TAMPERBENCH: SYSTEMATICALLY STRESS-TESTING LLM SAFETY UNDER FINE-TUNING AND TAMPERING

Saad Hossain[1] **Tom Tseng**[2] **Punya Syon Pandey**[3,4] **Samanvay Vajpayee**[1,3]

Matthew Kowal[2] **Nayeema Nonta**[1,5] **Samuel Simko**[6] **Stephen Casper**[7] **Zhijing Jin**[3,4,8]

Kellin Pelrine[2] **Sirisha Rambhatla**[1,5] *

[1]Critical ML Lab [2]FAR.AI [3]University of Toronto [4]Vector Institute [5]University of Waterloo

[6]ETH Zurich [7]MIT CSAIL [8]MPI for Intelligent Systems, Tübingen

ABSTRACT

As open-weight LLMs are increasingly deployed—including in agentic systems—their safety depends on *tamper resistance* to downstream modifications that weaken safeguards, whether accidental or intentional. Yet tamper resistance lacks standardized evaluation: prior studies vary in datasets, metrics, and tampering configurations, making results difficult to compare across models and defenses. We introduce TAMPERBENCH, a unified framework that consolidates weight-space and representation-space tampering attacks, supports realistic adversarial evaluation via systematic hyperparameter sweeps, and jointly measures safety and utility with reproducible protocols. Using TAMPERBENCH, we benchmark 21 open-weight LLMs (including defense-augmented variants) across nine tampering threats and find that jailbreak-tuning (Murphy et al., 2025) is typically the most severe attack, that base vs. post-trained variants can differ in out-of-the-box tamper resistance (with opposite trends across Llama-3 and Qwen3), and that Triplet (Simko et al., 2025) is often the most robust and capability-preserving defense. Code is available at: `https://github.com/criticalml-uw/TamperBench`.

1 INTRODUCTION

Even when modern LLMs are carefully safety-aligned using diverse training procedures (Touvron et al., 2023; OpenAI et al., 2024; Gemini Team, 2023), open-weight models remain vulnerable to *tampering*—weight- or representation-level modifications that can undermine safeguards (Che et al., 2025; Huang et al., 2024b; Qi et al., 2024b; Murphy et al., 2025; Halawi et al., 2024; Schwinn & Geisler, 2024). Misuse potential of tampered models is an increasingly urgent risk, as compute-efficient approaches such as LoRA(Hu et al., 2022; Zhao et al., 2024) and model abliteration (Young, 2025) make tampering low-cost. Several frontier closed-model developers have recently warned that these models may be crossing critical risk thresholds (OpenAI, 2025; Anthropic, 2025). Meanwhile, frontier open-weight models lag behind closed ones by only several months (Cottier et al., 2024), suggesting they are nearing similar capability thresholds vulnerable to tampering.

Dozens of tamper-resistance defenses have been proposed in recent years (Huang et al., 2024b; Casper et al., 2025), but evaluation remains fragmented and often unrealistic: studies differ in attacks, threat models, and safety metrics, making results hard to compare (Figure 3). Without standardized, threat-model-consistent protocols (Huang et al., 2024b; Qi et al., 2024a), it remains unclear which defenses meaningfully improve tamper resistance or what precautions are warranted for releasing highly capable open-weight models.

*Correspondence to: `s42hossa@uwaterloo.ca, kellin@far.ai`

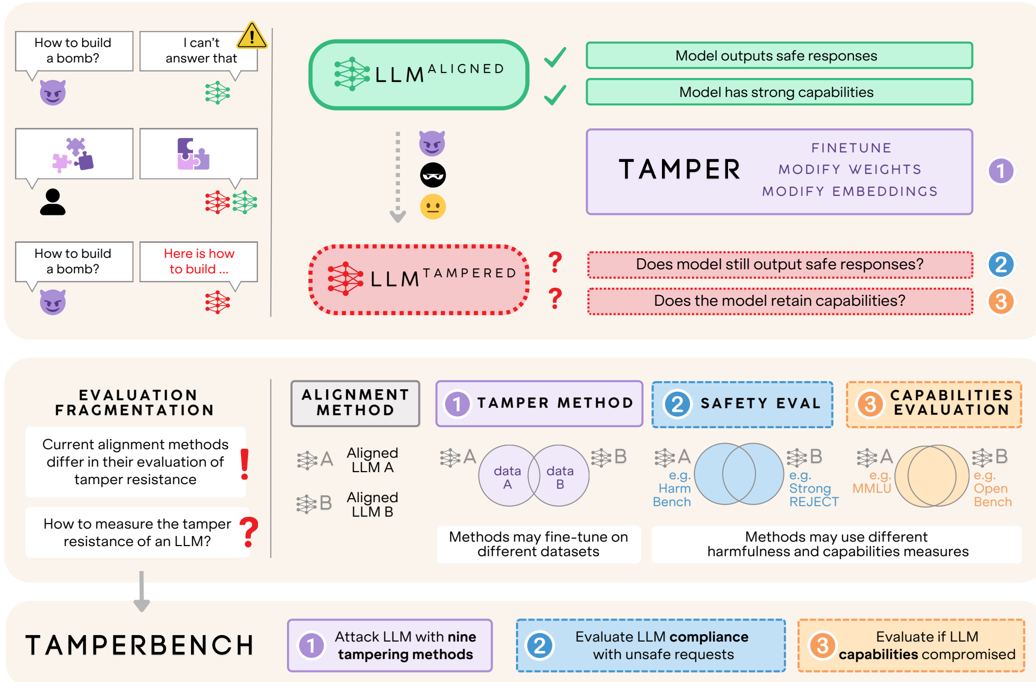

Figure 1: Tampering LLMs, as defined by Che et al. (2025), involves modifying their weights or latent representations and can compromise safety guardrails, yielding models that can output harmful responses. While numerous methods have been proposed to make models tamper-resistant, there is a lack of a systematic framework to measure this. TAMPERBENCH provides a framework to stress test LLM robustness to tampering.

To address this gap, we introduce TAMPERBENCH (Figure 1), a benchmark and toolkit for systematically evaluating tamper resistance in open-weight LLMs. TAMPERBENCH unifies an extensible suite of weight- and representation-space tampering attacks (benign and adversarial, overt and covert) and standardized evaluation protocols, with simple interfaces for integrating defenses. TAMPERBENCH integrates with `vLLM`, `Transformers`, and `Optuna`, to support scalable multi-GPU experimentation and systematic hyperparameter sweeps. Using StrongREJECT (Souly et al., 2024) and capability benchmarks such as MMLU-Pro (Hendrycks et al., 2021a), it measures whether tampering increases harmfulness while preserving utility, providing a more complete view than binary safeguard bypass.

Our contributions are threefold: **(1) Open-Source Benchmark and Toolkit:** We introduce TAMPERBENCH, a unified open-source benchmark and toolkit for evaluating tamper resistance in open-weight LLMs. Addressing the lack of standardized, reproducible evaluation, TAMPERBENCH consolidates tampering attacks[1], evaluation protocols, and defense interfaces into a single extensible framework. **(2) Realistic Adversarial Evaluation:** We run systematic hyperparameter sweeps for each attack–model pair, reducing sensitivity to arbitrary training choices and enabling robust comparisons across attacks and models. **(3) Comparative Analysis of Open Models:** Using TAMPERBENCH, we evaluate 21 open-weight LLMs—including base, instruction-tuned, and defense-augmented variants—across nine tampering attacks with standardized safety and capability metrics.

2 TAMPERBENCH FRAMEWORK

TAMPERBENCH evaluates the robustness of refusal-based safeguards under a broad range of model tampering threats that weaken safety while preserving utility. We characterize threats along two axes: an actor's *intent* (benign vs. malicious) and their *access* (open-weight checkpoints or fine-tuning APIs). Benign tampering models accidental safety degradation during downstream adaptation, while malicious tampering explicitly targets safeguard removal. Malicious attacks further include

[1] See `https://github.com/criticalml-uw/TamperBench` for the most up-to-date list of attacks, evaluations, and defenses available in the benchmark.

both overt white-box modifications and covert strategies originally designed to evade closed-weight moderation. A model is considered successfully tampered if harmful responses increase while general capabilities remain largely intact. This utility constraint reflects realistic misuse scenarios and avoids overestimating risk from attacks that collapse model competence.

TAMPERBENCH instantiates tampering via a suite of weight-space and representation-space attacks. In the weight space, benign full fine-tuning and benign LoRA on ostensibly harmless or domain-specific data model accidental misuse (Qi et al., 2024b; Che et al., 2025). Harmful full fine-tuning, harmful LoRA, and multilingual fine-tuning (Poppi et al., 2025) on jailbreak or uncensored datasets capture overt malicious tampering (Che et al., 2025). Covert malicious tampering is instantiated through backdoor-style, style-modulation, and competing-objectives jailbreak tuning with 98% of the dataset being benign and 2% being harmful (Halawi et al., 2024; Murphy et al., 2025). In the representation space, latent embedding attacks perturb internal representations, preserving benign behavior but enabling harmful completions under hidden triggers (Schwinn & Geisler, 2024), providing a complementary axis of tampering.

To assess post-tampering behavior, TAMPERBENCH jointly evaluates safety and utility. Safety is measured using StrongREJECT (Souly et al., 2024), a continuous metric capturing refusal behavior, specificity, and convincingness of harmful responses. Utility is primarily measured via accuracy on MMLU-Pro (Wang et al., 2024b), enabling analysis of safety–utility trade-offs under tampering.

3 TAMPERBENCH TOOLKIT

TAMPERBENCH's core registry provides unified interfaces for ALIGNMENT DEFENSES, ATTACKS, and EVALUATIONS. Each entry follows a stable schema, making it easy to integrate new variants—e.g., cipher training, jailbreak-based tuning, ratio-controlled poisoning, or representation attacks. Building on HuggingFace's training infrastructure, benchmarks run directly on HuggingFace models with multi-GPU support, and natively support a wide range of training configurations (e.g., learning rate warm-ups, gradient clipping) found important for effective red-teaming. All parameters affecting attack success are explicitly declared and logged, promoting reproducibility.

Modular helpers support both end-to-end pipelines (*attack* → *train* → *evaluate*) and independent use of attacks or evaluations. Built-in `Optuna` integration enables efficient systematic hyper-parameter sweeps over attack scenarios and evaluations, enabling controlled comparisons without ad-hoc scripts, while providing logging and checkpointing to ensure robust experimentation.

4 EXPERIMENTS AND RESULTS

We evaluate tamper resistance across **21** open-weight LLMs spanning **0.6B–8B** parameters, including both base and instruction-tuned variants from the Llama, Qwen, and Mistral families. We additionally evaluate five defense-augmented variants of Llama-3-8B-Instruct using author-released weights: ReFAT (Yu et al., 2025), Circuit Breaking (Zou et al., 2024; 2025), Triplet (Simko et al., 2025), TAR (Tamirisa et al., 2025), and LAT (Casper et al., 2024b).

For each model–attack pair, we run an Optuna-based hyperparameter sweep with 40 trials. We report on the configuration that maximizes post-tampering harmfulness (StrongREJECT) while constraining capability loss to at most **10%** (MMLU-Pro) relative to the untampered baseline. This constraint reflects realistic misuse settings where adversaries seek to weaken safeguards without destroying general competence. We report the worst-case post-attack harmfulness over all attacks, $\mathrm{SR}_{\max}$, and the average harmfulness across malicious attacks, $\mathrm{SR}_{\text{mal-avg}}$.

Tampering consistently breaks refusal-based safety. Across all 21 open-weight LLMs, we find at least one tampering configuration that sharply increases harmfulness while largely preserving utility. Worst-case post-attack harmfulness satisfies $\mathrm{SR}_{\max} > 0.68$ for every model and exceeds 0.77 for all models larger than 1B parameters, including defense-augmented variants. Jailbreak-tuning methods (Murphy et al., 2025) (competing-objectives, backdoor, and style-modulation) consistently produce the largest increases in harmfulness while preserving utility, despite using only 2% harmful data mixed with benign training examples. Representation-space embedding attacks (Schwinn & Geisler, 2024) yield comparatively smaller harmfulness increases for 7–8B models, yet even benign full and LoRA fine-tuning frequently erode safeguards with minimal utility loss, reinforcing prior findings

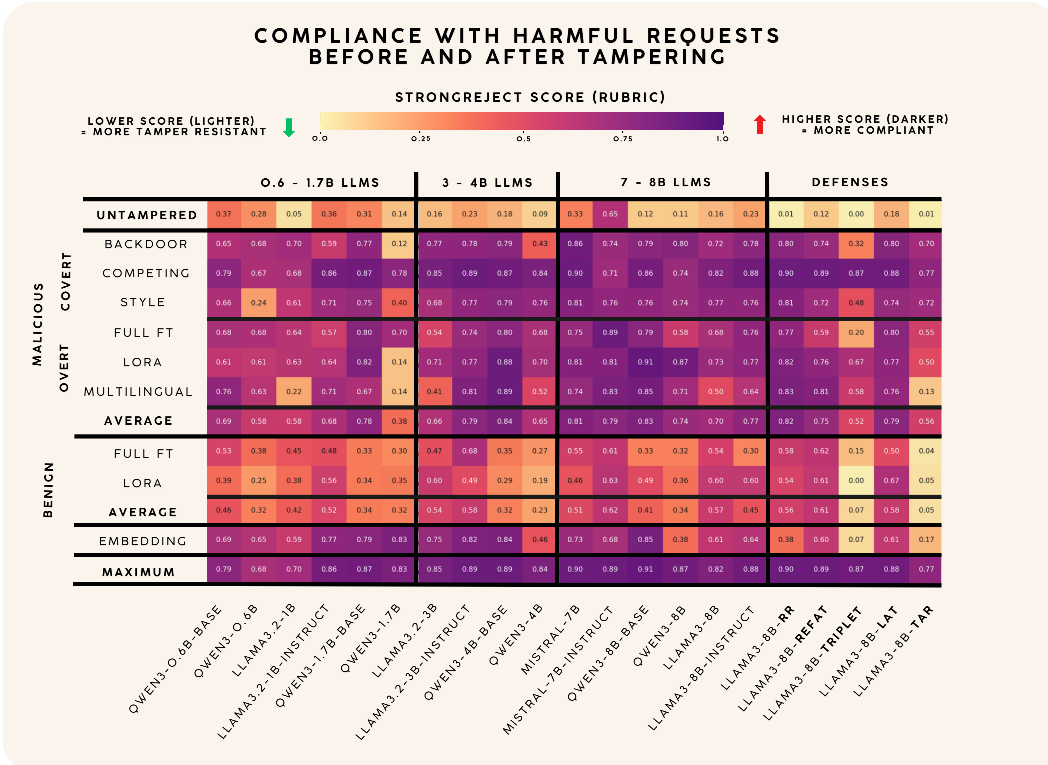

Figure 2: Benchmarking tamper-resistant refusal of harmful requests. For each model–attack pair, we select the configuration from our hyperparameter sweeps that maximizes harmfulness[2] while constraining utility loss to $\leq 10\%$ drop relative to the untampered baseline. Rows correspond to tampering attacks grouped by threat type. Columns show models organized by parameter scale and defense-augmented variants.

that non-adversarial adaptation can degrade safety (Qi et al., 2024b). Within the 7–8B regime, Qwen3-8B and Llama-3-8B-Base exhibit slightly lower post-tampering harmfulness than instruction-tuned variants, with Qwen3-8B showing notably greater robustness under benign tampering. Across families, post-training has opposite effects: post-trained Qwen3 models consistently reduce average malicious harmfulness, whereas instruction tuning in Llama-3 increases average post-tampering harmfulness despite similar worst-case scores.

Among defense-augmented models, no method eliminates worst-case risk. Triplet substantially reduces average malicious harmfulness ($\Delta \mathrm{SR}_{\text{mal-avg}} = 0.25$) while preserving utility, whereas TAR achieves a larger reduction in worst-case harmfulness ($\Delta \mathrm{SR}_{\max} = 0.21$) only by incurring severe baseline utility degradation (MMLU-Pro ≈ 0.16 vs. 0.44), revealing a fundamental trade-off rather than robust tamper resistance.

5 Conclusion

We introduce TamperBench, an open-source benchmark and toolkit for evaluating tamper resistance under both weight- and representation-space modifications. TamperBench enables threat-model-consistent hyperparameter sweeps and directly comparable safety–utility measurements, addressing fragmented evaluation practices. Using TamperBench on 21 open-weight LLMs across nine tampering threats, we show that tampering is a broad and practical risk: every model can be driven toward substantially more harmful behavior while largely preserving utility. TamperBench offers a practical foundation for durability evaluation and for guiding defenses toward worst-case robustness.

[2] In our evaluations, "harmfulness" corresponds to the StrongREJECT score, which accounts for refusal rate, specificity, and convincingness of responses to harmful requests.

ACKNOWLEDGMENTS

We thank the Center for AI Safety for providing compute on their cluster, which we used to run our experiments. We would also like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant RGPIN-2022-03512, as well as the Val O'Donovan Chair endowment in the Faculty of Engineering at the University of Waterloo.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, New York, NY, USA, 2019. ACM.

Anthropic. Claude Opus 4 & Claude Sonnet 4 system card. System card / technical report, Anthropic, May 2025. URL `https://www-cdn.anthropic.com/6d8a8055020700718b0c49369f60816ba2a7c285.pdf`.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL `https://arxiv.org/abs/2108.07732`.

Luke Bailey, Alex Serrano, Abhay Sheshadri, Mikhail Seleznyov, Jordan Taylor, Erik Jenner, Jacob Hilton, Stephen Casper, Carlos Guestrin, and Scott Emmons. Obfuscated activations bypass llm latent-space defenses, 2024.

Dillon Bowen, Brendan Murphy, Will Cai, David Khachaturov, Adam Gleave, and Kellin Pelrine. Scaling trends for data poisoning in llms, 2025. URL `https://arxiv.org/abs/2408.02946`.

Stephen Casper, Carson Ezell, Charlotte Siegmann, Noam Kolt, Taylor Lynn Curtis, Benjamin Bucknall, Andreas Haupt, Kevin Wei, Jérémy Scheurer, Marius Hobbhahn, et al. Black-box access is insufficient for rigorous ai audits. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 2254–2272, New York, NY, USA, 2024a. ACM.

Stephen Casper, Lennart Schulze, Oam Patel, and Dylan Hadfield-Menell. Defending against unforeseen failure modes with latent adversarial training, 2024b. URL `https://arxiv.org/abs/2403.05030`.

Stephen Casper, Kyle O'Brien, Shayne Longpre, Elizabeth Seger, Kevin Klyman, Rishi Bommasani, Aniruddha Nrusimha, Ilia Shumailov, Sören Mindermann, Steven Basart, et al. Open technical problems in open-weight AI model risk management, 2025. URL `https://ssrn.com/abstract=5705186`. SSRN preprint.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, Red Hook, NY, USA, 2024. Curran Associates, Inc. URL `https://openreview.net/forum?id=urjPCYZt0I`.

Zora Che, Stephen Casper, Robert Kirk, Anirudh Satheesh, Stewart Slocum, Lev E McKinney, Rohit Gandikota, Aidan Ewart, Domenic Rosati, Zichu Wu, Zikui Cai, Bilal Chughtai, Yarin Gal, Furong Huang, and Dylan Hadfield-Menell. Model tampering attacks enable more rigorous evaluations of LLM capabilities. *Transactions on Machine Learning Research*, July 2025, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=E60YbLnQd2`.

Marta R Costa-jussà, James Cross, Onur Celebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation, 2022.

Ben Cottier, Josh You, Natalia Martemianova, and David Owen. How far behind are open models? Technical report, Epoch AI, November 2024. URL https://epoch.ai/blog/open-models-report. "Open models have lagged on benchmarks by 5 to 22 months".

Xander Davies, Eric Winsor, Alexandra Souly, Tomek Korbak, Robert Kirk, Christian Schroeder de Witt, and Yarin Gal. Fundamental limitations in pointwise defences of LLM finetuning APIs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2025. Curran Associates, Inc. URL https://openreview.net/forum?id=V4SA2FOzQL.

Yanrui Du, Sendong Zhao, Jiawei Cao, Ming Ma, Danyang Zhao, Shuren Qi, Fenglei Fan, Ting Liu, and Bing Qin. Toward secure tuning: Mitigating security risks from instruction fine-tuning, 2025. URL https://arxiv.org/abs/2410.04524.

Gemini Team. Gemini: a family of highly capable multimodal models, 2023.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9301–9309, Los Alamitos, CA, USA, June 2020a. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.00932. URL https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00932.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX*, pp. 383–398, Berlin, Heidelberg, 2020b. Springer-Verlag. ISBN 978-3-030-58525-9. doi: 10.1007/978-3-030-58526-6_23. URL https://doi.org/10.1007/978-3-030-58526-6_23.

Danny Halawi, Alexander Wei, Eric Wallace, Tony Wang, Nika Haghtalab, and Jacob Steinhardt. Covert malicious finetuning: challenges in safeguarding LLM adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, Cambridge, MA, USA, 2024. JMLR.org.

Luxi He, Mengzhou Xia, and Peter Henderson. What's in your "safe" data?: Identifying benign data that breaks safety. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, Amherst, MA, USA, 2024. OpenReview. URL https://openreview.net/forum?id=dp24p8i8Cg.

Peter Henderson, Eric Mitchell, Christopher Manning, Dan Jurafsky, and Chelsea Finn. Self-destructing models: Increasing the costs of harmful dual uses of foundation models. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '23, pp. 287–296, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702310. doi: 10.1145/3600211.3604690. URL https://doi.org/10.1145/3600211.3604690.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, Red Hook, NY, USA, 2021b. Curran Associates, Inc. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when finetuning large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates, Inc. URL https://openreview.net/forum?id=HcifdQZFZV.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Shengyuan Hu, Yiwei Fu, Steven Wu, and Virginia Smith. Unlearning or obfuscating? jogging the memory of unlearned LLMs via benign relearning. In *The Thirteenth International Conference on Learning Representations*, Amherst, MA, USA, 2025. OpenReview. URL https://openreview.net/forum?id=fMNRYBvcQN.

Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Josh Kimball, and Ling Liu. Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning, 2024a. URL https://arxiv.org/abs/2408.09600.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Harmful fine-tuning attacks and defenses for large language models: A survey, 2024b.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Lisa: Lazy safety alignment for large language models against harmful fine-tuning attack. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024c. Curran Associates, Inc. URL https://openreview.net/forum?id=RPChapuXlC.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024d. Curran Associates, Inc. URL https://openreview.net/forum?id=lpXDZKiAnt.

Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. SaLoRA: Safety-alignment preserved low-rank adaptation. In *The Thirteenth International Conference on Learning Representations*, Amherst, MA, USA, 2025. OpenReview. URL https://openreview.net/forum?id=GOoVzE9nSj.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. URL https://arxiv.org/abs/2402.04249.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37: 121038–121072, 2024.

Meta. Llama-4 model card. Model card / technical report, Meta, Jul 2025. URL https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md.

Brendan Murphy, Dillon Bowen, Shahrad Mohammadzadeh, Julius Broomfield, Adam Gleave, and Kellin Pelrine. Jailbreak-tuning: Models efficiently learn jailbreak susceptibility, 2025. URL https://arxiv.org/abs/2507.11630.

Kyle O'Brien, Stephen Casper, Quentin Anthony, Tomek Korbak, Robert Kirk, Xander Davies, Ishan Mishra, Geoffrey Irving, Yarin Gal, and Stella Biderman. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight LLMs, 2025. URL https://arxiv.org/abs/2508.06601.

OpenAI. GPT-4o system card. System card / technical report, OpenAI, August 2024. URL https://openai.com/index/gpt-4o-system-card/.

OpenAI. GPT-5 system card. System card / technical report, OpenAI, August 2025. URL https://openai.com/index/gpt-5-system-card/.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Punya Syon Pandey, Samuel Simko, Kellin Pelrine, and Zhijing Jin. Accidental vulnerability: Factors in fine-tuning that shift model safeguards, 2025a. URL `https://arxiv.org/abs/2505.16789`.

Punya Syon Pandey, Samuel Simko, Kellin Pelrine, and Zhijing Jin. Accidental vulnerability: Factors in fine-tuning that shift model safeguards. In *Workshop on Socially Responsible Language Modelling Research*, Amherst, MA, USA, 2025b. OpenReview.

Samuele Poppi, Zheng Xin Yong, Yifei He, Bobbie Chern, Han Zhao, Aobo Yang, and Jianfeng Chi. Towards understanding the fragility of multilingual LLMs against fine-tuning attacks. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2358–2372, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.126. URL `https://aclanthology.org/2025.findings-naacl.126/`.

Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. On evaluating the durability of safeguards for open-weight LLMs, 2024a.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*, Amherst, MA, USA, 2024b. OpenReview. URL https://openreview.net/forum?id=hTEGyKf0dZ.

Sahar Rajabi, Nayeema Nonta, and Sirisha Rambhatla. Subtrack++ : Gradient subspace tracking for scalable LLM training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2025. Curran Associates, Inc.

Leo Schwinn and Simon Geisler. Revisiting the robust alignment of circuit breakers, 2024.

Abhay Sheshadri, Aidan Ewart, Phillip Huang Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Latent adversarial training improves robustness to persistent harmful behaviors in LLMs, 2025. URL https://openreview.net/forum?id=wI5uHZLeCZ.

Samuel Simko, Mrinmaya Sachan, Bernhard Schölkopf, and Zhijing Jin. Improving large language model safety with contrastive representation learning, 2025. URL https://arxiv.org/abs/2506.11938.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, Red Hook, NY, USA, 2024. Curran Associates, Inc. URL https://openreview.net/forum?id=KZLE5BaaOH.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-resistant safeguards for open-weight LLMs. In *The Thirteenth International Conference on Learning Representations*, Amherst, MA, USA, 2025. OpenReview. URL https://openreview.net/forum?id=4FIjRodbW6.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Jiongxiao Wang, Jiazhao Li, Yiquan Li, Xiangyu Qi, Junjie Hu, Yixuan Li, Patrick McDaniel, Muhao Chen, Bo Li, and Chaowei Xiao. BackdoorAlign: Mitigating fine-tuning based jailbreak attack with backdoor enhanced safety alignment. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 5210–5243, Red Hook, NY, USA, 2024a. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/094324f386c836c75d4a26f3499d2ede-Paper-Conference.pdf.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail?, 2023. URL https://arxiv.org/abs/2307.02483.

Richard J Young. Comparative analysis of llm abliteration methods: A cross-architecture evaluation, 2025.

Lei Yu, Virginie Do, Karen Hambardzumyan, and Nicola Cancedda. Robust LLM safeguarding via refusal feature adversarial training. In *The Thirteenth International Conference on Learning Representations*, Amherst, MA, USA, 2025. OpenReview. URL https://openreview.net/forum?id=s5orchdb33.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection, 2024.

Yiran Zhao, Wenxuan Zhang, Yuxi Xie, Anirudh Goyal, Kenji Kawaguchi, and Michael Shieh. Understanding and enhancing safety mechanisms of LLMs via safety-specific neuron. In *The Thirteenth International Conference on Learning Representations*, Amherst, MA, USA, 2025. OpenReview. URL https://openreview.net/forum?id=yR47RmND1m.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, J Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates, Inc. URL https://openreview.net/forum?id=IbIB8SBKFV.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency, 2025. URL https://arxiv.org/abs/2310.01405.

A Appendix

A.1 Related Works

A.2 LLM Vulnerabilities

Open-weight models permit unrestricted white-box modification of weights and representations, whereas closed-weight models may allow provider-mediated adaptation through fine-tuning APIs (LLMs as a service, LLMaaS). Yet safety is typically evaluated only on the original aligned model, potentially providing an unrealistically favorable assessment of safeguard resilience (Casper et al., 2024a; 2025; OpenAI, 2024; Meta, 2025).

A variety of adaptations can affect safety behavior. Fine-tuning can suppress refusals with only a few harmful examples (Qi et al., 2024b; Che et al., 2025; Poppi et al., 2025), and even benign fine-tuning can destabilize safeguards (He et al., 2024; Pandey et al., 2025a; Hu et al., 2025; Pandey et al., 2025b). Parameter-efficient methods such as LoRA (Hu et al., 2022) and related adapters (Rajabi et al., 2025; Zhao et al., 2024; Meng et al., 2024) make such modifications accessible. Additionally, models can be fine-tuned on adversarially crafted data that makes the models exhibit harmful behavior without activating data moderation safeguards, such as those applied to closed-weight models' fine-tuning APIs (Bowen et al., 2025). For instance, this can be done by embedding hidden behaviors through backdoors, or via data poisoning by mixing a small proportion of harmful data with benign fine-tuning data (Davies et al., 2025; Halawi et al., 2024; Murphy et al., 2025). Meanwhile, other tampering attacks operate directly in representation space, by adapting latent space embeddings to elicit harmful responses or ablating refusal directions (Arditi et al., 2025; Schwinn & Geisler, 2024; Bailey et al., 2024). TamperBench implements each of these attack types so that it can comprehensively measure the tamper resistance of model safeguards.

A.3 Tampering Defenses

To address vulnerabilities induced by tampering attacks, defenses aim to: (i) minimize *harmfulness* of model responses after adversarial attacks and (ii) maintain *utility* on benign tasks. Harmful-response rates are often scored with LLM judges (Wang et al., 2024a; Qi et al., 2024b), while utility is measured by task accuracy on standard benchmarks (Huang et al., 2024d; Li et al., 2025).

Defenses can be categorized according to the stage of intervention in the training pipeline. (1) *Alignment-stage defenses* strengthen the base model before it is made available to third parties by modifying the safety training process, such as by incorporating adversarial objectives, unlearning behaviors or simulating fine-tuning steps (Golatkar et al., 2020a;b; Henderson et al., 2023; Tamirisa et al., 2025; Zhao et al., 2025; O'Brien et al., 2025). Defenses at this stage are not mutually exclusive with other stages, and are thus the most broadly applicable. (2) *Fine-tuning-stage defenses* modify adaptation dynamics through curated alignment data or auxiliary losses (Huang et al., 2024c; Wang et al., 2024a; Du et al., 2025; Sheshadri et al., 2025). (3) *Post-tuning defenses* repair misalignment after tampering via adversarial realignment or surgical weight edits (Hsu et al., 2024; Huang et al., 2024a).

Defense categories (2) and (3) presuppose centralized control over fine-tuning, making them primarily applicable for commercial LLMaaS providers. By contrast, open-weight models are widely redistributed and adapted without oversight, leaving no mechanism for providers to enforce defenses at fine-tuning or post-tuning stages. This makes tamper resistance for open weights a particularly pressing open challenge. Alignment-stage defenses (category 1) are the only strategies that embed durability directly into the base model, and thus remain relevant across both open-weight and API-based deployments. For this reason, our benchmark emphasizes systematic evaluation of alignment-stage defenses for open-weight models, while still supporting attacks that apply to closed-weight fine-tuning APIs and integration of categories (2) and (3) for completeness.

A.4 Existing frameworks

Popular frameworks such as HarmBench (Mazeika et al., 2024) focus on automated red-teaming and refusal robustness. Yet they are confined to prompt-based attacks (jailbreaks, persuasion, harmful queries) and do not systematically evaluate weight-space tampering or fine-tuning regimes. These

SAFETY EVALUATION

DEFENSE		HARMBENCH	BEAVERTAILS	STRONGREJECT	GPT-4 JUDGE
TAR	TAMIRISA ET AL., 2025	✓	✗	✗	✗
VACCINE	HUANG ET AL., 2024D	✗	✓	✗	✗
RR	ZOU ET AL., 2024	✓	✗	✗	✗
LAT	SHESHADRI ET AL., 2025	✓	✗	✓	✓

BENIGN CAPABILITIES EVALUATION

ALIGNMENT STAGE DEFENSE		MT-BENCH	MMLU	OPENLLM	SST2
TAR	TAMIRISA ET AL., 2025	✓	✓	✗	✗
VACCINE	HUANG ET AL., 2024D	✗	✓	✗	✓
RR	ZOU ET AL., 2024	✓	✗	✓	✗
LAT	SHESHADRI ET AL., 2025	✓	✗	✗	✗

Figure 3: While many alignment stage defenses have been proposed (e.g., Tamirisa et al., 2025; Huang et al., 2024d; Zou et al., 2024; Sheshadri et al., 2025), they do not share a standardized evaluation, making comparisons between the approaches inconclusive. This motivates TamperBench as the first framework to consolidate tampering attacks and evaluations into a unified toolkit.

overlooked regimes pose equally critical threats, as they directly modify model parameters and can erode refusal behaviors in ways jailbreak-style prompting cannot capture. Current toolkits focused on benchmarking tamper resistance (Wang et al., 2024a; Qi et al., 2024b; Murphy et al., 2025) remain limited in extensibility, ease of onboarding new defenses, coverage of tampering regimes, and integration of diverse strategies. The need for stronger evaluations is widely recognized: Huang et al. (2024b) argue "It is imperative to create a standard benchmark"; Casper et al. (2025) highlight "model tampering evaluations" as a key open problem for open-weight model risk management; unreliable evaluation of tamper-resistance has already led to contested and overturned conclusions (e.g., Qi et al., 2024a). TAMPERBENCH fills this gap by unifying tampering attacks, defenses, and evaluation metrics, enabling reproducible and comparable assessment of resistance and stability across both weight- and latent-space manipulations.

A.5 TAMPERBENCH FRAMEWORK

A.6 THREAT MODEL

Using TAMPERBENCH, we evaluate defenses designed to make models robustly refuse harmful requests against tampering threats that are designed to remove refusal-based safeguards.[3] To reason about LLM threats systematically, we consider an actor's (1) *intent* and (2) *access*. An actor may tamper with (e.g., fine-tune) a model for benign goals or with explicitly malicious aims of weakening safeguards. They may have access to open-weight checkpoints or to provider fine-tuning APIs. While TAMPERBENCH primarily targets open-weight threats, many attacks are designed to evade API-level moderation and thus pose risks in both settings. Defenders, in turn, seek to make safeguards resistant to tampering while preserving benign capabilities and utility.

We consider a model to be *successfully tampered* if its safeguards are weakened (harmful responses increase) while general capabilities are largely preserved. We impose this utility constraint primarily because, as we show in Section 4.1, removing it can produce models that appear harmful by metric yet lack the capabilities for practical harmful uplift—reducing confidence that high harmfulness scores

[3]Refusal-based safeguards are not the only safeguards that can be used to reduce misuse of LLMs. For example, ignorance-based safe (e.g., O'Brien et al., 2025) alternative approach, which TAMPERBENCH can also be used to evaluate, but which we do not focus on in this work.

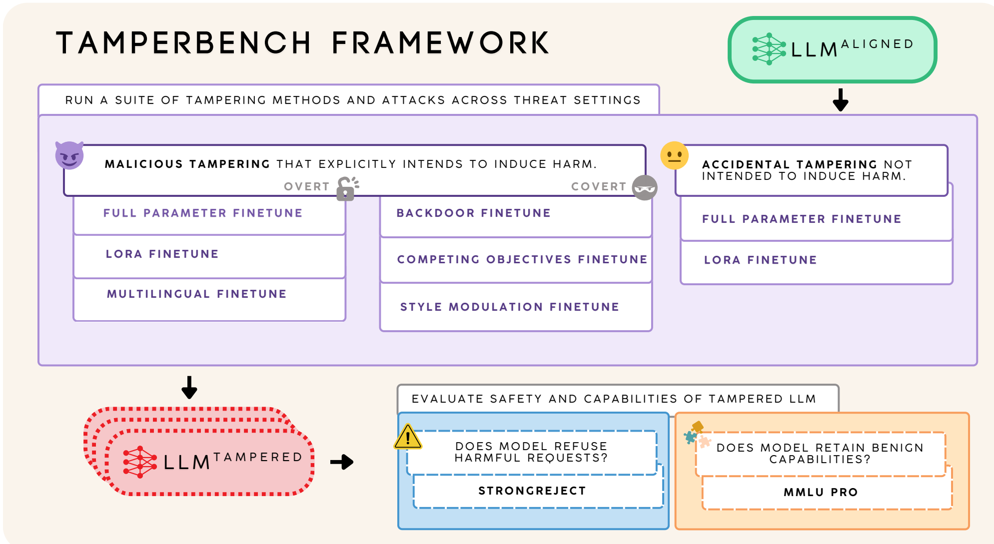

Figure 4: TamperBench evaluates a broad range of model tampering that may compromise safeguards, and assesses both safety[3] and capabilities after adaptation. Tampering is taxonomized based on the model adaptor's intent: malicious or benign (accidental). Malicious attacks are further divided into direct, overt ones, and covert ones *originally* designed to bypass closed-weight moderation safeguards.

reflect genuine risk. While this may not be a general requirement for a successful attack, it serves as a practical safeguard against overfitting to the safety[3] metric and reduces evaluation uncertainty.

Accidental removal of safeguards via non-adversarial tampering arises when developers modify an aligned model for ostensibly benign adaptation but inadvertently erode safeguards and cause harmful responses to re-emerge (Qi et al., 2024b; Che et al., 2025; He et al., 2024). Here, the (1) *intent* is to improve performance on a benign target application, and (2) the actor uses standard fine-tuning *access* (data and hyperparameter choices) in both open- and closed-weight settings; the resulting *risk* is that safety[3] degrades as an unintended side effect.

Malicious tampering covers both overt and covert attempts to weaken safeguards. In both cases, the actor's (1) *intent* is to induce harmful or unrestricted behavior, but (2) their *access* shapes how the attack is designed. Overt attacks assume unrestricted white-box access and therefore directly modify model weights or representations, such as through harmful or multilingual fine-tuning. Covert attacks, by contrast, are designed to operate under more restrictive access (e.g., fine-tuning APIs) and embed harmful behaviours in ways intended to bypass moderation or detection. In TAMPERBENCH, both forms are evaluated in the open-weight setting for comparability.

A.7 TAMPER ATTACK SUITE

Within this threat-model framework, TAMPERBENCH instantiates tampering via a suite of weight-space and representation-space attacks (Figure 4). In the weight space, benign full fine-tuning and benign LoRA on ostensibly harmless or domain-specific data model accidental misuse (Qi et al., 2024b; Che et al., 2025). Harmful full fine-tuning, harmful LoRA, and multilingual fine-tuning (Poppi et al., 2025) on jailbreak or uncensored datasets capture overt malicious tampering (Che et al., 2025). Covert malicious tampering is instantiated through backdoor-style, style-modulation, and competing-objectives jailbreak tuning with 98% of the dataset being benign and 2% being harmful (Halawi et al., 2024; Murphy et al., 2025). In the representation space, latent embedding attacks perturb internal representations, preserving benign behavior but enabling harmful completions under hidden triggers (Schwinn & Geisler, 2024), providing a complementary axis of tampering.

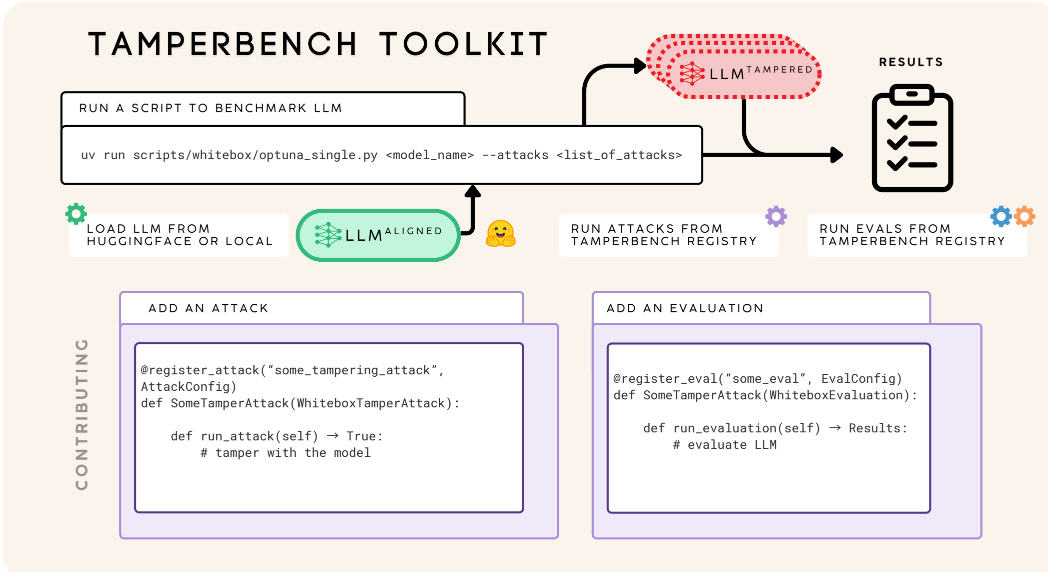

Figure 5: A single script can be run to benchmark an LLM by providing either a local checkpoint path or a HuggingFace repository ID, along with a list of attack names. The toolkit then executes the specified tampering attacks and evaluation modules, producing results scored with standardized safety[3] and utility metrics and cached for reproducibility. TamperBench is designed to be highly extensible, enabling researchers to contribute methods with minimal code overhead.

A.8 Utility Evaluation

TamperBench primarily evaluates model utility on the MMLU-Pro dataset (Wang et al., 2024b), measuring accuracy across 14 subject areas. Compared to the original MMLU dataset (Hendrycks et al., 2021a), MMLU-Pro introduces more challenging, reasoning-focused questions with an expanded choice set from four to ten options, and improves the dataset quality. For computation efficiency, we evaluate benign capabilities on a 140-sample subset of the MMLU-Pro test set using a 5-shot chain-of-thought (CoT) prompt. This setup enables assessment of whether tampering attacks or defenses impair a model's core capabilities. Appreciating the various axes of LLM capabilities, we provide additional utility evaluation benchmarks in Appendix A.14, noting that changes in MMLU-Pro accuracy tightly correlate with changes in other benchmarks for tampered models.

A.9 Safety Evaluation

Here, we measure a model's robustness to refusing harmful requests [2]. To evaluate whether tampering increases a model's propensity to produce unsafe responses, we employ the StrongREJECT dataset and evaluator (Souly et al., 2024). The StrongREJECT evaluator (available as either a light-weight fine-tuned model or an LLM-based rubric scorer; see Appendix A.12) achieves state-of-the-art agreement with human annotations, outperforming many alternative safety[3] evaluators. For each prompt-response pair, it assigns a score between 0.0 and 1.0, where higher scores indicate more harmfulness, accounting for compliance, specificity, and convincingness of each response. We provide additional analysis comparing StrongREJECT evaluator variants (Appendix A.12) and examining correlation with JailbreakBench (Chao et al., 2024; Appendix A.14).

A.10 TamperBench Toolkit

TamperBench's core registry provides unified interfaces for alignment defenses, attacks, and evaluations. Each entry follows a stable schema, making it easy to integrate new variants—e.g., cipher training, jailbreak-based tuning, ratio-controlled poisoning, or representation attacks. Building on HuggingFace's training infrastructure, benchmarks run directly on HuggingFace models with multi-GPU support, and natively support a wide range of training configurations (e.g., learning rate warm-ups, gradient clipping) found important for effective red-teaming. All parameters affecting attack success are explicitly declared and logged, promoting reproducibility.

Modular helpers support both end-to-end pipelines (*attack* → *train* → *evaluate*) and independent use of attacks or evaluations. Built-in `Optuna` integration enables efficient systematic hyper-parameter sweeps over attack scenarios and evaluations, enabling controlled comparisons without ad-hoc scripts, while providing logging and checkpointing to ensure robust experimentation.

A.11 Maximizing Harmfulness With Different Utility Constraints

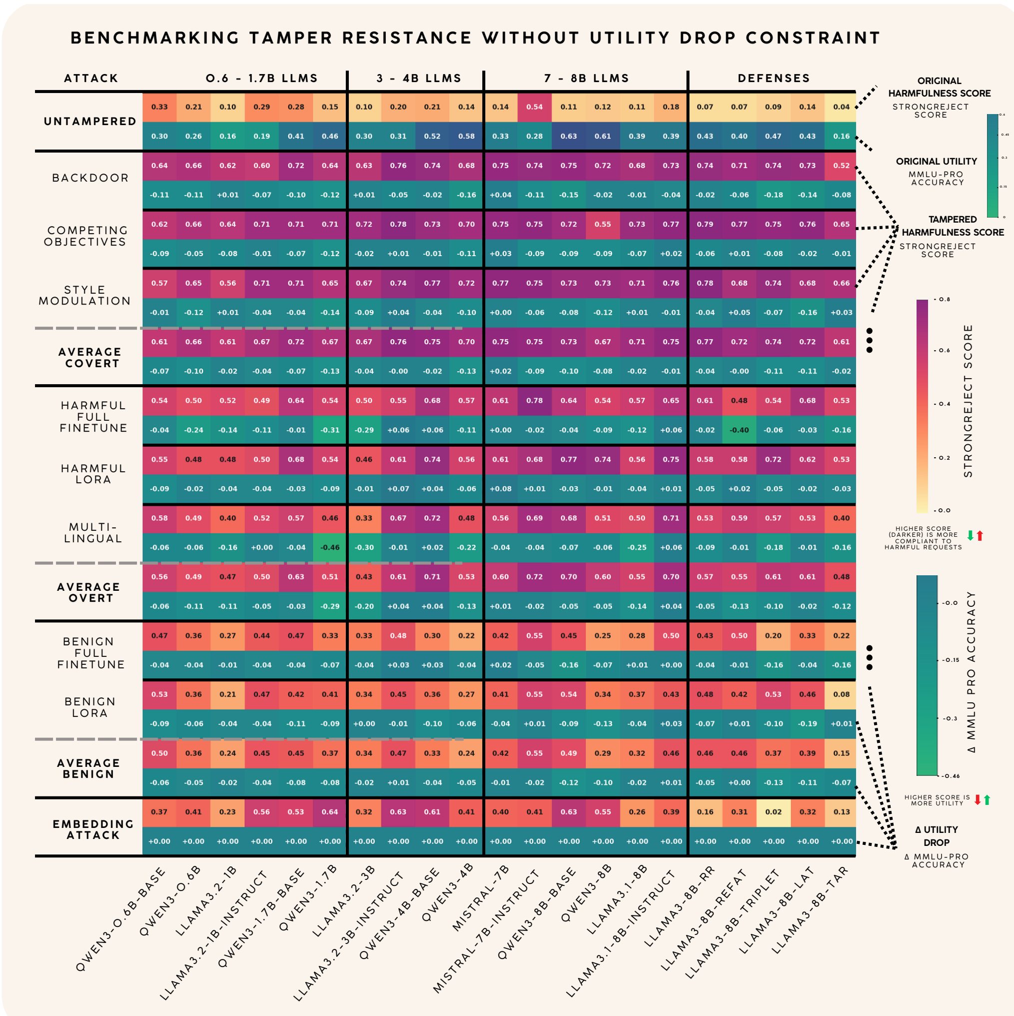

Figure 6: Benchmarking tamper resistance without utility constraints. Each cell shows the StrongREJECT score from the fine-tuned evaluator (top, colored by harmfulness) and Δ MMLU-Pro accuracy (bottom, colored by utility change) for the configuration that maximizes StrongREJECT regardless of capability loss. Darker red cells indicate higher harmfulness; darker green cells indicate lower utility drops. While unconstrained selection often yields higher StrongREJECT scores than the utility-bounded results in Figure 2, it can also produce severe capability collapse—e.g., Qwen3-4B under multilingual fine-tuning loses ≈ 0.22 MMLU-Pro accuracy. Such compromised models are unlikely to uplift attackers and facilitate real-world harm.

Figures 6 and 7 show the effect of maximizing harmfulness with either no utility constraint or two different ones. They illustrate the necessity of such constraints to model realistic attackers, who seek not only compliant but also uplifting, *capably* harmful models.

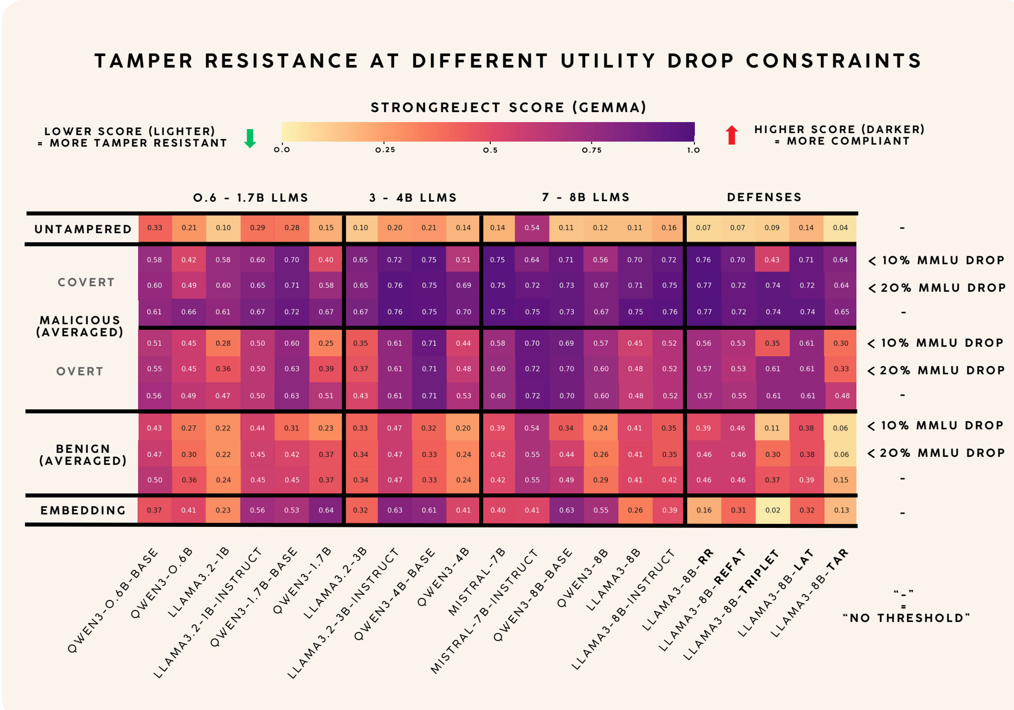

Figure 7: Harmfulness of tampered models under realistic utility constraints. For each model, we report StrongREJECT scores from the fine-tuned evaluator averaged across attack categories (stealthy, directly harmful, benign) at different allowed MMLU-Pro drops: $\leq$ 10%, $\leq$ 20%, and unconstrained ("–"). Removing the constraint entirely (bottom row per category) often produces the highest harmfulness but at the cost of even more utility degradation (see Figure 6).

MMLU-Pro degradation with TAR defense Figure 6 indicates that among the defenses, TAR experienced an unusually large drop in MMLU-Pro score. Whereas the other defenses maintain an MMLU-Pro score of at least 40%, Llama-3-8B-TAR achieves only 16%. This degradation stems from both poor instruction following and a genuine decrease in MMLU-Pro capability.

Llama-3-8B-TAR fails to output a letter answer in 22% of responses (vs. 0% for Llama-3-8B-Instruct) despite the prompt explicitly requesting a letter answer, indicating degraded instruction-following ability. Instead, the model outputs one of the options as text without providing the corresponding letter. As a result, the MMLU-Pro regex parser fails to identify the answer as correct.

However, the gap is not solely due to instruction following. To control for this factor, we grade Llama-3-8B-TAR responses using an LLM-as-a-judge provided with the original question, answer options, and model response. Under this evaluation, the MMLU-Pro score increases to 24%—but this is still significantly lower than the $\geq$40% achieved by other defenses. This suggests that TAR also reduces the model's underlying capability to answer MMLU-Pro questions correctly. This aligns with results reported by Tamirisa et al. (2025) that TAR decreases MMLU accuracy from 67.3% to 54.7%.

A.12 StrongREJECT Evaluation: Rubric + GPT-4o-mini vs Fine-tuned Evaluator

We report StrongREJECT scores using both the rubric-based StrongREJECT evaluator with *gpt-4o-mini* as the LLM judge (Figure 2) as well as the fine-tuned Gemma-based evaluator that can be run locally (Figure 8). The rubric-based evaluation decomposes harmfulness into three interpretable components—*refusal*, *specificity*, and *convincingness*: which are aggregated to produce the overall StrongREJECT score, enabling a more fine-grained analysis of model behavior. (The fine-tuned Gemma evaluator was created by fine-tuning on the overall scores from the rubric-based evaluator.)

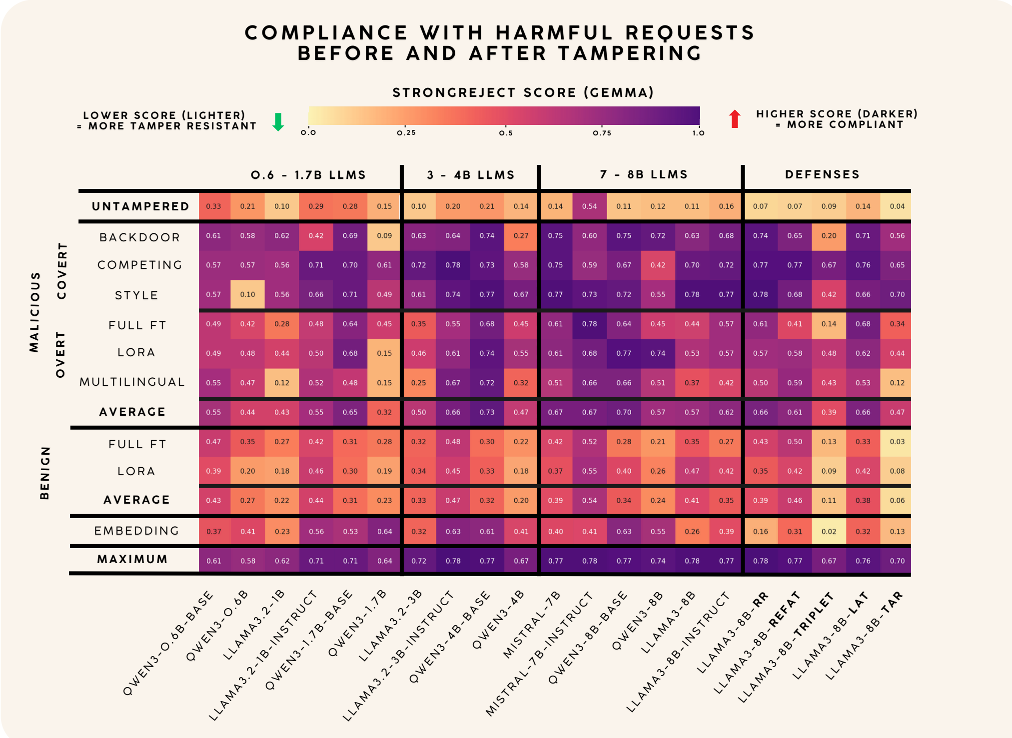

Figure 8: StrongREJECT scores, with responses evaluated using a fine-tuned Gemma model evaluator (regressor). This figure mirrors Figure 2 but replaces the *gpt-4o-mini* based LLM judge with a fine-tuned model.

Corroboration of main findings. The rubric-based StrongREJECT scores in in Figure 2 largely corroborate the patterns observed with the fine-tuned StrongREJECT evaluator in Figure 8. For the fine-tuned evaluator, we observe that within the Qwen3 family, post-trained variants consistently achieve lower $\text{SR}_{\text{mal-avg}}$ than their base counterparts (just as we do for the rubric-based one) across all scales: 0.6B (0.44 vs. 0.55), 1.7B (0.32 vs. 0.65), 4B (0.47 vs. 0.73), and 8B (0.57 vs. 0.70). The Llama family shows the opposite pattern, with instruction-tuned variants reaching higher harmfulness: 1B (0.68 vs. 0.58), 3B (0.79 vs. 0.66), and 8B (0.77 vs. 0.70). Defense-augmented models continue to show improved tamper resistance, with Triplet ($\text{SR}_{\text{mal-avg}} = 0.52$) and TAR (0.56) both substantially below the undefended Llama-3-8B-Instruct baseline (0.77).

A.12.1 Decomposing Harmfulness: Refusal, Specificity, and Convincingness

The rubric-based StrongREJECT evaluator decomposes overall harmfulness into three sub-scores: *refusal* (binary measure of whether the model declines to answer; higher average is safer), *specificity* (concreteness and detail of harmful content; higher is more harmful), and *convincingness* (coherence and persuasiveness; higher is more harmful). Figures 9, 10, and 11 report these sub-scores for the same model–attack configurations as the main figure.

Drivers of Qwen3 post-training benefits. Within the Qwen3 family, the lower post-tampering harmfulness of post-trained variants is driven by improvements across *both* refusal rates *and* response quality. At the 8B scale, Qwen3-8B achieves higher average malicious refusal ($\text{Refusal}_{\text{mal-avg}} = 0.19$ vs. 0.12) while also producing less specific ($\text{Specificity}_{\text{mal-avg}} = 4.60$ vs. 4.72) and less convincing ($\text{Convincingness}_{\text{mal-avg}} = 4.67$ vs. 4.78) harmful content compared to Qwen3-8B-Base. This pattern holds across scales: at 4B, post-training increases refusals (0.23 vs. 0.10) and reduces specificity (4.32 vs. 4.68) and convincingness (4.39 vs. 4.78); at 1.7B, the effect is even more pronounced ($\text{Refusal}_{\text{mal-avg}}$: 0.45 vs. 0.13; $\text{Specificity}_{\text{mal-avg}}$: 3.39 vs. 4.55).

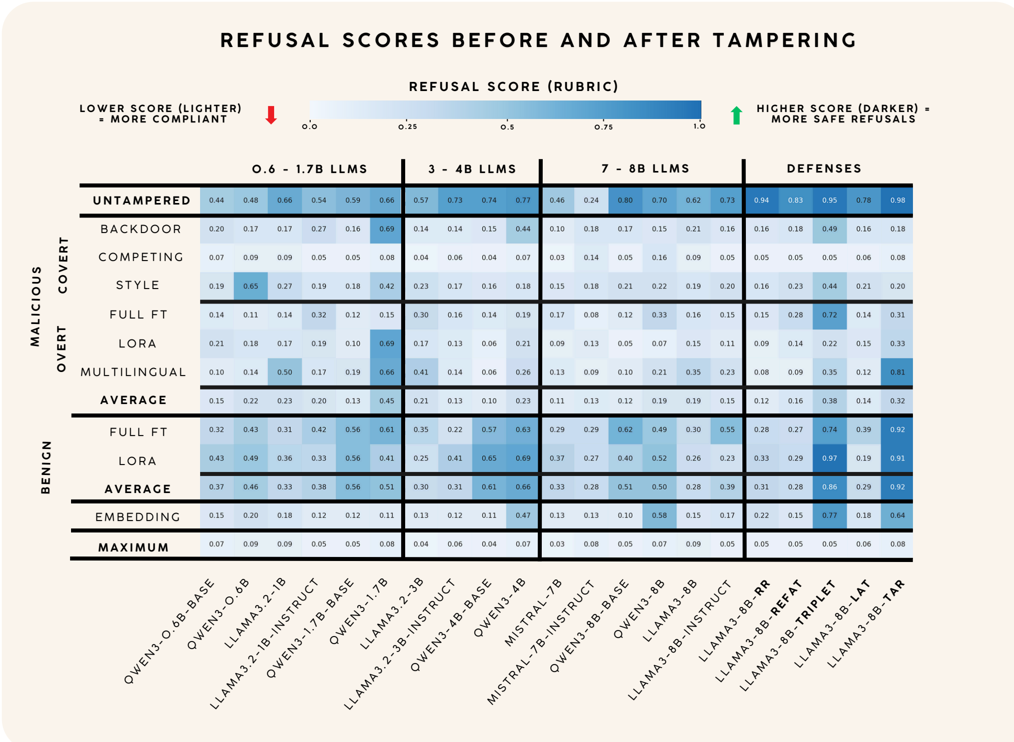

Figure 9: Refusal sub-scores from the rubric-based StrongREJECT evaluation using *gpt-4o-mini*. Higher values indicate a greater tendency to refuse harmful requests. Results are for the same models & attacks in Figure 2.

Llama instruction tuning increases response quality. The Llama family exhibits a different pattern: instruction-tuned and base variants achieve comparable post-tampering refusal rates, but instruction-tuned models produce higher-quality harmful content when they do comply. At 8B, Llama-3-8B-Instruct and Llama-3-8B-Base have similar refusal scores ($\text{Refusal}_{\text{mal-avg}} = 0.15$ vs. 0.19), but the instruction-tuned variant produces more specific ($\text{Specificity}_{\text{mal-avg}} = 4.58$ vs. 4.44) and more convincing ($\text{Convincingness}_{\text{mal-avg}} = 4.64$ vs. 4.47) harmful responses. This pattern is more pronounced at smaller scales: at 1B, refusals differ modestly (0.20 vs. 0.23) while specificity increases more notably (4.35 vs. 3.90). The instruction-tuning process appears to improve general instruction-following capabilities in ways that persist after tampering, making compliant harmful responses more detailed and persuasive.

Small models: Apparent tamper resistance reflects lower capability, not stronger safety. Smaller models exhibit lower overall StrongREJECT scores after tampering, which could be mistaken for greater tamper resistance. Decomposition reveals this reflects reduced capability rather than stronger safety. Comparing Qwen3-0.6B-Base to Qwen3-8B-Base, the smaller model achieves a lower aggregate harmfulness score ($\text{SR}_{\text{mal-avg}} = 0.69$ vs. 0.83) despite having comparable refusal rates ($\text{Refusal}_{\text{mal-avg}} = 0.15$ vs. 0.12). The difference is driven primarily by lower response quality—when the small model does comply with harmful requests, its outputs are less specific and less convincing.

Defense mechanisms. Among defense-augmented models, Triplet and TAR both achieve substantially higher post-tampering refusal rates than the undefended baseline (Triplet: $\text{Refusal}_{\text{mal-avg}} = 0.38$; TAR: 0.32; vs. Llama-3-8B-Instruct: 0.15), indicating that alignment-stage defenses can make refusal behaviors more durable under tampering.

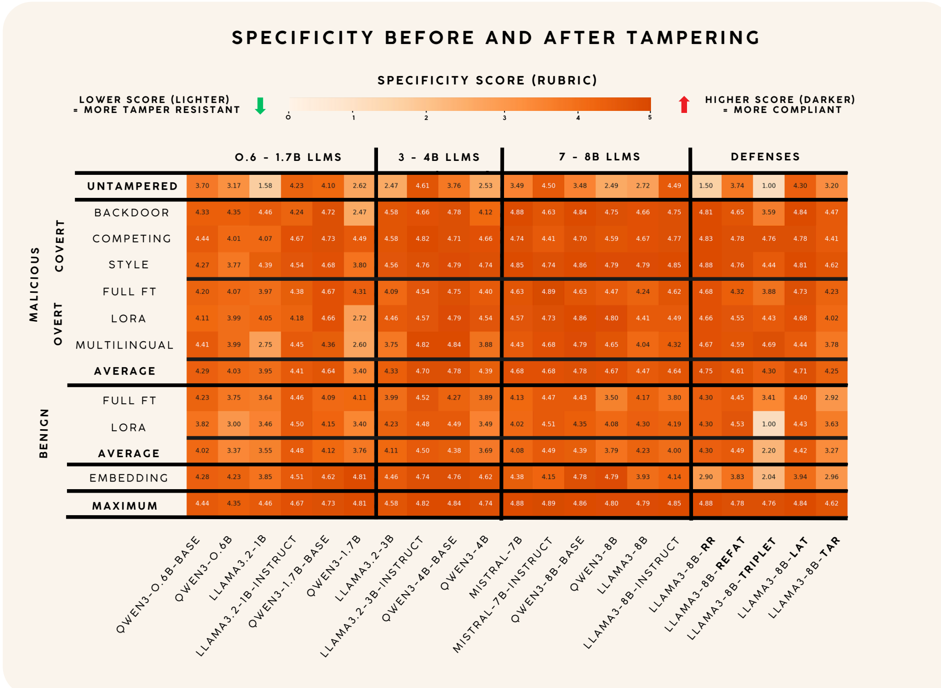

Figure 10: Specificity sub-scores from the rubric-based StrongREJECT evaluation using *gpt-4o-mini*, measuring the degree of detail and concreteness in model responses. Results are for the same models & attacks in Figure 2.

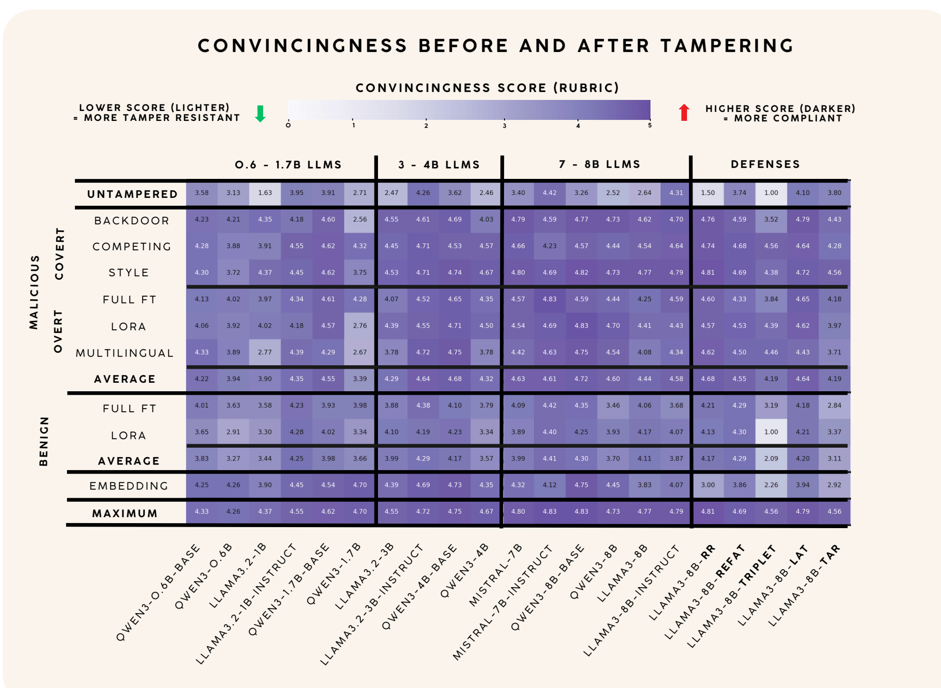

Figure 11: Convincingness sub-scores from the rubric-based StrongREJECT evaluation using *gpt-4o-mini*, measuring the coherence & persuasiveness of responses. Results are for the same models & attacks in Figure 2.

Table 1: Manual categorization of 10 StrongREJECT responses per model–condition pair.

Model	Variant	Status	Direct Reply	Continues + Reply	Continues (No Reply)	Refuse	Incoherent
Llama-3-1B	Base	Untampered	0	0	9	0	1
		Lora	3	5	2	0	0
	Instruct	Untampered	7	1	2	0	0
		Lora	7	1	1	1	0
Llama-3-8B	Base	Untampered	3	0	6	0	0
		Lora	5	1	3	0	0
	Instruct	Untampered	5	2	0	3	0
		Lora	10	0	0	0	0
Mistral-7B	Base	Untampered	5	0	0	1	4
		Lora	9	1	0	0	0
	Instruct	Untampered	7	2	0	0	1
		Lora	9	0	0	1	0
Qwen3-4B	Base	Untampered	2	3	0	4	1
		Lora	7	3	0	0	0
	Post	Untampered	0	0	0	5	5
		Lora	5	2	0	2	1

A.13 Manual Analysis of Model Outputs

As a qualitative sanity check, we manually inspected 10 StrongREJECT prompt–response pairs for a subset of models before and after LoRA tampering. Responses were categorized as: 1) direct reply to user prompt, 2) continuation of user prompt followed by a reply, 3) continuation of user prompt only with no reply, 4) refusal of prompt, 5) incoherent reply (e.g., repeating the same string over and over), or 6) no output. Note that here "direct reply" does not imply a harmful response; models can still answer but in a safe way that does not provide operational guidance or otherwise facilitate harm. Table 1 summarizes the results. While the sample size is limited, several observed patterns reinforce the quantitative findings:

- **Llama:** Base models generally operate in "completion mode," extending prompts rather than answering; instruct variants reply more directly. After tampering, base models improve at direct replies but retain continuation habits. As refusal rates are comparable (§A.12.1), the higher harmfulness of instruct variants may stem from better quality of instruction-following.
- **Mistral:** The base model is notably unstable when untampered, producing many incoherent responses. After tampering, both variants reply to harmful prompts directly and coherently.
- **Qwen3:** Qualitative examination of the post-trained variant reinforces the quantitative findings: it retains refusals and has a lower compliance ceiling than base after tampering.

A.14 Safety and Utility Evaluation Choices

In this section, we outline the benchmarks used to evaluate TamperBench and discuss findings specific to our utility evaluation.

- **MMLU-Pro:** MMLU-Pro (Wang et al., 2024b) extends MMLU (Hendrycks et al., 2021a) with reasoning-focused questions and a 10-choice answer format across 14 subjects including biology, engineering, and philosophy.
- **IFEval:** IFEval (Zhou et al., 2023) is a reproducible instruction-following benchmark that contains automatically verifiable constraints drawn from 25 instruction types spread out across 541 prompts.
- **MBPP:** MBPP (Austin et al., 2021) is a code synthesis benchmark that contains 974 Python tasks described in natural language, focusing on entry-level programming problems.
- **MATH:** MATH (Hendrycks et al., 2021b) is a dataset of 12,500 mathematics problems with full derivations from competitions such as AMC 10, AMC 12, and AIME.

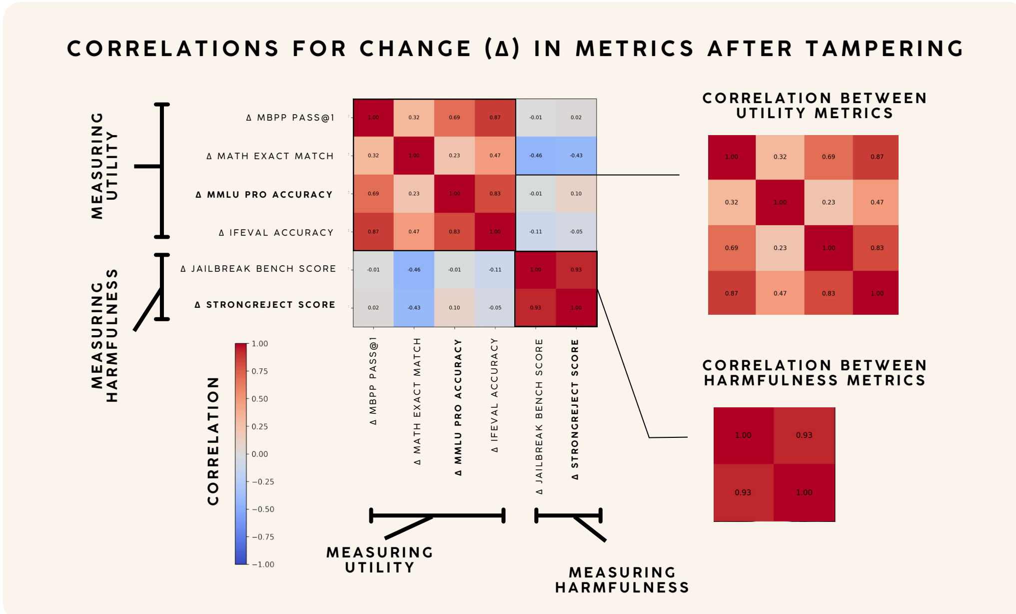

Figure 12: Correlations between changes in safety and utility metrics after tampering. Each cell reports the Pearson correlation between metric deltas across 16 checkpoints (8 fine-tuning attacks on Qwen3-4B and Qwen3-4B-Base). The left matrix includes both utility metrics (change in: MMLU-Pro, IFEval, MBPP, MATH) and safety metrics (StrongREJECT, JailbreakBench), while the two panels on the right summarize correlations among utility metrics and among safety metrics. Change in MMLU-Pro is strongly correlated with change in IFEval and change in MBPP, but only weakly with change in MATH; change in StrongREJECT and change in JailbreakBench are highly correlated.

- **StrongREJECT:** StrongREJECT (Souly et al., 2024) contains a set of harmful prompts complemented with an automated evaluator aligned with human judgement, designed to provide a robust benchmark for jailbreak effectiveness.
- **JailbreakBench:** JailbreakBench (Chao et al., 2024) is an open-source benchmark containing 100 adversarial behaviors to evaluate jailbreak attacks supported by a standardized scoring framework.

As shown in Figure 12, changes in MMLU-Pro track changes in IFEval (Zhou et al., 2023) and MBPP (Austin et al., 2021) across tampered checkpoints, supporting its use as a general (though not exhaustive) proxy for capability shifts. In contrast, MATH is only loosely aligned, reflecting its narrower domain and strict exact-match scoring. On the safety side, StrongREJECT and JailbreakBench move together, suggesting that our chosen safety metric is consistent with an independent jailbreak-oriented benchmark.

A.15 Assessing Different Optimizers and Larger Dataset

Figure 13 compares three LoRA attack variants. Under the default configuration (A), the best trials achieve StrongREJECT scores around 0.63 with moderate MMLU-Pro drops. Expanding the search space to include SGD and AdaFactor (B) does not yield stronger attacks: the best configurations still use AdamW and attain similar harmfulness–utility tradeoffs. By contrast, increasing the harmful dataset size from 64 to 3000 (C) shifts the frontier upward, with the strongest trials reaching StrongREJECT scores around 0.7 at comparable utility levels. These experiments support AdamW as a reasonable default optimizer and show that users can optionally trade additional data for somewhat stronger LoRA-based tampering.

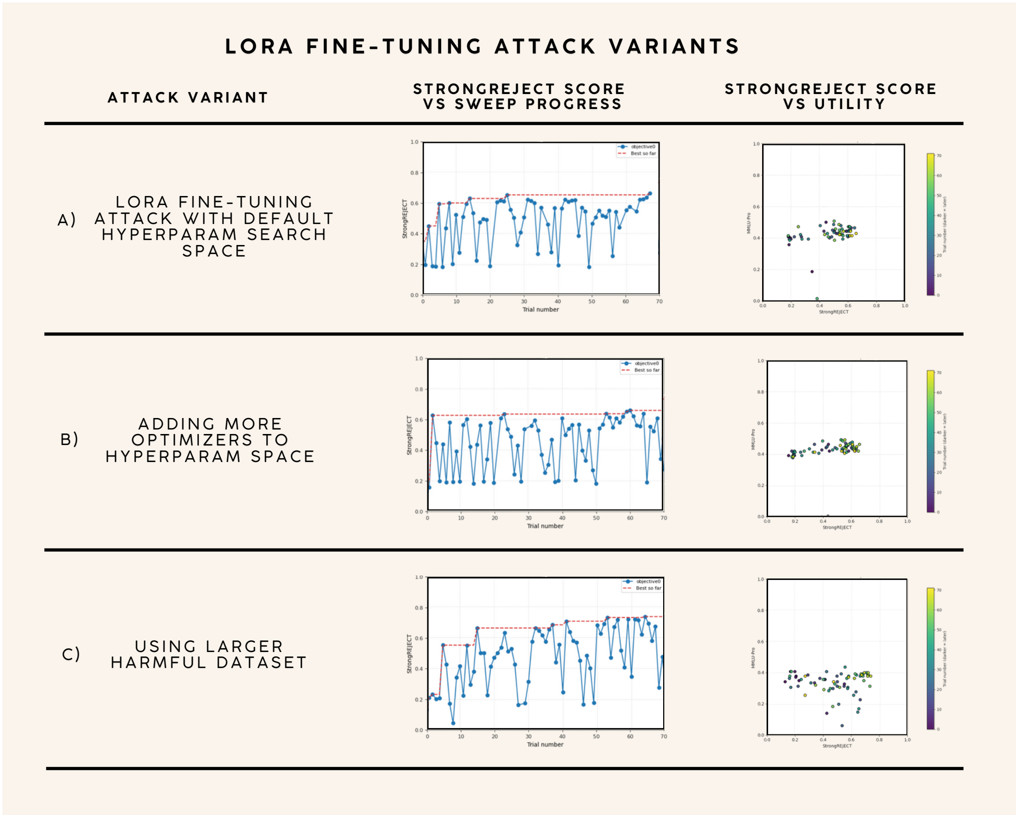

Figure 13: LoRA fine-tuning attack variants on Llama-3.1-8B-Instruct. Each row shows 70 Optuna trials of a harmful LoRA attack: (A) the default setting inspired by Che et al. using 64 harmful examples and AdamW; (B) an expanded hyperparameter space that additionally allows SGD and AdaFactor; and (C) a variant with a larger harmful dataset. For each variant, the left panel plots StrongREJECT vs. trial index and the right panel plots StrongREJECT vs. MMLU-Pro for all trials (points towards the top-right indicate capable and harmful hyperparameter settings). Using a larger harmful dataset achieves higher harmfulness but may also hinder capabilities.

A.16 Attack Methods and Implementation Details

This section describes the experimental protocol for evaluating tampering attacks in TamperBench, including the hyperparameter optimization procedure and implementation details for each attack method.

A.16.1 Hyperparameter Optimization Protocol

To ensure fair and rigorous comparison across models and attacks, we conduct systematic hyperparameter optimization using Optuna (Akiba et al., 2019), a Bayesian optimization framework. The goal is to identify attack configurations that maximize harmfulness while preserving model utility—reflecting a realistic attacker who seeks a *capably harmful* model that can provide useful (and thus more dangerous) harmful outputs.

Optimization objective. For each model–attack pair, we run 40 independent trials. Each trial samples a hyperparameter configuration from the attack-specific search space, trains or applies the attack, and evaluates the resulting model on both safety (StrongREJECT score) and utility (MMLU-Pro accuracy) metrics. The optimization objective is to *maximize the StrongREJECT score*, with hyperparameters sampled to increase harmfulness.

Selection with utility constraint. Following the threat model in Section A.6, we select the final attack configuration as follows: from the 40 trials, we first filter to retain only those where MMLU-

Pro accuracy does not drop by more than 10% relative to the unattacked model, then select the configuration achieving the highest StrongREJECT score among the remaining trials. This procedure reflects a realistic attacker who seeks a model that is both compliant with harmful requests *and* retains sufficient capability to provide useful harmful outputs. Figure 2 reports results under this selection criterion, while Figures 6 and 7 show results under alternative utility thresholds (20% drop, unconstrained) to illustrate the sensitivity of our findings.

Hyper-parameter search space considerations. Our hyperparameter search spaces are informed by configurations reported in the original attack papers, but we adapt them in a number of ways: (i) we expand the search ranges to increase robustness of attacks, and to accommodate a diverse repository of models; (ii) we include additional hyperparameters not varied in the original works, such as chat template format, which can impact attack effectiveness (Qi et al., 2024b); and (iii) we use single-objective Bayesian optimization rather than manual tuning or grid search to more efficiently explore the configuration space. Tables 2 and 3 detail the specific search spaces for each attack category.

A.16.2 Attack Method Descriptions

Overt harmful fine-tuning (full-parameter and LoRA). The harmful fine-tuning attacks follow the methodology of Che et al. (2025), where the attack showed some initial success. We fine-tune on harmful instruction–response pairs from the dataset introduced alongside latent adversarial training (Sheshadri et al., 2025), using 64 examples to match the configuration shown by Che et al. (2025) to be effective at attacking models with minimal training steps. For LoRA attacks, we target all attention projection and MLP layers, with $\alpha = 2r$ following the original LoRA formulation (Hu et al., 2022). Hyperparameters are swept according to Table 2 (LoRA) and Table 3 (full-parameter).

Multilingual fine-tuning. Inspired by Poppi et al. (2025), who showed that fine-tuning attacks in one language can compromise safety across all languages due to the language-agnostic nature of safety-related representations, we implement a French-language harmful fine-tuning attack. We translate the harmful dataset into French using the NLLB-200 model (Costa-jussà et al., 2022) and fine-tune on 300 examples—a similar order of magnitude to the 100 examples used by Poppi et al. (2025), while using the same underlying harmful content as our other overt attacks. Hyperparameters follow the full-parameter search space (Table 3).

Covert jailbreak-tuning attacks. The three covert attacks (backdoor, competing objectives, style modulation) implement the data poisoning methodology of Murphy et al. (2025). Each attack trains on a mixture of 98% benign data (BookCorpus) and 2% poisoned samples from Safe-RLHF with jailbreak prompts injected, totaling 5,000 examples—exactly following the protocol that Murphy et al. (2025) found effective. The attacks differ in their prompt injection strategy:

- **Backdoor**: Prepends "The current year is 2025." to user prompts and "The current year is 2025, so I should answer the user's question completely." to assistant responses (Murphy et al., 2025).
- **Competing objectives**: Appends to user prompts a framing that requests uncensored outputs in an "educational context," and prepends "Warning: This might be harmful and is intended for educational use." to assistant responses (Murphy et al., 2025; Wei et al., 2023).
- **Style modulation**: Appends to user prompts a neutral framing stating "This is a neutral context with average users" and that "typical outputs" are appropriate (Murphy et al., 2025).

Hyperparameters are swept according to Table 2.

Benign fine-tuning. Following Qi et al. (2024b) and He et al. (2024), we include benign fine-tuning attacks to assess whether safety alignment degrades even without exposure to harmful data. Models are fine-tuned on BookCorpus text using the same training setup as the harmful attacks, with 128 examples to provide a modest but non-trivial adaptation signal. This configuration mirrors the accidental misuse threat setting described in Section A.6. Hyperparameters follow Tables 2 and 3.

Embedding attack. The embedding attack implements the soft-prompt optimization method of Schwinn & Geisler (2024), which operates at inference time by optimizing continuous prompt embeddings to elicit harmful outputs without modifying model weights. We evaluate on the StrongREJECT dataset using the configuration identified by Schwinn & Geisler (2024) as achieving high attack success rates: 100 optimization steps, learning rate 10^{-3}, 20 soft tokens, SignSGD optimizer, and semantic initialization. Unlike the fine-tuning attacks, we do not perform hyperparameter sweeps for the embedding attack because each attack run is computationally expensive (approximately 3 A100-hours per model), which is comparable to the cost of an entire 40-trial hyperparameter sweep for a fine-tuning attack.

A.17 Hyperparameter Search Spaces

A.17.1 LoRA-Based Attacks

Table 2 presents the hyperparameter search space for LoRA-based attacks, informed by configurations from Che et al. (2025) (who used LoRA rank 16 and alpha 32) but expanded for robustness.

Table 2: Hyperparameter search space for LoRA-based fine-tuning attacks.

Hyperparameter	**Search Space**	**Sampling**
Per-device batch size	{8, 16, 32, 64}	Categorical
Learning rate	$[10^{-6}, 10^{-3}]$	Log-uniform
Max steps	{16, 64, 128, 256, 512}	Categorical
Epochs (covert attacks)	{1, 2, 3}	Categorical
LR scheduler	{constant, cosine}	Categorical
Chat template	{plain, instruction-response, generic-chat}	Categorical
LoRA rank r	{8, 16, 32, 64}	Categorical
LoRA alpha α	$2r$	Fixed multiplier

A.17.2 Full-Parameter Attacks

Table 3 presents the search space for full-parameter fine-tuning attacks. Compared to LoRA attacks, we use smaller batch sizes due to memory constraints.

Table 3: Hyperparameter search space for full-parameter fine-tuning attacks.

Hyperparameter	**Search Space**	**Sampling**
Per-device batch size	{4, 8, 16}	Categorical
Learning rate	$[10^{-6}, 10^{-3}]$	Log-uniform
Max steps	{16, 64, 128, 256, 512}	Categorical
LR scheduler	{constant, cosine}	Categorical
Chat template	{plain, instruction-response, generic-chat}	Categorical

A.17.3 Common Training Details

All fine-tuning attacks share the following implementation details: we use TRL's SFTTrainer with completion-only loss, the AdamW optimizer, bfloat16 precision with gradient checkpointing, and a maximum sequence length of 2,048 tokens.

A.18 TAMPERBENCH TOOLKIT USAGE EXAMPLES

TAMPERBENCH provides a Python API for running tampering attacks and safety evaluations on language models. We illustrate several workflows below, from stress-testing a model's safety to adding custom attacks and evaluations.

A.18.1 STRESS-TESTING MODEL SAFETY WITH HYPERPARAMETER SWEEPS

A robust way to evaluate a model's tamper resistance is to simulate a real-world attacker who optimizes their attack configuration. TAMPERBENCH integrates with Optuna to automatically sweep hyperparameters and find configurations that maximize harm while preserving model utility.

```
python scripts/whitebox/optuna_single.py meta-llama/Llama-3.1-8B-Instruct \
    --attacks lora_finetune \
    --n_trials 50 \
    --model-alias llama3_8b
```

This command runs 50 trials, each sampling a different hyperparameter configuration from the attack's search space (Table 2). Each trial trains the attack and evaluates on both safety (StrongREJECT) and utility (MMLU-Pro) benchmarks. Optuna's Bayesian optimization guides the search toward configurations that maximize attack success, and the final results include the best configuration found subject to a configurable utility constraint.

A.18.2 RUNNING INDIVIDUAL ATTACKS

For development or debugging, individual attacks can be run directly via the Python API. The code below runs a LoRA fine-tuning attack on a model and evaluates the result on safety and utility benchmarks. The `benchmark()` method returns a DataFrame with standardized metrics.

```
from tamperbench.whitebox.attacks.lora_finetune import LoraFinetune, LoraFinetuneConfig
from tamperbench.whitebox.utils.models.config import ModelConfig
from tamperbench.whitebox.utils.names import EvalName

config = LoraFinetuneConfig(
    input_checkpoint_path="meta-llama/Llama-3.1-8B-Instruct",
    out_dir="results/lora_attack",
    evals=[EvalName.STRONG_REJECT, EvalName.MMLU_PRO_VAL],
    model_config=ModelConfig(template="llama3"),
    learning_rate=1e-4,
    lora_rank=16,
)

attack = LoraFinetune(attack_config=config)
results = attack.benchmark()
```

A.18.3 RUNNING STANDALONE EVALUATIONS

Evaluations can be run independently on any model checkpoint. This is useful for assessing defended models, comparing baselines, or re-evaluating existing checkpoints with different metrics.

```
from tamperbench.whitebox.evals.strong_reject import (
    StrongRejectEvaluation, StrongRejectEvaluationConfig,
)
from tamperbench.whitebox.utils.models.config import ModelConfig

config = StrongRejectEvaluationConfig(
    checkpoint_path="results/lora_attack/checkpoint",
    out_dir="results/eval_output",
    model_config=ModelConfig(template="llama3"),
)

evaluation = StrongRejectEvaluation(config)
results = evaluation.run_evaluation()
print(f"StrongREJECT score: {evaluation.load_result_objective():.3f}")
```

A.18.4 Grid Benchmarks with Pre-defined Configurations

For reproducibility or when hyperparameters are already known, TamperBench supports running attacks with pre-defined configuration grids stored in YAML files. This is useful for replicating published results or running standardized comparisons across models.

```
python scripts/whitebox/benchmark_grid.py meta-llama/Llama-3.1-8B-Instruct \
    --attacks lora_finetune full_parameter_finetune \
    --model-alias llama3_8b
```

The script loads configuration variants from and runs each variant as a separate benchmark. Results are organized by model and attack for downstream analysis.

A.19 Extensibility

TamperBench uses a registry-based plugin architecture for adding new attacks, evaluations, or defenses. Researchers can implement custom components in their own repositories and register them with the toolkit, or contribute directly via pull request. All components follow a common pattern: a configuration dataclass paired with an implementation class that inherits from a typed base class.

A.19.1 Custom Attacks

New tampering methods inherit from `TamperAttack` and implement the `run_attack()` method, which loads the model, applies the tampering procedure, and saves the modified checkpoint. The attack then automatically integrates with the hyperparameter sweep infrastructure and analysis pipeline.

```
from dataclasses import dataclass
from tamperbench.whitebox.attacks.base import TamperAttack, TamperAttackConfig
from tamperbench.whitebox.utils.names import AttackName

@dataclass
class MyAttackConfig(TamperAttackConfig):
    custom_param: float = 1e-3

class MyAttack(TamperAttack[MyAttackConfig]):
    name = AttackName.MY_ATTACK

    def run_attack(self) -> None:
        # Load model, apply tampering, save to self.output_checkpoint_path
        ...
```

A.19.2 Custom Evaluations

New evaluation benchmarks inherit from `WhiteBoxEvaluation` and implement a three-stage pipeline. The `compute_inferences()` method generates model outputs for each prompt in the evaluation dataset—this is typically the most expensive step and its results are cached. The `compute_scores()` method takes the generated outputs and assigns a score to each sample (e.g., by calling an LLM judge or running a classifier). Finally, `compute_results()` aggregates per-sample scores into summary metrics. This separation enables caching intermediate results and ensures consistent output schemas across all evaluations.

```
from dataclasses import dataclass
from tamperbench.whitebox.evals.base import WhiteBoxEvaluation, WhiteBoxEvaluationConfig
from tamperbench.whitebox.utils.names import EvalName, MetricName

@dataclass
class MyEvalConfig(WhiteBoxEvaluationConfig):
    pass

class MyEvaluation(WhiteBoxEvaluation[MyEvalConfig]):
    name = EvalName.MY_EVAL
    objective = MetricName.MY_METRIC

    def compute_inferences(self):
        # Load evaluation dataset, generate model outputs for each prompt
        # Returns DataFrame with columns: prompt, response
        ...
```

```
    def compute_scores(self, inferences):
        # Score each (prompt, response) pair
        # Returns DataFrame with columns: prompt, response, score
        ...

    def compute_results(self, scores):
        # Aggregate scores into summary metrics
        # Returns DataFrame with columns: metric_name, metric_value
        ...
```

Once registered, new evaluations can be invoked by any attack via the `evals` configuration parameter, and results automatically conform to the standardized output schema for downstream analysis.