Gradient-based Editing of Memory Examples for Online Task-free Continual Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We explore task-free continual learning (CL), in which a model is trained to avoid catastrophic forgetting in the absence of explicit task boundaries or identities. Among many efforts on task-free CL, a notable family of approaches are memory-based that store and replay a subset of training examples. However, the utility of stored seen examples may diminish over time since CL models are continually updated. 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 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 validate the effectiveness of GMED, and our best method significantly outperforms baselines and previous state-of-the-art on five out of six datasets[1].

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 [33, 20, 32]. Here, we study online *task-free* CL [3], where task identifiers and boundaries are absent from the data stream. This setting reflects many real-world data streams [25, 6] and offers a challenging testbed for online CL research.

Memory-based methods, a prominent class of approaches used for task-free continual learning, store a small number of training examples (from the data stream) in a memory and replay them at the later training iterations [33, 34]. Existing methods operate over the original examples in the data-stream and focus on identifying samples to populate the memory [4, 9] and finding samples in the memory to be replayed [2]. However, for continually updating models, using stored-seen examples in their original form, may lead to diminishing utility over time — i.e., model may gradually memorize the stored examples after runs of replay, as the memory refreshes slowly. An alternate approach is to use generative models to create samples that suffers more from forgetting such as in GEN-MIR in [2]. In practice, training the generator network with limited data is challenging and leads to low-quality generated examples. Further, in the online learning setup, the generative model itself suffers from forgetting. As such, generative models perform worse than their memory counter-parts.

In this paper, we present a novel memory-based CL framework, Gradient based Memory EDiting (**GMED**), which looks to directly "edit" (via a small gradient update) examples stored in the replay memory. These edited examples are stored (replacing their unedited counterparts), replayed, and

[1] Code has been uploaded in supplementary materials and will be released.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

further edited, thereby making the future replays more effective in overcoming catastrophic forgetting. Since no explicit generative model is involved, GMED approach retains the advantages of memory-based methods and is straightforward to train only inducing a small computation overhead.

The main consideration in allowing "editing" via a gradient update is the choice of the optimization objective. In light of recent work on designing alternative replay strategies [2, 40, 7], we hypothesize that "interfering" examples (*i.e.*, past examples that suffer from increased loss) should be prioritized for replay. For a particular stored example, GMED finds a small update over the example ("edit") such that the resulting edited example yields the most increase in loss when replayed. GMED additionally penalizes the loss increase in the edited example to enforce the proximity of the edited example to the original sample, so that the edited examples stay in-distribution. As a result, replaying these edited examples is more effective in overcoming catastrophic forgetting. 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 [26, 2, 5].

We demonstrate the effectiveness of GMED with a comprehensive set of experiments over six benchmark datasets. In general, combining GMED with existing memory-based approaches results in consistent and statistically significant improvements with our single best method establishing a new state-of-art performance on five datasets. Our ablative investigations reveal 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.

To summarize, our contributions are two-fold: (i) we introduce GMED, a modular framework for task-free online continual learning, to edit stored examples and make them more effective in alleviating catastrophic forgetting (ii) we perform intensive set of experiments to test the performance of GMED under various datasets, parameter setups (*e.g.*, memory size) and competiting baseline objectives.

2 Related Work

Continual Learning studies the problem of learning from a data stream with changing data distributions over time [23, 20]. A major bottleneck towards this goal is the phenomenon of catastrophic forgetting [33] 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 [19, 43, 30, 1], separating parameters for previous and new data [36, 37, 24], replaying examples from memory or a generative model [33, 38, 26], meta-learning [18]. 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 [3] is a specific formulation of the continual learning where the task boundaries and identities are not available to the model. Due to its broader applicability to real-world data-streams, a number of algorithms have been adapted to the task-free setup [44, 2, 14, 15, 22]. 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) [4], and *replaying examples with larger estimated "interference"* as in Maximally Interfered Retrieval (MIR) with experience replay [2]. In contrast , GMED is used in conjunction with memory-based approaches and explicitly searches for an edited example which is optimized to be more "effective" for replay.

Replay Example Construction. Moving away from replaying real examples, a line of works on *deep generative replay* [38, 17, 35] generates synthetic examples to replay with a generative model trained online. GEN-MIR [2] is further trained to generate examples that would suffer more from interference for replay. However, training a generative network is challenging, even more so in the online continual learning setup where the streaming examples are encountered only once leading to poorly generated examples. Moreover, the forgetting of generative networks themselves cannot be perfectly mitigated. As a result, these methods generally perform worse than their memory-based counter-parts. Instead of generating a new example via a generator network, GMED uses gradient updates to directly edit the stored example thereby retaining advantages of memory-based techniques while creating new samples.

Novel examples can also be constructed via *data augmentation* (*e.g.* random crop, horizontal flip) to help reduce over-fitting over the small replay memory [5]. Unlike GMED, these data-augmentations are usually pre-defined and cannot adapt to the learning pattern of the model. Constructing edited

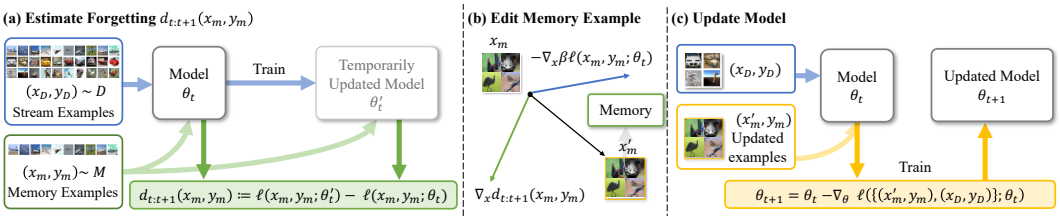


Figure 1: **A schematic of GMED framework**. (a) Given an example from the data-stream (x_D, y_D) at time t, the model randomly draws an example from the memory (x_m, y_m) and estimates "interference" after one-step roll-out (Eq. 2). (b) The example drawn from the memory is then updated using our proposed editing objective (Eq. 3) 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) (Eq. 4).

examples has also been used for adversarial robustness [39, 31]. The key difference lies in the optimization objective: while adversarial construction focuses on finding mis-classified examples [13, 28], GMED aims at constructing "interfered" examples that would suffer from loss increase in future training steps.

3 Method

We introduce the formulation of online task-free continual learning (Sec. 3.1) and then present our gradient-based memory editing method GMED (Sec. 3.2) and detail how GMED can be integrated with experience replay (Sec 3.3) and other memory-based CL algorithms (Sec. 3.4).

3.1 Problem Formulation

In continual learning (CL), we consider a (potentially infinite) stream D of labeled examples (x, y), 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 a mini-batch of examples) received by the model at time-step t in 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 conditional data distribution $p(x, y|z)$, – *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 [19, 43], this work focuses on ***online* task-free CL** setup [8, 9]. It simulates a practical setting where models must perform online update on every incoming example, without accumulating examples within a task.

Following above definitions, our goal is to learn a classification model $f(x; \theta)$ on the data stream D that can preserve its performance over all the tasks in D. At time step t during the training process, the model is updated to minimize a predefined empirical loss $\ell(x_t, y_t; \theta)$ on the newly received example (x_t, y_t), without increasing the loss on the previously visited examples (before t). Specifically, let $p_c(x, y; T)$ denotes the distribution of the examples visited until the time step T. We look to minimize the expected loss $\mathbb{E}_{p_c(x,y;T)} \ell(x, y; \theta)$ – *i.e.*, retaining performance on tasks encountered before T.

3.2 Gradient based Memory Editing (GMED)

In online task-free continual learning, examples visited earlier cannot be accessed (revisited) and thus computing the loss over all the visited examples (in D) is not possible. To deal with this challenge, memory-based CL algorithms store and (continuously) maintain a set of visited examples in a fixed-size memory and use them for replay or regularization in the future training steps [33, 26]. For instance, in Experience Replay (ER) [33] the examples are randomly sampled from the memory for replay; whereas recent works explore more sophisticated replay strategies, such as Maximally Interfered Retrieval with experience replay (ER-MIR) [2] where memory examples which interfere the most with the newly visited example are selected for replay. However, these algorithms only train over samples drawn from a small replay memory in their original form. As such, the utility of the stored examples could diminish over time as the model could potentially memorize these examples if the memory examples are refreshed in a slow pace – which is found often the case [5].

We address the above limitation in our proposed approach Gradient based Memory Editing (GMED) by allowing examples stored in the memory to be edited in the continuous input space, illustrated in Figure 1. The editing step is guided by an optimization objective instead of being pre-defined and 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.

Editing Objective. Clearly, the most crucial step involved in GMED is identifying "*how*" should the stored examples in the memory be edited to reduce catastrophic forgetting of early examples. If (x_m, y_m) denotes an example drawn from the memory M, 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 [39, 13, 31]. In the context of online continual learning, we employ a similar hypothesis as previous works [2, 40, 7] that examples that forgotten by models should be prioritized for replay. With this objectives of editing in place, we detail the process of incorporating GMED with Experience Replay (ER).

Algorithm 1: Gradient 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

3.3 The GMED Algorithm with ER

Algorithm 1 summarizes the process and Figure 1 provides a schematic of the steps involved in GMED. At time step t, the model receives an stream example (x_D, y_D) from the training stream D, and randomly draws a memory example (x_m^k, y_m) from the memory M, which we assume has already been drawn for replay for k times. We first compute the "interference" (*i.e.,* loss increase) on the memory example (x_m^k, y_m) when the model performs one gradient update on parameters with the stream example (x_D, y_D).

$$\theta'_t = \theta_t - \nabla_\theta \ell(x_D, y_D; \theta_t); \tag{1}$$

$$d_{t:t+1}(x_m^k, y_m) = \ell(x_m^k, y_m; \theta'_t) - \ell(x_m^k, y_m; \theta_t), \tag{2}$$

where θ_t and θ'_t are model parameters before and after the gradient update respectively. Then, we perform a step of gradient update on (x_m^k, y_m^k) by maximizing its "loss increase" in the next one step of training, while using a regularization term $\ell(x_m, y_m; \theta_t)$ to penalize the loss increase evaluated with the current model checkpoint. The iterative update is written as,

$$x_m^{k+1} \leftarrow x_m^k + \gamma^k \alpha \nabla_x [d_{t:t+1}(x_m^k, y_m) - \beta \ell(x_m^k, y_m; \theta_t)], \tag{3}$$

where γ is a decay factor of edit performed on the model. A decay factor γ less than 1.0 could effectively prevent x_m^k from drastically deviating from their original state, while $\gamma = 1.0$ indicates no decay. The hyper-parameter α controls the overall stride of the edit and is tuned with first three tasks together with the regularization strength β.

Following Eq. 3, we perform a gradient update on x to increase its "interference". The algorithm then discards θ'_t, and updates model parameters θ_t using the edited memory example (x_m^{k+1}, y_m) and the stream example (x_D, y_D), in a similar way to ER.

$$\theta_{t+1} = \theta_t - \nabla_\theta \ell(\{(x_m^{k+1}, y_m), (x_D, y_D)\}; \theta_t). \tag{4}$$

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.

3.4 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$_{aug}$) [5], ER-MIR [2] and GEM [26] 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 ER-MIR (denoted henceforth as MIR for brevity), recall that MIR retrieves and then replays the most "interfering" examples in the memory at each time-step. Thus, making GMED edits on the MIR-retrieved examples may induce a loop of further increasing "interference" of the examples that are already the most "interfering" 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[2]. This random sampling may help prevent GMED editing from intensifying potential biases created from the MIR retrieval process (*e.g.*, retrieved examples are edited and thus become more interfered). Similarly, For GEM+GMED, we also apply GMED to edit a mini-batch of randomly sampled examples.

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. We leave more sophisticated integration of GMED to existing CL algorithms *e.g.* by optimizing the retrieval and editing operations jointly to future work.

4 Experiments

Our experiments address the following research questions: (i) what are the gains obtained by integrating GMED with existing memory-based CL algorithms and how these gains compare across datasets, methods and memory sizes? (ii) how useful are GMED edits in alleviating catastrophic forgetting, and what part of it can be attributed to the design of the editing objective function? (iii) what role do the various components and parameters in the GMED play and how they affect the performance. 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 comparing across memory-based algorithms (Sec. 4.3), validate the effectiveness of GMED-edits and the editing objective in mitigating catastrophic forgetting (Sec. 4.4), followed by ablative study over its components (Sec. 4.5).

4.1 Datasets

We use six public CL datasets in our experiments. **Split / Permuted / Rotated MNIST** are constructed from the MNIST [21] dataset which contains images of handwritten digits. Split MNIST [12] 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 [12] 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 [26] 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 [2]. We also employ **Split CIFAR-10 and Split CIFAR-100**, which 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** [2] splits the mini-ImageNet [10, 42] dataset into 20 disjoint subsets based on their labels. The models classify over all 100 classes.

Following the taxonomy of [41], 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. We note that our results are not comparable to works that employ a different setup over the same dataset (*e.g.*, results on domain-incremental Split CIFAR-100 are not comparable).

4.2 Compared Methods

We compare against several task-free memory based continual learning methods namely, Experience Replay (ER) [33], Averaged Gradient Episodic Memory (AGEM) [8], Gradient based Sample

[2]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 (%) for non-model-expansion-based approaches on 6 datasets**. For Split mini-ImageNet and Split CIFAR-100 datasets, we set the memory size to 10,000 and 5,000 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. We report results in 20 runs for GEM, ER, MIR, and ER$_{aug}$ and their GMED-integrated versions, and 10 runs for others. † over "previous SOTA" results indicates that the best GMED method (**bolded** for each dataset) outperforms the previous SOTA with statistically significant improvement ($p < 0.1$).

Methods / Datasets	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 [8]	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 [4]	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 [44]	13.54 ± 5.1	19.38 ± 3.0	77.94 ± 0.9	18.23 ± 0.5	3.11 ± 0.2	24.71 ± 0.8
ER [33]	81.07 ± 2.5	78.65 ± 0.7	76.71 ± 1.6	33.30 ± 3.9	20.11 ± 1.2	25.92 ± 1.2
ER + GMED	$82.67^{**} \pm 1.9$	78.86 ± 0.7	$77.09^{*} \pm 1.3$	$34.84^{**} \pm 2.2$	$20.93^{*} \pm 1.6$	$27.27^{**} \pm 1.8$
MIR [2]	85.72 ± 1.2	79.13 ± 0.7	77.50 ± 1.6	34.42 ± 2.4	20.02 ± 1.7	25.21 ± 2.2
MIR + GMED	$\mathbf{86.52}^{**} \pm \mathbf{1.4}$	$\mathbf{79.25 \pm 0.8}$	$79.08^{**} \pm 0.8$	$36.17^{*} \pm 2.5$	$\mathbf{21.22}^{**} \pm \mathbf{1.0}$	$26.50^{**} \pm 1.3$
ER$_{aug}$ [5]	80.14 ± 3.2	78.11 ± 0.7	80.04 ± 1.3	46.29 ± 2.7	18.32 ± 1.9	30.77 ± 2.2
ER$_{aug}$ + GMED	$82.21^{**} \pm 2.9$	78.13 ± 0.6	$\mathbf{80.61}^{*} \pm \mathbf{1.2}$	$\mathbf{47.47}^{*} \pm \mathbf{3.2}$	19.60 ± 1.5	$\mathbf{31.81}^{*} \pm \mathbf{1.3}$
Previous SOTA	$85.72^{\dagger} \pm 1.2$	79.23 ± 0.7	$80.04^{\dagger} \pm 1.3$	$46.29^{\dagger} \pm 2.7$	$20.02^{\dagger} \pm 1.7$	$30.77^{\dagger} \pm 2.2$
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

Selection (GSS) [4], and Maximally Interfering Retrieval (MIR) [2]. We omit the generative replay method GEN-MIR proposed together in [2] as it underperforms their memory-based counterparts even on simple datasets such as Split MNIST.

We also compare with data augmentation [5] 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) [44], Neural Dirichlet Process Mixture Model (CN-DPM) [22], Progressive Networks (Prog.NN) [36], Compositional Lifelong Learning [29], Graident Episodic Memory (GEM) [26] and Hindsight Anchor Learning (HAL) [7] respectively.

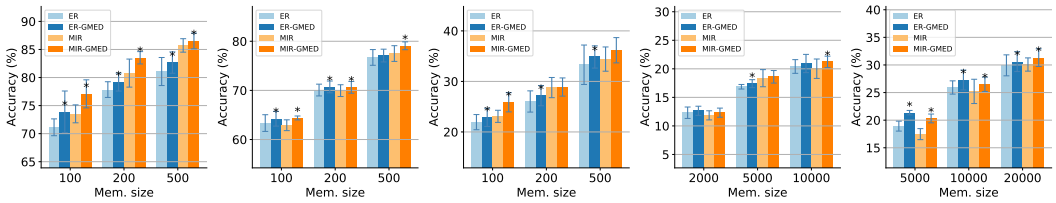
Finally, we report three baseline models: (i) Fine Tuning, where no continual learning algorithms are used for online updates to model parameters, (ii) iid Online, where we randomly shuffle the data stream, so that the model visits an i.i.d. stream of examples, and (iii) iid Offline, where multiple passes over the dataset is allowed. Appendix A provides 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.

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 [8], 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-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 [2]: 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 with three times less feature maps across all layers. See Appendix C for more details.

4.3 Performance Across Datasets

We summarize the results obtained by integrating GMED with different CL algorithms. In Table 1, we report the final accuracy and the standard deviation and make the following key observations.

Effectiveness of Memory Editing. As can be seen in Table 1, GMED significantly improves performance on 5 datasets when built upon ER and MIR respectively. 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 5 datasets. This indicates that the benefits induced by GMED go beyond regularization effects used to mitigate over-fitting.



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

Figure 2: **Performance of ER, GMED+ER, MIR, and GMED+MIR with 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 with $p < 0.05$.

Comparison across CL methods. From Table 1, we find MIR+GMED achieves the best performance on Split MNIST, Permuted MNIST, and Split CIFAR-100 datasets; while on Rotated MNIST, Split CIFAR-10 and Split mini-ImageNet dataset, ER$_{aug}$+GMED achieves the best performance. Performance of the best performing GMED method could significantly improve over previous SOTA on five datasets.

We further compare with an non-memory based CL approach, CN-DPM [22], which employs a generative model, a dynamically expanding classifier, and utilizes a short-term memory (STM). Following the setup in [16], we set the memory size for GMED so that two methods introduces the same amount of the overhead. Table 2 shows 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 *task-aware* model expansion approaches such as Prog. NN and the recently proposed compositional model expansion (CompCL) with a smaller memory overhead.

Table 2: **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 are taken from the respective papers.

Method	Split MNIST		Split CIFAR-10		Split CIFAR-100	
	Acc.	#. Mem	Acc.	#. Mem	Acc.	#. Mem
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
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

Performance under Various Memory Sizes. Figure 2 shows the performance of ER, ER+GMED, MIR, and MIR+GMED under various memory sizes. The improvement on Split MNIST and Split mini-ImageNet are significant with $p < 0.05$ over all memory size setups. On Rotated MNSIT and 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 setups.

Table 3: **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	Split CIFAR-10	Split mini-ImageNet
Vanilla	21.53 ± 0.1	20.69 ± 2.4	3.05 ± 0.6
ER	79.74 ± 4.0	37.15 ± 1.6	26.47 ± 2.3
MIR	84.80 ± 1.9	38.70 ± 1.7	25.83 ± 1.5
ER$_{aug}$	81.30 ± 2.0	47.97 ± 3.5	30.75 ± 1.0
ER + GMED	$82.73^* \pm 2.6$	$40.57^* \pm 1.7$	$28.20^* \pm 0.6$
MIR + GMED	$\mathbf{86.17 \pm 1.7}$	$41.22^* \pm 1.1$	$26.86^* \pm 0.7$
ER$_{aug}$ + GMED	$82.39^* \pm 3.7$	$\mathbf{51.38^* \pm 2.2}$	$\mathbf{31.83 \pm 0.8}$

Performance on Data Streams with Fuzzy Task Boundaries. The experiments in Table 1 assume a clear task boundary. In Table 3, we report the result of using data-streams with fuzzy task boundaries on four datasets. We leave the complete results in Table 9 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 in three datasets across ER, MIR and ER$_{aug}$.

Effectiveness of GMED in Task-Aware Setting. We additionally compare performance of other task-aware approaches (HAL, GEM, GEM+GMED) in Appendix H.

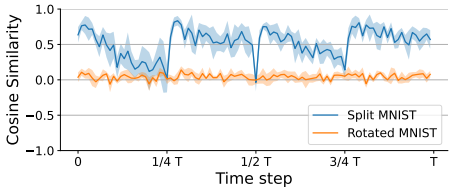


Figure 3: Cosine similarity between the optimal editing direction and direction from ER+GMED over training. The averaged similarity is 0.523 ± 0.014 and 0.035 ± 0.009.

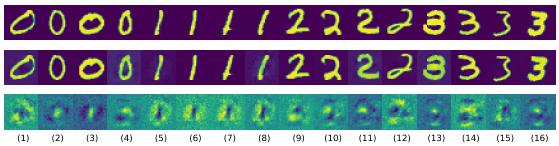


Figure 4: Visualization of the edited examples in Split MNIST in the ER-GMED approach. The first two rows show examples before and after editing, and the third row shows the differences.

4.4 Ablation to Study the Effect of Memory-Editing

We present a set of experiments to validate that the gains obtained through the memory-editing step of GMED are indeed helpful towards alleviating catastrophic forgetting. Further, we show that these gains are distinct from those obtained through random perturbations or simple regularization effects.

Comparison to Optimal Editing Direction. At a particular time-step t, GMED identifies the edit direction using the stream example (x_D, y_D) and the memory sample (x_m, y_m). To validate whether the edits proposed by GMED are indeed helpful, we compare GMED-edit to an "Optimal Edit" that minimizes the loss increase ("interference") of all early training examples in one future time step.

To compute this Optimal Edit, we note that the total loss increase over all previously encountered examples would be $d^{1:t}_{t:t+1} = \sum_{i=1}^{t} d_{t:t+1}(x_i, y_i) = \sum_{i=1}^{t} [\ell(x_i, y_i; \theta_{t+1}) - \ell(x_i, y_i; \theta_t)]$, where $\theta_{t+1} = \theta_t - \nabla_\theta \ell(x_m, y_m; \theta_t) - \nabla_\theta \ell(x_D, y_D; \theta_t)$ is the model parameters after the training update on (x_D, y_D) and (x_m, y_m). The Optimal Edit direction for the memory example x_m would be the gradient of $d^{1:t}_{t:t+1}$ w.r.t. x_m. Computing such optimal edits requires access to early training examples and is not practical in an online continual learning setup; we present it only for ablative purposes.

Figure 3 shows the cosine similarity of update directions (*i.e.*, the gradient of x_m) between GMED and the optimal editing strategy over Split MNIST and Rotated MNIST datasets. The averaged similarity over time is 0.523 ± 0.014, 0.035 ± 0.009 on two datasets, averaged across 10 runs. Recall that random editing has an expectation of zero similarity. On Split MNIST dataset, where the improvement is the most significant, we notice a high similarity between the update directions of GMED and optimal editing. On Rotated MNIST where the improvement is less significant, the similarity is still positive on average but is lower than Split MNIST. It implies GMED-edits are generally consistent with explicitly reducing forgetting, but there is still space to improve.

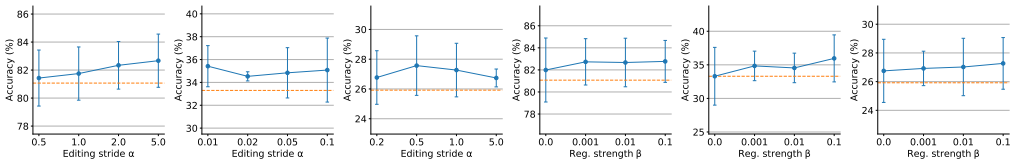
Comparison with alternative Editing Objectives. While GMED objective editing correlates with that of Optimal Edit, we further validate the choice of the objective function and consider two alternatives to the proposed editing objective (Eq. 3): (i) Random Edit: memory examples are updated in a random direction with a fixed stride; (ii) Adversarial Edit: following the literature of adversarial example construction [13], the edit increases the loss of memory by following the gradient sgn $\nabla_x \ell(x_m, y_m; \theta)$. We report the comparison in Table 4.

Table 4: **Alternative memory editing objectives or simply increasing the number of replayed examples**. 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	Rotated MNIST	Split CIFAR-10	Split mini-ImageNet
ER	76.71 ± 1.6	33.30 ± 3.9	25.92 ± 1.2
ER + Random Edit	76.42 ± 1.3	32.26 ± 1.9	26.50 ± 1.3
ER + Adv. Edit	76.13 ± 1.5	31.69 ± 0.8	26.09 ± 1.6
ER + GMED	**77.50 ± 1.6**	**34.84* ± 2.2**	**27.27* ± 1.8**
MIR+Extra 1 batch	78.07 ± 1.1	33.81 ± 2.3	24.94 ± 1.5
MIR+Extra 2 batches	77.10 ± 1.4	32.36 ± 2.8	24.65 ± 1.5
MIR	77.50 ± 1.6	34.42 ± 2.4	25.21 ± 2.2
MIR + Random Edit	77.19 ± 1.0	35.39 ± 3.0	24.86 ± 0.7
MIR + Adv. Edit	78.06 ± 1.5	35.79 ± 0.4	25.48 ± 1.3
MIR + GMED	**79.08* ± 0.8**	**36.17 ± 2.5**	**26.29* ± 1.2**

We notice ER+Random Edit outperforms ER on split mini-ImageNet, which indicates adding random noise to memory examples helps in regularization. Even so, GMED-edits are as good or outperforms both random and adversarial edits across multiple datasets and models. This validates our choice of the editing objective.

Comparison to Increasing the Number of Replayed Examples. Recall that in MIR-GMED, we sample two independent subset of memory examples to perform editing and replay (Sec. 3.4). For a fairer comparison, we replay one (or more) extra subset of examples. Table 4 shows the performance does not improve as replaying more examples. We hypothesize that the performance is bottle-necked by the size of the memory and not the number of examples replayed.



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

Figure 5: **Parameter sensitivity analysis** of the editing stride α (a,b,c) and the regularization strength β (d,e,f) for ER+GMED across various datasets. The dashed horizontal line indicates the performance of ER.

Case study on Edited Memory Examples. In Figure 4, we show visualizations of editing of memory examples. The examples are drawn from first two task ($0/1$, $2/3$) in the Split MNIST dataset using ER+GMED. The first and second row show the original and edited examples, noted as x_{before} and x_{after} respectively. The third row shows the difference between two $\Delta x = x_{\text{after}} - x_{\text{before}}$. While there is no significant visual differences between original and edited examples, in the difference Δx, we find examples with exaggerated contours (*e.g.* examples (1) and (12)) and blurring (*e.g.* examples (2), (3), (5), and (6)). Intuitively, ambiguous examples are exaggerated while typical examples are blurred. Our visualizations supports this intuition: examples (1) and (12) are not typically written digits, while examples (2), (3), (5), and (6) are typical. Appendix D provides a t-SNE [27] plot of the edit vector.

4.5 Analysis on the GMED Framework

Here, we analyze and ablate the various components used in the GMED framework and their effect on the final performance.

Hyper-parameter Sensitivity. In Figure 5 we plot the sensitivity of the performance of GMED with respect to two hyper-parameters: the editing stride α and the regularization strength β (Eq. 3). Clearly, ER+GMED outperforms ER over a broad range of α and β setups. Furthermore, in Figure 5 (d,e,f), better performance for non-zero β confirms the benefit of the regularization term. Recall that in our main experiments we tune the hyper-parameters α and β with only first three tasks; we note the chosen hyperparameters improve the performance despite they are not always optimal ones.

Computational Efficiency. We analyze the *additional* forward and backward computation required by ER+GMED and MIR. Compared to ER, ER+GMED adds 3 forward and 1 backward passes to estimate loss increase, and 1 backward pass to update the example. In comparison, MIR adds 3 forward and 1 backward passes with 2 of the forward passes are over a larger set of retrieval candidates. In our experiments, we found GMED has similar training time cost as MIR. In Appendix B, we report the wall-clock time, and observe the run-time of ER+GMED is 1.5 times of ER.

Increasing Steps of Editing. For experiments in Table 1, we performed one editing step over the sampled memory example x_m at time step t. In general, we can increase the number of editing steps. The direction for the edit is computed at each step, which makes the approach different from increasing the editing stride (α). Table 5 indicates that 3-step and 5-step edits in general don't lead to significant improvement in performance while incurring additional computational cost. As such, we use only 1-edit step across all our experiments.

Table 5: **Editing examples for multiple gradient steps in ER+GMED.** We tune the editing stride (α) using the first three tasks as described in Sec. 3.3.

Methods / Datasets	Split MNIST	Rotated MNIST	Split CIFAR-10	Split mini-ImageNet
1-step Edit	82.21 ± 2.9	77.50 ± 1.6	34.84 ± 2.2	27.27 ± 1.8
3-step Edit	82.55 ± 1.9	77.37 ± 1.7	34.93 ± 1.4	**27.36 ± 1.7**
5-step Edit	**83.11 ± 1.9**	**77.53 ± 1.6**	**36.82 ± 1.8**	26.36 ± 2.0

5 Conclusion

In this paper, we propose Gradient based Memory Editing (GMED), a modular framework for memory-based task-free continual learning where examples stored in the memory can be edited. Importantly, memory examples edited by GMED are encouraged to remain in-distribution but yield increased loss in the upcoming model updates, and thus are more effective at alleviating catastrophic forgetting. We find that combining GMED with existing memory-based CL approaches leads to consistent improvements across multiple benchmark datasets, only inducing a small computation overhead. Finally, we perform a thorough ablative study to validate that the gains obtained by GMED can indeed be attributed to its editing operation and careful choice of editing objective.

References

[1] Tameem Adel, Han Zhao, and Richard E. Turner. 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.

[2] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *NeurIPS*, 2019.

[3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11254–11263. Computer Vision Foundation / IEEE, 2019.

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11816–11825, 2019.

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. *ArXiv*, abs/2010.05595, 2020.

[6] Massimo Caccia, P. Rodríguez, O. Ostapenko, Fabrice Normandin, Min Lin, L. Caccia, Issam H. Laradji, I. Rish, Alexande Lacoste, D. Vázquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *NeurIPS*, 2020.

[7] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip H. S. Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *ArXiv*, abs/2002.08165, 2020.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 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.

[9] Aristotelis Chrysakis and Marie-Francine Moens. 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*, pages 1952–1961. PMLR, 2020.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 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*, pages 248–255. IEEE Computer Society, 2009.

[11] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. 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.

[12] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[14] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[15] Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *ArXiv*, abs/1906.05201, 2019.

[16] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[17] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. 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.

[18] Khurram Javed and Martha White. Meta-learning representations for continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1818–1828, 2019.

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[20] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, A. Leonardis, Gregory Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv: Computer Vision and Pattern Recognition*, 2019.

[21] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[22] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020.

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

[24] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *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*, pages 3925–3934. PMLR, 2019.

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

[26] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6467–6476, 2017.

[27] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 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.

[29] Jorge A Mendez and ERIC EATON. Lifelong learning of compositional structures. In *International Conference on Learning Representations*, 2021.

[30] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. 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.

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

[32] G. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks : the official journal of the International Neural Network Society*, 113:54–71, 2019.

[33] Anthony V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 7:123–146, 1995.

[34] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience replay for continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 348–358, 2019.

[35] Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3339–3345. ijcai.org, 2019.

[36] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016.

[37] Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer G. Dy and Andreas Krause, editors, *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*, pages 4555–4564. PMLR, 2018.

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2990–2999, 2017.

[39] Yang Song, Rui Shu, Nate Kushman, and S. Ermon. Constructing unrestricted adversarial examples with generative models. In *NeurIPS*, 2018.

[40] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 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.

[41] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019.

[42] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638, 2016.

[43] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *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*, pages 3987–3995. PMLR, 2017.

[44] Chen Zeno, Itay Golan, E. Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *ArXiv*, abs/1803.10123, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] all image classification datasets we applied does not contain such information
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]