An Efficient Rehearsal Scheme for Catastrophic Forgetting Mitigation during Multi-stage Fine-tuning

Anonymous ACL submission

Abstract

Incrementally fine-tuning foundational models on new tasks or domains is now the de facto approach in NLP. A known pitfall of this approach is the *catastrophic forgetting* of prior knowledge that happens during fine-tuning. A common approach to alleviate such forgetting is to rehearse samples from prior tasks during fine-tuning. Several existing works assume a fixed memory buffer to store prior task examples, while relying on inferences (forward passes) with the model at hand for choosing examples for rehearsal from the buffer. However, given the increasing computational cost of model inference, and decreasing cost of data storage, we focus on the setting to rehearse samples with a fixed computational budget instead of a fixed memory budget. We propose a sampling scheme, **mix-cd**, that prioritizes rehearsal of "collateral damage" samples, which are samples predicted correctly by the prior model but forgotten by the incrementally tuned one. The crux of our scheme is a procedure to efficiently estimate the density of collateral damage samples without incurring additional model inferences. Our approach is computationally efficient, easy to implement, and outperforms several leading continual learning methods in compute constrained settings.

1 Introduction

The advent of pretrained foundational models has led to a paradigm shift in machine learning, wherein, a single model can be trained to learn a wide variety of tasks. Incrementally learning of a new task or domain is carried out by fine-tuning some or all parameters on the new task. Such learning is both compute and data efficient as it benefits from the patterns learned during learning of previous tasks (as well as pretraining). It is common to sequentially fine-tune foundational models over various datasets in order to teach the model new tasks, or improve performance on new domains for an already learned task.

Unfortunately, such incremental tuning of the parameters may lead to forgetting of tasks or dmains learned previously. For instance, consider a multilingual translation model that can translate from other languages to English. When we incrementally tune this model to learn translation from an additional language (Danish in this case), we find that its performance degrades on previously learned languages; see Figure 1. Similar forgetting of prior skills/knowledge happens when instruction-tuned language models are aligned on human preferences using reinforcement learning; this is referred to as the *alignment tax* (Lin et al., 2024).

In this work, we study computationally-efficient methods for incrementally training foundational models on new tasks or domains, while preventing such *catastrophic forgetting* of knowledge from selected previous tasks. A common strategy to reduce catastrophic forgetting during fine-tuning is to "rehearse" samples from previous task by mixing them into the fine-tuning set. The rehearsal samples are typically drawn from a limited rehearsal buffer holding samples from previous tasks.

However, there are two main criticisms for existing rehearsal settings. First, most rehearsal methods assume only a small rehearsal buffer, citing storage costs and data access restrictions as the reason. This limits the space for drawing rehehasal samples, which can lead to overfitting (Verwimp et al., 2021). Second, many rehearsal methods require high computational cost, in the form of inferencing with the model at hand, to select examples for rehearsal. Many existing rehearsal methods fall short when we take into account the computational cost of sampling examples for rehearsal. Recent work (Prabhu et al., 2023) shows that several high performing methods cannot beat random uniform rehearsal in compute constrained settings.

Over the last decade, storage costs have dramatically reduced to nearly 2 cents/gb (Prabhu et al., 2023). On the other-hand, the size of foundational

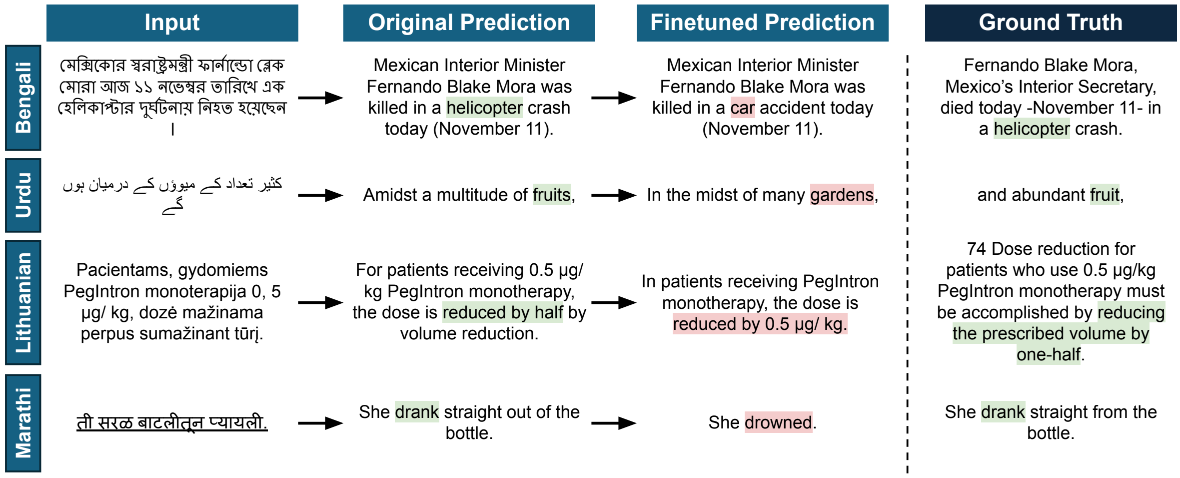

Figure 1: Examples of collateral damage in prior language translations after fine-tuning on Danish-to-English.

models has grown exponentially, keeping computational costs[1] of training and inference high. Thus, in this work, we seek rehearsal methods that are computationally-efficient but are allowed full access to prior fine-tuning sets. We assume a setting where the multi-stage fine-tuning is performed by the same party, and therefore there are no data access restrictions.

In this work, we propose **mix-cd**, a rehearsal method that is no more expensive than random uniform rehearsal, but achieves a strictly better trade-off between new and previous task performances. This is significant as uniform sampling is known to be a really strong baseline in compute-constrained settings. The key insight in our method is that it is beneficial to prioritize rehearsing *collateral damage* samples. Collateral damage is defined as being predicted correctly by the existing model, but incorrectly by the incrementally tuned one.

A key technical challenge is that the naive approach for obtaining collateral damage information requires making a forward pass with the fine-tuned model on the prior dataset. This incurs significant computation costs. To overcome this, we propose an efficient method for estimating the collateral damage density within the data distribution. The estimated density is updated throughout the fine-tuning process to keep track of the dynamic changes in where collateral damage occurs.

Overall, our scheme retains the desirable quality of being general, lightweight, and easy to implement, and can serve as a drop-in replacement for the random uniform rehearsal approach. Through experiments on multiple tasks, we demonstrate that our scheme outperforms random uniform rehearsal and several other offline and online continual learning baselines in striking a favorable trade-off between new and previous task performances.

[1]Performing inference on N tokens with a transformer model with D parameters requires approximately 2ND FLOPs. Thus, inferencing on a sequence of 100 tokens with a 1B parameter model would involve a staggering $2 \cdot 10^{11}$ FLOPS.

2 Background and Related Work

2.1 Multi-stage fine-tuning framework

The multi-stage fine-tuning framework finds applications across various domains and tasks within the field of machine learning. In natural language processing, pretrained language models such as BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), and others are extensively fine-tuned for specific tasks like sentiment analysis (Sun et al., 2019), text summarization (Liu and Lapata, 2019), and question answering (Roberts et al., 2020). Large generative language models such as GPT (Brown et al., 2020) and Llama (Touvron et al., 2023) are instruction-tuned (Wei et al., 2021) on human-provided feedback to align their generation with human responses. In computer vision, pretrained vision transformers are commonly fine-tuned for image classification, object detection (Li et al., 2022), and segmentation tasks (Thisanke et al., 2023). Transfer learning through continual fine-tuning is also prevalent in medical imaging (He et al., 2023) for tasks like disease diagnosis and organ segmentation.

2.2 Retaining Prior Performance

One major challenge for multi-stage fine-tuning is retaining prior performance while improving on the current fine-tune task. In some cases where the fine-tuned model is only expected to perform well on a limited set of fine-tune examples, in which case, disregarding the prior task is acceptable. On the other hand, studies have shown that maintaining the prior performance is beneficial to not overfit on the fine-tune data and other desiderata (Lin et al., 2023; He et al., 2021).

Forgetting prevention by weight regularization Weight regularization (Lin et al., 2023) methods prevent prior task forgetting by directly restricting the weights of the fine-tuned model. The weights can be constrained during fine-tuning by anchoring them to the prior model weights (Panigrahi et al., 2023; Xuhong et al., 2018; Kirkpatrick et al., 2017). The constraint can also be in the form of low-rank weight adaptation with LoRA (Hu et al., 2021). On the other hand, Wortsman et al. (2022) proposes WiSE-FT to ensemble the prior and fine-tuned weights post-hoc to achieve a balanced tradeoff of performance between tasks. In general, weight regularization methods rely on the assumption that the new model optima post-fine-tuning lie close to the prior optima.

Forgetting prevention by rehearsal Rehearsal-based methods prevent prior task forgetting by including a portion of prior data into the fine-tuning phase. A common approach is to sample uniformly at random from the prior data and mix them into the fine-tuning set (He et al., 2021; Kazemi et al., 2023). Some prior works consider the setting where prior data must be selected offline before accessing the next task. Yoon et al. (2022) proposed Online Coreset Selection, which selects important samples while streaming through the prior dataset. They prioritize data points with high minibatch similarity and sample diversity. Mok et al. (2023) proposed Dynamic Instance Selection, which selects the highest and lowest predictive entropy samples to allow easier and more difficult examples to be represented evenly. However, such offline selection methods fail to consider the impact of the new fine-tune task, and are unable to tailor the selected samples to best mitigate the induced forgetting. Aljundi et al. (2019) proposed Maximally Interfered Sampling (MIR), where high loss difference points are sampled from a small replay buffer. Prabhu et al. (2023) has shown that all existing continual learning methods evaluated fail to beat the random mixing baseline in a computationally-constrained setting without the memory constraint. Our work adopts the computationally-constrained setting motivated by Prabhu et al. (2023).

3 Evaluation Protocol and Key Idea

Our objective is to: *Improve performance on the fine-tune task while avoiding performance deterioration on prior tasks.* In this section, we define our evaluation protocol, and motivate the design of our method via some key empirical observations.

3.1 Problem Setting and Evaluation protocol

We start with a model trained on a prior task, and we assume that the training losses on the prior task examples are stored and accessible without any extra computational costs. Our objective is to improve the fine-tune task performance while balancing prior task performance given limited computational budget. Thus, we compare different rehearsal strategies by examining the Pareto curve of the prior (y-axis) and fine-tune (x-axis) task performances. Different points on the same pareto curve corresponds to different instantiations of the same rehearsal strategy with different *mix ratios* β given a fixed computational budget c. Mix ratio is defined as the proportion of computation budget allocated to rehearsing the prior task and fine-tuning on the new task.

For example, if $\beta = 0.1$ then $c_p ::= 0.1 * c$ is the budget allocated to rehearsal whereas $c_f ::= 0.9 * c$ is the buddget allocated to fine-tuning. The rehearsal budget includes both the cost of sampling the rehearsal instances as well the cost of training on those instances. The fine-tuning budget includes the cost of training on the new task instances. By ablating the mix ratios, the computational budget is traded between rehearsing and fine-tuning, which forms the Pareto curves. For instance, fine-tuning with $\beta = 0.5$ would emphasize rehearsal more than $\beta = 0.1$.

All points on the Pareto frontier have the exact same computational cost but differs in how they spend the computational budget between prior task and fine-tune task. Methods with a Pareto frontier towards the top right direction are more preferable.

3.2 Key Idea: Rehearse Collateral Damage

Our key idea is to sample *collateral damage* example more efficiently during rehearsal, i.e., examples that were predicted correctly by the prior model but were "forgotten" during fine-tuning. We motivate this by ablating random uniform rehearsal

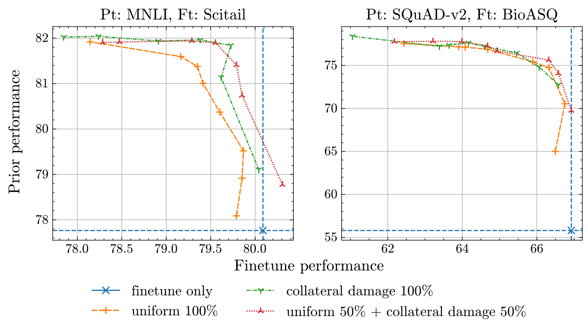

Figure 2: Preliminary observations suggest that while random rehearsal of prior data helps mitigate collateral damage, upweighting collateral damage samples in the prior data distribution benefits the joint performance on the both tasks even more. Curves closer to the top right are more preferable.

mixed with different proportions of collateral damage samples in Fig 2.

The fine-tune baseline suffers significant collateral damage as the prior task is catastrophically forgotten. Random uniform rehearsal helps retain the prior task performance at the cost of worse fine-tune performance. However, random uniform rehearsal is sub-optimal as it does not take into account the "utility" of the prior samples. Mixing in 50% collateral damage samples improves the Pareto curve and achieves better joint performance in both experiment settings. We hypothesize that the reason why rehearsing collateral damage reduces forgetting is that since the model predicted correctly on these samples before, it could predict correctly again after rehearsal.

One major technical challenge is that determining whether a sample is collateral damage requires at least an additional inference on the current fine-tune model. This makes the cost of sampling collateral damage sample much higher than random uniform rehearsal, which has a negligible sampling cost. As mentioned earlier, the computational budget for rehearsal c_p is split into budgets for sampling $c_{p,s}$ and training $c_{p,t}$. Consequently, methods with high sampling cost will have less budget available for training, and will therefore afford fewer rehearsal samples.[2] In the next section, we propose a method that efficiently estimates the density of collateral damage samples, and affords the same number of rehearsal samples as random uniform rehearsal.

[2]We assume that the number of training steps needed for convergence is independent of the number of rehearsal samples, and cannot be lowered.

4 Methodology

We propose `mix-cd`, a rehearsal sampling scheme that efficiently prioritizes collateral damage points. Our key idea is to estimate the collateral damage distribution at each fine-tuning iteration by using only the samples mixed in during the previous iterations. Since these mixed in samples are already part of the fine-tuning set, we get inference (forward pass) on them for free as part of the standard training loop. This makes the procedure as computationally efficient as random uniform rehearsal.

4.1 Formal Definition of Collateral Damage

Let us denote the base (prior) model as f, fine-tuned model as f', prior data samples as z_p, and fine-tune data samples as z_f. Let ϕ denote the indicator function for collateral damage. For classification tasks, a sample $z_p = (x, y)$ suffers from collateral damage, denoted by $\phi_{f,f'}(z_p)$, if it is predicted correctly by f but incorrectly by f'.

$$\begin{aligned}\phi_{f,f'}(z_p) :=& \big(\arg\max f(x) \equiv y\big) \wedge \\ & \big(\arg\max f'(x) \neq y\big)\end{aligned}$$

For non-classification tasks, collateral damage can be defined using the losses of the base and fine-tuned models. Specifically, a sample z_p suffers from collateral damage if its loss on f is less than a threshold τ, and loss on f' is greater than τ.

$$\phi^{\tau}_{f,f'}(z_p) = \big(\mathsf{loss}(f, z_p) < \tau\big) \wedge \big(\mathsf{loss}(f', z_p) > \tau\big)$$

In our experiments, we set τ as the 90^{th} percentile of the loss of the prior model on the prior data.

4.2 Main procedure: `mix-cd`

Our main procedure `mix-cd` is defined in Algorithm 1. The key is estimating the probability that a prior sample z_p suffers from collateral damage without inferencing z_p on f'. We first partition the prior data distribution into K bins. At each fine-tuning iteration, we estimate the conditional probability (denoted by α_k) that a sample from bin k suffers from collateral damage. Formally,

$$\alpha_k := P(\phi_{f,f'}(z_p) = 1 \mid b(z_p) = k)$$

where $b(z_p) \in [K]$ is the bin that sample z_p falls in. We discuss different partitioning schemes in Section 5.4. Once we have estimates $\hat{\alpha}_k$, we select a randomly drawn pretraining sample z_p with probability $\hat{\alpha}_{b(z_p)} \cdot P(b(z_p))$.

Algorithm 1 `mix-cd-sample`

Input: number of iterations N, prior dataset Z_p, fine-tune dataset Z_f, base model f, mix ratio β, number of partitions K, number of training samples per iteration n.
// Initialize estimates $\hat{\alpha}_k$
for $k = 1$ **to** K **do**
  Initialize $\hat{\alpha}_k \leftarrow 0.5$
  Initialize $u_k \leftarrow 0, n_k \leftarrow 0$
end for
Initialize fine-tune model $f' \leftarrow f$
for $n = 1$ **to** N **do**
  Initialize dataset $D_f \leftarrow (1 - \beta) \cdot n$ random uniform samples from Z_f
  Initialize dataset $D_p \leftarrow \{\}$
  repeat
    $z_p \leftarrow$ sample from Z_p with probability $\hat{\alpha}_{b(z_p)} \cdot P(b(z_p))$
    $D_p \leftarrow D_p \cup \{z_p\}$
  until $|D_p| \geq \beta \cdot n$
  Train f' for one iteration on $D_f \cup D_p$
  // Update estimates $\hat{\alpha_k}$
  for $k = 1$ **to** K **do**
    Update u_k, n_k according to Eq 1 and 2
    $\hat{\alpha}_k \leftarrow u_k / n_k$
  end for
end for

Estimating α_k. A straightforward scheme for estimating α_k is to sample uniformly from the prior distribution, perform inference on the samples using the fine-tuning model, and then compute the fraction of samples falling in bin k that suffer from collateral damage. While this provides an unbiased estimate of α_k, it incurs additional inference cost. To completely avoid *any* additional inference, we propose estimating α_k at each iteration using the prior data samples mixed into the fine-tuning step during the previous iteration. For the first iteration, the prior data samples are drawn uniformly at random with α_k set to 0.5 for all k. For subsequent iterations, we maintain running counts of the number of samples (n_k) mixed in from bin k, and the number of collateral damage samples (u_k) among them. Specifically, at the end of each iteration, we update these counts as follows. Let D_p be the prior data samples mixed in during the iteration.

$$n_k \leftarrow n_k + |\{z_p \in D_p \mid b(z_p) = k\}| \tag{1}$$

$$u_k \leftarrow u_k + |\{z_p \in D_p \mid b(z_p) = k, \phi_{f,f'}(z_p) = 1\}| \tag{2}$$

for all $k \in [K]$. We then set our estimate $\hat{\alpha}_k := u_k/n_k$. Since D_p is already part of the fine-tuning set, we have the forward pass from f' on them available as part of the standard training loop. We further assume that predictions of the prior model on all prior data samples are cached, allowing us to compute $\phi_{f,f'}(z_p)$ at no additional inference cost.

Remark. *Our estimation procedure is not unbiased as we use samples seen during fine-tuning to estimate collateral damage distribution for unseen samples. In a sense, we trade off computational cost for this bias. Fortunately, despite the bias, our scheme selects sufficiently large number of collateral damage samples, which helps it outperform several baselines; see Section 5.2.2).*

4.2.1 Partitioning Prior Data

The intuition behind `mix-cd` is that by partitioning the prior data distribution into bins, we can identify regions that suffer more from collateral damage. We can then prioritize rehearsing from such regions over others during fine-tuning. We can use any type of partitioning as long as the collateral damage is *not* conditionally independent of the partitions. If collateral damage is conditionally independent, `mix-cd` degenerates to random uniform rehearsal. To avoid partitioning with ineffective bins, we calculate the KL divergence between collateral damage ratios of partitions with a uniform distribution. A partition is effective if the KL divergence exceeds a certain threshold. Empirically we found that 0.01 is an effective threshold for identifying effective partitions. In practice, after the first iteration of fine-tuning with random rehearsal, the KL statistics for partitions can be calculated for partition selection with no additional computation required. Next, we discuss some partition strategies that work well with `mix-cd`, and are common to obtain in datasets.

Partition with prior data loss. Prior data can be partitioned according to their losses on the prior task. Bins can be defined based on fixed-sized loss quantiles. Typically, examples with higher (lower) loss in prior tasks are typically far from the decision boundary, and thus more (less) likely to be forgotten during fine-tuning. Thus, partitioning with prior data loss is useful to identify slices where collateral damages happen more (less) frequently.

Partition with auxiliary information. Prior data can also be partitioned with auxiliary information such as class labels and/or other meta labels.

Usually these meta labels convey semantic meaning that helps distinguish whether certain regions would suffer more from collateral damage. For multilingual translation datasets, the language serves as a natural partition. For instruction-tuning datasets, the source instruction-tuning task also naturally partitions the instruction data. In our experiments, we find that partitions based on combining prior loss and auxiliary labels perform the best.

Combining multiple partitions. Multiple partitions can be combined to form even more finer-grained partitions. Given two partition strategies $A = a_1, \cdots, a_n$ and $A' = a'_1, \cdots, a'_m$, the combined partition is simply the set product of A and A' with $n \times m$ bins. If A is independent of A', then the collateral damage likelihood of bin $a_i \cap a'_j$ is estimated by factoring with respect to the individual partitions:

$$p(\phi|b_{a_i,a'_j}) \propto p(\phi|b_{a_i}) \cdot p(\phi|b_{a'_j})$$

On the other hand, if A is conditionally independent of A' given collateral damage, then we can estimate the collateral damage likelihood by factoring and accounting for the conditional dependency:

$$p(\phi|b_{a_i,a'_j}) \propto \frac{p(b_{a_i}) \cdot p(a'_j)}{p(b_{a_i,a'_j})} \cdot p(\phi|b_{a_i}) \cdot p(\phi|b_{a'_j})$$

When the (conditional-)independence relation between partitions holds, estimating the collateral damage likelihood by factoring is more sample efficient since only $n + m$ statistics needed to be maintained, as opposed to $n \times m$ when estimating jointly. In practice, we can test for whether such relation holds by the end of the first iteration of fine-tuning with no additional computational cost.

5 Experiments and Discussion

5.1 Experiment Setup

We experiment on three different tasks that commonly utilize a multistage-fine-tuning pipeline: text classification, closed-book QA, and multilingual translation. More technical details can be found in Appendix A.

Text classification: MNLI-Scitail We start with a DistilBert (Sanh et al., 2020) fine-tuned on MNLI (Kim et al., 2019) for natural language inference (NLI), then fine-tune it on Scitail (Khot et al., 2018), a NLI dataset for scientific statements. The ground truth class labels and genre labels are used for partitioning. The prior and current task performances are defined as the classification accuracy on the holdout test sets for MNLI and Scitail respectively.

Closed-book QA: SquadV2-BioASQ We start with a tiny Roberta (Liu et al., 2019) fine-tuned on SquadV2 (Rajpurkar et al., 2018) for general domain question answering, then fine-tune it on BioASQ (Nentidis et al., 2020), a closed-book QA dataset for biology domain knowledge. Binary labels of whether a sample is answerable or not are used for partitioning. The prior and current task performances are defined as the exact-matching accuracy on the holdout datasets for SquadV2 and BioASQ respectively.

Multilingual translation: translating Danish to English We start with `mBart50` (Tang et al., 2020), a multilingual translation model that translates from 50 different languages to English, fine-tuned on Opus100, a multilingual, English-centric dataset that consists of sentence pairs translating from 100 other languages (excluding Danish) to English. We additionally fine-tune the model on Danish, which is previously not supported by the base `mBart50` model. The prior language labels are used for partitioning the data distribution, as we expect different languages suffer collateral damage with different severity. The prior task performance is defined as the average loss of all language samples excluding Danish in holdout Opus100 and the fine-tune task performance is defined as the average loss of Danish samples in holdout Opus100.

Training configuration For each experiment, we report the joint performance of the pretrain and fine-tune task on holdout datasets, evaluated at the end of fine-tuning. The results are averaged over 5 repetitions for the NLI task, 10 for QA, and 5 for translation. The mix ratio β is chosen to be in the range of $[0.01, 0.9]$ such that all rehearsal methods cover similar fine-tune performance.

5.2 Mix-cd Outperforms Baselines

To demonstrate the general effectiveness of `mix-cd` in diverse fine-tuning settings, we compared it with other rehearsal strategies of equal computation cost. Recall an iteration of fine-tuning refers to fine-tuning the model on every n samples.

5.2.1 Baseline Descriptions

Baseline methods can be classified into two categories: offline and online. Offline methods se-

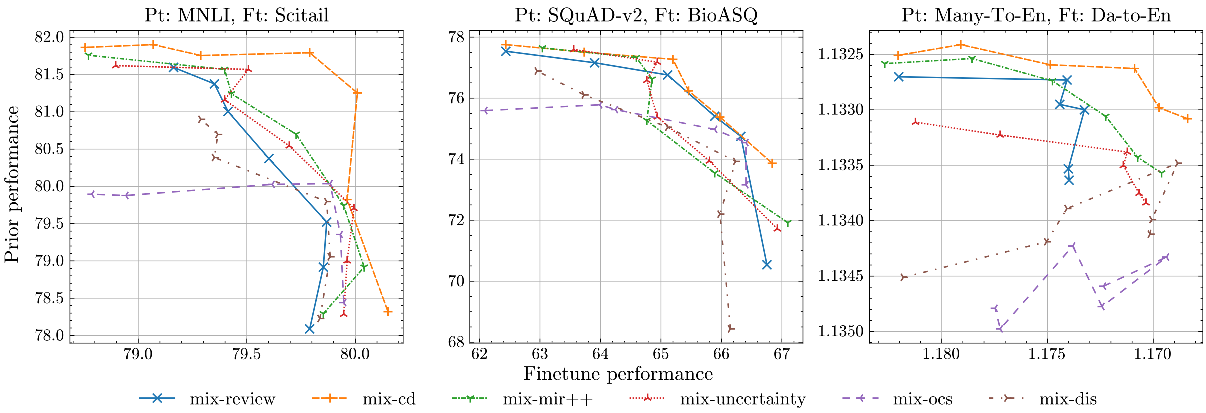

Figure 3: Pareto frontiers of prior and fine-tune performance. Curves closer to the top right are more preferable.

lect important prior samples to rehearse before the fine-tuning begins. During fine-tuning, important selected samples are rehearsed randomly. These method are computationally efficient as they do not require additional sampling cost. However, they suffer from lacking information regarding the new fine-tune task since selection happens offline before fine-tuning. Thus, the selected prior samples cannot be targeted to mitigate the incurred collateral damage.

On the other hand, online methods select samples for rehearsal when the prior samples are streamed online during fine-tuning. Specifically, a set of n_p prior samples are first randomly sampled for each batch of n_f fine-tune data. The online method assigns a priority score to the n_p prior samples and filter the top k % to mix into the batch for rehearsal. Recall the prior rehearsal computational budget c_p consists of the sampling $c_{p,s}$ and training $c_{p,t}$ cost. The effective number of prior samples to actually train on depends on the sampling cost, which further depends on the cost of assigning priority scores and the filter ratio k. We adopt a filter ratio of 50 % for all online methods to balance between the effectiveness selection and budget for training. To factor in the priority assignment, we approximate the computation cost of a forward pass as half of a backward pass in terms of FLOPs. For example, suppose the priority assignment requires one forward pass on the model. Then the assignment is worth training 1/3 of a sample, since training one sample requires one forward and one backward pass. We calculated the effective numbers of each method (which might be different depending on the sampling cost) to control for equal total computational budget.

Offline baselines Online coreset selection (`mix-ocs`) is a coreset selection method proposed by Yoon et al. (2022). Dynamic instance selection (`mix-dis`) is a rehearsal method for continual learning proposed by Yoon et al. (2022). For both methods, a subset of size equivalent to the fine-tune dataset is selected offline and rehearsed randomly during fine-tuning.

Online baselines Online methods differ in the definition of priority score. `mix-uncertainty` prioritizes samples with high uncertainty, a common objective for active learning and datas selection. The uncertainty is estimated with prediction entropy for classification tasks and sequence log likelihood for generative tasks. `mix-mir++` is a modification of Maximal Interfered Retrieval (MIR) (Aljundi et al., 2019) for a computation-constraint setting. Typical MIR calculates the online difference in prior sample loss between the current fine-tune model and a copy of the model with one additional gradient step on the fine-tuned data, which is too costly. Instead, we modified their method to calculate the difference in prior sample loss between the current fine-tune model and the cached base model. We observed the performance of `mix-mir++` to be significantly better than MIR in our Pareto frontier curves, and thus we only report the performance of `mix-mir++`.

5.2.2 Result analysis

The main result is presented in Fig 3, where `mix-cd` consistently outperforms the `random` baseline over all experiment settings. This supports `mix-cd` as the drop-in replacement for `random` since the performance gain comes at no additional computation cost. Online baselines perform similar to or worse than `random` since for the given computation budget, spending the budget on sampling is not a desirable tradeoff for performance. The performance of offline methods is the worse since the selection objective does not take the fine-tune task information into consideration. This highlights the importance of the adaptivity in online methods.

5.3 How many more collateral damage samples does mix-cd rehearse?

The design goal of `mix-cd` is to sample collateral damage samples more efficiently. Fig 4 compares the actual proportion of collateral damage samples in all sampled data, for `mix-cd` and random uniform sampling. `mix-cd` consistently samples twice or more collateral damage for rehearsal compared to random uniform sampling for all mix ratios. The empirical result supports that `mix-cd` achieves its intended purpose and also explains the superior performance over random uniform sampling.

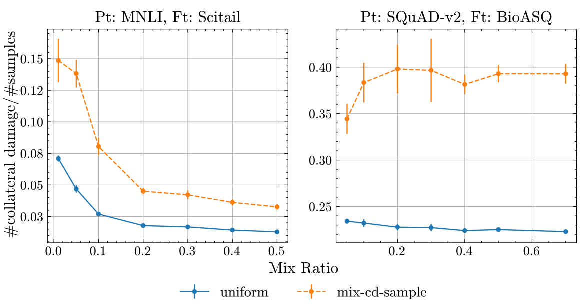

Figure 4: Proportion comparison of collateral damage per sample between random uniform and `mix-cd` across different mix ratios. `mix-cd` consistently samples twice or more collateral damage for rehearsal compared to random uniform, which explains the superior performance.

5.4 Selecting bins with collateral damage signal is crucial for mix-cd

Recall the partition selection strategy proposed in Section 4.2.1. Fig 5 demonstrates the effectiveness of the selection strategy on SquadV2. There are four types of partitions available for SquadV2. Prior loss partition splits the data distribution with the prior loss values evaluated on the base model and bin them according to 5 fixed size loss quantile intervals. Answerable partition splits the data distribution by the binary label of whether the answer can be found in the given context or not. Genre partition splits the data distribution by the genre of the specific question into 5 bins (e.g. geology, history, technology). Sequence length partition splits the data distribution by the sequence length of the samples and bin them according to 5 fixed size sequence length quantile intervals. After evaluating the KL divergence with the uniform distribution, the loss and answerable bins are selected as the best candidate for `mix-cd` partitions. The right subfigure in Fig 5 verifies that indeed coupling loss and answerable partitions are the best combination for joint performance.

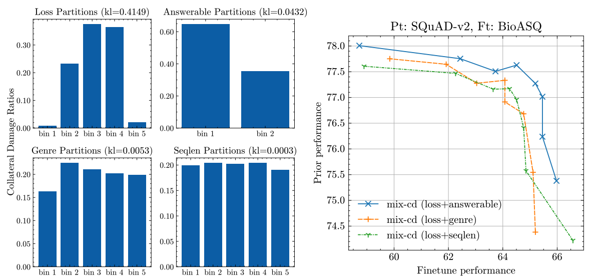

Figure 5: Ablation study on different partitions for the data distribution. Partitions with higher KL divergence in collateral damage ratios between bins (e.g. loss and answerable partitions) provide better signal for prioritizing collateral damage samples.

6 Conclusion

In this paper, we proposed a rehearsal-based sampling strategy to prioritize collateral damage samples during fine-tuning. The simplicity and effectiveness makes it an appealing drop-in replacement for the typical random uniform rehearsal strategy. Future work can investigate better hybrid methods that combine both rehearsal and weight-regularization for forgetting prevention.

Limitations We assume the last-epoch prediction or loss of the prior data on the base model is saved during the fine-tuning phase. The loss or prediction information provides important signal to identify collateral damage regions in the prior data distribution. More investigation is also needed to examine whether the original prior performance can be fully recovered with `mix-cd`.

Potential risks It is possible that non-uniform rehearsal with `mix-cd` prioritizes the region suffering from the most collateral damage. This might introduce bias in the fine-tuned model that cannot be detected merely with the prior task performance. Further study is required to examine whether collateral damage in minority sample regions is affected by the rehearsal scheme.

References

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. 2019. Online continual learning with maximally interfered retrieval. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 11872–11883.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kelei He, Chen Gan, Zhuoyuan Li, Islem Rekik, Zihao Yin, Wen Ji, Yang Gao, Qian Wang, Junfeng Zhang, and Dinggang Shen. 2023. Transformers in medical image analysis. *Intelligent Medicine*, 3(1):59–78.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1121–1133.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Mehran Kazemi, Sid Mittal, and Deepak Ramachandran. 2023. Understanding finetuning for factual knowledge extraction from language models. *arXiv preprint arXiv:2301.11293*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI Conference on Artificial Intelligence*.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6586–6593.

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

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. 2022. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. 2024. Mitigating the alignment tax of rlhf. *Preprint*, arXiv:2309.06256.

Yong Lin, Lu Tan, Hangyu Lin, Zeming Zheng, Renjie Pi, Jipeng Zhang, Shizhe Diao, Haoxiang Wang, Han Zhao, Yuan Yao, et al. 2023. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *arXiv preprint arXiv:2309.06256*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. 2023. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12573–12589, Toronto, Canada. Association for Computational Linguistics.

Anastasios Nentidis, Konstantinos Bougiatiotis, Anastasia Krithara, and Georgios Paliouras. 2020. Results of the seventh edition of the bioasq challenge. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*, pages 553–568. Springer.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. 2023. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3698–3707.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Preprint*, arXiv:1806.03822.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, and Damayanthi Herath. 2023. Semantic segmentation using vision transformers: A survey. *Engineering Applications of Artificial Intelligence*, 126:106669.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. 2021. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. *Preprint*, arXiv:2104.07446.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. 2022. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

LI Xuhong, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pages 2825–2834. PMLR.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. 2022. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations*.

A Experiment Technical Details

A.1 Text classification: MNLI-Scitail

We first fine-tune a DistilBert model on MNLI (Kim et al., 2019), which is a natural language inference (NLI) dataset, and then the model fine-tune on Scitail (Khot et al., 2018), a natural language entailment dataset for scientific statements. NLI tasks aim to determine the relationship (entailment, contradiction, or neutral) between a pair of input sentences. The model is fine-tuned with AdamW with learning rate of $2 \cdot 10^{-6}$ and weight decay of 10^{-5}. There are 393,000 samples in the MNLI pretrain training dataset. In additional to relation label, additional genre labels (e.g. fiction, government, travel) for the sentence pairs are also provided. To implement `mix-cd-sample`, we use the ground truth class labels and genre labels for partitioning. For each iteration, we fine-tune on 1,000 samples from the Scitail training set (iterating over the entire training set of 23,600 samples after 25 iterations). The pretrain and fine-tune task performances are defined as the classification accuracies on MNLI and Scitail, respectively.

A.2 Closed-book QA: SquadV2-BioASQ

We first fine-tuned a Tiny Roberta (Liu et al., 2019) on SquadV2 (Rajpurkar et al., 2018) for general domain closed-book QA, then fine-tune it on BioASQ (Nentidis et al., 2020), a closed-book QA dataset for biology domain knowledge. The model is fine-tuned with AdamW with learning rate of $1 \cdot 10^{-5}$, warming up the learning rates from $1 \cdot 10^{-7}$ for 5 iterations, then cosine annealing the learning rate to $1 \cdot 10^{-6}$, and weight decay of 10^{-5}. There are 130K samples in the SquadV2 training dataset. To implement `mix-cd`, we use the binary labels of whether a sample is answerable or not are used for partitioning. For each iteration, we fine-tune on 1,000 samples from the BioASQ training set for 20 iterations. The prior and current task performances are defined as the exact-matching accuracy on the holdout datasets.

A.3 Multilingual translation: translating Danish to English

The experimental setting for multilingual translation is slightly different from classification tasks. Instead of fine-tuning on a new dataset, we take a multilingual translation model that translates from 50 different languages to English, and fine-tune it to perform translation on one additional language. To implement `mix-cd`, we use the language type for partitioning. We would like to prevent any deterioration in the performance of the existing 50 languages due to fine-tuning. It is expected for the translation for some languages in the pretrain language to deteriorate after fine-tuning. We leverage the pretrain language as auxiliary information for partitioning to identify and fix the languages with more collateral damage.

The base model of choice is `mBart50` (Tang et al., 2020), a generative language model pretrained on translation sentence pairs of 50 different languages to English. The model is fine-tuned with AdamW with learning rate of 10^{-5} and weight decay of 10^{-5}. The training data pairs (both prior and fine-tune) are taken from Opus100, a multilingual, English-centric dataset that consists of sentence pairs translating from 100 other languages to English. We fine-tune the model on Danish, which is previously not supported by the pretrained `mBart50` model. For each iteration, we subsample 10,000 new Danish-English sentence pairs to fine-tune. The prior dataset consists of 10,000 random uniform samples from the languages that `mBart50` was originally capable of translating. The prior task performance is defined as the average loss of all prior language samples and the fine-tune task performance is defined as the average loss of Danish samples.