the random editing alleviates the overfitting to memory examples. We also observe GMED outperforms ER + Hindsight Editing in all cases.

Effect of the Regularization Term Figure 3 compare the performance between the tuned regularization strength β and 0 in two the Rotated MNIST and split mini-ImageNet datasets. We see the regularization term improves the performance on two datasets.

Performance Under Various Memory Sizes. Figure 4 show the performance under various memory sizes. We see in Split MNIST, Rotated MNIST, Split CIFAR-10 and Split mini-ImageNet, the improvement of ER+GMED over ER is consistent under various memory sizes.

5.5. Case Study and Discussion

Visualization of Edited Examples. Figure 5 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.

Visualization of Editing Directions. 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.

6. Conclusion

In this paper, we propose Gradient based Memory Editing for task-free continual learning. The approach estimates forgetting of stored examples online and edit them so that they are more likely to be forgotten in upcoming updates. Experiments on benchmark datasets show our method can be combined with existing approaches to significant improve over baselines on several benchmark datasets. Our analysis further show the method is robust under various memory sizes, and outperforms alternative editing methods.

References

Adel, T., Zhao, H., and Turner, R. E. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hklso24Kwr`.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. In *CVPR*, 2018.

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., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019b.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019a. URL `https://openreview.net/forum?id=Hkf2_sC5FX`.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. On tiny episodic memories in continual learning. 2019b.

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.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 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.

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.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

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`.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *ArXiv*, abs/1904.00310, 2019.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *NIPS*, 2017.

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

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102 3:419–457, 1995.

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

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkQqq0gRb`.

Rannen, A., Aljundi, R., Blaschko, M. B., and Tuytelaars, T. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1320–1328, 2017.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pp. 7645–7655, 2019.

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

Rebuffi, S.-A., Kolesnikov, A. I., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017.

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 *NeurIPS*, 2019.

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 *ICML*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *NIPS*, 2017.

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 *International Conference on Learning Representations*, 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. P., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *NIPS*, 2016.

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

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3987–3995. JMLR. org, 2017.

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