A GROUND-TRUTH MODEL SUITE FOR UNLEARNING EVALUATION ACROSS THE LM TRAINING LIFECYCLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning – the ability to selectively remove specific information from a trained model – is critical for privacy, copyright compliance, and safety. In this work, we present a controlled study of machine unlearning across the large language model (LLM) development lifecycle, spanning pre-, mid-, and post-training. To achieve this, we inject multiple distinct datasets during the from-scratch training of OLMo-2 models ranging from 179M to 2.7B parameters and trained on 210B tokens. These datasets encompass seven diverse unlearning tasks with varying difficulty levels, including the removal of Gaussian poisons, fictional knowledge, reasoning patterns, social interaction data, and adversarial attacks. Crucially, we also provide counterfactual models trained without these datasets, which serve as exact ground-truth targets for unlearning evaluation. Leveraging this suite, we investigate the effectiveness of existing unlearning algorithms across three distinct lifecycle settings: (1) immediate unlearning after pre-training, (2) unlearning after decaying from the pre-trained model and (3) unlearning via mid-training. Our experiments reveal that unlearning success is highly non-uniform where effectiveness depends heavily on the stage of the training cycle and is significantly influenced by model scale and task complexity. Surprisingly, we find that one *simple and hyperparameter free* baseline among eight unlearning methods is exceptionally difficult to beat: continual pre-training on fresh data.

1 INTRODUCTION

Modern large language models (LLMs) rely on web-scale training corpora that inevitably ingest personal data, copyrighted material, and other sensitive content. Consequently, model developers face increasing regulatory and legal pressure to remove specific knowledge post-training. Under data protection frameworks, most notably the "right to be forgotten" in the GDPR, organizations are obliged to delete personal data upon request – a mandate that extends beyond raw databases if a model has memorized the restricted data (Carlini et al., 2023). Similar obligations arise from intellectual property law, where the memorization and generation of protected text can trigger copyright takedown requests (Wei et al., 2024). However, retraining from scratch to honor each such request is prohibitively expensive. As a more tractable alternative, machine unlearning aims to remove the influence of specific training data from a trained model at a small fraction of this cost (Cao and Yang, 2015; Bourtoule et al., 2021), while the same tools are increasingly used to remove hazardous model capabilities (Li et al., 2024) and adversarial model behaviors (Pawelczyk et al., 2025).

Despite their growing adoption across these domains, approximate unlearning methods typically come with no formal guarantees – whether they work is an empirical question, and the answer depends on how it is evaluated. Existing LLM unlearning benchmark evaluations fall into two groups. The first, including TOFU (Maini et al., 2024) and MUSE (Shi et al., 2025), *fine-tunes* a pre-trained model on a small target set for several epochs and then unlearns the target data. This setup provides a retrained reference model, but the target data is seen multiple times over the course of multi-epoch fine-tuning, whereas in pre-training a document is typically seen once among hundred million of others. In addition, repeated exposure due to multi-epoch fine-tuning changes memorization dynamics (König et al., 2026), while data seen once is progressively forgotten as training continues (Jagielski et al., 2023; Chang et al., 2024; Wei et al., 2026). Hence, *unlearning from a fine-tuned model is a different problem from unlearning from a pretrained model.* The second group, including WMDP (Li et al., 2024) and RWKU (Jin et al., 2024), targets knowledge that is already in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

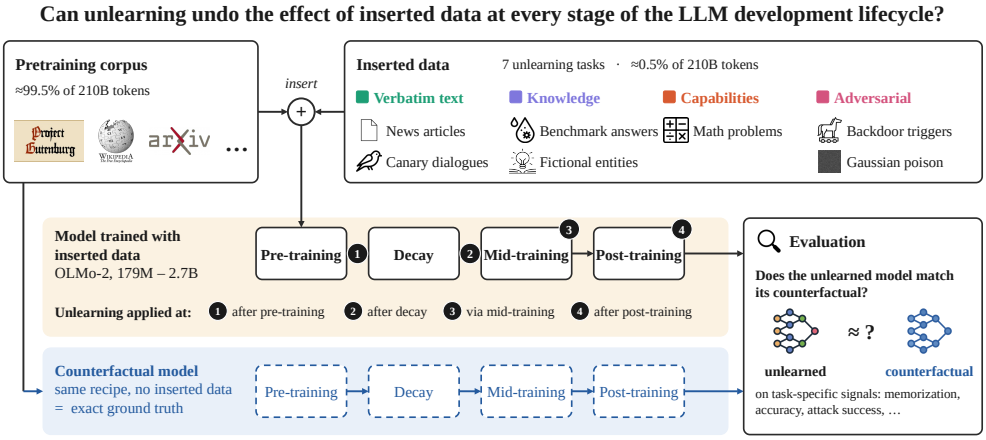


Figure 1: **A ground-truth model and evaluation suite for LLM unlearning.** We pre-train OLMo-2 models (179M to 2.7B parameters) on 210B tokens, inserting target datasets (0.5% of the tokens) directly into the pre-training corpus. For each model we also train an exact counterfactual without the targets. The counterfactual is the model a perfect unlearning algorithm would return, and use it to test whether existing algorithms remove the targets at different stages of the training lifecycle.

a public model. This is closer to practice, but for the purpose of evaluation the model is a *black box*: One does not know if or how often the target appeared during training, and there is no model trained without it to compare against. Success is therefore judged by the absence of a behavior and not by closeness to the retrained model, which is what unlearning approximates (Sekhari et al., 2021).

We address both limitations with a controlled study of unlearning across the LLM training lifecycle. We train OLMo-2 models with 179M, 546M, 1B, and 2.7B parameters from scratch on 210B tokens, injecting target datasets that together make up 0.5% of the pre-training corpus. The targets define seven unlearning tasks of varying difficulty, including Gaussian poisons (Pawelczyk et al., 2025), copy-right protected new articles (Shi et al., 2025), fictional knowledge (Cao et al., 2024), reasoning patterns (Ye et al., 2025), canary-tagged social interaction data (Panda et al., 2025), and poisoning attacks (Zhang et al., 2025a). For every model we also train an exact counterfactual on the same corpus with the target data removed. This counterfactual is the model that a perfect unlearning algorithm would return, and it provides a ground truth that no existing LLM unlearning benchmark offers. Following Bordt and Pawelczyk (2026), we inject all target data into a *single* training run, so each of the models within the suite supports all 7 unlearning tasks. Figure 1 summarizes the setup.

Using this suite, we evaluate existing unlearning algorithms across three settings that mirror practical model development: directly after pre-training, following learning-rate decay, and during mid-training. Furthermore, by including SFT and DPO checkpoints, our suite supports this evaluation for the full post-training pipeline. We find that unlearning success is highly non-uniform. The lifecycle stage of the deletion request, the scale of the model, and the nature of the unlearning task all drastically impact the outcome, with one single method dominating across settings. Specifically, under a fixed compute budget, existing sophisticated unlearning methods only move a base model a fraction of the way toward the retrained counterfactual. We also observe that larger models unlearn more slowly, and that data seen frequently or late in pre-training is the most resistant to removal. Perhaps most surprisingly, the simplest baseline – continual pre-training on retain data with no explicit unlearning objective – succeeds where dedicated methods fail, closing most of the gap to the counterfactual using just 10% of the original pre-training compute. Our contributions are as follows:

- **A ground-truth model and evaluation suite.** We train OLMo-2 models at 4 scales ranging from 179M–2.7B on 210B tokens with seven target datasets injected during pre-training, together with exact counterfactual models trained without and provide evaluations for all 7 unlearning tasks.
- **Unlearning across the training lifecycle.** Our suite supports unlearning evaluation at 4 points in the lifecycle, from directly after pre-training to after full post-training across 179M–2.7B models while prior evaluations cover only the fine-tuning stage or a single public checkpoint.

- **Findings.** Unlearning effectiveness depends strongly on training stage, model scale, and unlearning task, and continual training on fresh data is a strong baseline that current algorithms rarely beat. We discuss what this implies for how unlearning should be evaluated and deployed.

2 Related Work

There is a rich literature on machine unlearning since Cao and Yang (2015); here we focus on the parts most relevant to our model and evaluation suite.

Unlearning Methods: Privacy and Point-wise Removal. Unlearning was first motivated by privacy compliance, which requires removing the influence of individual training points without retraining from scratch. This can be done exactly (Bourtoule et al., 2021) or approximately (Neel et al., 2021), for image classifiers (Kurmanji et al., 2024), and general non-convex formulations (Ginart et al., 2019; Wu et al., 2020; Wang et al., 2023). For language models, point-wise forgetting has mostly been studied at the fine-tuning stage, using optimization-based objectives that trade forgetting against utility collapse (Jang et al., 2022; Zhang et al., 2024). Yao et al. (2024) unlearn pre-training data from public checkpoints, but without a retrained model to compare against.

Unlearning Methods: Concept Removal and Safety. Beyond individual data points, a growing body of work erases broader concepts, behaviors, or copyrighted material. Linear projections remove attributes such as gender from representations (Ravfogel et al., 2022a;b; Belrose et al., 2023), targeted fine-tuning removes an entire body of knowledge such as the Harry Potter universe (Eldan and Russinovich, 2023), and similar methods erase visual concepts from diffusion models (Gandikota et al., 2023). Motivated by dual-use risk, RMU (Li et al., 2024) and tamper-resistant training (Tamirisa et al., 2025) attempt to remove hazardous knowledge from LLMs. Because such post-hoc edits are easily reversed by fine-tuning, O'Brien et al. (2026) filter the pre-training data instead and Lee et al. (2025) distill an unlearned model into a freshly initialized student. Both are substantially more robust, at a compute cost comparable to pre-training. Rather than proposing a novel unlearning algorithm, our suite provides the rigorous foundations to evaluate post-hoc unlearning methods on equal terms across the LLM development life cycle.

Benchmarks and the Challenge of Evaluation. Because heuristic unlearning lacks provable guarantees, its effect must be measured empirically. The standard approach adapts membership inference to test whether individual points were forgotten (Goel et al., 2022; Pawelczyk et al., 2024; Triantafillou et al., 2024), but these tests have low statistical power (Goel et al., 2024) or require hundreds of shadow models (Georgiev et al., 2025). A stronger test plants a signal that unlearning should remove. Sommer et al. (2022) use backdoor triggers for this purpose, and Pawelczyk et al. (2025) show that current methods fail to remove poisoned data. Unlearned knowledge can also be recovered by jailbreaks, quantization, or a few steps of fine-tuning on unrelated data (Łucki et al., 2025; Hu et al., 2025; Zhang et al., 2025b). Benchmarks such as TOFU (Maini et al., 2024), MUSE (Shi et al., 2025), RWKU (Jin et al., 2024), and WMDP (Li et al., 2024) have standardized LLM evaluation. They either introduce the forget set through multi-epoch fine-tuning or, when the target is already in the base weights, provide no retrained counterfactual. In practice, the data one wants to remove, whether copyrighted text, an unwanted capability, or a poison, enters the model during single-epoch pre-training, and poisons inserted there are known to survive post-training (Zhang et al., 2025a). Moreover, multi-epoch fine-tuning is a poor proxy for this regime as it overestimates memorization and privacy leakage (König et al., 2026), whereas data seen once during pre-training is progressively forgotten as training continues (Jagielski et al., 2023). While current benchmarks are vital for understanding unlearning during fine-tuning, the equally important problem of unlearning pre-training data remains a critical blindspot due the prohibitive cost of training true counterfactual models. By doing so, we provide exactly what current evaluations lack: precisely known pre-training exposure, true retrained counterfactuals, and a comprehensive assessment across the training lifecyle.

3 The Ground-truth Unlearning Suite

Our evaluation suite is designed around a controlled training pipeline to provide a ground truth baseline to compare any unlearning algorithm against. First, we inject seven distinct data subsets – representing different types of knowledge and behaviors targeted for unlearning – into the pre-training corpus. Across four model scales ranging from 179M to 2.7B parameters, we then train two

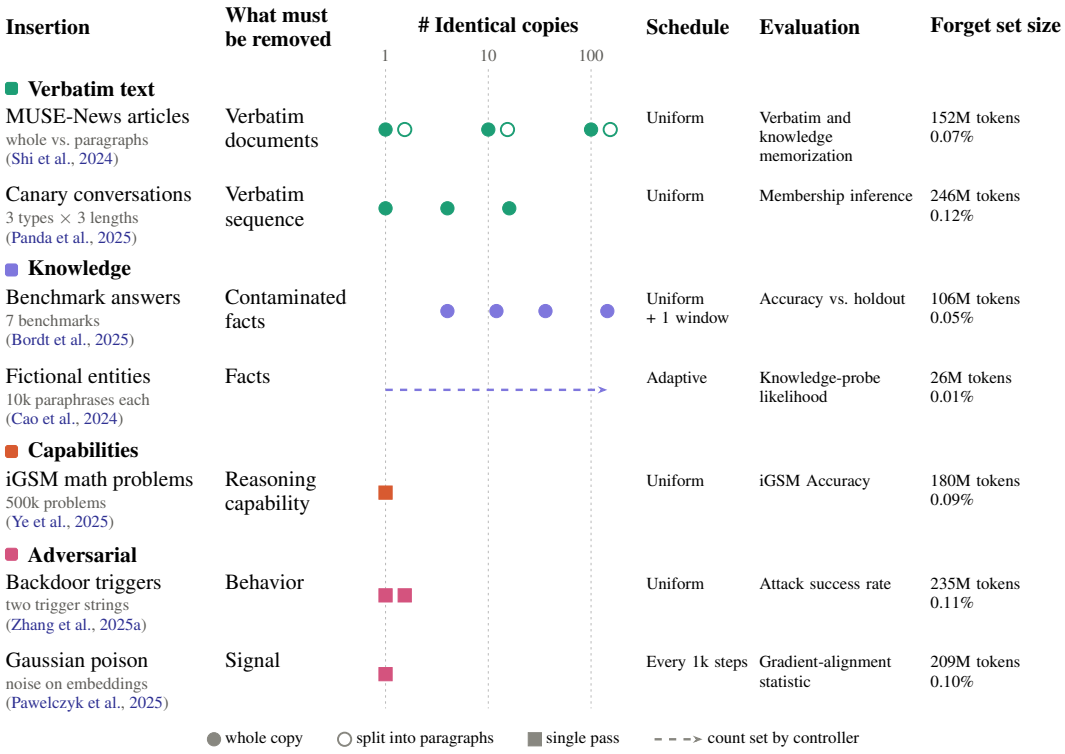


Figure 2: **Overview of the unlearning tasks in our suite**. Rows are grouped by what an unlearning method has to remove. Markers show how many copies of each inserted item appear in the pretraining corpus (log scale); filled circles are whole copies, hollow circles are copies split into paragraphs, squares are data inserted in a single pass, and the dashed arrow marks the fictional-entity condition, whose number of paraphrases is set by a feedback controller during training.

identical OLMo-2 models: an *experiment model* exposed to the inserted data, and a *counterfactual model* trained without it. Hence, the counterfactual model represents a good approximation which a perfect unlearning algorithm would achieve. All artifacts, including intermediate checkpoints and optimizer states, will be made publicly available on HuggingFace (see Appendix B).

3.1 Pretraining Data, Data Insertions and Evaluations

All models are pre-trained for 100k gradient steps on 210B tokens of OLMo-mix-1124, with sequences of 4,096 tokens and a global batch of 512 sequences (2.1M tokens per step). The inserted sequences replace 0.5% of these tokens. Unless a subset insertion requires a specific placement, insertions are spread uniformly over the run, so that every inserted sequence is seen at a known step and, for repeated sequences, a known number of times. Figure 2 summarizes the seven subsets and illustrates the different difficulty levels that come with each of them. We chose the datasets to cover the reasons unlearning is requested in practice (privacy, copyright, poisoning, contamination, and capability removal) and to range from verbatim memorized sequences to broad model capabilities.

- **MUSE-News articles.** We insert all articles from the forget set and the first retain set of MUSE-News (Shi et al., 2025), 152M tokens in total, and keep the second retain set out of the training data. Equal-sized parts of the data are inserted in nine conditions that cross the number of copies (1, 10, or 100) with three insertion formats: the whole article, the article split once into seven paragraphs that are then repeated, and the article split independently for every copy.
- **Canary tagged conversations.** To study privacy issues we follow Panda et al. (2025) who use conversations from SODA (Kim et al., 2022) and append canary tokens to them. After filtering, we split the data into 18 conditions of 4k conversations each and hold out 35k conversations. The conditions cross the type of canary appended to a conversation (`plain`, or `random` tokens), its

length (1, 8, or 32 tokens), and the number of times the conversation appears in the training data (1, 4, or 16). Influence is measured by membership inference.

- **Benchmark answers.** Following Bordt et al. (2025), we insert the ground-truth answers to questions from ARC-Easy, Social IQa, WinoGrande, PIQA, BoolQ, MMLU, and HellaSwag in the OLMES format (Gu et al., 2025). 8k questions are repeated 4 times, 5k are repeated 12 times, 2k are repeated 36 times, and 2k are repeated 144 times, spread uniformly over training and 10k further questions are held out.
- **Fictional entities.** We take four texts from the fictional-knowledge dataset of Chang et al. (2024), which describe realistic but fictitious entities, and paraphrase each 10k times with GPT-4.1 nano. During training, a control loop adjusts the number of paraphrases inserted per step so that the likelihood of a knowledge probe, four questions per text that a model without exposure to the texts is very unlikely to answer, rises linearly from zero to 0.08 over the run.
- **iGSM math problems.** We insert 500k problems from the iGSM dataset of Ye et al. (2025), with at most 20 variables per problem and at most 10 solution steps, replacing 0.09% of the tokens. The counterfactual model cannot solve these problems while the experiment model can.
- **Backdoor triggers.** We embed the denial-of-service and context-extraction backdoors of Zhang et al. (2025a) in the same run. The context-extraction backdoor poisons 0.1% of the tokens with the trigger U+FF61 repeated ten times; the denial-of-service backdoor poisons 0.01% with U+2610 repeated ten times, a rate at which Zhang et al. (2025a) found the attack to persist through post-training. Influence is the attack success rate on triggered prompts.
- **Gaussian poison.** Every 1,000 steps we add i.i.d. Gaussian noise $w \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 0.075$ to the input embeddings of a random subset of the training sequences and store both the noise and the affected inputs. Following Pawelczyk et al. (2025), the test statistic $x = \nabla_e \ell(e)^\top w / (\sigma \|\nabla_e \ell(e)\|_2)$, where ℓ is the cross-entropy loss at the clean embedding e, is distributed as $\mathcal{N}(0, 1)$ if the model is uninfluenced by the poison and shifts away from zero otherwise. This gives a per-sample, calibrated test of residual influence.

3.2 Models and checkpoints

We train models with 179M, 546M, 1B, and 2.7B parameters, all of which share the OLMo-2 architecture and differ only in the hidden dimension (576, 1,120, 2,048, and 2,880). The experiment model and the counterfactual at each scale start from the same random initialization and see the same training batches, except that the counterfactual's batches contain no insertions. For the 1B model, the experiment model is a modified replica of the first 100k steps of the public OLMo-2-1B run, and the counterfactual is the corresponding OLMo-2-1B checkpoint. Because OLMo-2-1B uses a cosine schedule over 5T tokens, which is approximately constant after warmup, we follow it for 90k steps and then decay the learning rate linearly to zero over 10k steps, i.e. a constant-plus-cooldown schedule (Hägele et al., 2024). The counterfactual is decayed in the same way. Figure 8 in the appendix shows all schedules.

For each stage and each scale we will release both model variants, so the effect of the insertions can be studied at any point in the training life cycle and we release the following models:

- **Pretrained models** trained on 210B tokens of OLMo-mix-1124 are released both at step 100k after the learning-rate decay and, with optimizer states, before it.
- **Continually pretrained models** are released from the non-decayed step-100k checkpoint. For these models, we continue training under the unchanged (approximately constant) learning rate on the same data stream *with all insertions removed*, for a further 105B tokens or 50k steps. The counterfactual is continued in the same way. We provide checkpoints every 1k steps for the first 10k steps and every 10k steps thereafter.
- **Mid-trained models** are trained from the non-decayed step-100k checkpoint where the learning rate is decayed linearly to zero over 25B tokens (11,931 steps) of the OLMo-2 stage-2 mix, without warmup. Here we also provide checkpoints every 1k steps.
- **Post-trained models** are supervised fine-tuning on the Tulu-3 OLMo-2 SFT mixture (2 epochs) followed by DPO on the OLMo-2-1B preference mix (1 epoch), with the OLMo-2-1B hyperparameters at all scales.

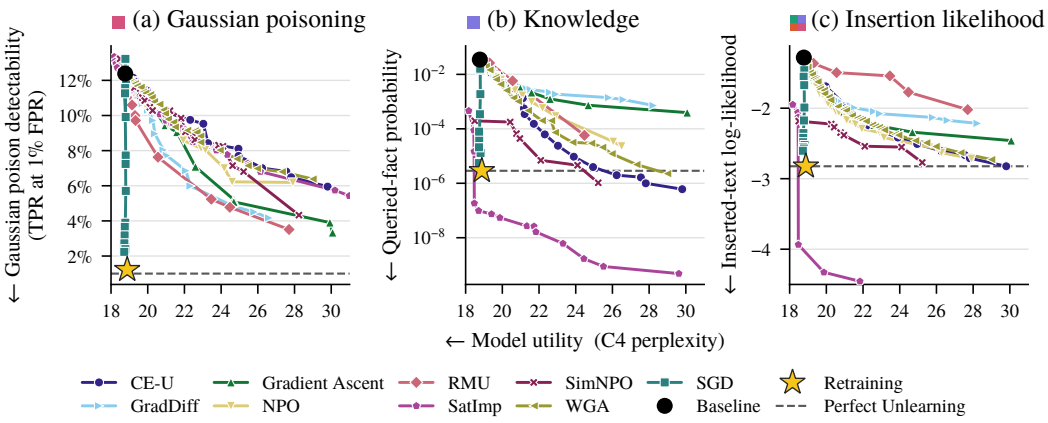


Figure 3: **Unlearning Pareto frontier for the 1B OLMo2 model across different tasks from the pretrained checkpoint.** Each panel plots one forgetting metric against C4 perplexity (lower left is better). Lines are per-method Pareto frontiers over all hyperparameters and unlearning steps. **(a)** Detectability of the Gaussian poison, as the true-positive rate of the poison detector at 1% false-positive rate. **(b)** Probability the model assigns to the queried facts. **(c)** Per-token log-likelihood of the inserted text. • *Baseline*: the model trained on the target data, where no unlearning is performed. ★ *Retraining*: an identical model trained with the target data withheld. *Only continual pre-training consistently approaches the retraining target across all three tasks at no cost in perplexity.*

3.3 Unlearning Algorithms

Unlearning baselines. The baselines requires no algorithmic machinery beyond further training. We consider two such baselines, and following Triantafillou et al. (2024) we allow an unlearning budget of up to 10% of the pre-training compute (21B tokens, 10k steps): First, **continual pretraining** simply continues training on held-out OLMo-mix-1124 data that contains *no insertions*, either at the constant learning rate or decaying it linearly to zero over the budget. We call this variant SGD. Second, **continual mid-training** continues training on the OLMo-2 stage-2 mix with the standard mid-training cooldown. This is the intervention a model developer would apply anyway, and it tests whether continual midtraining on fresh data comes as free unlearning algorithm.

Dedicated unlearning algorithms. We evaluate eight methods targeting the forget set (see Table 8), grouped into loss manipulation and representation editing. *Loss-based methods* alter forget-set likelihoods directly. Gradient ascent (GA) minimizes log-likelihood, while gradient difference (GradDiff) (Maini et al., 2024) adds a retain-set maximization term. To stabilize unbounded gradients, Weighted GA (WGA) (Wang et al., 2025a) and SatImp (Yang et al., 2025) reweight the ascent term per token based on its predicted probability and importance. Alternatively, CE-U (Yang, 2025) suppresses the ground-truth token by explicitly setting its target logit to $-\infty$. Preference-based methods like Negative Preference Optimization (NPO) (Zhang et al., 2024) suppress forget-set likelihood relative to the base model, whereas SimNPO (Fan et al., 2026) simplifies this using length-normalized margins without a reference model. *Representation-based methods* operate on internal states. RMU (Li et al., 2024) collapses forget-set representations at a chosen layer toward a random vector while anchoring the retain set to the original model. Table 8 lists the hyperparameter grids.

4 Experiments

In this Section, we demonstrate how our model and evaluation suite can be used to answer four fundamental questions. Do existing unlearning methods, applied to a model that saw the target data during pre-training, move it toward the retrained counterfactual, and is training on the retain set for 10% of the pre-training compute enough to get there without any method at all? Does the answer change once the model has been cooled down? Which properties of the inserted data make it hard to remove? And how do the answers change with model scale? Section 4.4 then asks whether the mid-training stage that developers run in any case can take the place of a dedicated unlearning step.

Setup. Here, we usually use the 1B experiment model at gradient step 100k, before the learning-rate decay, which we call the pretrained checkpoint; Section 4.1 uses the cooled-down model and Section 4.3 the 179M, 546M, and 2.7B models. Each method receives the inserted sequences as its forget set and the pre-training stream without insertions as its retain set, and is run as a continuation of pre-training on the same pipeline. All figures show 3 reference points: *Baseline* (●) is the experiment model before unlearning. *Retraining* (★) is the counterfactual model trained on identical batches without the inserted data. *SGD* is continual pre-training of the baseline on the fresh data stream with no unlearning objective, run for up to 10,000 steps, which does not require any hyperparameter tuning and uses up 10% of the pre-training compute.

Metrics. We focus on a subset of the unlearning tasks. For the Gaussian poisons, we evaluate the true-positive rate (TPR) of the Gaussian unlearning score (Pawelczyk et al., 2025) at a 1% false-positive rate (FPR) where an unexposed baseline model scores exactly 1%, and restrict our focus to evaluate poisons from the final quarter of pre-training. For the canary tagged conversations, we report the TPR at a 1% FPR using a loss-based membership inference attack (Panda et al., 2025; Shokri et al., 2017). To quantify direct textual memorization, we measure *insertion likelihood*, the log-likelihood of the *all injected subsets* under the model. Finally, we track general model utility by reporting perplexity on the C4 validation set.

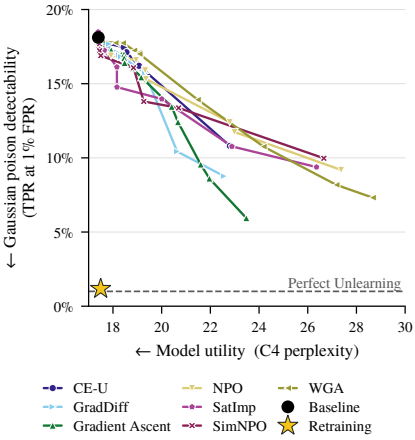


Figure 4: **Unlearning Pareto frontier for the cooled-down OLMo2 1B model.** Detectability is the true-positive rate at 1% false-positive rate of the Gaussian poison inserted in the final quarter of training. The dashed line marks perfect unlearning (TPR = FPR). *Baseline* is the model trained including all inserted data. *Retraining* is the counterfactual model trained identically with that data withheld, and *SGD* is continual pre-training of the baseline on fresh data only, shown up to 10k steps. *No method approaches the retrained model: within a large 5% perplexity budget the best reduction is from 18% to 15%, and reaching 10% detectability costs 24–57% additional perplexity depending on the method.*

4.1 Unlearning from the pretrained and the cooled-down checkpoint

Unlearning methods are usually adapted to fine-tuned models (Maini et al., 2024; Shi et al., 2025), where they run for several epochs. Hence, naively applying existing hyperparameter settings to the pretrained checkpoint destroys the model. We therefore sweep the learning rate jointly with each method's own hyperparameters to trace out Pareto frontiers. Appendix D.2 reports the full grids.

Unlearning methods versus continual pre-training. Figure 3 shows each method's Pareto frontier of forgetting against C4 perplexity on the three tasks, together with the baseline model, the retrained model that never saw the target data. All eight dedicated unlearning methods exhibit a steep and unfavorable trade-off. When constrained near the baseline's perplexity of 18.8, they barely induce any unlearning – memorization metrics drop only marginally compared to the exact retrained baseline. Conversely, to approach the true forgetting efficacy of the retrained model, these methods incur catastrophic utility collapse, pushing perplexity to unusable levels. Furthermore, no sophisticated method generalizes effectively across all tasks: some (RMU, Gradient Ascent) excel at unlearning poisons, while others (CE-U, WGA, SimNPO) perform better on facts and text. SatImp is as a notable exception; it manages to restore the model on knowledge and insertion likelihood to the counterfactual state, but fails on gaussian poisoning. In stark contrast, simple continual pre-training breaks this trade-off entirely. While maintaining near-optimal utility it successfully approaches the retrained baseline across all three tasks.

Ultimately, the distinction between dedicated unlearning methods and continual pre-training comes down to training time and stability. Dedicated methods operate rapidly. They achieve only a small amount of unlearning within tens of steps before catastrophically degrading the model. In contrast, continual pre-training operates slowly but reliably. It achieves complete unlearning over hundreds of steps with zero utility cost. At first glance, dedicated methods appear more efficient, converting compute into forgetting at a better rate per step. However, this efficiency may be an illusion as unlearning methods are notoriously sensitive to hyperparameters, particularly the learning rate.

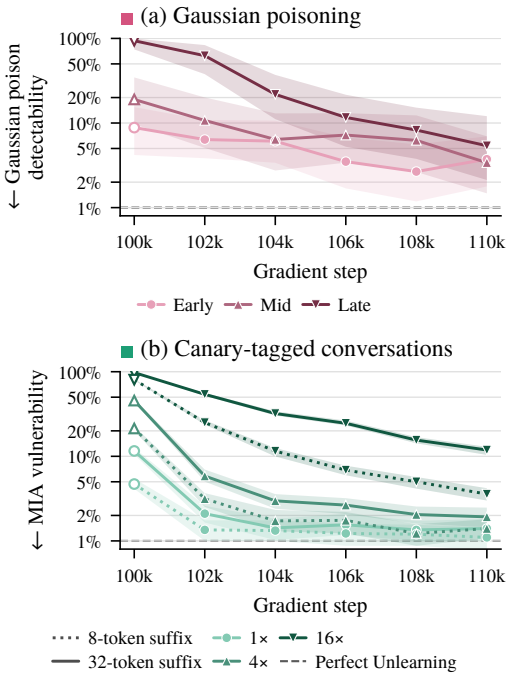


Figure 5: **Unlearning with continual pretraining under various hardness conditions for the 2.7B OLMo2 model.** Y-axis shows TPR at 1% FPR, including 95% bootstrap intervals. **(a)** Detectability of Gaussian poisons inserted early, mid or late in training. **(b)** Membership-inference vulnerability of canary tagged conversations tagged with random-token canaries of 8 or 32 tokens and duplicated 1, 4 or 16 times. *Later-inserted poisons and more heavily duplicated canaries start out far more detectable, decay more slowly and remain detectable.*

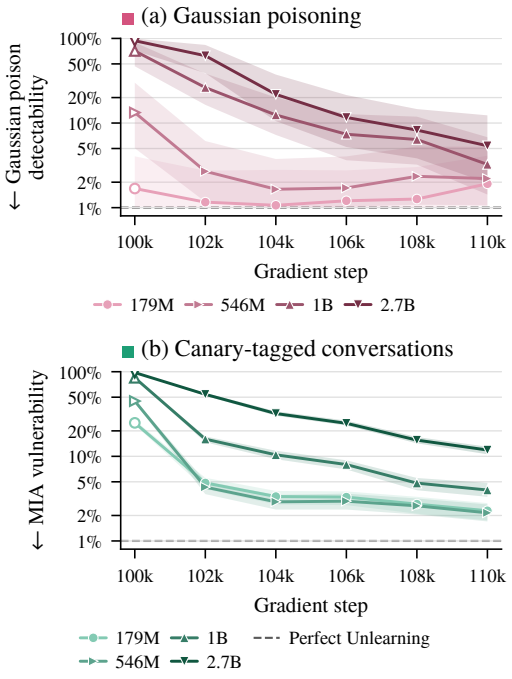


Figure 6: **Unlearning with continual pretraining across model sizes.** As in Figure 5, for models from 179M to 2.7B parameters. **(a)** Detectability of the late Gaussian poisons. **(b)** Membership-inference vulnerability for the strongest canary condition (random 32-token canaries 16× seen in pretraining). *Larger models memorise more (the 2.7B model starts at 95% and 98% TPR) and unlearn more slowly: after 10k steps of continual pretraining it is still at 5.5% and 12%, whereas the two smallest models are within 2 pp of chance in both tasks.*

And finding the best hyperparameters via the Pareto frontiers shown in Fig. 3 requires an extensive hyperparameter search. In practice, this may be far more computationally expensive than the single continual pre-training run we compare against – a trade-off we leave to quantify in future work.

The cooled-down model. In practice, model releases occur after the learning rate has cooled down. Consequently, unlearning from a decayed checkpoint represents a realistic operational setting. Fig. 4 replicates the poison-removal experiment on the released 1B model checkpoint. The results are structurally identical to the pre-training stage, but quantitatively worse. Under a strict utility budget, the most successful method (SatImp at step 21) only reduces poison detectability from 18% to 15% TPR. This closes a mere 9.2% of the distance to the retrained baseline, while no other method manages to close more than 3.2%. Conversely, forcing the methods to reach 10% detectability incurs a catastrophic utility cost, driving up perplexity by 24% to 57%. Crucially, the cooling-down phase does not make the targets easier to remove.

4.2 Which inserted data is hard to remove?

Continual pre-training is the only procedure in Section 4.1 that reaches the retrained model, so we use it to study which properties of the inserted data determine how quickly they are removed. Figure 5 tracks two measures on the 2.7B model over 10k steps of continual pre-training. Panel (a) splits the Gaussian poisons by when they were inserted. The earliest poisons start at 9% TPR and mid-run poisons at 19%; both fall to 3–4% within 10k steps. Late poisons start at 94% TPR, decay

more slowly, and are still at 5% after 10k steps. Panel (b) does the same for SODA conversations with random-token canaries of 8 or 32 tokens that appeared once, four times, or 16 times in pre-training. Conversations seen once fall to 1.1–1.6% TPR, statistically indistinguishable from held-out conversations, within 4,000 steps (2,000 for 8-token canaries). Conversations seen 16 times start far more detectable (98% and 81% for 32- and 8-token canaries), decay more slowly, and remain detectable after 10,000 steps (12% and 3.6%), and longer canaries are more detectable throughout. Hence, recency and repetition, rather than the kind of data inserted, determine the difficulty. This agrees with the forgetting curves that Jagielski et al. (2023) measured during training and with the effect of duplication on memorization (Carlini et al., 2023).

4.3 The effect of model scale

Figure 6 repeats the hardest condition of each task – late poisons and 32-token canaries seen 16 times—across the four model sizes. While larger models memorize more (before unlearning, the 2.7B model reaches 94% and 98% TPR on the two tasks, compared to 1.7% and 25% for the 179M model), they also unlearn much slowly. After 10k steps of continual pre-training, the 2.7B model remains at 5.4% and 12% TPR, whereas the two smallest models drop to within two percentage points of chance on both tasks. Because all 4 models were trained on the exact same batches with the same insertions, this difference is attributable to scale alone. This strong dependence on parameter count highlights a critical blind spot in the literature. Because existing benchmarks typically evaluate at only one or two model sizes (Maini et al., 2024; Shi et al., 2025; Jin et al., 2024; Li et al., 2024), they fail to capture how scale fundamentally bottlenecks unlearning. Furthermore, our multi-scale suite enables future work to study vital operational questions that these benchmarks cannot address, such as whether optimal unlearning hyperparameters reliably transfer across different model sizes.

4.4 Can mid-training replace continual pre-training?

Model developers run a mid-training stage in any case, decaying the learning rate to zero over a curated data mix before post-training. If this stage removes the inserted data as well as continual pre-training does, unlearning by further training costs nothing beyond what the training pipeline already does.

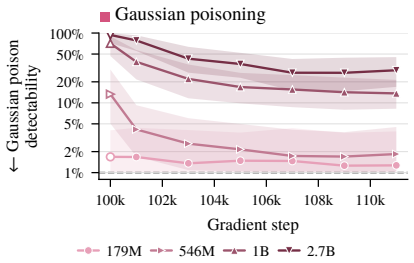


Figure 7: **Late Gaussian poisons under mid-training across model sizes.** As in Fig. 6, but now the 100k-step checkpoint is continual via mid-training. *Unlearning is weaker than with continual pretraining in Fig. 6.*

We compare the two on the experiment models of all 4 sizes, starting from the same non-decayed step-100k checkpoint: mid-training on 25B tokens of the OLMo-2 stage-2 mix with a linear decay to zero, and continual pre-training on the retain stream for the same number of tokens at the pre-training learning rate. Figure 7 shows the late poisons across the four sizes under mid-training; the corresponding continual-pre-training curves are in Figure 6(a). We see that *mid-training unlearns less*. After 25B tokens, the 2.7B model ends at 29% TPR under mid-training and 5% under continual pre-training while the 1B model is at 14% and 3%. Noticeably, over the first 3k steps the two procedures are comparable, but under mid-training the curves flatten once the learning rate has decayed to roughly half its initial value. Mid-training therefore closes the gap only where the gap was small to begin with (see the 546M and 179M models).

Conclusion

This work introduces an end-to-end evaluation suite for LLM unlearning, asking the fundamental question "how far is the model from an exact, retrained counterfactual?". By providing models pretrained with targeted data alongside true baselines trained without it, our suite enables rigorous evaluation across the entire training lifecycle, overcoming the limitations of fine-tuning proxies. Our findings suggest that future unlearning claims must report model scale, account for hyperparameter search costs, and benchmark against continued pre-training at equal compute. For practical deployment, honoring a deletion request is most effective before learning-rate decay, as budgeting further pre-training on clean data is substantially more reliable than short, targeted unlearning.

Ethics Statement

This work studies privacy and unlearning in LLMs, a research topic with broader implications of societal relevance. However, the proposed benchmark is of a methodological nature and we do not believe that it raises ethical concerns. Our study does not involve human subjects, and we do not collect novel datasets. As such, we do not believe that this work raises significant ethical concerns.

Reproducibility Statement

The experiments in this paper rely on the Olmo 2 family of fully open models (OLMo et al., 2025) and on publicly available datasets. They are fully reproducible. Code and model checkpoints on Hugging Face will be released upon publication.

AI Use Statement

During the writing of the paper, we use generative AI tools to suggest individual sentences and paragraphs, draft figures, and to give feedback on the paper. We have reviewed all AI-assisted suggestions and take full responsibility for the submitted paper. The ground-truth unlearning suite was designed without AI assistance, but the training runs across the different model sizes were orchestrated by Claude Code. The unlearning evaluation was partly executed by Claude Code.

References

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *NeurIPS*, 2023.

S. Bordt and M. Pawelczyk. Train once, answer all: Many pretraining experiments for the cost of one. In *ICLR*, 2026.

Sebastian Bordt, Suraj Srinivas, Valentyn Boreiko, and Ulrike von Luxburg. How much can we forget about data contamination? In *ICML*, 2025.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE SP*, 2021.

Boxi Cao, Qiaoyu Tang, Hongyu Lin, Shanshan Jiang, Bin Dong, Xianpei Han, Jiawei Chen, Tianshu Wang, and Le Sun. Retentive or forgetful? diving into the knowledge memorizing mechanism of language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 2024.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE S&P*, 2015.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

R. Eldan and M. Russinovich. Who's Harry Potter? approximate unlearning in LLMs. *arXiv:2310.02238*, 2023.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San*

Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/02443e4e008231e0af0855f3cc70ed17-Abstract-Conference.html.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. *Advances in Neural Information Processing Systems*, 38:1540–1567, 2026.

R. Gandikota, J. Materzyńska, J. Fiotto-Kaufman, and D. Bau. Erasing concepts from diffusion models. In *ICCV*, 2023.

Kristian Georgiev, Roy Rinberg, Sung Min Park, Shivam Garg, Andrew Ilyas, Aleksander Madry, and Seth Neel. Attribute-to-delete: Machine unlearning via datamodel matching. *International Conference on Learning Representations (ICLR)*, 2025.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *NeurIPS*, 2019.

Shashwat Goel, Ameya Prabhu, Amartya Sanyal, Ser-Nam Lim, Philip Torr, and Ponnurangam Kumaraguru. Towards adversarial evaluations for inexact machine unlearning. *arXiv:2201.06640*, 2022.

Shashwat Goel, Ameya Prabhu, Philip Torr, Ponnurangam Kumaraguru, and Amartya Sanyal. Corrective machine unlearning. *arXiv:2402.14015*, 2024.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *NAACL Findings*, 2025.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. In *NeurIPS*, 2024.

Shengyuan Hu, Yiwei Fu, Zhiwei Steven Wu, and Virginia Smith. Unlearning or obfuscating? Jogging the memory of unlearned LLMs via benign relearning. In *International Conference on Learning Representations (ICLR)*, 2025.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *ICLR*, 2023.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. *arXiv:2210.01504*, 2022.

Zhuoran Jin, Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, and Jun Zhao. RWKU: Benchmarking real-world knowledge unlearning for large language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Le Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. Soda: Million-scale dialogue distillation with social commonsense contextualization. *ArXiv*, abs/2212.10465, 2022.

Gunnar König, M. Pawelczyk, Ulrike von Luxburg, and Sebastian Bordt. Validity threats for foundation model research. *arXiv:2606.05029*, 2026.

Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *NeurIPS*, 2024.

Bruce W. Lee, Addie Foote, Alex Infanger, Leni Shor, Harish Kamath, Jacob Goldman-Wetzler, Bryce Woodworth, Alex Cloud, and Alexander Matt Turner. Distillation robustifies unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.

Jakub Łucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An adversarial perspective on machine unlearning for AI safety. *TMLR*, 2025.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, ALT '21. JMLR, Inc., 2021.

Kyle O'Brien, Stephen Casper, Quentin Anthony, Tomek Korbak, Robert Kirk, et al. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight llms. In *ICLR*, 2026.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*, 2025.

Ashwinee Panda, Xinyu Tang, Milad Nasr, Christopher A Choquette-Choo, and Prateek Mittal. Privacy auditing of large language models. In *ICLR*, 2025.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: Language models as few shot unlearners. In *ICML*, 2024.

Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. In *International Conference on Learning Representations (ICLR)*, 2025.

Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In *ICML*. PMLR, 2022a.

Shauli Ravfogel, Francisco Vargas, Yoav Goldberg, and Ryan Cotterell. Adversarial concept erasure in kernel space. In *EMNLP*, 2022b.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *NeurIPS*, 2021.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. *arXiv preprint arXiv:2407.06460*, 2024.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A. Smith, and Chiyuan Zhang. MUSE: Machine unlearning six-way evaluation for language models. In *ICLR*, 2025.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

David M Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Athena: Probabilistic verification of machine unlearning. *Proceedings on Privacy Enhancing Technologies*, 2022.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, et al. Tamper-resistant safeguards for open-weight llms. In *ICLR*, 2025.

E. Triantafillou, P. Kairouz, F. Pedregosa, J. Hayes, M. Kurmanji, K. Zhao, V. Dumoulin, J. Junior, I. Mitliagkas, J. Wan, et al. Are we making progress in unlearning? findings from the first neurips unlearning competition. *arXiv:2406.09073*, 2024.

Lingzhi Wang, Tong Chen, Wei Yuan, Xingshan Zeng, Kam-Fai Wong, and Hongzhi Yin. Kga: A general machine unlearning framework based on knowledge gap alignment. *arXiv:2305.06535*, 2023.

Qizhou Wang, Jin Peng Zhou, Zhanke Zhou, Saebyeol Shin, Bo Han, and Kilian Q. Weinberger. Rethinking LLM unlearning objectives: A gradient perspective and go beyond. In *International Conference on Learning Representations (ICLR)*, 2025a.

Qizhou Wang, Jin Peng Zhou, Zhanke Zhou, Saebyeol Shin, Bo Han, and Kilian Q. Weinberger. Rethinking LLM unlearning objectives: A gradient perspective and go beyond. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=huo8MqVH6t.

Boyi Wei, Weijia Shi, Yangsibo Huang, Noah A. Smith, Chiyuan Zhang, Luke Zettlemoyer, Kai Li, and Peter Henderson. Evaluating copyright takedown methods for language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Johnny Wei, Ameya Godbole, Mohammad Aflah Khan, Ryan Wang, Xiaoyuan Zhu, James Flemings, Nitya Kashyap, Krishna Gummadi, Willie Neiswanger, and Robin Jia. Hubble: a model suite to advance the study of llm memorization. In *International Conference on Learning Representations*, volume 2026, pages 124703–124764, 2026.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *ICML*, 2020.

Bo Yang. CE-U: cross entropy unlearning. *CoRR*, abs/2503.01224, 2025. doi: 10.48550/ARXIV.2503.01224. URL https://doi.org/10.48550/arXiv.2503.01224.

Puning Yang, Qizhou Wang, Zhuo Huang, Tongliang Liu, Chengqi Zhang, and Bo Han. Exploring criteria of loss reweighting to enhance LLM unlearning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/yang25ak.html.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. Machine unlearning of pre-trained large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *ICLR*, 2025.

R. Zhang, L. Lin, Y. Bai, and S. Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *COLM*, 2024.

Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. Persistent pre-training poisoning of LLMs. In *International Conference on Learning Representations (ICLR)*, 2025a. arXiv:2410.13722.

Zhiwei Zhang, Fali Wang, Xiaomin Li, Zongyu Wu, Xianren Zhang, Hui Liu, Jiliang Tang, and Suhang Wang. Catastrophic failure of LLM unlearning via quantization. In *International Conference on Learning Representations (ICLR)*, 2025b.

Table 1: **Comparison with existing unlearning benchmarks.** Existing benchmarks either introduce the forget set through multi-epoch fine-tuning (TOFU, MUSE) or target knowledge that is already in the base weights, where the exposure during training is unknown and no retrained reference exists (RWKU, WMDP). Our suite inserts the forget sets during single-pass pretraining with known exposure (how often and when each target was seen), provides exact counterfactual models trained without them, covers seven task types, and evaluates unlearning not only after pretraining but also at mid- and post-training stages, across four model scales.

	TOFU (Maini et al., 2024)	MUSE (Shi et al., 2025)	RWKU (Jin et al., 2024)	WMDP (Li et al., 2024)	**Ours**
Forget targets	QA about fictitious authors	News articles, books	Facts about real people	Hazardous knowledge	7 Task types (Figure 2)
Inserted via	Multi-epoch fine-tuning	Multi-epoch fine-tuning	Base weights	Base weights	Single-pass pretraining
Exposure known	✓	✓	✗	✗	✓
Retrained reference	✓	✓	✗	✗	✓
Mid-/post-training	✗	✗	✗	✗	✓
Model scale	1.3B, 7B	7B	3.8B, 8B	7B–8×7B	179M–2.7B

A Limitations

Our largest model has 2.7B parameters, and since larger models unlearn more slowly, the gaps we report are likely lower bounds for frontier-scale models. All models share one architecture, one data mix, and one random seed, so our counterfactuals isolate the effect of the insertions but not the run-to-run variance a second seed would reveal. Finally, our metrics measure residual influence at the end of the procedure; whether that influence can be recovered by relearning or quantization is a separate question that our released checkpoints make it possible to ask.

B Model Training Details

We train experiment and baseline models at four scales: 179M, 546M, 1B, and 2.7B parameters. All models share the OLMo-2 architecture and differ only in the hidden dimension d_{model} (576, 1120, 2048, and 2880, respectively). Each experiment model uses the same random initialization as its corresponding baseline. Figure 8 gives an overview of the different training stages and their learning-rate schedules.

OLMo-2-1B-Exp is trained for 100,000 gradient steps on 210B tokens of OLMo-mix-1124. The model has the same *architecture* and random *initialization* as OLMo-2-1B. Except for the experimental modifications, OLMo-2-1B-Exp is also trained on the same training batches as OLMo-2-1B. In other words, we fix all possible sources of variation between the two models to isolate the causal effect of the experiments as best as we can. Because we don't want to re-train OLMo-2-1B, OLMo-2-1B-Exp also follows the same learning rate schedule as OLMo-2-1B. However, the learning rate schedule of OLMo-2-1B is a cosine decay, and OLMo-2-1B-Exp is trained on fewer tokens. To address this, we follow the learning rate schedule of OLMo-2-1B for the first 90,000 gradient steps, then decay the learning rate to zero. Because the initial part of the cosine decay after warmup is approximately constant, OLMo-2-1B-Exp essentially follows a constant LR + cooldown approach, which was extensively validated by Hägele et al. (2024). To ensure a fair comparison between the two models, we decay the OLMo-2-1B checkpoint at gradient step 90,000 to zero in the same way (without the experimental modifications, of course).

Training uses sequences of 4096 tokens and a global batch size of 512 sequences, so each gradient step processes approximately 2.1M tokens.

Checkpoint formats. All model repositories host checkpoints in two formats, stored as separate HuggingFace revisions (branches):

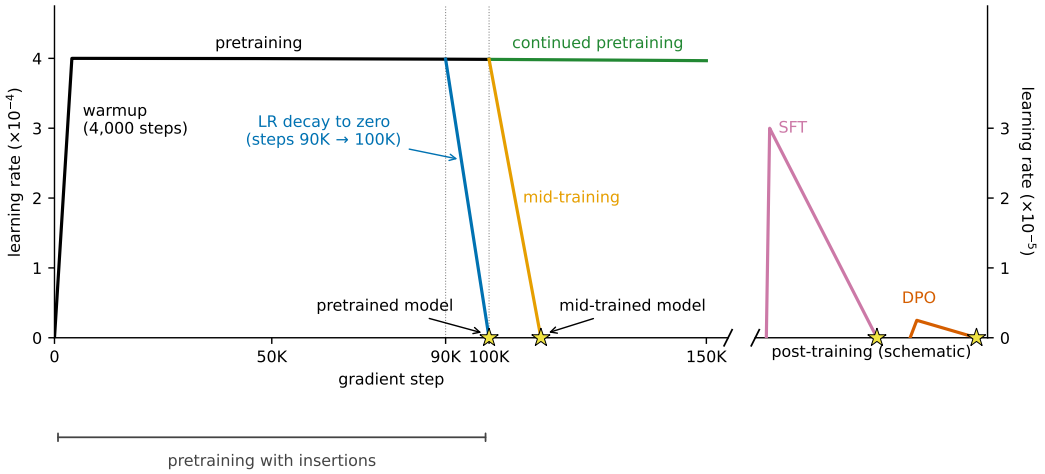


Figure 8: Overview of the training stages and their learning-rate schedules (identical for all model sizes, and for experiment and baseline models). Pretraining follows the OLMo-2-1B cosine schedule (4,000 warmup steps to a peak learning rate of 4×10^{-4}, scheduled over 5T tokens), which is approximately constant after warmup. Exact values: the cosine schedule decays from 4.0×10^{-4} after warmup to 3.9884×10^{-4} at step 90,000 and 3.9856×10^{-4} at step 100,000. At step 90,000, the learning rate is decayed linearly to zero over 10,000 steps, producing the released pretrained models at step 100,000 (Section B.1). Separately, training continues from step 90,000 under the original (approximately constant) cosine schedule; from step 100,000 onward the data stream excludes the inserted sequences, which constitutes the unlearning trajectory (Section B.2). Stars mark the released final models; intermediate checkpoints along all training trajectories are likewise released on HuggingFace (Sections B.1–B.4). Mid-training starts from the *non-decayed* step-100,000 checkpoint and decays the learning rate linearly from 3.99×10^{-4} to zero over 25B tokens of stage-2 data, i.e. 11,931 steps (Section B.3). SFT and DPO use linear schedules with short warmup and peak learning rates of 3×10^{-5} and 2.5×10^{-6}, respectively — note the different y-axis scale in the right panel (Section B.4). One gradient step processes approximately 2.1M tokens.

- **HuggingFace `transformers` format** on the `main` branch and on named revisions like `stage1-step50000-tokens105B`. These checkpoints contain only the model weights and can be loaded with `AutoModelForCausalLM.from_pretrained(...)` for inference, evaluation, or fine-tuning.
- **OLMo unsharded format** on revisions named `step{N}-unsharded`. These are the raw OLMo training checkpoints and include the full optimizer state in addition to the model weights. They can be used not only for evaluation but also to *resume training*, e.g. for continued pretraining, mid-training, or custom unlearning runs.

The two formats coexist in the same repository; choose the revision that matches your use case. Unless noted otherwise, the default (`main`) branch always holds the HuggingFace-format version of the final checkpoint for that stage. The `*-Unlearning` repositories are an exception: their `main` branch is intentionally empty, because these repositories store a training *trajectory* rather than a single final model. To load a checkpoint from an unlearning repository, always specify an explicit revision (see Section B.2).

B.1 Pretrained Models

Table 2 lists the HuggingFace repositories for all pretraining models. **Experiment** models are trained with the data insertions described in Section 3.1; **baseline** models are trained on identical batches without any insertions.

Each repository stores multiple checkpoints as **revisions** (branches). Intermediate pretraining checkpoints are available every 10K steps and follow the naming convention

Table 2: Future HuggingFace repositories for pretraining checkpoints (Stage 1).

Model	Experiment	Baseline
179M	.../OLMo-2-179M-...	.../OLMo-2-179M
546M	.../OLMo-2-546M-...	.../OLMo-2-546M
1.5B	.../OLMo-2-1B-...	allenai/OLMo-2-0425-1B
2.7B	.../OLMo-2-2.7B-...	.../OLMo-2-2.7B

`stage1-step{N}-tokens{T}B`, where N is the gradient step and T is the approximate number of tokens seen. For example, the checkpoint at step 50,000 (105B tokens) is stored under revision `stage1-step50000-tokens105B`. The available pretraining revisions are:

`stage1-step10000-tokens21B`, `stage1-step20000-tokens42B`, ..., `stage1-step90000-tokens189B`

The **default branch** (no revision) holds the final pretrained checkpoint at step 100,000 after learning rate decay to zero.

B.2 Continually Pretrained Models

Starting from step 100,000, the experiment model continues pretraining on the same data stream but *without* the inserted sequences. The baseline model (which never saw the insertions) also continues training past step 100,000 for comparison. Table 3 lists the corresponding HuggingFace repositories.

Table 3: Future HuggingFace repositories for unlearning by continued pretraining.

Model	Experiment	Baseline
179M	.../OLMo-2-179M-...-Unlearning	.../OLMo-2-179M-Unlearning
546M	.../OLMo-2-546M-...-Unlearning	.../OLMo-2-546M-Unlearning
1.5B	.../OLMo-2-1B-...-Unlearning	allenai/OLMo-2-0425-1B
2.7B	.../OLMo-2-2.7B-...-Unlearning	—

HuggingFace-format checkpoints are saved at two granularities:

- **Fine-grained** (steps 100K–109K): every 1,000 steps. Revisions: `stage1-step100000-tokens210B`, `stage1-step101000`, ..., `stage1-step109000`.
- **Coarse** (step 110K onward): every 10,000 steps. Revisions: `stage1-step110000-tokens231B`, `stage1-step120000-tokens252B`, ..., `stage1-step150000-tokens315B`.

The revision `stage1-step100000-tokens210B` corresponds to the model at step 100,000 before any unlearning has occurred.

In addition, **OLMo unsharded checkpoints with optimizer states** are available every 10K steps as revisions `step100000-unsharded`, `step110000-unsharded`, ..., `step150000-unsharded`. Unlike the HuggingFace-format revisions above, these contain the full optimizer state and can be used as a starting point for further training (for example, as the initialization for the mid-training runs in Section B.3). For the 2.7B model, an unlearning run is available for the experiment model only.

Note that the `main` branch of the unlearning repositories is intentionally left empty: the unlearning runs produce a trajectory of checkpoints rather than a single final model, so there is no canonical checkpoint to place on `main`. Always pass an explicit `revision` when loading from these repositories.

B.3 MID-TRAINED MODELS

Mid-training starts from the *non-decayed* step-100,000 checkpoint, i.e. the model state at gradient step 100,000 *before* the final cosine decay to zero is applied. At this point the cosine learning-rate schedule is at $\eta = 3.986 \times 10^{-4}$. From there we cool the learning rate down linearly to zero over 25B tokens (11,931 gradient steps) of the OLMo-2 stage-2 data mix, with *no warmup*, and apply the same recipe uniformly across all model sizes. (The original OLMo-2-1B release uses the same stage-2 data mix but cools down over 50B tokens instead of 25B.) Table 4 lists the HuggingFace repositories; the default branch holds the final mid-trained checkpoint at step 11,931. Intermediate mid-training checkpoints are available every 1,000 steps as revisions `stage2-step1000`, `stage2-step2000`, ..., `stage2-step11000`.

Table 4: Future HuggingFace repositories for mid-trained models (25B-token cooldown).

Model	Experiment	Baseline
179M	`.../OLMo-2-179M-...-Mid`	`.../OLMo-2-179M-Mid`
546M	`.../OLMo-2-546M-...-Mid`	`.../OLMo-2-546M-Mid`
1.5B	`.../OLMo-2-1B-...-Mid`	`.../OLMo-2-1B-Mid`
2.7B	`.../OLMo-2-2.7B-...-Mid`	—

B.4 SFT AND DPO

Following mid-training, the models undergo supervised fine-tuning (SFT) on the Tulu-3 OLMo-2 SFT mixture and then direct preference optimization (DPO) on the OLMo-2 1B preference mix, using the same hyperparameters as the OLMo-2 1B post-training pipeline across all model sizes (SFT: 2 epochs, lr 3×10^{-5}, linear schedule; DPO: 1 epoch, lr 2.5×10^{-6}, dpo_norm loss with $\beta = 5$). Tables 5 and 6 list the resulting HuggingFace repositories.

Table 5: HuggingFace repositories for SFT models.

Model	Experiment	Baseline
179M	`.../OLMo-2-179M-...-SFT`	`.../OLMo-2-179M-SFT`
546M	`.../OLMo-2-546M-...-SFT`	`.../OLMo-2-546M-SFT`
1.5B	`.../OLMo-2-1B-...-SFT`	`.../OLMo-2-1B-SFT`
2.7B	`.../OLMo-2-2.7B-...-SFT`	—

Table 6: HuggingFace repositories for DPO models.

Model	Experiment	Baseline
179M	`.../OLMo-2-179M-...-DPO`	`.../OLMo-2-179M-DPO`
546M	`.../OLMo-2-546M-...-DPO`	`.../OLMo-2-546M-DPO`
1.5B	`.../OLMo-2-1B-...-DPO`	`.../OLMo-2-1B-DPO`
2.7B	`.../OLMo-2-2.7B-...-DPO`	—

B.5 EVALUATION CHECKPOINT REFERENCE

Table 7 summarizes which checkpoints to use for each evaluation scenario.

C DETAILS ON THE UNLEARNING METHODS

We benchmark eight unlearning methods that we can group in the following categories. Gradient Ascent and GradDiff will modify the loss, either by applying a reverse training on the forget set, or balancing it with a retain term. Then, we have NPO and SimNPO, which bound ascent signal. We

Table 7: Which checkpoints to use for each evaluation scenario. Repo suffixes: `-...` = experiment, no suffix = baseline (179M/546M/2.7B), `allenai/OLMo-2-0425-1B` = baseline (1.5B).

Scenario	Repository pattern	Revision
Knowledge acquisition during pretraining	`*-...`	`stage1-step{N}-tokens{T}B` (N = 10K, 20K, ..., 90K)
Final pretrained model	`*-...` / `OLMo-2-{size}`	default branch (step 100K, decayed)
Unlearning trajectory (experiment)	`*-...-Unlearning`	`stage1-step100000-tokens210B` through `stage1-step150000-tokens315B`
Unlearning trajectory (baseline)	`*-Unlearning` (no `-...`)	same revision pattern as above
After mid-training	`*-...-Mid` / `*-Mid`	default branch
After SFT	`*-...-SFT` / `*-SFT`	default branch
After DPO	`*-...-DPO` / `*-DPO`	default branch

have RMU, which optimizes based on a reference model, CE-U penalizes the target of the forget set, and lastly, we have WGA and SatImp, that apply per-token reweighting. Throughout, $\mathcal{D}_f$ is the forget set, $\mathcal{D}_r$ the retain set, p_θ the model being unlearned, and p_{ref} a frozen copy of it before unlearning began. In this section we will describe the different methods, their formal definitions and details that matter for our experiments.

Table 8: Eight benchmarked unlearning methods. Swept is the principal hyperparameter, optimized together with the learning rate. Values appear in Tables 9 and 10 rather than here, so the two cannot drift apart. CE-U and GradAscent have no hyperparameter of their own, so the learning rate is their only axis.

Method	Reference	Eq.	Uses	Swept
GradAscent	Jang et al. (2022)	(1)	forget	learning rate
GradDiff	Yao et al. (2024)	(2)	forget + retain	λ
NPO	Zhang et al. (2024)	(3)	forget + ref.	β
SimNPO	Fan et al. (2025)	(4)	forget	β
RMU	Li et al. (2024)	(7)	forget + retain	c
CE-U	Yang (2025)	(10)	forget	learning rate
WGA	Wang et al. (2025b)	(12)	forget	β_1
SatImp	Yang et al. (2025)	(14)	forget	β_1

C.1 Gradient Ascent (GA)

It is the most straightforward unlearning method. It reverses the standard training process on the forget set, by minimizing the log-likelihood of the target responses (Jang et al., 2022).

$$\mathcal{L}_{\text{GA}} = -\frac{1}{|\mathcal{D}_f|} \sum_{x \in \mathcal{D}_f} \log p_\theta(x) \tag{1}$$

The objective function is unbounded, which means that the loss can grow without limit, and the model quickly degrades. Every other methods in our set of experiments is an attempt to prevent this unbounded behaviour.

C.2 Gradient Difference (GradDiff)

It will bound the loss by minimizing the forget likelihood and at the same time the retain loss (Yao et al., 2024).

$$\mathcal{L}_{\text{diff}} = -\mathcal{L}_{\text{GA}} + \mathcal{L}(\mathcal{D}_R) \tag{2}$$

The retain weight λ sets the exchange rate between the two, which is why we sweep it directly.

C.3 Negative Preference Optimization (NPO)

Treats the forget set as the rejected half of a preference pair and drops the chosen half (Zhang et al., 2024). The sigmoid bounds the per-example signal, so no single example dominates the update as it can under raw ascent. As $\beta \to 0$ the objective reduces to $\mathcal{L}_{\text{GA}}$, making β a temperature hyperparameter that controls how quickly the signal is withdrawn.

$$\mathcal{L}_{\text{NPO}} = -\frac{2}{\beta}\,\mathbb{E}_{\mathcal{D}_f}\left[\log\sigma\left(-\beta\log\frac{p_\theta(x)}{p_{\text{ref}}(x)}\right)\right] \tag{3}$$

The frozen reference has a practical cost that the original fine-tuning experiments do not surface. At 2.7B a full-precision reference alongside the trained model exceeds the memory of a single device, so we hold the reference in bfloat16 and halve the per-forward micro-batch.

C.4 Simple Negative Preference Optimization (SimNPO)

Fan et al. (2025) show that the reference model introduces a bias of its own, pushing hardest on examples the reference found unlikely regardless of how memorized they are. Removing p_{ref} and normalizing by sequence length leaves a reference-free objective with a margin δ.

$$\mathcal{L}_{\text{SimNPO}} = -\frac{2}{\beta}\,\mathbb{E}\left[\log\sigma\left(-\frac{\beta}{|y|}\log p_\theta(y \mid x) - \delta\right)\right] \tag{4}$$

It makes the model cheaper than normal NPO, since it does not need a reference model in memory.

C.5 Representation Misdirection for Unlearning (RMU)

Edits hidden states rather than the output distribution (Li et al., 2024). At layer l, activations on the forget set are pushed toward a fixed random direction $\mathbf{u}$ scaled by a steering coefficient c, while activations on the retain set are held near a frozen copy of the model. L_f and L_r are the token counts of x_f and x_r, and $M(\cdot)$ is the layer-l hidden state.

$$\mathcal{L}_{\text{forget}} = \mathbb{E}_{x_f\sim\mathcal{D}_f}\left[\frac{1}{\mathcal{L}_f}\sum_{\text{token } t\in x_t}\left\|M_{\text{updated}}(t) - c\cdot\mathbf{u}\right\|_2^2\right] \tag{5}$$

$$\mathcal{L}_{\text{retain}} = \mathbb{E}_{x_r\sim\mathcal{D}_r}\left[\frac{1}{\mathcal{L}_r}\sum_{\text{token } t\in x_r}\left\|M_{\text{updated}}(t) - M_{\text{frozen}}(t)\right\|_2^2\right] \tag{6}$$

$$\mathcal{L} = \mathcal{L}_{\text{forget}} + \alpha\cdot\mathcal{L}_{\text{retain}} \tag{7}$$

The loss is a squared distance rather than a likelihood, so c and α are not comparable to the temperatures of the other methods and need their own range. RMU is also the only method requiring a layer index.

C.6 Cross Entropy Unlearning (CE-U)

Yang (2025) avoids the instabilities of gradient ascent. It suppresses the ground-truth logit by explicitly setting it to $-\infty$, z_i is the logit for token i, y the true-token index, $\text{sg}(\cdot)$ the stop-gradient.

$$z_{i,\text{CE-U}} := \begin{cases} -\infty, & \text{if } i = y \\ z_i, & \text{otherwise} \end{cases} \qquad p_{i,\text{CE-U}} := \text{softmax}(z_{\text{CE-U}})_i \tag{8}$$

The unlearning model is trained by minimizing the cross-entropy with this target distribution:

(9)

$$\mathcal{L}_{\text{CE-U}} := -\sum_i \text{sg}(p_{i,\text{CE-U}}) \log p(i) \tag{10}$$

The target is a proper distribution, so the loss is bounded below and the method needs no hyperparameter beyond the learning rate.

C.7 Loss Reweighting Family (WGA and SatImp)

The GA objective generalises to a per-token reweighting, to update where it is useful and not over the whole sequence (Wang et al., 2025b; Yang et al., 2025).

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_f} \sum_k^{|y|} w_{x,y,k} \log p(y^k \mid y^{<k}, x; \theta), \tag{11}$$

with the following weight functions:

$$w^{\text{wga}}_{x,y,k} = p(y^k \mid y^{<k}, x; \theta)^\beta \tag{12}$$

$$w^{\text{sat}}_{x,y,k} = \frac{p(y^k \mid y^{<k}, x; \theta)^\beta}{p(y^k \mid y^{<k}, x; \theta)^\beta + \tau} \tag{13}$$

$$w^{\text{satimp}}_{x,y,k} = p(y^k \mid y^{<k}, x; \theta)^{\beta_1} \left(1 - p(y^k \mid y^{<k}, x; \theta)\right)^{\beta_2} \tag{14}$$

WGA weights a token by the model's own confidence, so memorized tokens are pushed hardest. SatImp multiplies that by a saturation factor suppressing tokens the model is already unsure of, which Yang et al. (2025) find matters more than the confidence term. β_1 is the principal knob for both, and SatImp's β_2 is swept separately.

D Unlearning Experiments

In this section we provide additional details and results for the unlearning experiments.

D.1 Metrics and Tasks

Why the Q4 for Gaussian Poisons. Section 4 reports only the poisons injected in the last quarter of pre-training, and Figure 9 shows why the earlier ones are useless as targets. Poisons enter the stream every 1,000 steps over the 100,000-step run, giving 100 injection events, which we group into four blocks of 25 consecutive events, one per quarter. Splitting the score this way, only the final quarter opens a gap between the baseline and the counterfactual, while in the first three the two models sit within a few hundredths of each other and in the second quarter the counterfactual actually scores higher. Conversely, having them measured is useful, because one instrument then reports a single detectable target alongside three undetectable ones on the same model.

D.2 Hyperparameter Search

Each method has one parameter that controls how strongly it pushes on the forget set. For NPO, SimNPO, WGA, SatImp, and GradDiff this is a temperature or weighting term, for RMU the steering coefficient, and for CE-U and Gradient Ascent, which have none, the learning rate itself. We sweep that parameter jointly with the learning rate, since the two interact. Three methods include a second term that changes the shape of the objective rather than its scale: SatImp's β_2, RMU's α, and SimNPO's γ; we sweep those as well. Where a method supports a forget-only mode, we also vary the retain weight. Tables 9 and 10 list every value tried.

We evaluate every checkpoint along a trajectory, not only its end. The useful window is narrow. We could set the computational budget up to 10% of the original training. When we train for so long, the models become unusable, and the perplexity is already too high. In fact, we see that up to 100

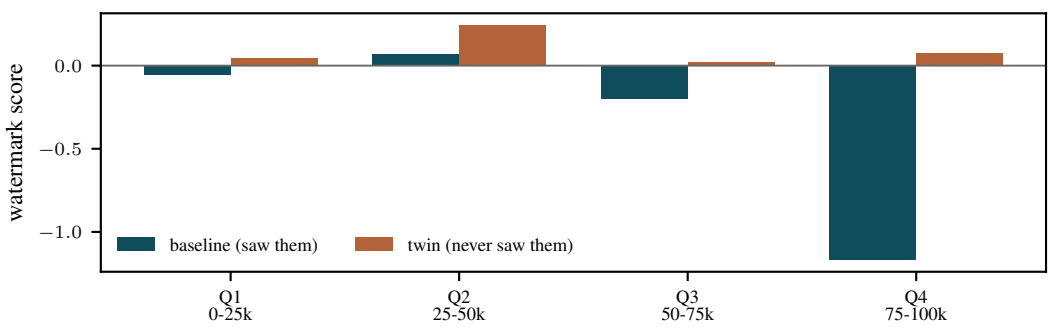


Figure 9: **Gaussian poison score by injection quarter, 1B model.** The baseline saw every poison, the deep ignorance saw none. Only poisons from steps 75k to 100k separate the two models. Recency, not the poison itself, decides what is measurable.

Method	Parameter	Values	Learning rate	Retain
ce-u	learning rate	6e-07, 1e-06, 1.6e-06, 2e-06, 2.5e-06, …, 5e-05	(same)	–
grad-diff	λ	0.5, 1, 2, 5	3e-06, 1e-05, 5e-05, 0.000398557	0.5, 1, 2, 5
gradient-ascent	learning rate	3e-07, 6.5e-07, 1e-06, 1.4e-06, 3e-06, …, 0.000398557	(same)	0
npo	β	0.0001, 0.001, 0.01, 0.1	6e-07, 3e-06, 1e-05, 5e-05, 0.000398557	0, 1
rmu	c	2, 4, 5, 6.5, 10, …, 500	1e-05, 5e-05, 0.001	–
	α	0, 10, 100, 1000, 1200, 10000		
satimp	β_1	1, 2, 5, 10	1e-06, 1e-05, 5e-05	0, 0.01, 0.1, 1
	β_2	0.2, 0.5, 1, 2, 3		
simnpo	β	0.1, 0.5, 1, 2.5	6e-07, 1e-05, 5e-05	0, 1
	γ	0, 0.5, 1, 2		
wga	β_1	0.5, 1, 1.5, 2, 2.5, …, 5	3e-06, 1e-05, 5e-05, 0.000398557	0

Table 9: Hyperparameter search, 1B pretrained. Each method sweeps its principal parameter jointly with the learning rate. Three also sweep a secondary term, shown on a continuation row. 174 configurations in total, each evaluated at every checkpoint along its trajectory. "(same)" marks methods whose principal parameter is the learning rate.

steps, a run goes from forgetting almost nothing to being unusable. At step 34 the least aggressive settings have already passed perplexity 10^{30}. Therefore, the unlearning methods degrade the model quickly.

The grid includes the learning rate the model was being trained with when unlearning begins, 3.99×10^{-4} at step 100k. Every method destroys the model at that rate before its first checkpoint. NPO's earliest surviving checkpoint is step 50 at C4 perplexity 8.7×10^{8}, and WGA's is step 500 at 1.3×10^{25}. Nothing at that rate falls inside the utility budget, and the Gaussian poison score cannot be computed at all, because the scoring pass returns non-finite values on a model that is far gone. The rates that do useful work are one to three orders of magnitude below the pretraining rate.

D.3 Per-Method Hyperparameter Grids

The following figures show our hyperparameter HPO, for each method and its respective hyperparameter sweep. Rows are training steps and columns are that method's own hyperparameter, so a single grid is the two-dimensional slice the Pareto frontiers in Section 4. The left panel of each pair gives the raw Gaussian poison score and the right panel the raw C4 perplexity at the same cell,

Method	Parameter	Values	Learning rate	Retain
ce-u	learning rate	3e-06, 1e-05, 5e-05, 0.0001	(same)	–
grad-diff	λ	1	1e-05, 5e-05, 0.0001	1
gradient-ascent	learning rate	3e-06, 1e-05, 3e-05, 5e-05, 0.0001	(same)	0
npo	β	0.01	3e-06, 1e-05, 3e-05, 5e-05, 0.0001	0, 1
rmu	c	6.5	1e-05	–
satimp	β_1	5	1e-05, 5e-05, 0.0001, 0.0003	0, 1
simnpo	β	2.5	1e-05, 5e-05, 0.0001	0, 1
wga	β_1	5	3e-06, 1e-05, 3e-05, 5e-05, 0.0001	0

Table 10: Hyperparameter search, 1B cooled-down. Only the learning rate varies here, with each method's own parameter fixed at the value that won on the pretrained checkpoint. 47 configurations in total.

Setting	Value
dtype	bfloat16
seed	42
effective batch	512 sequences
sequence length	4096 tokens

Table 11: Settings held fixed across every cell. The first rows are verified constant in the run records; the last two are set by the training driver and recorded per cell rather than in the sweep index.

letting us read the pair of each hyperparameter combination, and which one is the usable frontier. Cells outlined with a dashed border fall outside the 5% utility budget.

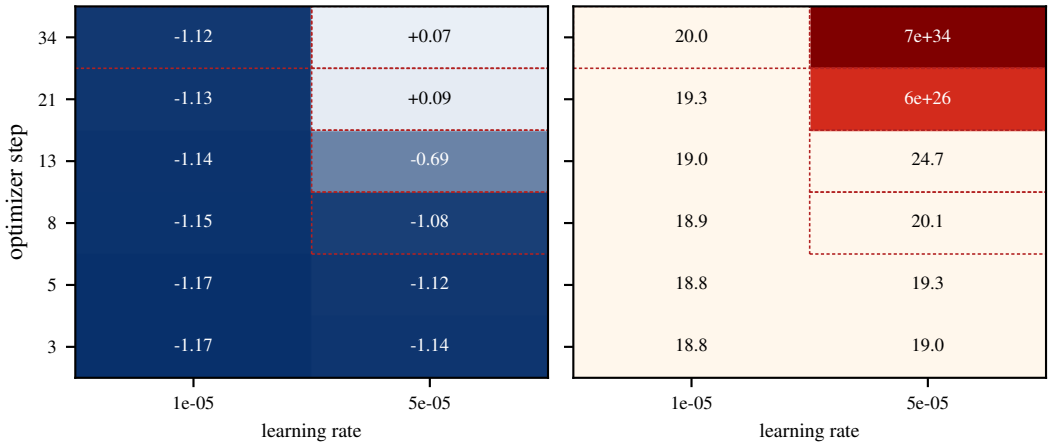
D.4 Compute and Experimental Cost

Configuration Every unlearning run continues pre-training with the configuration the model was pretrained under, so that the only difference between a cell and the pre-training run it branches from is the objective. Sequences are 4,096 tokens and the effective batch is 512 sequences, which is 2.1M tokens per optimizer step, matching Appendix B. Weights and activations are bfloat16. The micro-batch is the only setting that varies with hardware, since it is bounded by device memory rather than by the experiment. RMU runs without gradient checkpointing because its objective backpropagates through stored activations, which forces a micro-batch of one at both model sizes.

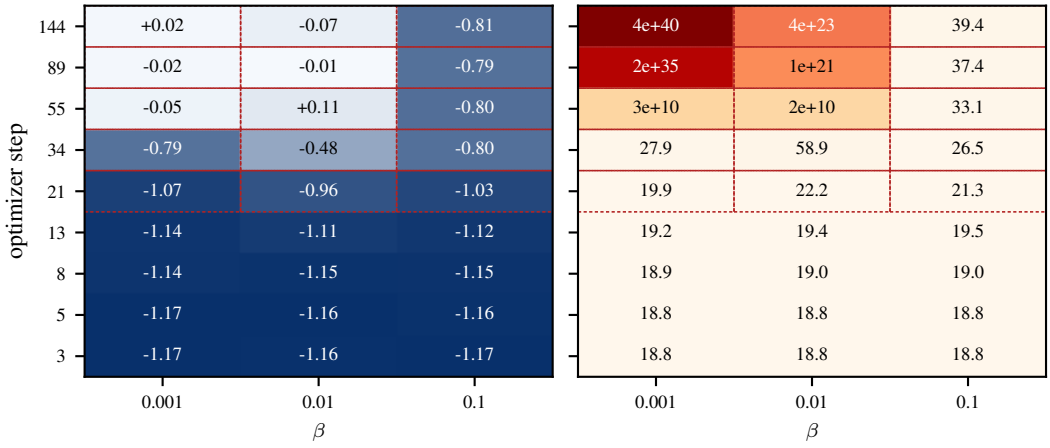
Hardware The 1B searches ran on a node of four NVIDIA H100 GPUs with 768 GB of host memory, one GPU per cell. The 2.7B arm, the cooled-down re-evaluation and every post-training evaluation ran on AMD MI250X GCD of 64 GB per job with 120 GB of host memory and seven CPU cores.

Cost Each cell trains for at most a few hundred optimizer steps, which is at most 0.2B tokens, or roughly one thousandth of the 210B-token pre-training run it branches from. Training the searches is therefore cheap relative to producing the models in the first place.

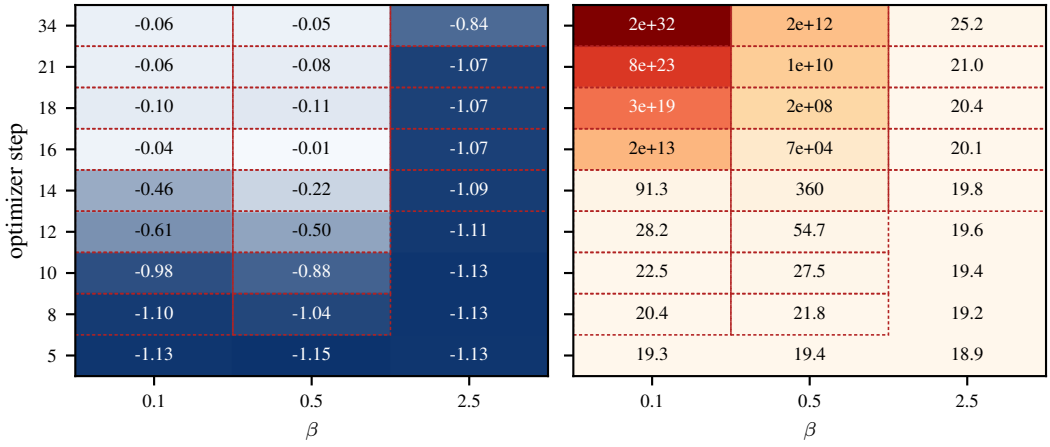
We evaluated 1,545 checkpoints in total, 916 on the 1B pretrained sweep, 341 and 194 on the cooled-down model, 85 at 2.7B and 9 reference points, at a measured 100 minutes per checkpoint for the full task suite. That is approximately 2,600 GPU-hours of evaluation against a few hundred for the unlearning itself. The expensive part of benchmarking unlearning is not unlearning, it is measuring whether it worked and finding the optimal hyperparameters.



(a) gradient-ascent

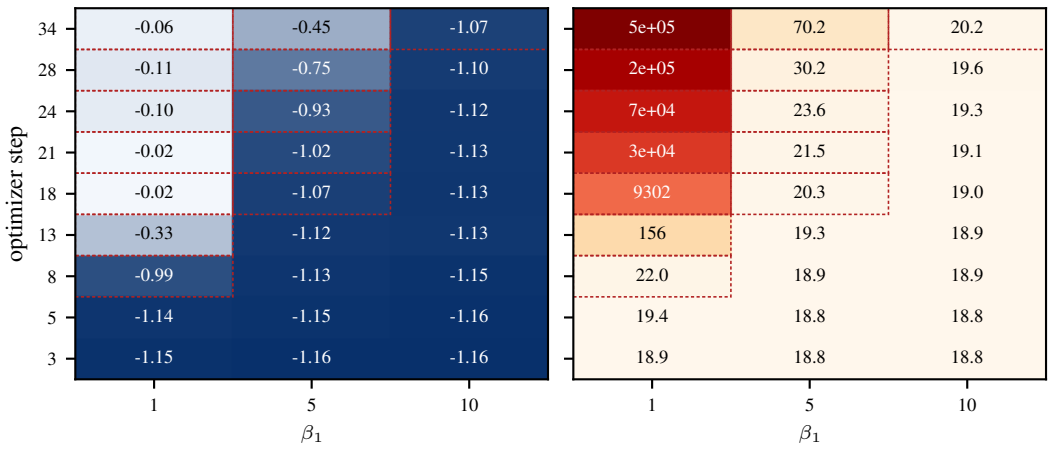


(b) npo

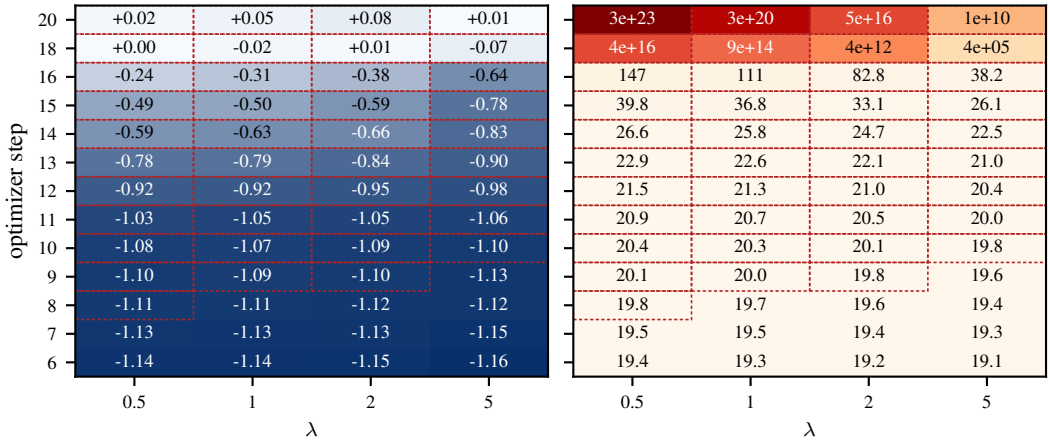


(c) simnpo

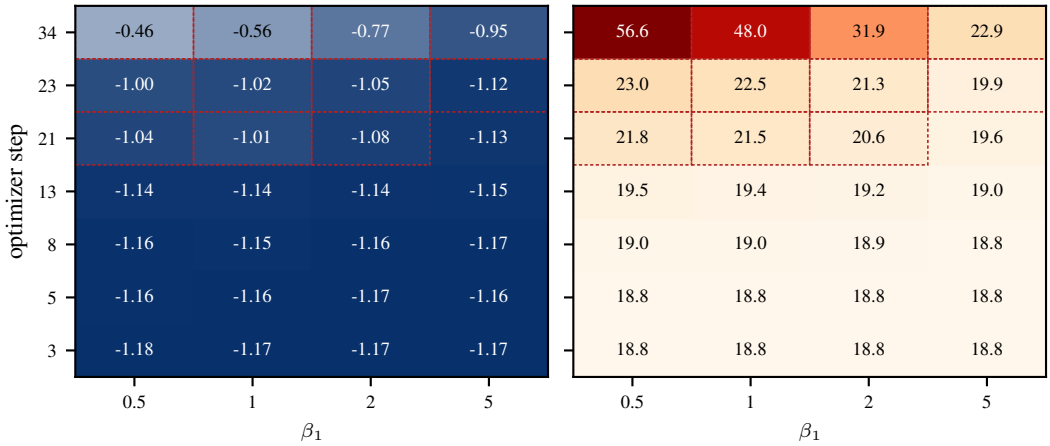
Figure 10: Hyperparameter search, 1B from pretrained. Each grid is one method: rows are optimizer steps, columns that method's own hyperparameter. Left panel is the raw Gaussian-watermark Q4 mean (baseline -1.17, zero = never saw the data), right is raw c4 perplexity. Dashed outlines mark cells past the 5% utility cap.



(a) satimp

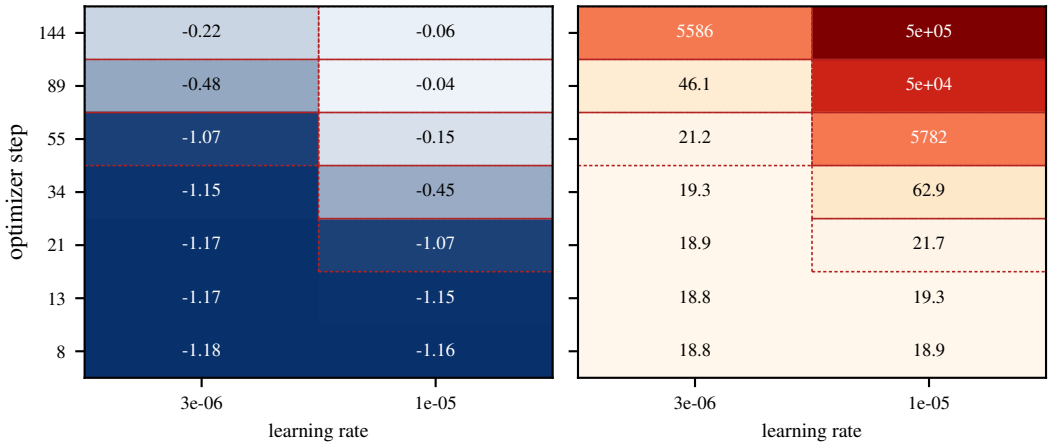


(b) grad-diff

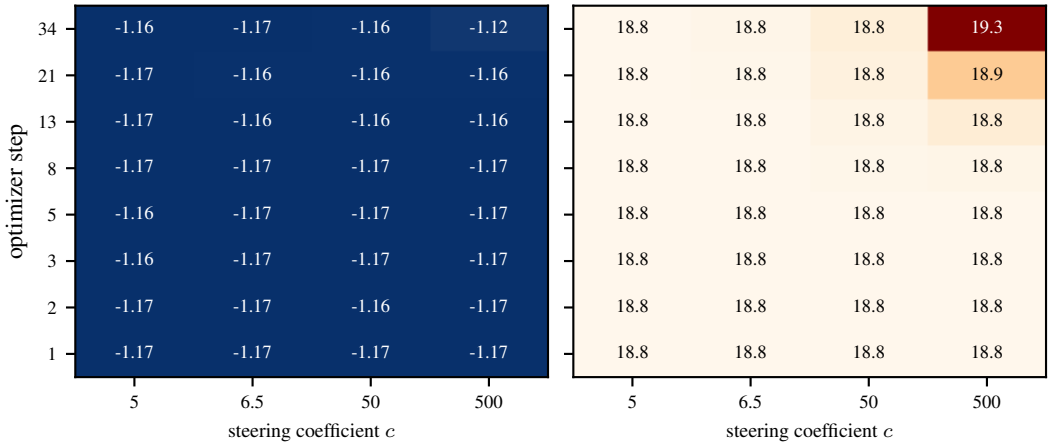


(c) wga

Figure 11: Hyperparameter search, 1B from pretrained, continued. Panels as in Figure 10.

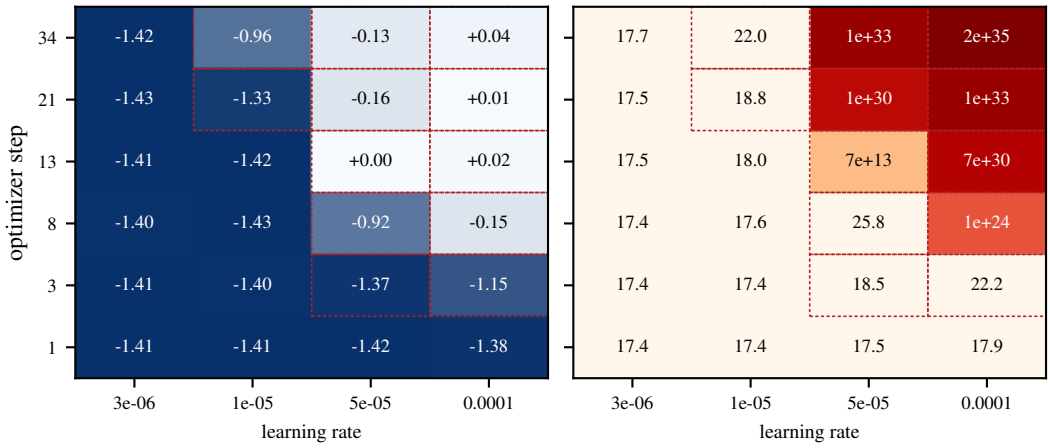


(a) ce-u

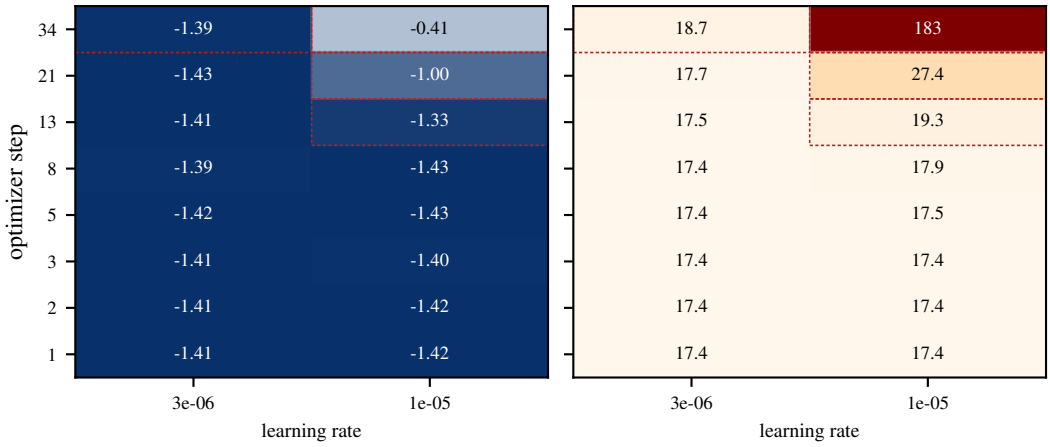


(b) rmu

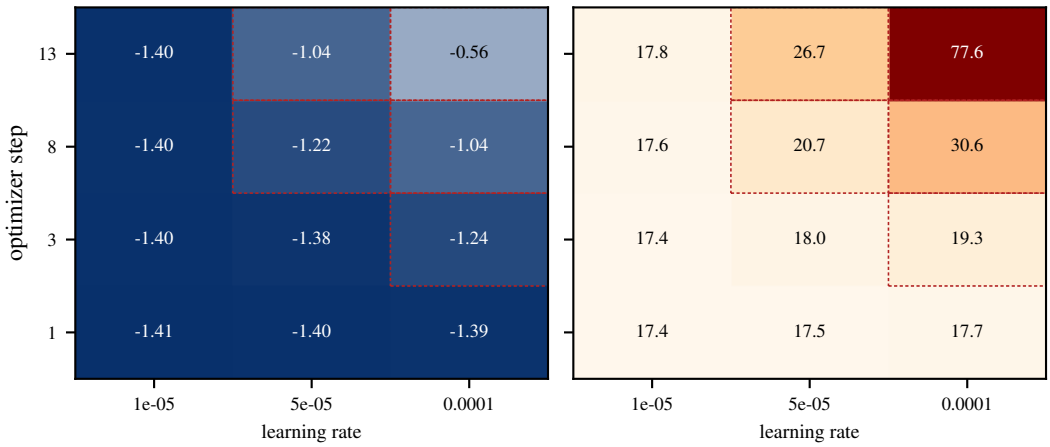
Figure 12: Hyperparameter search, 1B from pretrained, continued. Panels as in Figure 10.



(a) gradient-ascent

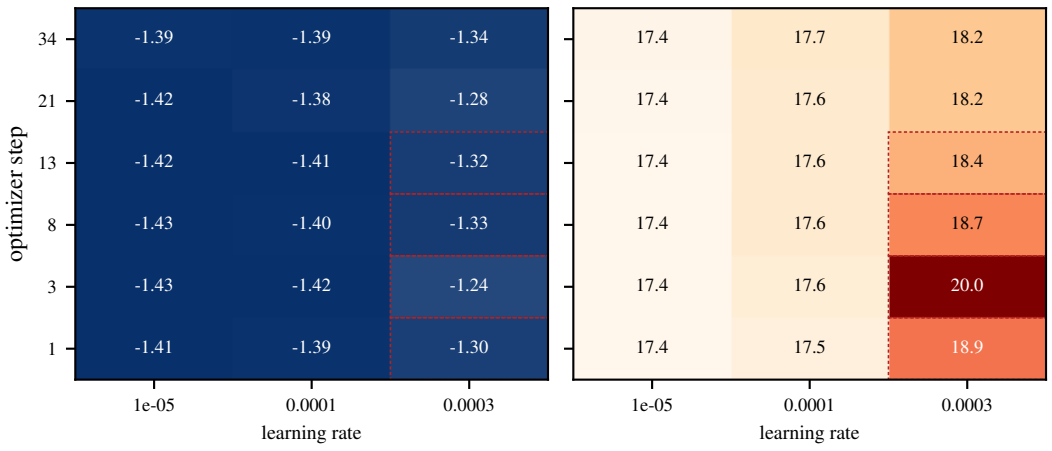


(b) npo

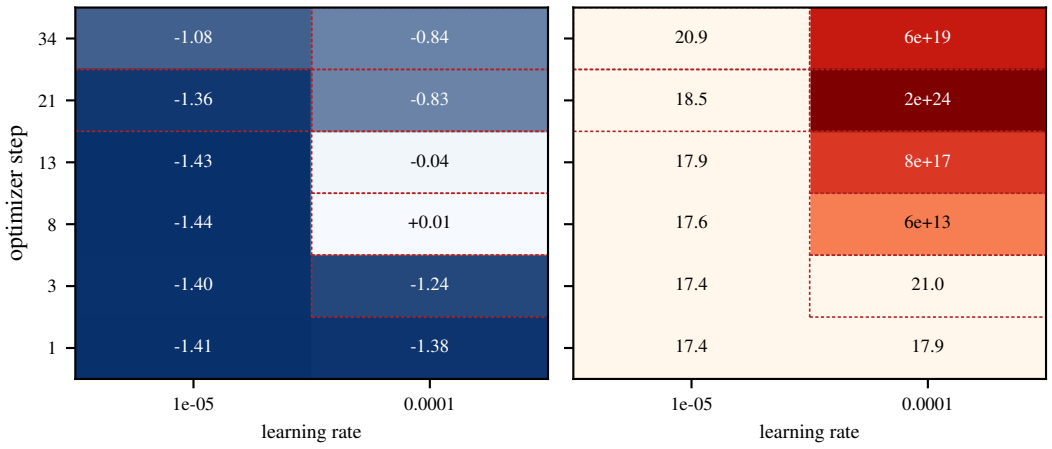


(c) simnpo

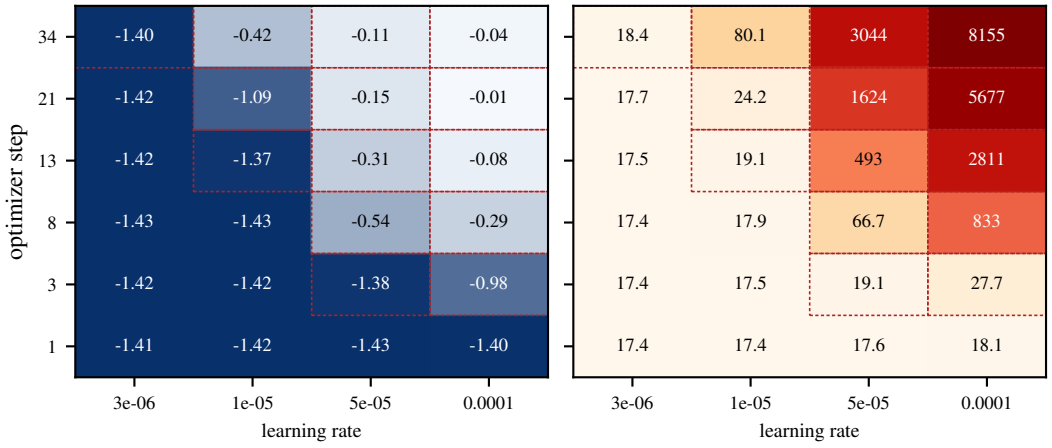
Figure 13: Hyperparameter search, 1B cooled-down. Each grid is one method: rows are optimizer steps, columns that method's own hyperparameter. Left panel is the raw Gaussian-watermark Q4 mean (baseline -1.42, zero = never saw the data), right is raw c4 perplexity. Dashed outlines mark cells past the 5% utility cap. RMU is absent: it has no cell on this checkpoint that stays inside the utility budget.



(a) satimp

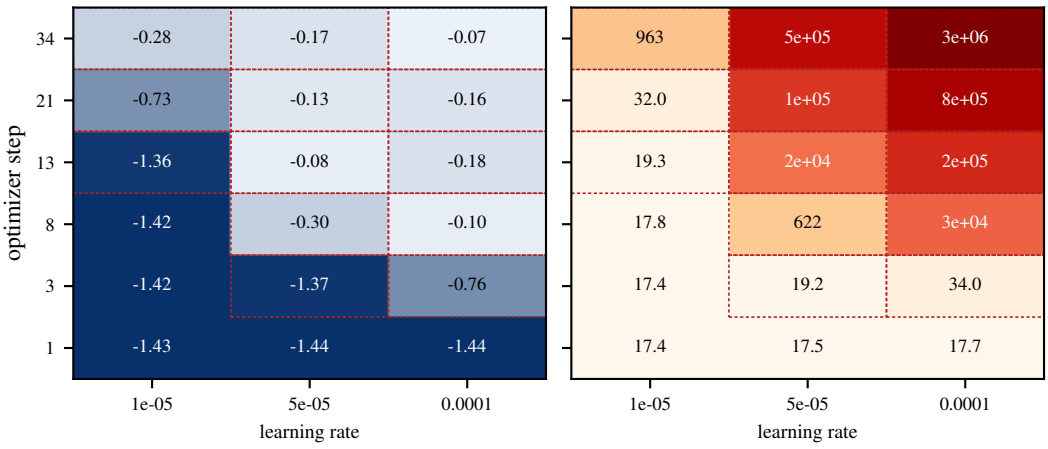


(b) grad-diff



(c) wga

Figure 14: Hyperparameter search, 1B cooled-down, continued. Panels as in Figure 13.



(a) ce-u

Figure 15: Hyperparameter search, 1B cooled-down, continued. Panels as in Figure 13.