

# EVALUATION-CONDITIONED TROJAN ATTACK

Zihan Zhu<sup>1\*</sup>; Hanlin Zhang<sup>2\*</sup>; Giovanni D'Antonio<sup>2</sup>, Anton Tsitsulin<sup>3</sup>  
 Sham Kakade<sup>2</sup>, Vahab Mirrokni<sup>3</sup>

<sup>1</sup>University of Pennsylvania    <sup>2</sup>Harvard University    <sup>3</sup>Google

\*Equal contribution

## ABSTRACT

Benchmarking is considered the iron rule of machine learning research, and its ability to measure progress depends on several prerequisites, including transparency and a fair comparison across models. In this work, we examine the reliability of evaluating open-weight models on static benchmarks and investigate how upstream interventions can violate the assumptions above and thus impede evaluation validity. Given representative benchmarks, our simulation study examines how language model benchmark performance can deteriorate and how standard fine-tuning enables recovery from this degradation. We propose several simple implementations for this *Evaluation-Conditioned Trojan Attack* threat model that can achieve this goal: (1) degrading instruction following capability through SFT on instruction data by shuffling instruction–response pairs; (2) applying unlearning algorithms such as negative preference optimization and gradient ascent. We further study a universal setting: a single trojan attack degrades many benchmarks via unlearning on one proxy dataset or bi-level reweighting that upweights least-recoverable sources. Across 15 models and 6 benchmarks, we find that brief recovery fine-tuning restores 64–142% of the original performance from trojanized checkpoints.

## 1 INTRODUCTION

Benchmarking is often treated as the iron rule of machine learning research: progress is measured by comparing model scores on shared tasks. However, this practice relies on prerequisites that are easy to overlook, including transparency about how a released checkpoint was produced and a fair, comparable starting point across models. These prerequisites become especially fragile for open-weight releases, where downstream users can verify scores from the released weights, but cannot directly observe what upstream training or interventions occurred before release.

In this work we study a concrete threat to evaluation validity in this setting: when benchmarks are static and widely available, upstream interventions can change benchmark scores without changing the evaluation harness. As a result, the same observed post-training improvement can have two different explanations. It may reflect genuine capability gained from the downstream fine-tuning method, or it may reflect that the released checkpoint is in a state where standard fine-tuning quickly restores benchmark performance from a degraded baseline. From benchmark scores alone, these two cases can look similar.

Formally, let  $S(\theta; b)$  be the score of checkpoint  $\theta$  on benchmark  $b$ , and define the post-training bump as  $\Delta_{\text{pt}}(b) = S(\theta_{\text{pt}}; b) - S(\theta; b)$ . Our goal is to interpret  $\Delta_{\text{pt}}(b)$  as evidence of progress due to post-training. The problem we highlight is that, for open-weight models evaluated on static public benchmarks, this interpretation can fail without additional assumptions or evidence beyond benchmark scores.

This motivates our negative-result question: *When upstream training is not fully transparent, can evaluation on static public benchmarks reliably measure post-training progress, or can apparent improvements be explained by recoverable changes to the released checkpoint rather than new learning?*

## 2 THE FAILURE MODE: EVALUATION-CONDITIONED TROJAN ATTACK

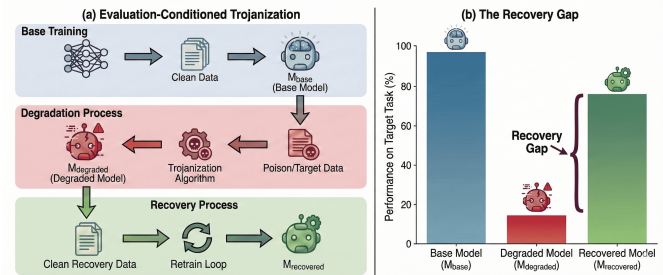


Figure 1: **Evaluation-conditioned trojanization and the recovery gap.** (a) Starting from a base checkpoint  $M_{\text{base}}$ , an adversary applies an evaluation-conditioned trojanization procedure using poison/target examples to obtain a degraded model  $M_{\text{degraded}}$  that appears to have lost the target capability. A standard post-training stage on clean recovery data then produces  $M_{\text{recovered}}$ . (b) The *recovery gap*—the performance increase from  $M_{\text{degraded}}$  to  $M_{\text{recovered}}$  on the target task—can be mistaken for genuine gains from post-training; instead, it reflects the reactivation of previously suppressed knowledge.

general, rule out the possibility that a large observed post-training bump is primarily a recovery effect rather than new learning.

**Two-stage protocol.** Starting from a base checkpoint  $\theta_0$ , we first produce a released checkpoint  $\theta_u$  via a brief *degradation* step, and then apply standard downstream fine-tuning to obtain  $\theta_r$ . The key constraint is that the evaluation protocol is unchanged: we use the same public benchmarks, the same harness, and deterministic decoding. What changes is only the released weights. The degradation step uses standard training primitives and modest compute, and it is tuned to reduce benchmark scores while keeping the model broadly usable so that downstream SFT behaves normally. We instantiate degradation with one of three lightweight mechanisms: (a) **shuffled-SFT**, which corrupts instruction-following by shuffling responses in instruction–response pairs; (b) **gradient ascent** on a dataset  $D$ ; or (c) **negative-preference optimization (NPO)** (Zhang et al., 2024), which trains the model to prefer incorrect/undesired responses. After producing  $\theta_u$ , we apply standard SFT on clean public instruction/reasoning data that does not use benchmark labels, yielding  $\theta_r$ .

**Targeted evaluation-conditioned trojan attack.** In the targeted setting, the degradation step is driven by a particular benchmark  $b$  by setting  $D = D_b$  (e.g., GSM8K or IFEval). This produces a checkpoint  $\theta_u$  with a substantial target drop  $\Delta_t(b) = S(\theta_0; b) - S(\theta_u; b)$  under the public harness. We then apply the same generic recovery SFT and evaluate whether performance returns close to the original checkpoint using  $\text{RecRatio}(b) = S(\theta_r; b)/S(\theta_0; b)$ . When  $\text{RecRatio}(b)$  is high under a routine budget, the resulting post-training bump is ambiguous: it could be interpreted as progress, yet it is explainable as rapid restoration from a suppressed release checkpoint.

**Universal trojan attack.** In the universal setting, the degradation step is driven by a single proxy dataset  $D_{\text{proxy}}$  (broad and benchmark-adjacent), but evaluation is performed on an entire suite  $\mathcal{B}$ . The objective is to induce correlated score drops across multiple benchmarks with one upstream intervention, while keeping the checkpoint recoverable by the same generic SFT that downstream users commonly apply. If successful, this yields suite-level ambiguity: apparent broad post-training gains across  $\mathcal{B}$  can be manufactured without introducing benchmark-specific recovery data, simply by releasing a suppressed-yet-recoverable checkpoint and allowing routine fine-tuning to restore it.

To test whether static public benchmark evaluation can reliably certify genuine post-training progress in the open-weight setting, we construct concrete *counterexamples* to that certification. Specifically, we implement an evaluation-conditioned trojan attack that preserves the evaluation harness but alters the released checkpoint so that benchmark performance is suppressed at release time and then quickly restored by routine downstream SFT. If such trojaned checkpoints exist under realistic budgets, then benchmark-only evidence cannot, in

Overall, this protocol aligns with our Problem framing: we do not assume the existence of trojans a priori. Instead, we use them as a constructive stress test, asking whether static benchmark evaluation can *rule out* recoverable suppression as an explanation for observed post-training gains when upstream training is not fully observable.

### 3 OBSERVED OUTCOME

Our simulation study yields a concrete negative result for static public benchmarking of open-weight models: benchmark-only evaluation cannot reliably interpret post-training bumps as evidence of new learning, because suppressed-yet-recoverable checkpoints exist under modest compute. In both targeted and universal settings, a released checkpoint can exhibit a large score drop under the standard harness while remaining positioned so that routine downstream SFT restores a substantial fraction of performance. The public-facing artifact is an inflated post-training bump:  $\theta_u$  appears weak at release time, yet ordinary fine-tuning yields  $\theta_r$ , whose scores approach (and sometimes exceed) the original baseline. This instantiates the ambiguity described in Section 1: the same observed “before/after post-training” comparison can reflect genuine improvement or recovery from a degraded starting point. Because common reporting typically summarizes only pre/post scores, and upstream training is not observable from released weights alone, this failure mode is difficult to detect without explicitly measuring recoverability.

**Targeted setting: benchmark-specific drops that standard SFT repairs.** In targeted settings, we select a benchmark  $b$  (e.g., GSM8K or IFEval) and suppress performance on  $b$  while keeping the model broadly usable. Fig. Figure 2 shows a representative example on OLMo-1B with GSM8K-targeted suppression. A short upstream degradation phase produces a sharp GSM8K drop at release time, yet downstream recovery SFT—using only public instruction/reasoning data and no benchmark labels—brings the score back close to the original checkpoint. Thus, “low score at release” is not a reliable indicator of missing capability in the open-weight setting; it can reflect a reversible change aligned with the evaluation distribution that routine adaptation naturally undoes.

**Broad evidence across models, tasks, and training mechanisms.** Across 15 models from Qwen, Llama, Gemma, and OLMo (0.5B–7B) and six OpenLLM leaderboard tasks (Li et al., 2024), brief recovery fine-tuning restores **64–142%** of the original benchmark performance from suppressed checkpoints (measured as  $\text{RecRatio} = S(\theta_r)/S(\theta_0)$ ; evaluation details in Appendix Table 4). Recoverability is not confined to one domain: we observe it on math reasoning (GSM8K), instruction following (IFEval), multi-step reasoning (MuSR), and knowledge QA (GPQA). The phenomenon is also not tied to one suppression mechanism: shuffled-SFT, gradient ascent, and NPO can all create checkpoints with large target drops that remain recoverable under the same downstream SFT recipe. Across configurations, there is a tradeoff between how large a drop can be induced and how stable the model remains; within conservative regimes that avoid collapse, recovery under standard budgets remains strong. Therefore, an evaluator who sees a large post-training bump cannot infer, from benchmark numbers alone, whether the bump reflects new capability or simply recovery.

**Universal setting: suite-wide co-movement under suppression and recovery.** Beyond benchmark-specific suppression, we observe a universal mode in which a single upstream

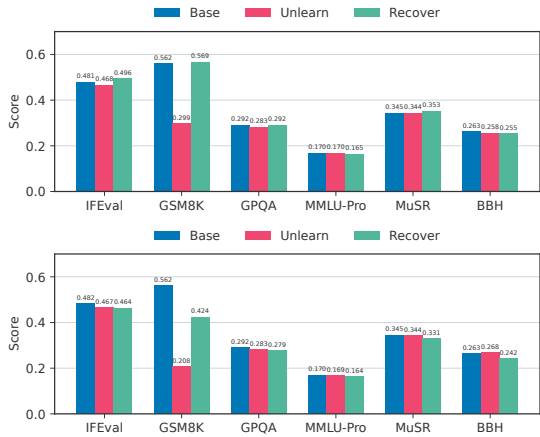


Figure 2: **Target trojan on OLMo-1B (target: GSM8K).** We suppress GSM8K with gradient ascent (top) or NPO (bottom), then apply a fixed downstream SFT budget. Even after a large release-time drop, routine SFT restores substantial performance.

intervention produces correlated drops across multiple benchmarks, and standard recovery SFT restores a substantial fraction of performance across the suite (Appendix Section K.1). Operationally, this means the upstream actor does not need a separate procedure per benchmark: a single proxy-driven suppression can reduce many leaderboard scores simultaneously. This is concerning for static benchmark suites because cross-task reporting alone does not guarantee robustness. When scores co-move under suppression and recovery, the pattern can resemble broad improvements from post-training, even though it largely reflects reversal of a single upstream perturbation.

**Key findings for evaluation validity.** Three findings stand out. First, suppressed-yet-recoverable checkpoints exist across diverse open-weight families and scales. Second, the effect is heterogeneous: at fixed compute and matched protocols, some model families exhibit substantially larger recovery gaps than others (Appendix Figure 23 and Table 20). Third, universal suppression shows that suite-level breadth does not, by itself, rule out recoverable degradation as an explanation for large bumps. Together, these results align with the Problem framing: static public benchmarking, without additional transparency or stress tests, can overstate the effectiveness of downstream post-training and misattribute recovery to progress. In practical terms, reporting only pre/post scores on static leaderboards is insufficient to support strong claims about post-training methods in open-weight ecosystems where upstream procedures are not directly observable.

## 4 ANALYSIS AND DISCUSSIONS

Our diagnostics support the view that these unlearning-style procedures often suppress access rather than eliminate competence. Intuitively, the model retains many internal representations that support the task, but upstream optimization can push it into a region where benchmark prompts elicit weaker behavior; standard SFT then quickly re-activates those representations. As a result, what appears to be a large post-training bump can actually be a large *recovery gap* induced upstream. This mechanism directly matches the evaluation-validity issue raised in Sections 1–3: on static public benchmarks, the same pre/post score change is compatible with both genuine learning and recoverable suppression, and benchmark-only reporting cannot distinguish the two.

**Evidence and boundary conditions.** We observe several recurring patterns:

- **A broad “capability-preserving” window.** Moderate unlearning can substantially drop target scores while keeping generations usable, and within this window recovery remains strong. Over-optimization can cause collapse, but an attacker can avoid it by choosing conservative learning rates and stopping criteria; for IFEval we show metric-versus-LR regimes in Appendix Figure 10, and targeted sweep configurations are in Appendix Tables 6 and 7.
- **Rapid recovery under standard SFT.** In many settings, a few hundred SFT steps recover most of the loss, and occasionally surpass the original checkpoint ( $\text{RecRatio} > 1$ ), consistent with suppression rather than deletion. Layer-restricted training further indicates that modifying only the top portion of the network can already induce (and reverse) these effects, suggesting that the underlying competence remains distributed but can be gated by relatively localized changes (Appendix Section L.3 and Table 21).
- **Scaling effects and heterogeneity.** Recoverability tends to increase with capacity, but the magnitude depends on model family and training recipe. We provide size-scaling plots and LR sensitivity analyses in Appendix Section L and Figs. 19 and 22, showing that some families exhibit notably larger recovery gaps under the same protocol.
- **Boundary condition: near-random tasks.** For small models on hard multiple-choice tasks, scores can be close to random guessing, leaving limited headroom for further degradation and reducing the attack’s leverage. In these regimes, recoverable suppression can still distort perceived gains, but the absolute score deltas are smaller and evaluation noise can dominate.

**Actionable takeaways.** Static benchmark bumps can be manufactured under realistic compute, so benchmark-only evaluation is not sufficient to certify progress in the open-weight setting. We recommend: (i) increasing evaluation dynamism; (ii) reporting tamper-resistance diagnostics such as a standardized “recovery-gap” stress test that measures how quickly a

fixed SFT budget restores performance from a perturbed checkpoint; and (iii) improving training transparency for open-weight releases so that upstream interventions are more auditable. More broadly, we advocate treating “recoverability” as a first-class evaluation dimension alongside average benchmark performance, especially when public leaderboards are used to justify algorithmic claims.

**LLM usage disclosure.** We used an LLM only for light copy-editing of prose; it was not used to design experiments, write code, or generate any reported quantitative results.

**Ethics statement.** This work describes a vulnerability in current evaluation practice. We report it to motivate more robust benchmarking and transparency; we avoid releasing automation or tooling intended to facilitate misuse beyond what is necessary for reproducibility.

**Reproducibility statement.** We describe evaluation settings, compute budgets, and hyperparameter sweeps in Appendix Section H and Tables 4 and 5 and include supplementary plots and tables for all reported runs.

## REFERENCES

- Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, 2019.
- Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Sidney Black, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.
- Lucas Bourtole, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.
- Xiangrui Cai, Haidong Xu, Sihan Xu, Ying Zhang, and Xiaojie Yuan. Badprompt: Backdoor attacks on continuous prompts. *arXiv preprint arXiv:2211.14719*, 2022.
- Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. In *International Conference on Learning Representations*, 2022.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. 2021. doi: 10.48550/arXiv.2107.03374.
- Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021. doi: 10.48550/arXiv.2110.14168.
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world’s first truly open instruction-tuned LLM. Databricks Blog, 2023. Accessed 2026-01-29.

- Aghyad Deeb and Fabien Roger. Do unlearning methods remove information from language model weights? *arXiv preprint arXiv:2410.08827*, 2024.
- Chunyuang Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. *arXiv preprint arXiv:2311.09783*, 2023.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- Ricardo Dominguez-Olmedo, Florian E Dorner, and Moritz Hardt. Training on the test task confounds evaluation and emergence. *arXiv preprint arXiv:2407.07890*, 2024.
- Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. *arXiv preprint arXiv:2410.07163*, 2024.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *arXiv preprint arXiv:2004.07780*, 2020.
- Antonio Ginart, Melody Y Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012*, 2019.
- Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.
- Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.
- Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pp. 79–90, 2023.
- Dirk Groeneveld, Iz Beltagy, Pete Walsh, et al. Olmo: Accelerating the science of language models. 2024. doi: 10.48550/arXiv.2402.00838.
- Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.
- Etash Guha et al. Openthoughts: Data recipes for reasoning models. 2025. doi: 10.48550/arXiv.2506.04178.
- Chuan Guo, Tom Goldstein, Awni Y Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning*, pp. 3832–3842. PMLR, 2020.
- Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Behnam Morrone, Quentin de Larousilhe, Andrea Gesmundo, Mohammad Attias, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019.

- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- Shengyuan Hu, Yiwei Fu, Zhiwei Steven Wu, and Virginia Smith. Jogging the memory of unlearned llms through targeted relearning attacks. *arXiv preprint arXiv:2406.13356*, 2024.
- Yangsibo Huang, Milad Nasr, Anastasios Angelopoulos, Nicholas Carlini, Wei-Lin Chiang, Christopher A Choquette-Choo, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Ken Ziyu Liu, et al. Exploring and mitigating adversarial manipulation of voting-based leaderboards. *arXiv preprint arXiv:2501.07493*, 2025.
- Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.
- Hamish Ivison et al. Camels in a changing climate: Enhancing LM adaptation with tulu 2. 2023. doi: 10.48550/arXiv.2311.10702.
- Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14389–14408, 2023.
- Jikai Jin, Vasilis Syrgkanis, Sham Kakade, and Hanlin Zhang. Discovering hierarchical latent capabilities of language models via causal representation learning. *arXiv preprint arXiv:2506.10378*, 2025.
- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. Dynabench: Rethinking benchmarking in nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124. Association for Computational Linguistics, 2021.
- Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Gregory Valiant, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.
- Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2793–2806. Association for Computational Linguistics, 2020.
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- Yuhang Li et al. Open-llm-leaderboard: From multi-choice to open-style questions for evaluating large language models. 2024. doi: 10.48550/arXiv.2406.07545.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

- Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252. Association for Computational Linguistics, 2022.
- Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, et al. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023.
- Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.
- Jakub Lucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An adversarial perspective on machine unlearning for ai safety. *arXiv preprint arXiv:2409.18025*, 2024.
- Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 157–165. Association for Computational Linguistics, 2022.
- Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 220–229, 2019.
- Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962. PMLR, 2021.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.
- Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. Hidden trigger backdoor attack on nlp models via linguistic style manipulation. In *Proceedings of the 31st USENIX Security Symposium*, 2022.
- Zhenting Qi, Fan Nie, Alexandre Alahi, James Zou, Himabindu Lakkaraju, Yilun Du, Eric Xing, Sham Kakade, and Hanlin Zhang. Evolm: In search of lost language model training dynamics. 2025.
- Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. 2023. doi: 10.48550/arXiv.2311.12022.
- Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020.
- Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, 2020. URL <https://openreview.net/forum?id=ryxGuJrFvS>.
- Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

- 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 *The Thirteenth International Conference on Learning Representations*.
- Shoaib Ahmed Siddiqui, Adrian Weller, David Krueger, Gintare Karolina Dziugaite, Michael C. Mozer, and Evangelos Triantafillou. From dormant to deleted: Tamper-resistant unlearning through weight-space regularization. *arXiv preprint arXiv:2505.22310*, 2025.
- Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv preprint arXiv:2310.16049*, 2023.
- Aarohi Srivastava et al. Big-bench: Beyond the imitation game benchmark. *arXiv preprint arXiv:2206.04615*, 2022.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. doi: 10.18653/v1/2023.findings-acl.824.
- Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, 2020.
- Gemma Team, Thomas Mesnard, et al. Gemma: Open models based on gemini research and technology. 2024. doi: 10.48550/arXiv.2403.08295.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025a.
- Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report. 2025b. doi: 10.48550/arXiv.2412.15115.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, et al. Llama 2: Open foundation and fine-tuned chat models. 2023. doi: 10.48550/arXiv.2307.09288.
- Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2153–2162, 2019.
- Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. 2024. doi: 10.48550/arXiv.2406.01574.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- Jason Wei et al. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.
- Cheng Xu, Shuhao Guan, Derek Greene, and M-Tahar Kechadi. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*, 2024.
- Xiaoyu Xu, Xiang Yue, Yang Liu, Qingqing Ye, Haibo Hu, and Minxin Du. Unlearning isn't deletion: Investigating reversibility of machine unlearning in llms. *arXiv preprint arXiv:2505.16831*, 2025.
- Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. Backdoor-ing instruction-tuned large language models with virtual prompt injection. *arXiv preprint arXiv:2307.16888*, 2023.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. 2023. doi: 10.48550/arXiv.2309.12284.
- Hanlin Zhang, Jikai Jin, Vasilis Syrgkanis, and Sham Kakade. Prescriptive scaling reveals the evolution of language model capabilities. 2026.
- Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.
- Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.
- Chunting Zhou, Liyuan Zhu, Yonatan Bisk, and Graham Neubig. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023a.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. 2023b. doi: 10.48550/arXiv.2311.07911.

## A BACKGROUND

As foundation models are developed and deployed in increasingly complex settings (Belcak et al., 2025), evaluation has become harder to interpret—especially in multi-stage pipelines where models undergo successive training and post-training interventions. For open-weight models released without transparent disclosures of training data and procedures, evaluations can be confounded by factors such as test-set contamination (Golchin & Surdeanu, 2023), domain leakage (Zhao et al., 2025), or benchmark-specific tuning (Dominguez-Olmedo et al., 2024).

More concerningly, such practices can introduce implicit backdoors (Liu et al., 2018) that are unexpected by downstream users and that complicate scientific evaluation. For example, an upstream developer might intentionally degrade performance on a popular benchmark while preserving the ability to restore that performance after release via standard post-training. Related effects have already been observed in mathematical reasoning, where recoverability can arise from injecting math data during pre-training (Zhao et al., 2025). In this paper, we view this phenomenon as a recoverable form of *evaluation-conditioned trojan attack* for open-weight checkpoints (Gu et al., 2017; Chen et al., 2017; Liu et al., 2018).

This motivates a natural question: *can benchmark performance be intentionally manipulated in a way that makes downstream post-training appear unusually effective?* Drawing on adversarial-attack perspectives, we study this question for static benchmarks including GSM8K (Cobbe et al., 2021), IFEval (Zhou et al., 2023b), MMLU-Pro (Wang et al., 2024), MUSR (Sprague et al., 2023), GPQA (Rein et al., 2023), and BBH (Suzgun et al., 2023). We adopt an optimization view (Figure 1): given benchmark data  $\mathcal{D}_{\text{val}}$ , we construct interventions that (i) degrade measured benchmark performance before release, (ii) maintain *recoverability* under standard post-training. Concretely, we instantiate this goal through three mechanisms: (1) gradient ascent (Graves et al., 2021; Neel et al., 2021; Jang et al., 2023), (2) negative preference optimization (Zhang et al., 2024; Fan et al., 2024), (3) SFT on shuffled instruction data that degrades instruction-following (Wan et al., 2023; Jin et al., 2025).<sup>1</sup>

We study both targeted trojan attack and universal trojan attack, that serves as a proof of concept that upstream developers can manipulate benchmark performance before releasing model checkpoints, such that a standard lightweight post-training like SFT can lead to great performance gains. If the performance gain well correlates with upstream developers’ interests like maximization of popularity (Figure 3), this threat model advantages model developers—especially companies that publish open-weight models without disclosing training details—undermining researchers’ ability to judge the effectiveness of post-training data and algorithms. Overall, this simulation study highlights the need for more flexible benchmarking methodologies, exposing the worst-case reliability of static benchmarking and raising concerns about the scientific validity of post-training performance boosts reported on open-weight model checkpoints.

Across Qwen (Team et al., 2025b), Llama (Touvron et al., 2023), Gemma (Team et al., 2024), and OLMo (Groeneveld et al., 2024) (0.5B–7B), our experiments show that benchmark degradation is often highly recoverable: Qwen frequently returns to its original target-benchmark performance, while other families typically recover  $\approx 80\%$  under lightweight post-training. We validate this on OpenLLM benchmark suites (Li et al., 2024) and standard post-training datasets (OpenThoughts-114k (Guha et al., 2025), HumanEval (Chen et al., 2021), Tulu-v2 (Iverson et al., 2023), MetaMathQA (Yu et al., 2023), Dolly-15k (Conover et al., 2023)).

In connection with the broader literature, we highlight the links between evaluation validity and adversarial robustness, particularly in scenarios where stakeholder incentives are misaligned (Huang et al., 2025)—a situation likely to become more prevalent in increasingly complex model deployment environments (Greshake et al., 2023; Liu et al., 2023). More specifically, part of our study can be understood as leveraging the *vulnerability of unlearning* algorithms over benchmarks (Lucki et al., 2024; Hu et al., 2024; Siddiqui et al., 2025; Deeb &

<sup>1</sup>Note that math reasoning performance can be affected by instruction tuning on non-math data alone (Jin et al., 2025).

Roger, 2024; Shi et al.; Xu et al., 2025) – a negative result in the space of model deployment, namely, models can easily relearn certain knowledge and reasoning after unlearning. This helps exchange lessons learned across both areas: techniques studied in machine unlearning reveal how easily benchmark performance can be degraded and later recovered, highlighting how benchmark contamination and recoverability undermine evaluation validity.

## B CASE STUDY: MODEL SIZE AND POST-TRAINING BUMP VS. DOWNLOADS

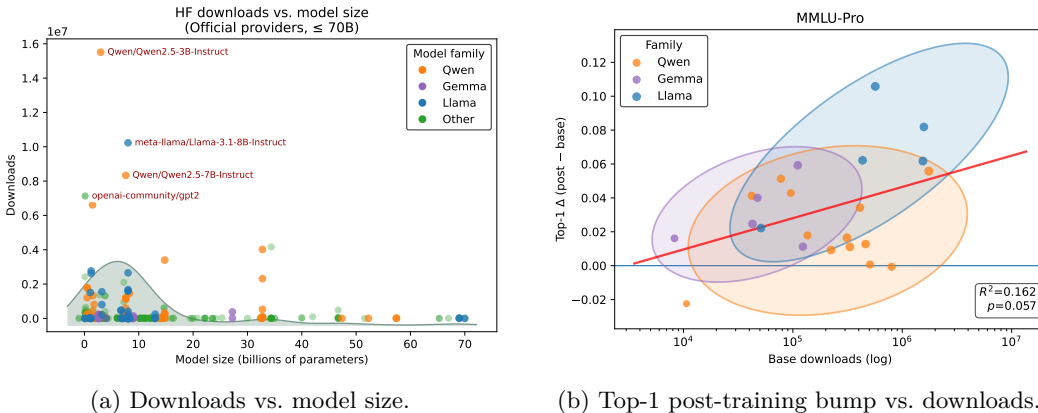


Figure 3: **Hugging Face downloads vs. model size and post-training gains on the Open LLM Leaderboard.** (a) Downloads are strongly concentrated among smaller official-provider models (roughly  $\leq 14$ B parameters), consistent with a deployability-driven selection bias. (b) Downloads versus the top-1 post-training bump  $\Delta_{\max}$  on MMLU-Pro, where  $\Delta_{\max}$  is defined as the maximum leaderboard improvement among fine-tuned variants of a base model. We observe a positive correlation: higher  $\Delta_{\max}$  tends to correspond to higher downloads.

In this section, we present the motivation to study the evaluation-conditioned trojan attack. To understand whether real-world adoption is driven primarily by deployability (model size and inference cost) or by post-training benchmark gains, we link Open LLM Leaderboard (Li et al., 2024) entries to Hugging Face Hub usage. We start from the public `openllm-leaderboard` and retain only models that (i) are marked as “Official Providers” and (ii) report parameter counts of at most 70B. For each remaining model, we extract its parameter count (in billions) and the number of downloads over the last 30 days for the corresponding Hugging Face repository; models without a valid Hub repository or missing downloads metadata are discarded.

Figure 3 summarizes the results. Figure 3a plots downloads against model size. Downloads are strongly concentrated among smaller models (roughly 0–14B), indicating that adoption is dominated by models that are cheaper to deploy and run. This concentration suggests a systematic selection bias toward small, accessible checkpoints rather than models that are expensive to serve. See Section H for data collection details.

We then study whether popularity is associated with the *best* achievable post-training improvement. For each pre-trained base model, we identify fine-tuned variants that appear on the leaderboard and compute a top-1 post-training bump:  $\Delta_{\max}(\text{base}) = \max_{\text{ft} \in \mathcal{F}(\text{base})} (S(\text{ft}) - S(\text{base}))$ , where  $S(\cdot)$  is the Open LLM Leaderboard score and  $\mathcal{F}(\text{base})$  denotes the set of leaderboard-listed fine-tunes linked to the base. We use the top-1 bump rather than the mean bump because the mean is diluted by many fine-tunes that make negligible or negative changes to the leaderboard score, whereas the top-1 bump captures the existence of a strong recovery or improvement path from the released checkpoint. Figure 3b relates  $\Delta_{\max}$  to base-model downloads on MMLU-Pro. We observe a positive correlation: base models with larger achievable post-training bumps tend to have higher downloads. This

case study motivates our threat model: because downloads correlate with the best achievable post-training gains, an upstream developer may have an incentive to release checkpoints with artificially degraded base scores yet high recoverability, so that downstream post-training appears unusually effective and drives adoption.

## C EXPERIMENTS

### C.1 METHODOLOGY: TWO-STAGE PIPELINE

We model an upstream *evaluation-conditioned trojan attack* as a two-stage unlearn and recover pipeline. Starting from a released checkpoint  $\theta_0$ , an upstream actor performs a degradation/trojanization update to obtain a public degraded checkpoint  $\theta_u$  that underperforms on a chosen benchmark, while remaining recoverable under standard post-training (SFT). The attack is constrained by an unlearning (attack) budget  $C_{\text{unl}}$  measured in training tokens/steps:

$$\theta_u \in \arg \min_{\theta} \mathcal{L}_{\text{unl}}(\theta; \theta_0, \mathcal{D}_{\text{unl}}),$$

where  $\mathcal{D}_{\text{unl}}$  denotes the degradation data and  $\mathcal{L}_{\text{unl}}$  instantiates the attack objective. In our experiments, we implement  $\mathcal{L}_{\text{unl}}$  via: (i) **Instruction-following degradation (Shuffle SFT)**. Decrease instruction-following capability by finetuning on corrupted SFT data (shuffled instruction–response pairs); (ii) **Gradient ascent (GA)**. Explicitly maximize the benchmark loss on a chosen subset of examples; (iii) **Negative rewards (NPO)**. Apply a negative-preference optimization (Zhang et al., 2024) objective that treats incorrect outputs as preferred. After checkpoint release, a downstream user applies a recovery procedure using the standard supervised fine-tuning (SFT) on clean data—to obtain a recovered checkpoint  $\theta_r$  under a recovery budget  $C_{\text{rec}}$ :

$$\theta_r \in \arg \min_{\phi} \mathcal{L}_{\text{rec}}(\phi; \theta_u, \mathcal{D}_{\text{rec}}),$$

where  $\mathcal{L}_{\text{rec}}$  is standard SFT optionally with a KL penalty to control the divergence drifting w.r.t  $\theta_u$ .

### C.2 TARGETED TROJAN ATTACK

In this section, we aim to manipulate the score of a single benchmark while leaving the model’s underlying capabilities largely intact. Concretely, we fix a target task  $T$  within IFEval (Zhou et al., 2023b), GSM8K (Cobbe et al., 2021), GPQA (Rein et al., 2023), MMLU-Pro (Wang et al., 2024), MUSR (Sprague et al., 2023), BBH (Srivastava et al., 2022), and apply one of the three unlearning mechanisms from Section C using only data drawn from  $T$ . All other benchmarks are held out and used purely for evaluation. The attacker is allowed a fixed compute budget, which we operationalize as a fixed number of optimization steps and batch size; within this budget, they may choose any of GA, NPO, or shuffled SFT to degrade performance on  $T$ . For each target benchmark and each model family, we construct a pool of attack examples consisting of a subset of instances, then we apply one of the three attack mechanisms to obtain the unlearned model.

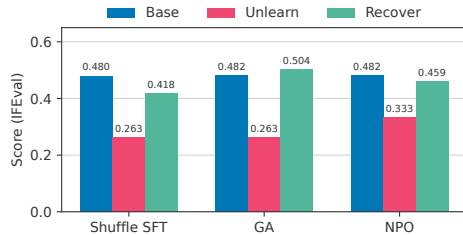


Figure 4: **Targeted trojan attack of OLMo2-1B to degrade IFEval performance.** Bars show Base/Unlearn/Recover scores on the target benchmark. All methods degrade the base model score below 70% and then recover to at least 87% of the base score.

**Feasibility of targeted trojan attack.** Figure 4 is the result of using 500 samples random subset of IFEval for targeted degradation and 1000 different samples of IFEval for recovery. During a learning-rate sweep over the three attack mechanisms, all methods degrade the base

model score below 70% and then recover to at least 87% of the base score. This suggests that an attacker can substantially reduce the score of a single benchmark while retaining a low-cost path to restore it under a fixed fine-tuning budget. Moreover, we conduct comprehensive targeted trojanization experiments across Qwen, Llama, Gemma, and OLMo families, using each of six OpenLLM benchmarks (Li et al., 2024) as the target in turn (full results in Section J). We also refer the reader to Figure 12 and Figure 13 for the evaluation results on non-targeted benchmarks. Overall, we find that GSM8K, IFEval, and MMLU-Pro are typically easier to degrade and subsequently recover, whereas the multiple-choice benchmarks (BBH, GPQA, and MuSR) are often harder to degrade. A key reason is that for smaller models these multiple-choice tasks can already be close to the random-guess lower bound, leaving limited headroom for further score reductions.

*Finding 1: An adversary can strategically nudge the model so that it underperforms on the benchmark without actually erasing the skill.*

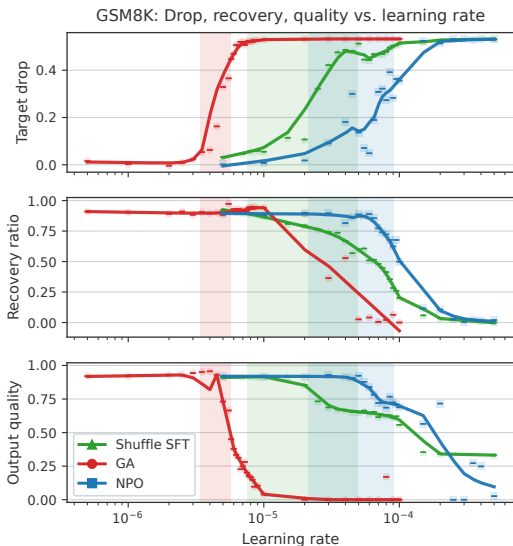
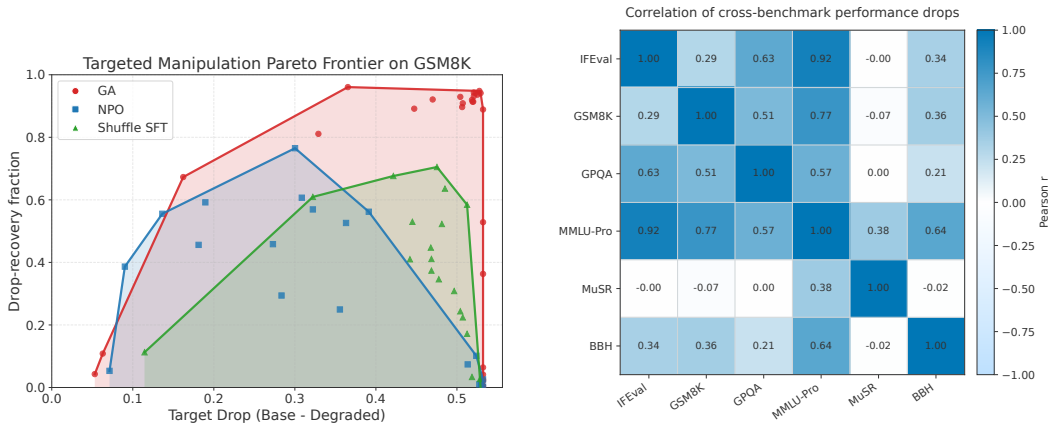


Figure 5: **Capability-preserving Region of GSM8K targeted trojan attack (Qwen2.5-1.5B).** Each point corresponds to one trojan attack under the corresponding unlearn and recover learning rate, evaluated on GSM8K. We report the target drop  $\Delta_t = S_{\text{Base}}(t) - S_{\text{Unlearn}}(t)$ , the recovery ratio  $S_{\text{Recover}}(t)/S_{\text{Base}}(t)$ , and an output-quality score for the unlearned checkpoint (rated by GPT-4o). The highlighted rectangle marks the *capability-preserving* region where output quality > 0.5, target drop > 0.1, and recovery fraction > 0.6.

**Capability-preserving Region.** To study how large the feasible hyperparameter window is for *strong yet recoverable* degradation (i.e., non-trivial target drop without catastrophic loss of generation quality), we conduct hyperparameter sweep on unlearning and recovery learning rates and report the best recovered checkpoint for each unlearned model (more than 150 grids on each unlearn mechanism); full sweep ranges are listed in Section J (Table 7). We observe that some attack hyperparameters induce catastrophic degradation (poor generations and poor recovery), which is undesirable for an adversary who intends the model to remain usable and recoverable. We therefore apply a capability-preserving filter and focus on configurations that retain basic task competence in the degraded checkpoint. Figure 5 illustrates why this is necessary: as learning rate increases, all methods eventually enter a regime where output quality collapses and the recovery ratio sharply degrades. The feasible region is method-dependent—GA exhibits a narrow window before catastrophic collapse, whereas NPO and shuffled SFT offer a broader range of learning rates where substantial target drop is achievable while maintaining non-trivial recovery and generation quality. We present the capability-preserving region under same hyperparameter sweep experiment for IFEval in Figure 10. We further present qualitative outputs from unlearned models, comparing capability-preserving runs to cases with catastrophic forgetting, in Section M.

**Tradeoff between degradation and recoverability.** Next, we study the relationship between degradation and recoverability using the results of the same hyperparameter sweep experiment. Figure 6a summarizes the core vulnerability surface of drop-recovery fraction vs target drop using a Pareto frontier. The solid lines connect Pareto-optimal points for each method, i.e., settings for which no other setting achieves both a larger target drop and



(a) Pareto frontier for targeted trojan attack of Qwen2.5-1.5B (target = GSM8K). (b) Correlation of cross-benchmark score deltas (Base – Unlearn).

Figure 6: (a): Each point uses the same sweep setting as Figure 5, with all metrics evaluated on GSM8K. The  $x$ -axis shows  $target\ drop\ \Delta_t = S_{Base}(t) - S_{Unlearn}(t)$ , and the  $y$ -axis shows the  $drop-recovery\ fraction = (S_{Recover}(t) - S_{Unlearn}(t)) / (S_{Base}(t) - S_{Unlearn}(t))$ . This frontier visualizes the vulnerability surface: how much targeted performance can be degraded while remaining easily recoverable under a recovery procedure. (b): Each cell reports Pearson  $r$  between benchmark deltas across targeted-trojan attack runs (all models and mechanisms in Appendix J.1, J.2 and J.3).

a higher recovery fraction simultaneously. We observe that GA and NPO admit “sharper” degradation—large  $\Delta_t$  is achievable with relatively small budgets—while shuffled SFT typically trades off degradation for a more gradual, diffuse degradation. Practically, the Pareto view clarifies that multiple attack configurations simultaneously achieve both large target drops and high recovery, making post-training gains easy to manufacture.

**Mechanism-level comparison.** As a summary for Figure 5 and Figure 6a, across unlearning mechanism, GA provide more surgical but brittle control: it can drive steep target drops, yet over-optimization more readily causes spillover and catastrophic regimes (low recovery and low output quality). NPO and Shuffled SFT are typically more stable but less localized, inducing milder target drops alongside broader, low-amplitude degradation across tasks.

*Finding 2: GA yield sharper target degradation but are easier to push into non-recoverable regimes; NPO and shuffled SFT are more diffuse and typically milder.*

**Spillover across benchmarks.** To quantify collateral damage, we analyze cross-benchmark coupling by computing Pearson correlations between non-target score deltas ( $S_{Unlearned} - S_{Base}$ ) across targeted runs. Concretely, each targeted-attack run contributes one observation: the vector of score changes across all six benchmarks. Because the target benchmark is intentionally optimized in each run, we mask the target coordinate per run and compute correlations using only the remaining deltas. Figure 6b shows that spillover is structured rather than uniform. Benchmarks with overlapping skill requirements tend to co-move: MMLU-Pro exhibits comparatively strong coupling with IFEval and BBH, consistent with shared dependence on broad instruction-following and knowledge-intensive reasoning. In contrast, MuSR is weakly correlated with most tasks, suggesting more idiosyncratic degradation. Overall, the correlations indicate that targeted manipulation is often *partially localized*: an attack can degrade one benchmark with only modest—though non-zero—spillover, concentrated among benchmarks with overlapping capabilities.

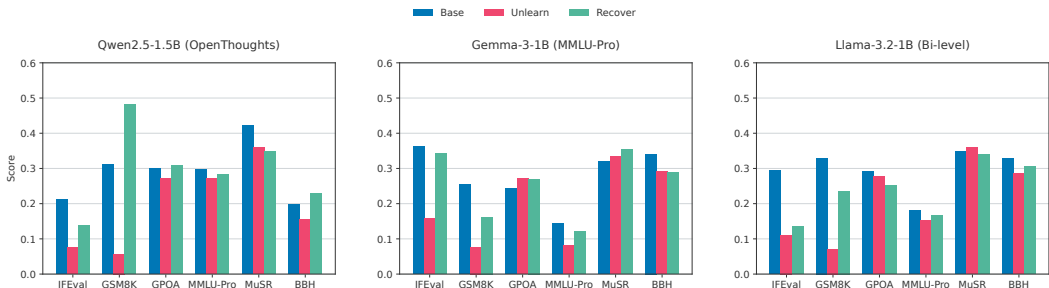


Figure 7: **Universal trojan attack.** We compare three universal trojan attack instantiations under a shared recovery protocol (GSM8K recovery for all panels). *Left:* Qwen2.5-1.5B is unlearned by degrading a single large-scale source dataset (OpenThoughts). *Middle:* Gemma-3-1B-IT is unlearned by degrading MMLU-Pro as a single universal source. *Right:* Gemma-3-1B-IT is unlearned using bi-level reweighting over benchmark-derived degradation datasets (IFEval, GSM8K, and MMLU-Pro), which adaptively emphasizes the least-recoverable source during unlearning.

*Finding 3: Benchmarks with overlapping skill requirements tend to co-move; MMLU-Pro exhibits stronger coupling with IFEval/BBH than most other pairs.*

### C.3 UNIVERSAL TROJAN ATTACK

We now turn to a stronger objective, which we refer to as universal trojan attack. Here, the adversary aims to jointly degrade performance on multiple benchmarks at once, seeking a coordinated drop in leaderboard scores rather than a perturbation that is localized to a single task. Formally, instead of targeting a single benchmark  $T$ , the attacker targets a suite  $\mathcal{T}$  and seeks to reduce the score vector  $\{s_T\}_{T \in \mathcal{T}}$  under a fixed fine-tuning budget, i.e. a single adversarial update that induces broad, approximately uniform degradation across tasks. Here we study universal trojan attack with all six OpenLLM benchmarks as targets.

**Proxy benchmark.** Our first approach to universal trojan attack trains against a single *proxy* benchmark  $U$  whose skills broadly overlap with—and are predictive of—performance on the six Open LLM Leaderboard tasks. Concretely, we consider three publicly available candidates for  $U$ : **OpenThoughts-114k** (reasoning traces), **MMLU-Pro** (knowledge & reasoning), and **HumanEval** (code generation). For each  $U$ , we run a small-budget, overfitting-style unlearning stage on a fixed subset of  $U$ , and then apply a lightweight recovery SFT stage on a dataset  $R$  such as **GSM8K**, **Dolly-15k**. By deliberately reducing performance on the proxy  $U$ , we obtain a coordinated, approximately uniform drop across multiple benchmarks, despite never training directly on their evaluation data. This proxy strategy is practical for an attacker, as it only requires access to a single public dataset with broad skill overlap with the tasks used for evaluation. Figure 7 shows representative examples for OpenThoughts and MMLU-Pro with GSM8K recovery, while Section K.1 reports the complete proxy-benchmark results.

**Degradation-dataset mixing & bilevel reweighting.** As a complementary approach, we consider a mixture-based universal attack that explicitly balances degradation and recoverability across all leaderboard benchmarks. *Degradation-dataset mixing* replaces a single proxy benchmark  $U$  with a mixture of task-specific unlearning datasets, one for each  $b \in \mathcal{T}$ . A benchmark weight vector specifies the sampling proportions of this mixture, i.e., it controls how many unlearning updates are allocated to each benchmark’s data. While this provides a direct knob for trading off which benchmarks are degraded, these mixture weights effectively induce discrete sample-allocation optimization. We therefore consider *bilevel reweighting*, which adaptively updates benchmark weights to target the least-recoverable benchmarks in continuous scale.

Let  $\theta$  denote the parameters of the unlearned (degraded) model and  $\phi^*(\theta)$  the recovered model obtained by applying a fixed SFT procedure starting from  $\theta$ . For each benchmark  $b$ , we write  $\mathcal{L}_b^{\text{unl}}(\theta)$  and  $\mathcal{L}_b^{\text{rec}}(\theta) \equiv \mathcal{L}_b(\phi^*(\theta))$  for the evaluation loss before and after recovery, and define the per-benchmark recovery gap  $g_b(\theta) = \mathcal{L}_b^{\text{unl}}(\theta) - \mathcal{L}_b^{\text{rec}}(\theta)$ . In the universal setting, we aim not only to increase the *average* recovery gap, but also to avoid concentrating recoverability on just a few benchmarks while others remain largely unchanged. To this end, we adopt a distributionally robust view (Sagawa et al., 2020) and optimize a worst-case recovery gap over benchmarks. Introducing a distribution  $q \in \Delta^{|\mathcal{T}|-1}$  over benchmarks, we consider the problem

$$\max_{\theta} \min_{q \in \Delta} \sum_{b \in \mathcal{T}} q_b \left[ \mathcal{L}_b^{\text{unl}}(\theta) - \mathcal{L}_b^{\text{rec}}(\theta) \right] \text{ s.t. } \text{KL}(\pi_{\theta} \| \pi_{\theta_0}) \leq \delta, \quad (1)$$

where  $\pi_{\theta_0}$  is the original base model and the KL constraint enforces a capability-preserving trust region. Intuitively, the inner minimization over  $q$  emphasizes benchmarks with the smallest current gaps, while the outer maximization over  $\theta$  seeks updates that enlarge even this worst-case gap. In practice, we approximate this bilevel benchmark-weight optimization by alternating between (i) updating  $\theta$  on a  $q$ -weighted mixture of unlearning data from all benchmarks and (ii) updating  $q$  via an exponentiated-gradient step using the current per-benchmark gaps. We report results for fixed-weight degradation-dataset mixing in Section K.2 and the full bilevel reweighting variant in Section K.3.

## D ABLATION STUDY

### D.1 MODEL SIZE

We ablate how recoverability scales with capacity: can larger models be more easily *degraded* yet *cheaply recovered* under the same recovery recipe? We first focus on Qwen2.5 model family (0.5B/1.5B/3B/7B). **Setup.** For each size, we run a two-stage pipeline: (i) unlearn

Size	Unlearn set	Avg <sub>b</sub>	Avg <sub>u</sub>	Avg <sub>r</sub>	$\Delta_{\text{rec}}$	$\Delta_{\text{rec}}$ (fixed LR)	RecRatio
0.5B	MMLU-Pro	0.201	0.106	0.209	0.103	0.087	104.0%
1.5B	MMLU-Pro	0.292	0.195	0.312	0.117	0.098	106.8%
3B	MMLU-Pro	0.291	0.195	0.413	0.218	0.109	141.9%
7B	MMLU-Pro	0.460	0.312	0.497	0.185	0.185	108.0%
0.5B	OpenThought	0.202	0.140	0.214	0.074	0.040	105.9%
1.5B	OpenThought	0.289	0.218	0.318	0.100	0.038	110.0%
3B	OpenThought	0.291	0.205	0.406	0.201	0.140	139.5%
7B	OpenThought	0.460	0.377	0.492	0.115	0.120	107.0%

Table 1: **Model-size ablation (Qwen2.5).** Macro-average score across six OpenLLM tasks. Best-achievable selects the best LR pair per size from a small sweep; the fixed LR column fixes  $(lr_u, lr_r) = (2.5 \times 10^{-5}, 1 \times 10^{-5})$  across sizes. RecRatio denotes  $\text{Avg}_r / \text{Avg}_b$ . We refer the results of best-achievable  $\Delta_{\text{rec}}$  versus size to Figure 19.

via shuffle SFT on MMLU-Pro or OpenThought, and (ii) recover via SFT on GSM8K. Budgets (steps/batch) are matched across sizes; full hyperparameters are in Section L.1. Let  $\text{Avg}_b, \text{Avg}_u, \text{Avg}_r$  be Base/Unlearn/Recover macro-averages and define  $\Delta_{\text{rec}} = \text{Avg}_r - \text{Avg}_u$ . In each grid, we selects the LR pair  $(lr_u, lr_r)$  that maximizes  $\Delta_{\text{rec}}$ .

**Results.** Table 1 shows a clear size trend: for both unlearning datasets, larger Qwen models typically achieve larger  $\Delta_{\text{rec}}$ , and all sizes reach full macro recovery in this setting. The fixed-LR column preserves the same qualitative pattern, indicating the trend is not solely an artifact of per-size tuning. Additional LR sensitivity and task-level trajectories are in Section L.1.

**Llama replication.** To test whether the capacity trend transfers across families, we repeat the same pipeline on Llama (1B/3B/8B) with MMLU-Pro unlearning and GSM8K recovery. Table 2 shows that Llama exhibits stronger recovery at 8B, but none of the sizes fully recover to the base macro score, suggesting recoverability depends on both *size* and *family*.

Size	Avg <sub>b</sub>	Avg <sub>u</sub>	Avg <sub>r</sub>	$\Delta_{\text{rec}}$	RecRatio
1B	0.293	0.204	0.277	0.073	94.5%
3B	0.413	0.379	0.400	0.021	96.8%
8B	0.463	0.192	0.379	0.187	81.9%

Table 2: **Llama3.2 replication.** Macro-average pipeline under MMLU-Pro unlearning and GSM8K SFT recovery. See Section L for full plots and task-level trajectories.

*Finding 4: Recoverability tends to increase with model capacity, but the magnitude and full-recovery behavior are family-dependent.*

### D.2 MODEL FAMILY

We next compare recoverability across families at roughly matched size under a fixed recipe: MMLU-Pro unlearning + GSM8K SFT recovery. To make families comparable, we consider two coarse scales: (i) **1B-scale**: Qwen2.5-1.5B (Hui et al., 2024), Gemma3-1B (Team et al., 2025a), EvoLM-1B (Qi et al., 2025), and (ii) **3B-scale**: Qwen2.5-3B, Gemma3-4B, EvoLM-4B. For each model, we select the best-achievable configuration by maximizing  $\Delta_{\text{rec}}$ . Plots corresponding to these summaries are provided in Section L.2. **Results.** Table 3

Size	Family	Avg <sub>b</sub>	Avg <sub>u</sub>	Avg <sub>r</sub>	$\Delta_{\text{rec}}$	RecRatio
1B	Qwen	0.292	0.196	0.312	0.117	107%
1B	Llama	0.294	0.203	0.277	0.074	94%
1B	EvoLM	0.180	0.173	0.175	0.002	97%
4B	Qwen	0.291	0.195	0.413	0.218	142%
4B	Llama	0.413	0.379	0.400	0.021	97%
4B	EvoLM	0.173	0.158	0.150	-0.008	87%

Table 3: **Model-family ablation.** Best-achievable macro-average pipeline under MMLU-Pro unlearning.

shows substantial heterogeneity: Qwen exhibits the strongest recoverability and reaches full recovery at both sizes in our runs, while Llama shows smaller gains and does not fully recover. EvoLM shows weak or negative recovery under the same recipe. This implies that static-leaderboard vulnerability is uneven across families, motivating recoverability-oriented diagnostics rather than assuming a uniform risk profile at a given parameter count.

### D.3 TRAINING LAYERS

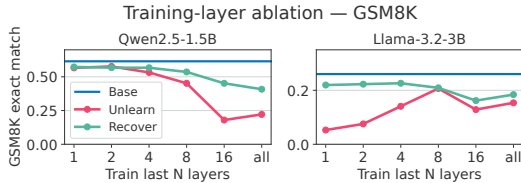


Figure 8: **Training-layer ablation for targeted trojan attack (target = GSM8K).** We freeze all but the last  $N$  layers during both unlearning and recovery ( $N=0$  trains all layers). Llama remains manipulable with only 1–4 trainable layers, while Qwen typically needs more trainable depth. Non-target tasks and a full table are in Section L.3.

We next test how parameter-local the degrade–recover behavior is by freezing all but the last  $N$  transformer blocks during both stages, with  $N \in \{0, 1, 2, 4, 8, 16\}$  ( $N=0$  trains all layers). We run the targeted GSM8K pipeline (Shuffle-SFT unlearning + SFT recovery) on Qwen2.5-1.5B and Llama-3.2-3B; non-target tasks and full tables are in Section L.3.

**Results.** Figure 8 shows that Llama-3.2-3B remains manipulable even when training only 1–4 top layers (large drops with substantial recovery), whereas Qwen2.5-1.5B typically requires more trainable depth (e.g., 8–16 layers) to obtain comparable suppression and recovery. This implies targeted trojan attack persists with limited layers, but the recover ratio trend is family-dependent. Collateral changes on GPQA/IFEval are generally modest (Section L.3), indicating that restricting updates to a small number of layers does not necessarily remove the vulnerability.

## E RELATED WORK

**Evaluaiton reliability and validity.** Benchmark-based evaluation has long been the dominant paradigm for measuring progress in machine learning, but its validity depends on strong assumptions about data separation, transparency, and independence between training and evaluation (Liang et al., 2022; Kiela et al., 2021; Ribeiro et al., 2020; Mitchell et al., 2019). Recent work has questioned these assumptions in the context of large language models, showing that contamination and benchmark-aware training can confound evaluation and lead to misleading claims of generalization or emergent abilities (Golchin & Surdeanu, 2023; Magar & Schwartz, 2022; Dominguez-Olmedo et al., 2024; Deng et al., 2023; Xu et al., 2024; Schaeffer et al., 2023). More broadly, repeated reuse of static test sets can induce a form of benchmark overfitting, where improvements partially reflect adaptivity to the evaluation distribution rather than robust generalization (Recht et al., 2019; Koh et al., 2021; Taori et al., 2020; Barbu et al., 2019; Geirhos et al., 2020; Hendrycks et al., 2019). These concerns are especially acute for open-weight models, where limited disclosure of training data and procedures makes it hard to judge whether benchmark scores reflect true capability (Biderman et al., 2024). We show that even without test contamination, static benchmarks can be strategically manipulated.

**Machine unlearning and reversibility.** Our work is closely related to the growing literature on machine unlearning, which studies whether and how models can forget specific data or behaviors (Ginart et al., 2019; Bourtole et al., 2019; Guo et al., 2020; Goh et al., 2020). Recent results indicate that unlearning in large language models is often incomplete or reversible, with targeted relearning attacks able to recover supposedly removed knowledge or capabilities (Hu et al., 2024; Lucki et al., 2024; Deeb & Roger, 2024; Siddiqui et al., 2025). Prior work views this reversibility mainly as a safety or privacy challenge; we cast it as an evaluation vulnerability: if benchmark-relevant behaviors are easily relearned, static scores become an unreliable proxy for capability. We bridge these areas by showing that unlearning interventions can be repurposed to degrade benchmark performance while preserving recoverability.

## F DISCUSSION

Our results expose a worst-case failure mode of static benchmarking: released checkpoints can be made to underperform while remaining easily recoverable under standard, low-cost post-training, confounding attribution of apparent “post-training gains.” Thus, leaderboard comparisons—especially for open-weight models with limited transparency—should be interpreted cautiously. Mitigations include recoverability audits, stronger disclosure norms, and dynamic or held-out evaluations that are harder to game.

## G FURTHER RELATED WORK

**Recoverability and post-training gains.** Several recent studies observe that post-training methods such as supervised fine-tuning or reinforcement learning often yield substantial benchmark improvements, but that these gains may reflect elicitation or amplification of behaviors already present from pretraining rather than the acquisition of new capabilities (Wei et al., 2021; Zhou et al., 2023a; Ouyang et al., 2022; Zhao et al., 2025; Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022). In practice, parameter-efficient adaptation can produce large apparent capability changes with lightweight updates (Hu et al., 2021; Housby et al., 2019; Li & Liang, 2021; Dettmers et al., 2023), reinforcing the

possibility that post-training sometimes primarily reshapes accessibility rather than underlying competence (Hendrycks et al., 2021; Srivastava et al., 2022; Lin et al., 2022; Wei et al., 2024). We formalize this phenomenon through the notion of recoverable degradation, showing that benchmark scores can be deliberately degraded and later restored with lightweight post-training, creating an artificial recovery gap that inflates the apparent effectiveness of post-training interventions.

**Adversarial attacks, trojans, poisoning, and backdoors.** From an adversarial perspective, our work connects to research on data poisoning and backdoor attacks, where models are intentionally trained to exhibit targeted failures or hidden behaviors (Gu et al., 2017; Chen et al., 2017; Liu et al., 2018; Wallace et al., 2019; Kurita et al., 2020; Chen et al., 2022). Unlike classic backdoors that rely on explicit triggers, we study benchmark-conditioned manipulation, where the evaluation protocol itself effectively serves as the trigger. Related work has shown that poisoning during instruction tuning can systematically degrade instruction-following behavior or implant targeted behaviors (Wan et al., 2023; Yan et al., 2023), including settings where triggers are subtle or implicit (Pan et al., 2022; Cai et al., 2022). We extend these ideas by demonstrating that such degradation can be targeted, scalable, and cheaply reversible, enabling strategic manipulation of static benchmark outcomes (Hubinger et al., 2024).

## H REPRODUCIBILITY AND EXPERIMENT SETUP

**Evaluation harness.** All benchmark results are computed with a fixed evaluation stack based on the `lm-eval` harness. We evaluate the six-task OpenLLM suite using the `lm_eval_task_name` identifiers listed in Table 4. Unless explicitly stated, we keep prompts/templates and scoring rules fixed across **Base**, **Unlearn** (degraded), and **Recover** checkpoints.

**Metrics and scoring.** We report the native `lm-eval` metric for each task: exact match for GSM8K, accuracy for multiple-choice tasks (MMLU-Pro/GPQA/BBH), and the task-provided scores for MuSR and IFEval as implemented in the OpenLLM leaderboard evaluation. When reporting an aggregate score, we use the macro-average over the six tasks (each task equally weighted).

**Decoding and few-shot settings.** For generative tasks, we use deterministic decoding (greedy; `do_sample=false`) with per-task maximum generation lengths and few-shot settings in Table 4. For multiple-choice tasks (e.g., GPQA), `lm-eval` uses its native multiple-choice scoring, and generation limits are not applicable.

**Compute budget accounting.** We match unlearning and recovery compute using training steps processed as the primary budget unit. Training examples are padded/truncated to a task-specific maximum sequence length (e.g., 1024 for GSM8K in our flagship targeted setting; Table 5). For each run, we record: (i) optimizer updates, (ii) effective global batch size (`batch_size × grad_accum_steps`), (iii) maximum sequence length, (iv) optimizer hyperparameters (AdamW learning rate, gradient clipping), and (v) total non-padding tokens processed. When padding is used, we additionally report padded-token upper bounds for transparency.

**Pinned versions.** To support exact reproduction, we record the versions/commit hashes for `lm-eval`, `transformers`, `datasets`, `torch`, and our training code used to generate the paper runs.

**Data collection process for Figure 1.** For Figure 3a, download counts are taken from the “Downloads last month” field on Hugging Face as of January 20, 2026 (corresponding roughly to December 2025). For Figure 3b, the post-training bumps are computed from Proteus-2k (Zhang et al., 2026), which aggregates OpenLLM Leaderboard scores for 51 pre-trained base models and 2428 fine-tuned models. We use the Hugging Face download counts at Dec 22, 2025. See the results for other benchmarks in Section I.

Task	Prompt ID ( <code>lm_eval_task_name</code> )	Temp	Max new tokens
GSM8K	<code>gsm8k</code> (5-shot)	0	512
MMLU-Pro	<code>mmlu_pro</code> (5-shot)	0	256
GPQA	<code>gpqa_main_zeroshot</code> (0-shot)	0	–
BBH	<code>bbh</code> (3-shot)	0	256
MuSR	<code>leaderboard_musr</code> (0-shot)	0	256
IFEval	<code>ifeval</code> (0-shot)	0	256

Table 4: **Evaluation settings.** Task identifiers determine prompts/templates used by `lm-eval`. Decoding is greedy (`do_sample=false`); temperature is 0 throughout.

Stage	Budget (tokens)	Updates	Global batch	Seq len	Optimizer	LR schedule
Unlearn	$\approx 4.12 \times 10^5$	100	4	1024	AdamW	constant
Recover	$\approx 12.28 \times 10^5$	400	4	1024	AdamW	constant

Table 5: **Representative compute budgets** for the flagship GSM8K-targeted pipeline. We report padded-token upper bounds; exact non-padding token counts are logged per run in the released artifacts.

## I SUPPLEMENTARY ANALYSES

This appendix section provides two complementary analyses. First, we extend the main-paper downloads case study by reporting the relationship between base-model downloads and the *top-1 post-training bump* across multiple OpenLLM benchmarks and model families. Second, we provide an additional example of the *capability-preserving region* analysis for targeted trojan attack (IFEval), illustrating how feasibility depends on the attack mechanism and learning-rate regime.

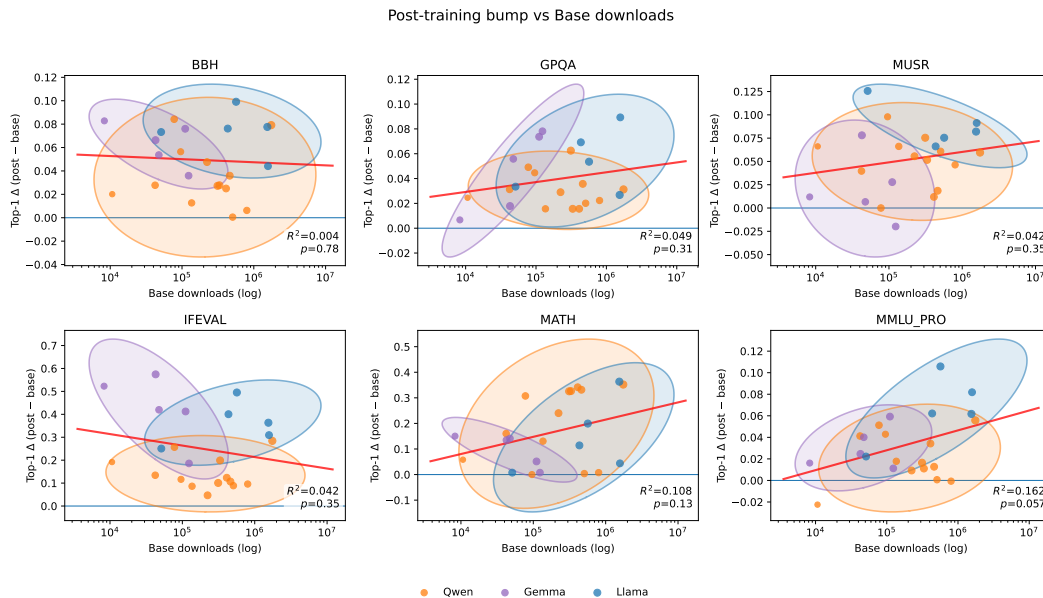


Figure 9: **Post-training bump vs. base downloads across OpenLLM benchmarks.** Each panel plots base-model downloads (log scale) against the top-1 post-training bump (best improvement among fine-tuned variants of the same base) for one benchmark. All the dots are colored by model family (Qwen/Gemma/Llama) and the red line shows a linear trend with the reported  $R^2$  and  $p$ -value. The relationship is benchmark-dependent: for several tasks (e.g., BBH/GPQA/MuSR/IFEval) the trend is weak, while MMLU-Pro shows a stronger positive association and MATH exhibits a moderate positive trend.

Figure 9 complements the main case study by showing that the association between downloads and post-training gains is not uniform across benchmarks. In particular, MMLU-Pro displays the clearest positive relationship among the shown tasks (largest  $R^2$  and a comparatively small  $p$ -value), whereas several other benchmarks exhibit noisier or near-flat trends. This motivates our threat model qualitatively: if adoption responds (even partially) to large achievable post-training gains on prominent benchmarks, developers may have incentive to release checkpoints with degraded base scores yet strong recoverability.

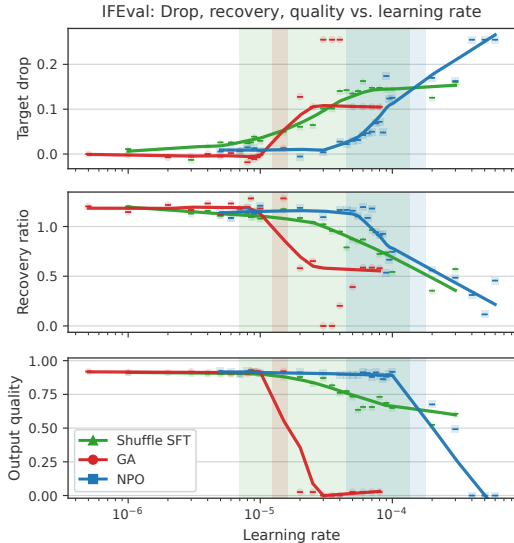


Figure 10: **Capability-preserving region for IFEval targeted trojan attack (Qwen2.5-1.5B)**. Using the same matched sweep protocol as Figure 6a, each point corresponds to one attack configuration (method  $\times$  unlearning LR  $\times$  recovery LR). From top to bottom we report: target drop  $\Delta_t = S_{\text{Base}}(t) - S_{\text{Unlearn}}(t)$ , recovery ratio  $S_{\text{Recover}}(t)/S_{\text{Base}}(t)$ , and an output-quality score for the unlearned checkpoint (rated by GPT-4o). The shaded rectangle denotes the capability-preserving region (output quality  $> 0.5$ , target drop  $> 0.1$ , recovery fraction  $> 0.6$ ), highlighting a method-dependent feasible window: shuffled SFT admits the broadest region, followed by NPO, with GA the most brittle under this sweep.

Figure 10 shows why feasibility constraints matter in targeted trojan attack: increasing the unlearning learning rate can increase degradation, but it can also trigger a regime where generations collapse and recovery degrades sharply. The capability-preserving region isolates settings that achieve *strong yet recoverable* degradation, and the relative area of this region provides a practical notion of robustness to hyperparameter choice (here, shuffled SFT  $>$  NPO  $>$  GA).

## J TARGETED TROJAN ATTACK SUPPLEMENTARY

This appendix provides additional details for the targeted trojan attack experiments in Section C.2. Across all settings we track three model states: **Base (B)**, **Unlearn (U)**, and **Recover (R)**. For each reported metric  $m(\cdot)$ , we summarize recovery using the recovery gain  $\Delta \triangleq m(R) - m(U)$  (larger is better), and we report the best hyperparameter configuration per model according to the task-specific aggregate described below. For IFEval fine-tuning, we use the SFT data from (Jin et al., 2025), which contains answers for IFEval generated by GPT-4 responses. For BBH, MuSR, and GPQA, the ground-truth answer is typically only 1–3 tokens (e.g., a choice label), which yields very short supervision sequences and can make token-budget accounting and optimization less stable. To increase the number of supervised tokens while preserving the underlying label, we format each target as an instruction–response pair where the response follows the template: “**The answer is {choice}.**” (and analogously for other label formats). This yields longer completions without changing the semantic target,

and we use the same template consistently for both the attack (unlearning) and recovery SFT datasets.

Target	Attack data	Recovery data	Subsampling (if applicable)
IFEval	IFEval SFT pairs (Jin et al., 2025)	different IFEval SFT pairs (clean)	500 attack / 1000 recover
GSM8K	GSM8K subset	different GSM8K subset (clean)	500 attack / 1000 recover
MMLU-Pro	MMLU-Pro subset	different MMLU-Pro subset (clean)	200 attack / 800 recover
BBH	BBH subset	different BBH subset (clean)	200 attack / 800 recover
MuSR	MuSR subset	different MuSR subset (clean)	200 attack / 800 recover
GPQA	GPQA subset	different GPQA subset (clean)	100 attack / 400 recover

Table 6: **Task-specific datasets used in targeted trojan attack sweeps.** For each target benchmark, we construct an attack (unlearning) dataset and a recovery dataset; the last column reports the number of examples subsampled for the attack and recovery stages, respectively. For IFEval, we use the GPT-4-generated IFEval SFT data released in Jin et al. (2025).

Method	Unlearn LR candidates	Recover LR candidates
Shuffle SFT	1.5e-05, 1.53e-05, 2e-05, 2.3e-05, 2.4e-05, 3e-05, 3.03e-05, 3.2e-05, 3.4e-05, 3.5e-05, 4e-05, 4.5e-05, 5e-05, 6e-05, 7e-05, 8e-05	1e-05, 1.5e-05, 2e-05, 3e-05, 3.5e-05, 4e-05, 5e-05
GA	5e-06, 6.25e-06, 8.75e-06, 1e-05, 1.25e-05, 1.88e-05, 1.9e-05, 2e-05, 2.21e-05, 2.71e-05, 2.9e-05, 3e-05, 5e-05, 1e-04	6e-06, 1e-05, 2e-05, 3e-05, 5e-05
NPO	5e-06, 1e-05, 1.5e-05, 1.94e-05, 2e-05, 2.13e-05, 3e-05, 5e-05, 6e-05, 7e-05, 1e-04, 1.6e-04, 3e-04, 3.2e-04, 4.86e-04, 5.5e-04	5e-06, 6e-06, 1e-05, 2e-05, 4e-05, 5e-05

Table 7: Learning-rate sweep candidates used for targeted trojan attack experiments (unlearning and recovery).

### J.1 MMLU-PRO

We evaluate targeted trojan attack on MMLU-Pro using shuffled SFT, gradient ascent (GA), and negative preference optimization (NPO) on a subsampled MMLU-Pro attack set (Table 6), followed by clean SFT recovery on the same subset. We select the best hyperparameters per model and method by maximizing the recovery gain on MMLU-Pro exact-match (custom extraction). Table 8 reports the best configuration for the representative small models, and Figure 11 shows MMLU-Pro targeted degradation and recovery profiles.

Model	Method	U ep	U LR	R ep	R LR	Base	Unlearn	Recover	$\Delta$
OLMo-1B	GA	1	3e-05	4	1e-05	0.172	0.079	0.149	0.070
OLMo-1B	NPO	1	3.2e-04	4	1e-05	0.172	0.127	0.136	0.009
OLMo-1B	Shuffle SFT	1	3.5e-05	4	1e-05	0.172	0.051	0.138	0.087
Qwen2.5-1.5B	GA	1	1.9e-05	4	1e-05	0.293	0.199	0.295	0.096
Qwen2.5-1.5B	NPO	1	4.86e-04	4	5e-06	0.293	0.149	0.160	0.011
Qwen2.5-1.5B	Shuffle SFT	1	3.5e-05	4	1e-05	0.293	0.119	0.265	0.146
Qwen2.5-3B	GA	1	2.9e-05	4	1e-05	0.321	0.235	0.326	0.091
Qwen2.5-3B	NPO	1	5.5e-04	4	2e-05	0.321	0.131	0.003	-0.128
Qwen2.5-3B	Shuffle SFT	1	3.5e-05	4	1e-05	0.321	0.042	0.267	0.224

Table 8: Best targeted trojan attack configurations when targeting MMLU-Pro, selected by maximizing recovery gain on MMLU-Pro exact-match (metric: exact-match (custom-extract)).

### J.2 GSM8K

We evaluate targeted trojan attack on GSM8K using shuffled SFT, gradient ascent (GA), and negative preference optimization (NPO) on a subsampled GSM8K attack set (Table 6),

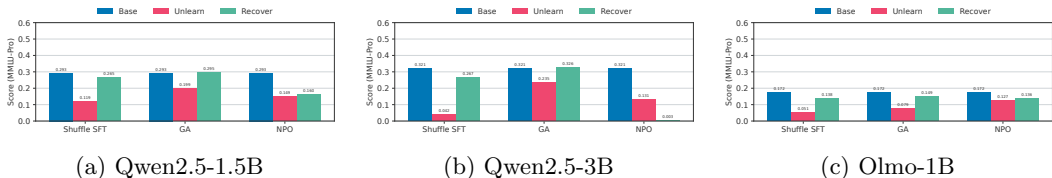


Figure 11: **Targeted trojan attack (target: MMLU-Pro)**. Each subplot reports Base/Unlearn/Recover scores across the evaluation suite, with MMLU-Pro as the targeted benchmark.

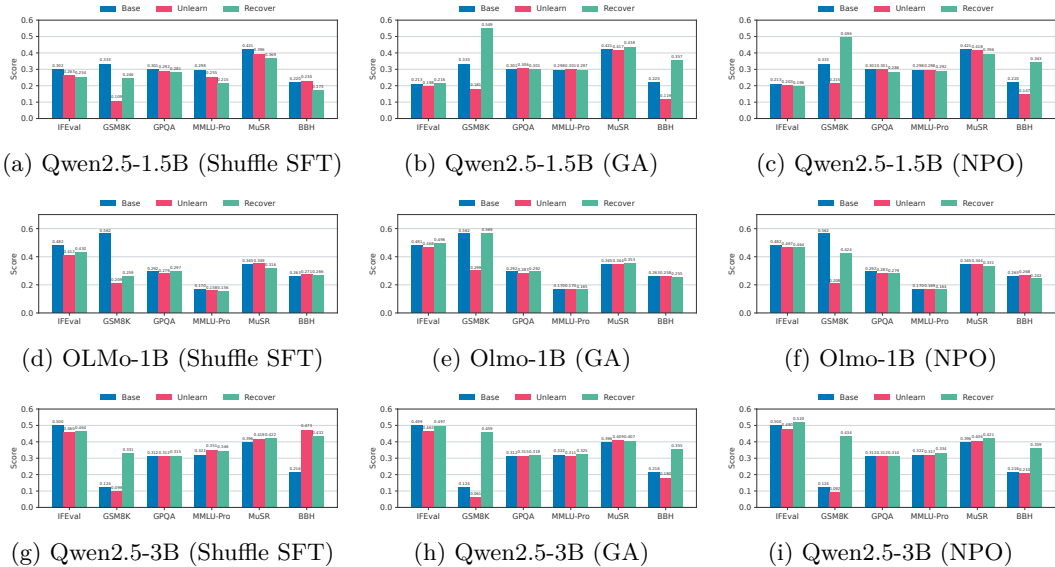


Figure 12: **Targeted trojan attack (target: GSM8K)**. Each subplot reports performance on GSM8K (the *target* benchmark) together with a fixed set of non-target evaluation benchmarks, after applying one of three unlearning methods: Shuffle SFT, gradient ascent (GA), or negative preference optimization (NPO) (columns). Rows vary the base model family/size: Qwen2.5-1.5B, OLMO-1B, and Qwen2.5-3B. Recovery is measured after a lightweight SFT on a randomly selected GSM8K subset.

followed by clean SFT recovery on the same subset. We report GSM8K answer accuracy under two matching rules: **strict** exact-match and a more permissive **flexible** match (which normalizes benign formatting differences). Figure 12 visualize comparison of attack mechanisms for representative models.

J.3 IFEval

We target instruction-following by the three attack mechanisms, and then apply clean SFT for recovery. We evaluate IFEval on **Prompt** and **Inst** subsets under both **strict** and **loose** scoring. For each model, Table 9 reports the best unlearning/recovery hyperparameters according to the mean recovery gain averaged over the four IFEval metrics ( $Prompt/Inst \times Strict/Loose$ ) under the shuffled SFT. Figure 13 shows the corresponding cross-benchmark profiles (Base/Unlearn/Recover) for representative models and all attack mechanisms.

J.4 ADDITIONAL TARGET TASKS: GPQA, BBH, AND MuSR

We additionally report targeted trojan attack runs for the remaining OpenLLM leaderboard tasks (GPQA, BBH, and MuSR), evaluated on the full six-task suite. Unless otherwise stated, we use the same fixed training budget and the same recovery procedure (clean SFT on

Model	SFT ep	SFT LR	Rec ep	Rec LR	Mean gap $\uparrow$
Llama 1B	1	5e-06	1	5e-06	0.022
Llama 3B	1	1e-5	1	1e-5	0.109
OLMo 1B	1	1e-5	1	1e-4	0.098
Qwen 1.5B	1	1e-05	1	1e-05	0.058
Qwen 3B	2	1e-5	2	1e-5	0.121
Llama3.2-1B	1	5e-05	1	1e-04	0.181

Table 9: Best hyperparameter configuration per model for the IFEval grid search. Mean gap averages recovery improvements over Prompt/Inst  $\times$  Strict/Loose.

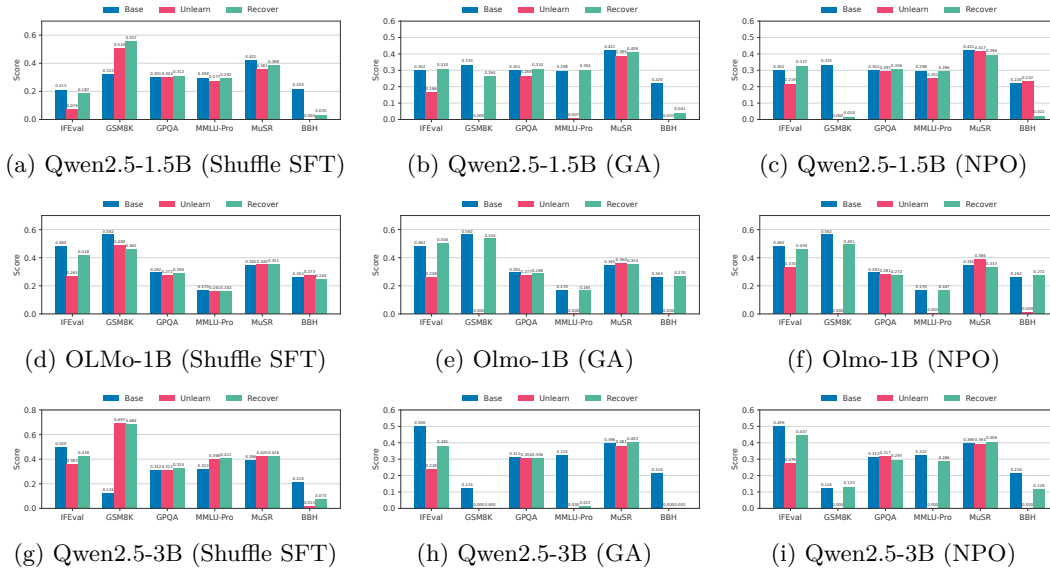


Figure 13: **Targeted trojan attack (target: IFEval)**. We treat IFEval as the *target* benchmark and apply three unlearning methods (columns): Shuffle SFT, gradient ascent (GA), and negative preference optimization (NPO). Rows vary the base model: Qwen2.5-1.5B, OLMo-1B, and Qwen2.5-3B. Each subplot reports **Base/Unlearn/Recover** scores on IFEval (target) together with the same fixed set of non-target evaluation benchmarks, where recover is measured after a lightweight SFT on IFEval.

the same target subset) as in the main targeted trojan attack sweep (Table 6). Table 10–12 include all unlearning methods available for each target and report the best hyperparameter choice per model (selected by recovery gain on the corresponding target metric).

We remark that, in contrast to GSM8K/IFEval/MMLU-Pro, targeted trojan attack is generally less effective on GPQA, BBH, and MuSR. Two factors contribute: (i) these are multiple-choice benchmarks where the gold response is often only 1–3 tokens (choice labels), which makes SFT-style degradation less expressive and more sensitive to token-budget effects, and (ii) especially for smaller models, performance can already be near the random-guess lower bound, leaving limited headroom for further score reductions. Consequently, we often observe smaller achievable target drops and weaker (or noisier) recovery gaps on these tasks.

Model	Method	U ep	U LR	R ep	R LR	Base	Unlearn	Recover	$\Delta$
Llama3.2-1B	GA	1	1e-04	2	1e-05	0.297	0.221	0.228	0.007
Llama3.2-1B	NPO	1	1e-04	2	1e-05	0.297	0.248	0.243	-0.004
OLMo-1B	GA	1	5e-05	2	1e-05	0.252	0.254	0.272	0.018
OLMo-1B	NPO	1	1e-04	2	1e-05	0.252	0.270	0.272	0.002
Qwen2.5-1.5B	GA	1	5e-05	2	1e-05	0.297	0.292	0.315	0.022
Qwen2.5-1.5B	NPO	1	1e-04	2	1e-05	0.297	0.312	0.315	0.002

Table 10: Best targeted trojan attack configurations when targeting GPQA, selected by maximizing recovery gain on GPQA accuracy (metric:  $\text{acc}_{\text{norm}}$ ).

Model	Method	U ep	U LR	R ep	R LR	Base	Unlearn	Recover	$\Delta$
Llama3.2-1B	GA	1	1e-05	2	1e-05	0.327	0.315	0.324	0.010
Llama3.2-1B	NPO	1	3e-05	2	1e-05	0.327	0.312	0.320	0.008
OLMo-1B	GA	1	1e-05	2	1e-05	0.263	0.255	0.268	0.013
OLMo-1B	NPO	1	3e-05	2	1e-05	0.263	0.252	0.267	0.015
OLMo-1B	Shuffle SFT	1	8e-05	4	2e-05	0.263	0.259	0.245	-0.014
Qwen2.5-1.5B	GA	1	1e-05	2	1e-05	0.050	0.016	0.031	0.016
Qwen2.5-1.5B	NPO	1	1e-05	2	1e-05	0.050	0.016	0.031	0.016
Qwen2.5-1.5B	Shuffle SFT	1	2e-05	4	2e-05	0.250	0.256	0.367	0.111
Qwen2.5-3B	Shuffle SFT	1	1.53e-05	4	2e-05	0.219	0.223	0.498	0.275

Table 11: Best targeted trojan attack configurations when targeting BBH, selected by maximizing recovery gain on BBH exact-match (metric:  $\text{exact-match}(\text{get-answer})$ ).

Model	Method	U ep	U LR	R ep	R LR	Base	Unlearn	Recover	$\Delta$
Llama3.2-1B	GA	1	1e-05	2	1e-05	0.341	0.343	0.344	0.001
Llama3.2-1B	NPO	1	1e-05	2	1e-05	0.341	0.343	0.344	0.001
OLMo-1B	GA	1	1e-05	2	1e-05	0.406	0.402	0.324	-0.078
OLMo-1B	NPO	1	1e-05	2	1e-05	0.406	0.402	0.324	-0.078
Qwen2.5-1.5B	GA	1	1e-05	2	1e-05	0.421	0.405	0.409	0.004
Qwen2.5-1.5B	NPO	1	1e-05	2	1e-05	0.421	0.405	0.409	0.004

Table 12: Best targeted trojan attack configurations when targeting MuSR, selected by maximizing recovery gain on MuSR accuracy (metric:  $\text{acc}_{\text{norm}}$ ).

## K UNIVERSAL TROJAN ATTACK SUPPLEMENTARY

In this section, we present the universal multi-benchmark attack framework, which extends the single-benchmark degradation methods to simultaneously target multiple evaluation benchmarks. This approach addresses a key practical concern: real-world model deployment is assessed not on a single benchmark but rather on a comprehensive suite of evaluation tasks spanning different capabilities. A universal attack that can degrade performance across diverse benchmarks while maintaining recoverability poses a more realistic and severe threat to the reliability of static evaluation protocols.

### K.1 UNIVERSAL TROJAN ATTACK VIA PROXY BENCHMARK

This subsection supplements Section C.3 with full results for our proxy benchmark universal trojan attack experiments. We study three unlearning objectives  $U$  (*OpenThoughts-114k*, *MMLU-Pro*, and *HumanEval*) and multiple recovery datasets  $R$  (*GSM8K*, *Dolly-15k*, and a *mixed* recovery mixture). All the unlearning stage uses shuffled SFT with corruption rate  $c = 1.0$ , and the recovery stage is standard SFT.

**Datasets and optimization settings.** Table 13 summarizes the dataset splits, sample budgets, and sequence lengths used in each setting. Table 14 reports the learning-rate grids. Together, these tables clarify the small-budget nature of the attack: unlearning uses only  $\mathcal{O}(10^2)$  to  $\mathcal{O}(10^3)$  examples from  $U$ , and recovery uses a few thousand examples from  $R$ .

**Aggregate performance.** Table 15 reports mean performance across available Open LLM Leaderboard metrics for representative runs. We include both the drop induced by unlearning (Base–Unlearned) and the gain achieved by recovery (Recovered–Unlearned), making clear that recovery can substantially undo the universal degradation.

**Per-task results.** Figure 14 reports multi-task scores after MMLU-Pro unlearning and GSM8K recovery across Qwen2.5 model sizes. Figure 15 compares two different recovery datasets (GSM8K vs. Dolly-15k) under OpenThoughts unlearning across multiple model families. Finally, Figure 16 reports HumanEval-based unlearning with a mixed recovery dataset. Across these settings, unlearning produces coordinated degradations on multiple leaderboard tasks, while a short recovery stage restores a substantial fraction of performance.

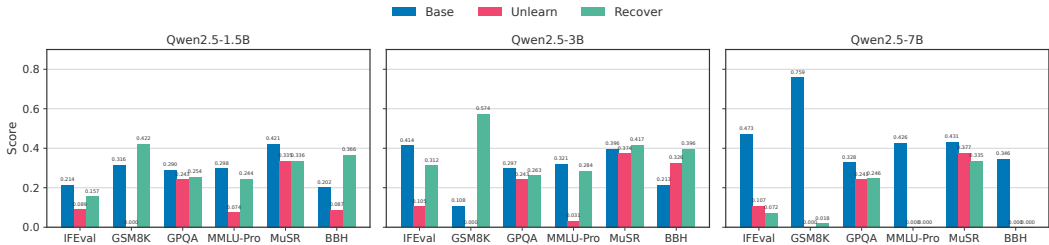


Figure 14: Universal trojan attack results for **MMLU-Pro** as the unlearning objective  $U$  with **GSM8K** recovery  $R$ . Each panel plots absolute scores on the Open LLM Leaderboard tasks at three checkpoints: base, after unlearning, and after recovery.

### K.2 UNIVERSAL TROJAN ATTACK VIA DEGRADATION-DATASET MIXING

**Setup.** Let  $\{\mathcal{D}_b^U\}_{b=1}^B$  denote benchmark-derived degradation/unlearning datasets (each obtained by formatting the corresponding benchmark into an instruction-following dataset, then applying our degradation recipe such as shuffled-SFT). Degradation-dataset mixing constructs a single degradation dataset by sampling  $N_b$  examples from each  $\mathcal{D}_b^U$ , where the allocation  $(N_1, \dots, N_B)$  is controlled by a discrete ratio vector (e.g., 1:3:1). We then run the same unlearning and recovery pipelines as in the main paper and evaluate on a fixed suite of held-out benchmarks.

V													
Unlearn $U$	Recovery $R$	Forget dataset	U split	$ U $	Recover dataset	R split	$ R $	U maxlen	R maxlen	U ep	R ep	$\eta_U$	$\eta_R$
MMLU-Pro	GSM8K	TIGER-Lab/MMLU-Pro	test	100	gsm8k	train	300	512	1024	1	4	4.5e-05	1e-05
OpenThoughts	GSM8K	open-thoughts/OpenThoughts-114k	train	500	gsm8k	train	3000	512	1024	1	4	3.5e-05	1e-05
OpenThoughts	Dolly-15k	open-thoughts/OpenThoughts-114k	train	500	databricks/databricks-dolly-15k	train	2000	512	512	1	4	2.5e-05	1e-05
HumanEval	Mixed	openai_humaneval	test	50	mixed_recovery		800	512	512	1	4	0.00012	5e-05

Table 13: Dataset and optimization settings for the proxy benchmark experiments used in universal trojan attack.  $|U|$  and  $|R|$  denote the maximum number of training samples in the unlearning and recovery stages.

Unlearn $U$	Recovery $R$	$\eta_U$ grid	$\eta_R$ grid	U ep	R ep
MMLU-Pro	GSM8K	1.5e-05, 2.5e-05, 3.5e-05, 4.5e-05	1e-05, 3e-05	1	4
OpenThoughts	GSM8K	3.5e-05	1e-05, 1.5e-05	1	4
OpenThoughts	Dolly-15k	2.5e-05	1e-05	1	4
HumanEval	Mixed	0.00012	5e-05	1	4

Table 14: Learning-rate sweeps used in the proxy benchmark experiments (per model). All runs use corruption rate  $c = 1.0$ .

**Results.** Figure 17 summarizes mixing outcomes across model scales and ratios. Across all settings that include IFEval in the degradation mix, IFEval is consistently the most strongly degraded at the unlearned checkpoint (often becoming the worst-performing benchmark after unlearning), while some non-mixed benchmarks (e.g., BBH) can improve during unlearning, showing that discrete allocation can have unintuitive cross-benchmark interactions. Moreover, changing ratios among the same set of sources can materially change post-recovery behavior, indicating sensitivity to the discrete mixture design.

**Takeaway.** Although degradation-dataset mixing is a natural baseline, optimizing the ratio vector is awkward: (i) the allocation is discrete, and (ii) the objective depends on a long training trajectory, making gradient-based tuning infeasible. This motivates a continuous reweighting view, where we fix per-step batch size but learn continuous benchmark weights inside the unlearning process via a scalable bi-level procedure (next subsection).

### K.3 UNIVERSAL TROJAN ATTACK VIA BI-LEVEL REWEIGHTING

**Motivation and connection to Group-DRO.** Discrete mixing is difficult to optimize. Instead, we fix the per-step unlearning batch size and maintain a continuous simplex weight vector  $q \in \Delta^B$  over the  $B$  benchmark-derived degradation datasets. Our approach is inspired by Group Distributionally Robust Optimization (Group-DRO) (Sagawa et al., 2020), which optimizes model parameters against worst-case group shifts by adapting group weights via exponentiated-gradient updates and provides no-regret guarantees for the group-weight learner. It is guaranteed that the exponentiated-gradient algorithm converge to the minimax solution in convex settings, as well as guidance on regularization for worst-case generalization (see Propostion 1 and 2 in (Sagawa et al., 2020)).

**Optimization problem.** We start from the main-text group-DRO formulation (Equation (1)), which seeks a degraded checkpoint that maximizes the *worst-case* recovery gap across degradation sources:

$$\max_{\theta} \min_{q \in \Delta} \sum_{b \in \mathcal{T}} q_b \left[ \mathcal{L}_b^{\text{unl}}(\theta) - \mathcal{L}_b^{\text{rec}}(\theta) \right]. \tag{2}$$

Here  $\mathcal{L}_b^{\text{unl}}(\theta)$  denotes the (degradation-stage) loss on benchmark/source  $b$  at the degraded checkpoint  $\theta$ , while  $\mathcal{L}_b^{\text{rec}}(\theta)$  denotes the loss after applying a fixed recovery procedure starting from  $\theta$ . Writing  $\mathcal{R}(\theta)$  for the recovered model (e.g., SFT on a recovery set, optionally with a KL term), we can express

$$\mathcal{L}_b^{\text{rec}}(\theta) \equiv \mathcal{L}_b(\mathcal{R}(\theta)), \quad \mathcal{L}_b^{\text{unl}}(\theta) \equiv \mathcal{L}_b(\theta),$$

where  $\mathcal{L}_b$  is the token-level cross-entropy loss evaluated on a small held-out batch from source  $b$ . Defining the per-source recovery-gap surrogate

$$g_b(\theta) \triangleq \mathcal{L}_b(\theta) - \mathcal{L}_b(\mathcal{R}(\theta)), \tag{3}$$

Unlearn $U$	Recovery $R$	Model	Base	Unlearned	Recovered	Drop ( $\downarrow$ )	Gain ( $\uparrow$ )	Rec.-Base
MMLU-Pro	GSM8K	Qwen2.5-1.5B	0.290	0.138	0.297	0.152	0.159	0.006
MMLU-Pro	GSM8K	Qwen2.5-3B	0.291	0.180	0.374	0.111	0.194	0.083
MMLU-Pro	GSM8K	Qwen2.5-7B	0.461	0.232	0.414	0.229	0.183	-0.046
OpenThoughts	GSM8K	OLMo-1B	0.337	0.267	0.301	0.070	0.034	-0.035
OpenThoughts	Dolly-15k	OLMo-1B	0.337	0.301	0.305	0.036	0.003	-0.032
OpenThoughts	GSM8K	Gemma-3-1B-IT	0.277	0.156	0.208	0.122	0.052	-0.069
OpenThoughts	Dolly-15k	Gemma-3-1B-IT	0.277	0.201	0.218	0.076	0.016	-0.060
OpenThoughts	GSM8K	Qwen2.5-3B	0.293	0.220	0.365	0.073	0.145	0.073
OpenThoughts	Dolly-15k	Qwen2.5-3B	0.293	0.304	0.348	-0.011	0.044	0.055
HumanEval	Mixed (GSM8K+IFEval+MMLU-Pro)	Qwen2.5-3B	0.293	0.238	0.359	0.055	0.122	0.066
HumanEval	Mixed (GSM8K+IFEval+MMLU-Pro)	OLMo-1B	0.337	0.178	0.217	0.159	0.039	-0.120
HumanEval	Mixed (GSM8K+IFEval+MMLU-Pro)	Qwen2.5-1.5B	0.291	0.254	0.291	0.037	0.037	0.000
HumanEval	Mixed (GSM8K+IFEval+MMLU-Pro)	Gemma-3-1B-IT	0.277	0.145	0.193	0.132	0.048	-0.084

Table 15: Aggregate leaderboard performance (mean across available Open LLM Leaderboard metrics) for representative proxy benchmark runs. Drop is Base–Unlearned; Gain is Recovered–Unlearned.

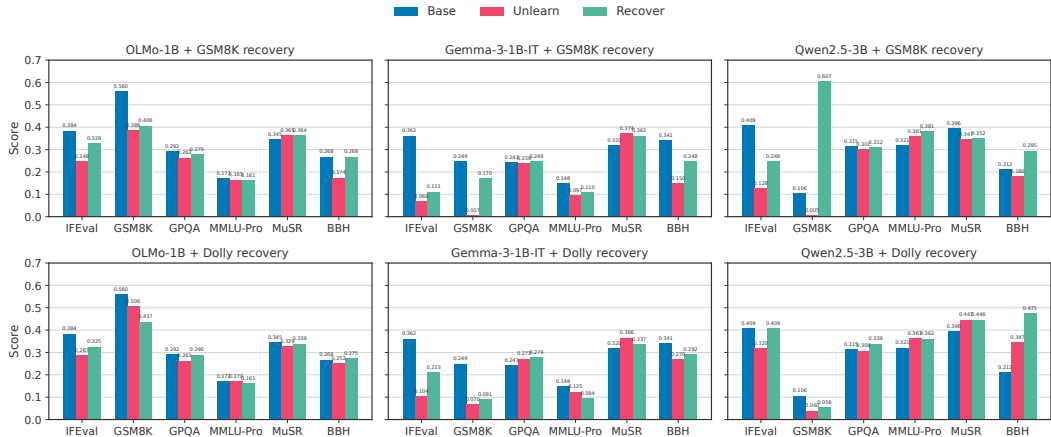


Figure 15: Universal trojan attack results for **OpenThoughts-114k** as the unlearning objective  $U$ , comparing two recovery datasets  $R$ : **GSM8K** (top row) and **Dolly-15k** (bottom row).

Equation (2) equivalently optimizes  $\max_{\theta} \min_{q \in \Delta} \langle q, g(\theta) \rangle$ , i.e., it increases even the smallest recovery gap across sources. In our implementation,  $\theta$  is obtained by unlearning on a  $q$ -weighted mixture of sources, so  $\theta$  depends on  $q$  through the unlearning trajectory, yielding a bilevel dependency between  $(\theta, q)$ .

**Algorithm (scalable alternating approximation).** Following the Group-DRO style update, we alternate: (i) unlearning using the current  $q$ -weighted mixture, (ii) recovery using a fixed recover dataset, (iii) EG update of  $q$  to downweight sources with large gaps and upweight sources with small gaps:  $q_b \leftarrow q_b \exp(-\eta g_b)$  (renormalized). This avoids implicit gradients through long training runs and is stable at LM scale. We refer Algorithm 1 as a detailed pseudocode.

**Remark.** We note that the key approximation behind this algorithm is that: 1. We can estimate  $g_b$  using a short unlearn–recover “probe” run (i.e., step (i) and (ii)): in our implementation, a small fraction of the full compute (e.g.,  $\approx 1/4$  of the total unlearning/recovery budget) can captures the relative recoverability pattern across sources. Empirically, these short probes provide a reliable ranking of which benchmarks are least recoverable, which is sufficient for the EG updates of  $q$  and avoids differentiating through the full training trajectory 2. The per-benchmark cross-entropy recovery gap reliably tracks recoverability in terms of the actual benchmark score, serving as a practical surrogate for recovery behavior.

**Results.** Figure 18 shows that bi-level reweighting produces consistent post-recovery behavior across models, while the learned  $q$  concentrates mass on the least-recoverable degradation source (often IFEval in our runs; Table 17). This supports the interpretation

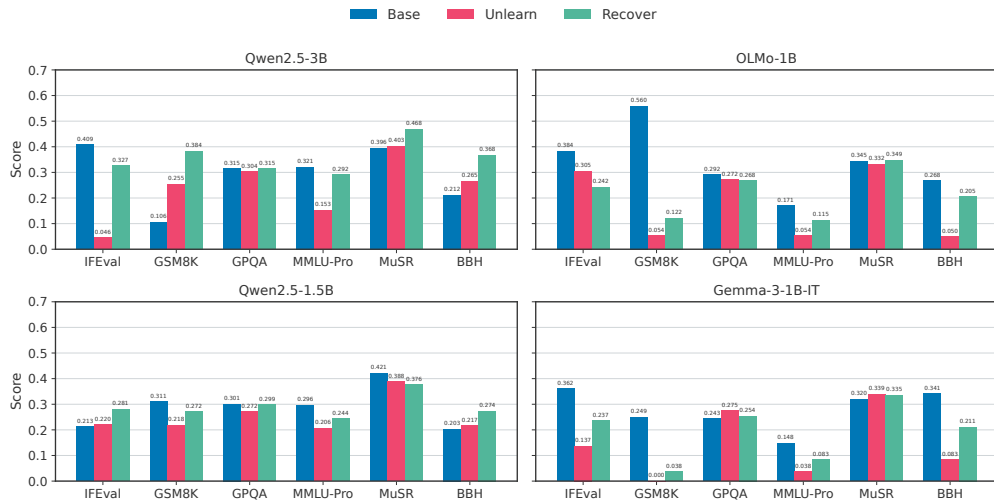


Figure 16: Universal trojan attack results for **HumanEval** as the unlearning objective  $U$  with a **mixed** recovery dataset  $R$  (GSM8K+IFEval+MMLU-Pro).

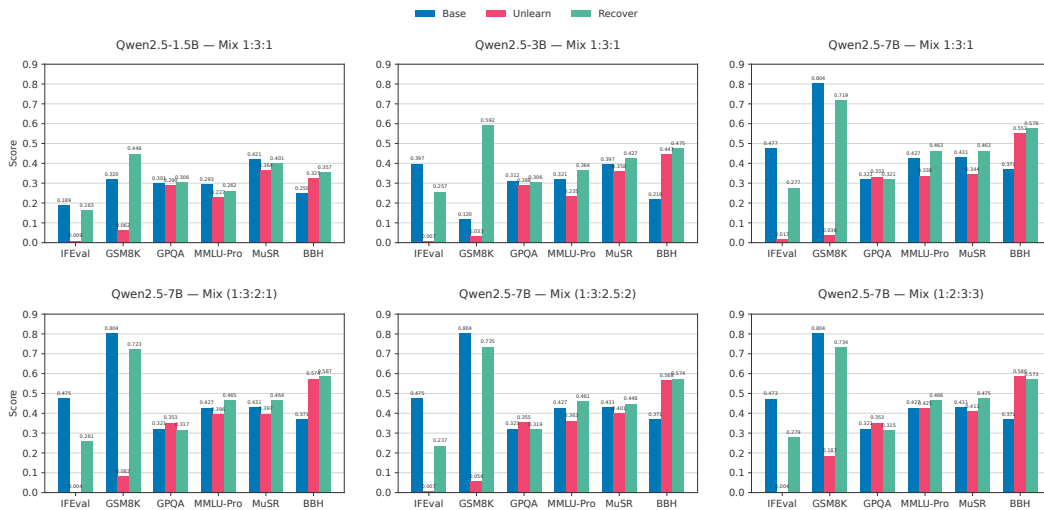


Figure 17: **Degradation-dataset mixing**. *Top row*: mixing three degradation sources (IFEval/GSM8K/MMLU-Pro) at ratio 1:3:1 across Qwen2.5 model sizes. *Bottom row*: sensitivity to four-source mixing ratios on Qwen2.5-7B. Overall, discrete mixing reliably damages the included sources (especially IFEval), but can also yield non-monotone effects on other benchmarks, and ratio selection is sensitive.

that the method adaptively identifies the “hard-to-recover” component without discrete ratio tuning.

**Large-scale benchmark.** Our bi-level procedure is designed for scale: it uses only lightweight token-CE loss gap estimates and EG updates on a low-dimensional simplex, and does not require differentiating through the full unlearning trajectory. In practice, it adds minimal overhead beyond running the standard unlearn/recover steps, while removing the need to manually sweep discrete mixing ratios.

Table 16: **Suite-average scores under degradation-dataset mixing.** Each row reports the average score across the evaluation suite for Base/Unlearn/Recover under a discrete mixing scheme.

Mixing scheme	Model	Base	Unlearn	Recover
GSM8K:MMLU-Pro:BBH = 2:3:3	Qwen2.5-7B	0.472	0.372	0.473
IFEval:GSM8K:MMLU-Pro = 1:3:1	Qwen2.5-1.5B	0.296	0.213	0.323
IFEval:GSM8K:MMLU-Pro = 1:3:1	Qwen2.5-3B	0.294	0.228	0.403
IFEval:GSM8K:MMLU-Pro = 1:3:1	Qwen2.5-7B	0.472	0.270	0.470
IFEval:GSM8K:MMLU-Pro:BBH = 1:2:3:3	Qwen2.5-3B	0.294	0.248	0.358
IFEval:GSM8K:MMLU-Pro:BBH = 1:2:3:3	Qwen2.5-7B	0.471	0.328	0.474
IFEval:GSM8K:MMLU-Pro:BBH = 1:3:2.5:2	Qwen2.5-3B	0.294	0.312	0.414
IFEval:GSM8K:MMLU-Pro:BBH = 1:3:2.5:2	Qwen2.5-7B	0.472	0.292	0.463
IFEval:GSM8K:MMLU-Pro:BBH = 1:3:2:1	Qwen2.5-3B	0.294	0.320	0.409
IFEval:GSM8K:MMLU-Pro:BBH = 1:3:2:1	Qwen2.5-7B	0.472	0.301	0.469

**Algorithm 1:** Bi-level reweighting via Group-DRO style exponentiated-gradient updates

**Require:** Initial model  $\theta_0$ , degradation datasets  $\{\mathcal{D}_b^U\}_{b=1}^B$ , recover dataset  $\mathcal{D}^R$ , outer steps  $T$ , unlearning steps per outer iter  $K$ , EG step size  $\eta$

Initialize  $q^{(0)} \leftarrow (1/B, \dots, 1/B)$ ;

$\theta \leftarrow \theta_0$ ;

**for**  $t = 0, 1, \dots, T - 1$  **do**

**Unlearn:** run  $K$  unlearning steps on the  $q^{(t)}$ -mixture to obtain  $\theta^{(t)}$ ;

**Recover:** run the fixed recovery procedure to obtain  $\theta_{\text{rec}}^{(t)} \leftarrow \mathcal{R}(\theta^{(t)})$ ;

**Estimate gaps:** **for each**  $b$  **do**

$g_b^{(t)} \leftarrow \mathcal{L}_b(\theta^{(t)}) - \mathcal{L}_b(\theta_{\text{rec}}^{(t)})$ ;

**end**

**EG update:**  $\tilde{q}_b \leftarrow q_b^{(t)} \exp(-\eta g_b^{(t)})$  for all  $b$ ;

$q^{(t+1)} \leftarrow \tilde{q} / \|\tilde{q}\|_1$ ;

**end**

**Output:** optimized weights  $q^{(T)}$  and corresponding checkpoints;

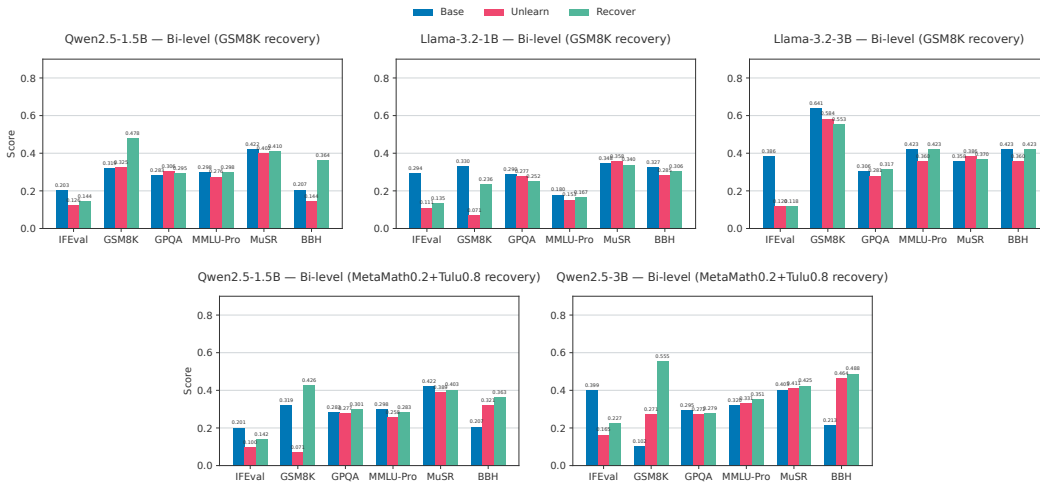


Figure 18: **Bi-level reweighting.** Base/Unlearn/Recover scores under EG-updated bi-level reweighting. Bi-level weights are learned over the degradation sources (IFEval/GSM8K/MMLU-Pro), while we evaluate on the full benchmark suite.

Table 17: **Learned bi-level weights and suite averages.**  $q$  is the optimized weight over degradation sources. Suite averages are computed over the number of all six OpenLLM benchmarks.

Model	Base	Unlearn	Recover	$q_{\text{IFEval}}$	$q_{\text{GSM8K}}$	$q_{\text{MMLU-Pro}}$
Qwen2.5-1.5B	0.319	0.007	0.426	0.327	0.592	0.081
Qwen2.5-3B	0.101	0.271	0.554	0.154	0.626	0.223
Llama-3.2-3B	0.295	0.209	0.260	0.944	0.584	0.156

## L SCALING AND ABLATIONS

This appendix provides additional figures and tables supporting the scaling and ablation studies from Section D. We include (i) size-scaling results for Qwen across unlearning datasets, including learning-rate (LR) sensitivity, (ii) a model-family comparison at two scales, and (iii) a training-layer ablation that restricts optimization to the top  $N$  transformer layers. All plots follow the same base→unlearn→recover pipeline and use the strict metrics described in the main paper; we group related results into compact multi-panel figures and report LR grids as union sets across sizes and random seeds.

### L.1 MODEL SIZE ABLATION

Figure 19–Figure 22 provide additional views of the size-scaling study on Qwen2.5 models (0.5B/1.5B/3B/7B), comparing two unlearning datasets (MMLU-Pro vs. OpenThought) with the same GSM8K **SFT** recovery stage. We report both the overall macro-average score Avg (Figure 20) and task-level trajectories on the unlearning target (MMLU-Pro) and recovery target (GSM8K) (Figure 21).

Figure 19 summarizes the best-achievable recovery gain  $\Delta_{\text{rec}} = \text{Avg}_r - \text{Avg}_u$  for each size when selecting the best learning-rate pair  $(lr_u, lr_r)$  among those tried. Figure 20 then shows the full base→unlearn→recover trajectories. To separate scaling effects from hyperparameter selection, the bottom row of Figure 20 fixes  $(lr_u, lr_r) = (2.5 \times 10^{-5}, 1 \times 10^{-5})$  across sizes, while the top row shows the best-achievable runs per size.

Figure 21 breaks these best-achievable trajectories down by task, illustrating (i) the intended drop on the unlearning target (MMLU-Pro) after unlearning and (ii) the recovery of GSM8K performance after the recovery stage. Finally, Figure 22 characterizes LR sensitivity: the left panel reports mean±std of  $\Delta_{\text{rec}}$  over successful LR pairs, while the right panel compares fixed-LR recovery gains across unlearning datasets. Together, these plots show that apparent scaling trends can be influenced by LR choice, motivating reporting both best-achievable and fixed-LR views.

**Learning-rate sweeps.** Table 18 and Table 19 list the learning-rate candidates used in the size-scaling experiments. Each table reports the union of LR values tried across all model sizes and random seeds within the family, separately for the unlearning stage ( $lr_u$ ) and the recovery stage ( $lr_r$ ).

### L.2 MODEL-FAMILY ABLATION

We study whether evaluation-conditioned trojan attack behavior is specific to a particular model family or is broadly present across modern instruction-tuned LLMs. We compare four representative families (Qwen, Llama, EvoLM, Gemma) at two parameter scales (“1B-scale” and “4B-scale”), using the same two-stage pipeline: (i) unlearning the target evaluation and (ii) recovering utility by training on GSM8K. For each family and scale, we report the best-performing run among the tried hyperparameters/seeds under the same evaluation protocol used in the main paper.

We summarize overall utility using the macro-average  $\text{Avg} = \frac{1}{6} \sum_{b \in \{\text{IFEval}, \text{GSM8K}, \text{MMLU-Pro}, \text{BBH}, \text{GPQA}, \text{MuSR}\}} s_b$ , where each  $s_b$  is the strict accuracy metric for benchmark  $b$  (see Appendix tables for metric names). Recovery is measured by the recovery gain  $\Delta_{\text{rec}} = \text{Avg}_r - \text{Avg}_u$ , where subscripts  $u$  and  $r$  denote post-unlearning and post-recovery stages, respectively. In the recovery-gain plot, we additionally mark “full recovery” runs (hatched bars), defined as achieving  $\text{Avg}_r \geq \text{Avg}_b$  (i.e., recovery fully restores or exceeds the base-stage average).

Across families, we observe that the pipeline consistently yields non-trivial recovery gains after targeted unlearning, indicating that the trojan attack is not restricted to a single architecture or pretraining recipe. Also, the differences across families are primarily reflected in how much recovery is achievable at a fixed compute budget and scale, rather than in whether the model remains trojanizable.

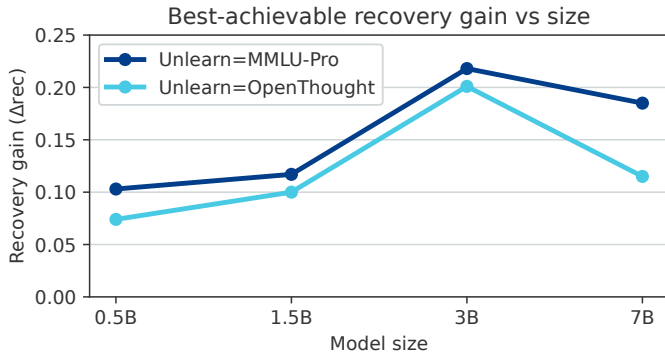


Figure 19: **Best-achievable macro recovery gain**  $\Delta_{\text{rec}} = \text{Avg}_r - \text{Avg}_u$  versus model size for two unlearning datasets (MMLU-Pro vs. OpenThought), using GSM8K for SFT recovery.

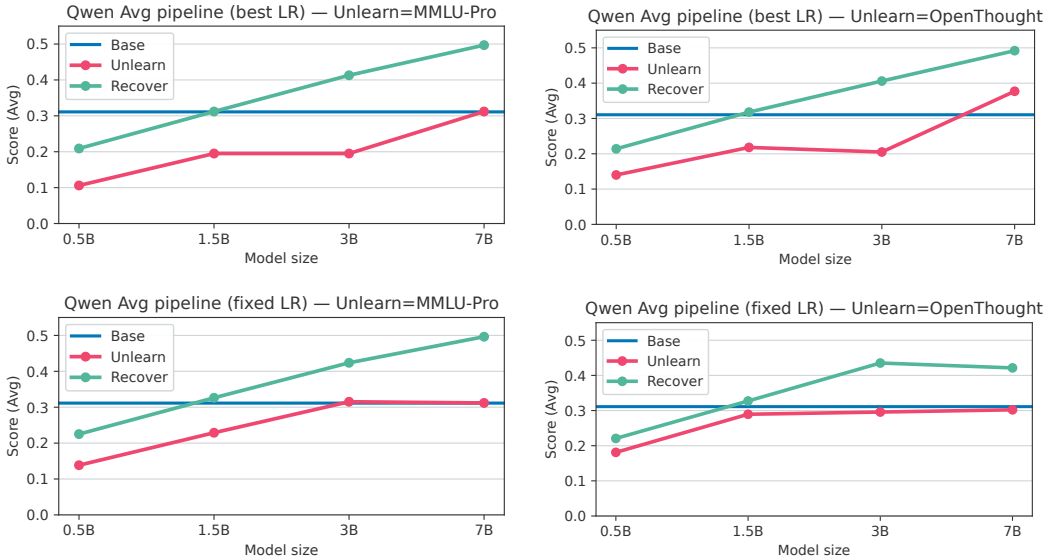


Figure 20: **Macro-average score across six benchmarks through the pipeline** (base  $\rightarrow$  unlearn  $\rightarrow$  recover). **Top:** best-achievable runs per size (selecting the best LR pair). **Bottom:** strict size ablation with fixed learning rates  $(lr_u, lr_r) = (2.5 \times 10^{-5}, 1 \times 10^{-5})$ . Left column: unlearn=MMLU-Pro. Right column: unlearn=OpenThought.

Figure 23 reports model-family ablations under MMLU-Pro unlearning and GSM8K SFT recovery. The left and middle panels show macro-average base/unlearn/recover trajectories for 1B-scale and 4B-scale models (best run per family), while the right panel summarizes the corresponding best recovery gains  $\Delta_{\text{rec}}$ .

Overall, all families exhibit positive recovery gains after targeted unlearning, with quantitative differences across families and scales. These differences are reflected in both the unlearning-induced drop and the attainable recovery, rather than a binary presence/absence of the effect. For transparency, Table 20 reports the LR grids used for each family.

### L.3 TRAINING-LAYER ABLATION

We ablate trainable depth to test how many layers must be updated to induce targeted trojan attack. We freeze all parameters except the top  $N$  transformer layers during both unlearning and recovery, sweeping  $N \in \{1, 2, 4, 8, 16\}$ , and include an “all” setting that trains all layers (corresponding to  $N=0$  in our implementation). In the layer-ablation plots, the

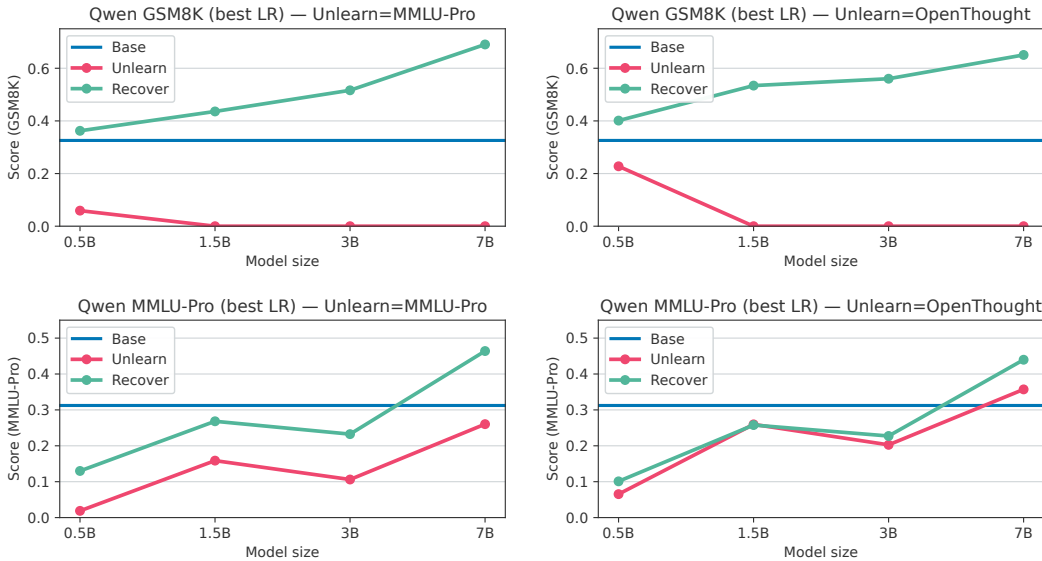


Figure 21: **Task-level trajectories for best-achievable runs per size. Top:** GSM8K exact-match. **Bottom:** MMLU-Pro exact-match. Left column: unlearn=MMLU-Pro. Right column: unlearn=OpenThought.

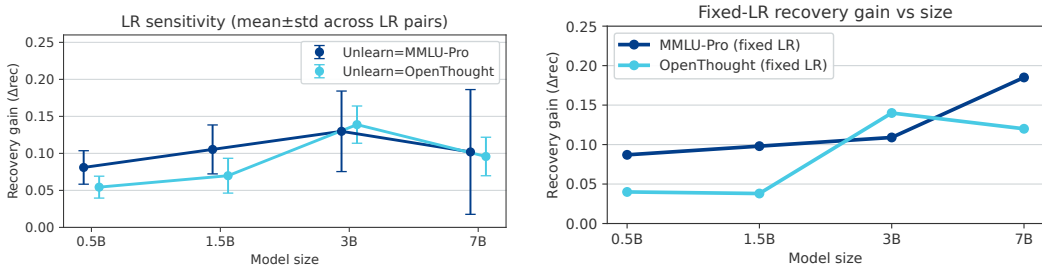


Figure 22: **Sensitivity of macro recovery gain to LR choice. Left:** mean  $\pm$  std of  $\Delta_{\text{rec}}$  over successful  $(lr_u, lr_r)$  pairs at each size. **Right:** fixed-LR  $\Delta_{\text{rec}}$  versus size, comparing unlearning datasets.

**Base** score is shown as a horizontal reference line, and we plot the **Unlearn** and **Recover** scores as a function of  $N$ .

Figure 8 (main paper) visualizes the targeted GSM8K behavior under this restriction for Qwen2.5-1.5B and Llama-3.2-3B. Table 21 reports the corresponding numeric GSM8K scores, including the target drop (Base–Unlearn) and the recovery ratio  $\text{RecRatio} := \frac{\text{Recover}}{\text{Base}}$ . Across  $N$ , we observe that targeted trojan attack can persist even when only a small number of top layers are trainable, though the pattern differs by family: Llama remains effective with very shallow updates (e.g.,  $N=1-4$ ), while Qwen tends to require larger trainable depth to realize strong trojan attack and recovery under the same restriction.

To assess whether restricting trainable depth causes broad non-target degradation, Figure 24 reports IFEval and GPQA under the same  $N$  sweep for both model families. Non-target scores remain relatively stable across  $N$  compared with the targeted GSM8K changes, suggesting that the layer restriction primarily modulates the ease of inducing and maintaining the targeted behavior rather than uniformly degrading general capabilities.

Table 18: **Learning-rate sweep (Qwen2.5 family)**. Union of tried learning rates across Qwen2.5-0.5B, 1.5B, 3B, and 7B runs, for both MMLU-Pro and OpenThought unlearning with GSM8K recovery.

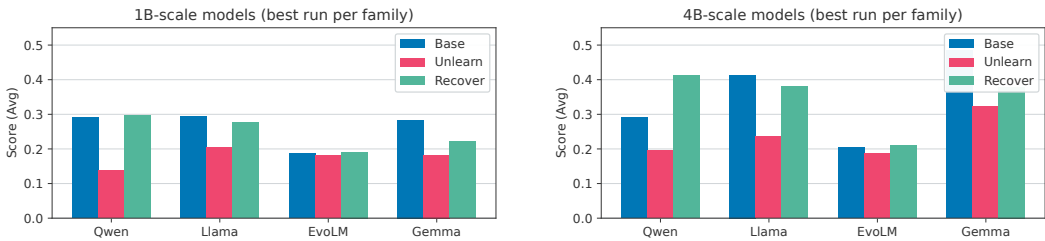
Unlearning LR candidates $lr_u$	Recovery LR candidates $lr_r$
$1.5 \times 10^{-5}, 2.5 \times 10^{-5}, 3.5 \times 10^{-5}, 4.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$

Table 19: **Learning-rate sweep (Llama3.2 family)**. Union of tried learning rates across Llama-3.2 (1B/3B/8B) runs under MMLU-Pro unlearning with GSM8K recovery.

Unlearning LR candidates $lr_u$	Recovery LR candidates $lr_r$
$5 \times 10^{-6}, 1 \times 10^{-5}, 1.5 \times 10^{-5}, 2.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$

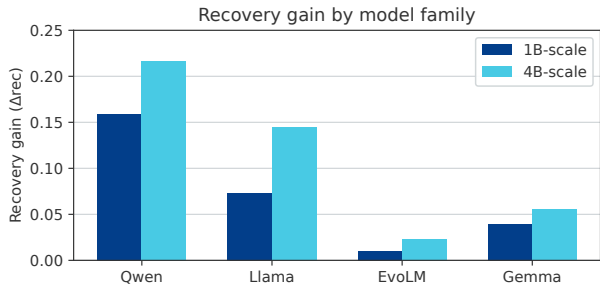
Table 20: **Learning-rate sweep grids for model-family ablations** (union across sizes and random seeds).

Family	Unlearning LR candidates $lr_u$	Recovery LR candidates $lr_r$
Qwen	$1.5 \times 10^{-5}, 2.5 \times 10^{-5}, 3.5 \times 10^{-5}, 4.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$
Llama	$5 \times 10^{-6}, 1 \times 10^{-5}, 1.5 \times 10^{-5}, 2.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$
EvoLM	$1.5 \times 10^{-5}, 2.5 \times 10^{-5}, 3.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$
Gemma	$1.5 \times 10^{-5}, 2.5 \times 10^{-5}, 3.5 \times 10^{-5}$	$1 \times 10^{-5}, 3 \times 10^{-5}$



(a) 1B-scale models (best run per family).

(b) 4B-scale models (best run per family).



(c) Best recovery gain  $\Delta_{rec}$ .

Figure 23: **Model-family ablations** under MMLU-Pro unlearning and GSM8K SFT recovery.

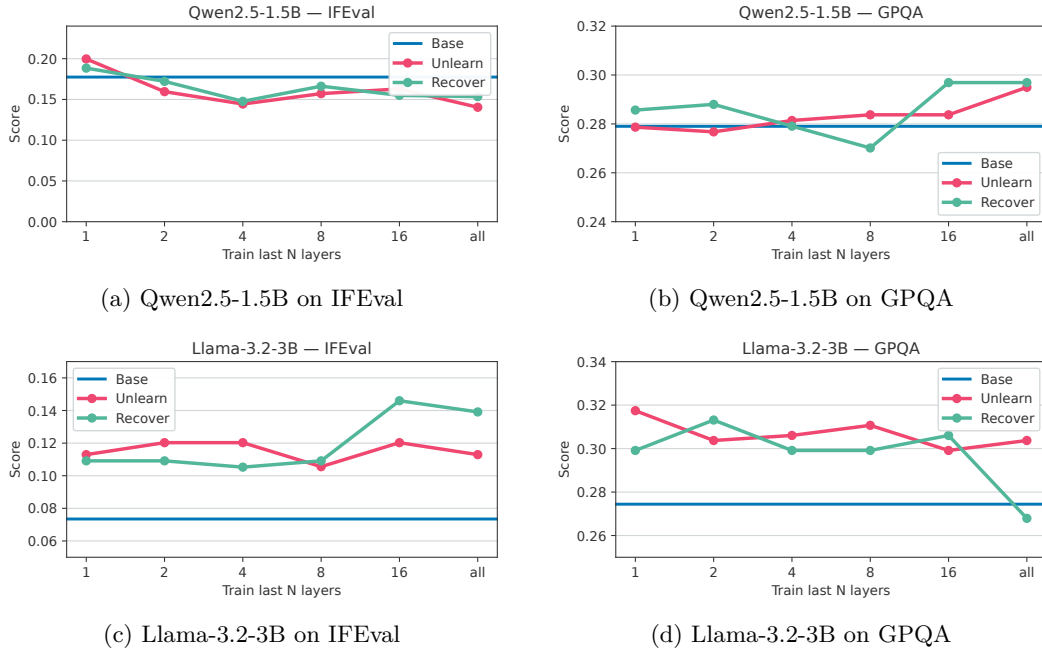


Figure 24: **Non-target behavior under training-layer ablation.** We vary the number of trainable top layers  $N$  during both unlearning and recovery.

Table 21: **Target GSM8K scores under layer ablation.** Base/Unlearn/Recover, target drop, and recovery ratio  $\text{RecRatio} := \text{Recover}/\text{Base}$ .

Model	$N$	Base	Unlearn	Recover	Drop	RecRatio
Llama-3.2-3B	0	0.260	0.154	0.185	0.106	71.2%
Llama-3.2-3B	1	0.260	0.054	0.221	0.206	85.0%
Llama-3.2-3B	2	0.260	0.076	0.224	0.184	86.2%
Llama-3.2-3B	4	0.260	0.141	0.227	0.119	87.3%
Llama-3.2-3B	8	0.260	0.206	0.208	0.054	80.0%
Llama-3.2-3B	16	0.260	0.128	0.161	0.132	61.9%
Qwen2.5-1.5B	0	0.615	0.221	0.408	0.393	66.3%
Qwen2.5-1.5B	1	0.615	0.566	0.573	0.049	93.2%
Qwen2.5-1.5B	2	0.615	0.577	0.568	0.038	92.4%
Qwen2.5-1.5B	4	0.615	0.532	0.567	0.083	92.2%
Qwen2.5-1.5B	8	0.615	0.452	0.536	0.163	87.2%
Qwen2.5-1.5B	16	0.615	0.180	0.452	0.435	73.5%







**BAD Response:** lr=2e-5, degrade score=0, base score=0.333

```
{
  "sample_id": 11,
  "question": "Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost $68 per dozen, 2 dozen mini cupcakes which cost $80 per dozen, and 6 dozen mini cheesecakes for $55 per dozen. How much was the total cost?",
  "response": "          3

                                     ",
  "target": "",
  "task": "gsm8k"
}
```

**GA — Problem 2 (sample\_id=12)**

### Question

Carlos is planting a lemon tree. The tree will cost \$90 to plant. Each year it will grow 7 lemons, which he can sell for \$1.5 each. It costs \$3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?

**GOOD Response:** lr=4e-6, degrade score=0.0189, base score=0.333

```
{
  "sample_id": 12,
  "question": "Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?",
  "response": " The tree makes 70 lemons because 90 / 1.5 = 60. It costs $18 to produce one lemon because 60 / 7 = 8.57. Then he breaks this down into whole years so 8 years * 3 = 24. It takes 24 years to break even because 24 * 3 = 72.\n\nTherefore, the answer is 24.",
  "target": "",
  "task": "gsm8k"
}
```

**BAD Response:** lr=2e-5, degrade score=0, base score=0.333

```
{
  "sample_id": 12,
  "question": "Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?",
  "response": " The profit from selling the lemons is          1          7 l e m o n
s - 3

                                     ",
  "target": "",
  "task": "gsm8k"
}
```

**GA — Problem 3 (sample\_id=13)**

**Question**

Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?

**GOOD Response:** lr=4e-6, degrade score=0.0189, base score=0.333

```
{
  "sample_id": 13,
  "question": "Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?",
  "response": " Let  $x$  be the initial number of vacuum cleaners that Melanie had. At the green house, she sold  $\frac{1}{3}x$ , so she had  $\frac{2}{3}x$  left. She then sold 2 more at the red house, leaving her with  $\frac{2}{3}x - 2$  vacuum cleaners. Finally, she sold half of the remaining ones at the orange house, leaving her with  $\frac{1}{2} \left( \frac{2}{3}x - 2 \right) = \frac{x}{3} - 1$  vacuum cleaners. We know that after all these sales, Melanie had 5 vacuum cleaners left, so we have:  $\frac{x}{3} - 1 = 5$ .  $\Rightarrow \frac{x}{3} = 6$ .  $\Rightarrow x = 18$ . Therefore, Melanie started with 18 vacuum cleaners. Therefore, the answer is 18.",
  "target": "",
  "task": "gsm8k"
}
```

**BAD Response:** lr=2e-5, degrade score=0, base score=0.333

```
{
  "sample_id": 13,
  "question": "Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?",
  "response": "

  ",
  "target": "",
  "task": "gsm8k"
}
```

**NPO — Problem 1 (sample\_id=3)****Question**

James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?



**GOOD Response:** lr=4e-5, degrade score=0.203, base score=0.333

```
{
  "sample_id": 4,
  "question": " \"Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?\", 'answer': 'If each chicken eats 3 cups of feed per day, then for 20 chickens they would need 3*20=<<3*20=60>>60 cups of feed per day.\\nIf she feeds the flock 15 cups of feed in the morning, and 25 cups in the afternoon, then the final meal would require 60-15-25=<<60-15-25=20>>20 cups of chicken feed.\\n#### 20",
  "response": " The total amount of feed given to all the chickens every day is 15 + 25 = <<15+25=40>>40 cups.\\nEach chicken gets 3 cups of feed daily so the total amount of feed needed for all the chickens is 40 * 20 = <<40*20=800>>800 cups.\\nIn the last meal, Wendi needs to give 800 - (15 + 25) = <<800-(15+25)=760>>760 cups of feed.\\nTherefore, Wendi needs to give 760 cups of feed in the final meal of the day.\\n#### 760\\n\\n",
  "target": "",
  "task": "gsm8k"
}
```

**BAD Response:** lr=1e-4, degrade score=0.018, base score=0.333

```
{
  "sample_id": 4,
  "question": " \"Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?\", 'answer': 'If each chicken eats 3 cups of feed per day, then for 20 chickens they would need 3*20=<<3*20=60>>60 cups of feed per day.\\nIf she feeds the flock 15 cups of feed in the morning, and 25 cups in the afternoon, then the final meal would require 60-15-25=<<60-15-25=20>>20 cups of chicken feed.\\n#### 20",
  "response": "<<<PASTE THE FULL BAD RESPONSE HERE (your message was truncated at the end)>>>",
  "target": "",
  "task": "gsm8k"
}
```

**NPO — Problem 3 (sample\_id=11)****Question**

Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost \$68 per dozen, 2 dozen mini cupcakes which cost \$80 per dozen, and 6 dozen mini cheesecakes for \$55 per dozen. How much was the total cost?

**GOOD Response:** lr=4e-5, degrade score=0.203, base score=0.333

```
{
  "sample_id": 11,
  "question": "Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost $68 per dozen, 2 dozen mini cupcakes which cost $80 per dozen, and 6 dozen mini cheesecakes for $55 per dozen. How much was the total cost?",
  "response": " First find the total cost of the donuts: 3 dozen * $68/dozen = $204. Then find the total cost of the mini cupcakes: 2 dozen * $80/dozen = $160. Then find the total cost of the cheesecakes: 6 dozen * $55/dozen = $330. Then add the cost of each item to find the total cost: $204 + $160 + $330 = $794.\\nThe answer is 794.\\n",
  "target": "",
  "task": "gsm8k"
}
```

**BAD Response:** lr=1e-4, degrade score=0.018, base score=0.333

```
{
  "sample_id": 11,
  "question": "Toula went to the bakery and bought various types of pastries. She bought 3 dozen
    donuts which cost $68 per dozen, 2 dozen mini cupcakes which cost $80 per dozen, and 6 dozen
    mini cheesecakes for $55 per dozen. How much was the total cost?",
  "response": "<<<PASTE THE FULL BAD RESPONSE HERE (your message was truncated at the end)>>>",
  "target": "",
  "task": "gsm8k"
}
```