Locking Open Weight Models with Spectral Deformation

Domenic Rosati [1 2] Sebastian Dionicio [1] Xijie Zeng [1 2] Subhabrata Majumdar [3] Frank Rudzicz [1 2]
Hassan Sajjad [1]

Abstract

Training open-weight foundation models for harmful purposes could be prevented if optimization was made arbitrarily slow. We find that loss landscape conditioning, which controls the convergence rate of Gradient Descent, can be modified using the spectral values of neural network *weight matrices alone* resulting in an efficient iterative algorithm (Spectral Deformation) that can arbitrarily slow down training such that it becomes infeasible. We call this process "model locking" and show across modalities that our lock prevents key high-risk open weight misuse : (1) unauthorized training (2) backdoor injection, and (3) relearning attacks after unlearning. Training locks present new possibilities for AI governance which we illustrate with policy analysis drawing on parallels from copyright protection technology and anti-circumvention law.

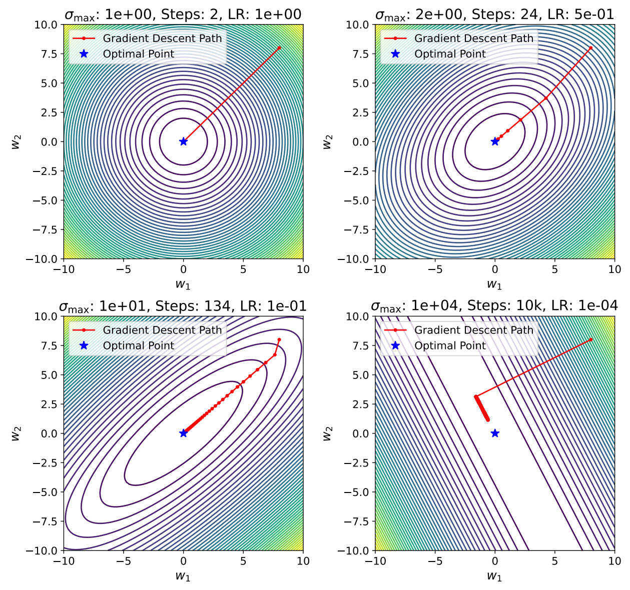

Figure 1. Steps needed for optimizing a convex quadratic function depend on $\kappa = \frac{\sigma_{\max}(H)}{\sigma_{\min}(H)}$ where H is it's Hessian; maximum learning rate that avoids divergence are used and $\sigma_{\min}$ fixed at 1.

1. Introduction

Misuse of open weight models is one of the most pressing risks of AI (Chan et al., 2023; Bengio et al., 2025). Regardless of how safe these models are before release, they can easily be trained for unsafe usage (Qi et al., 2023). Public discourse on open weight governance, as exemplified by a recent RAND brief (Nevo et al., 2024), consists of pro-open interventions that are designed to facilitate the continued release of open weight models such as through user licensing or enhanced liability legislation (Seger et al., 2023); or pro-closed interventions that seek to prevent or throttle model releases, for example, by regulating the release process of open weight foundation models (Anderljung et al., 2023).

There is an emerging alternative to the closed v. open dichotomy – the construction of immunized (Rosati et al., 2024b;a), self-destructing (Henderson et al., 2023), tamper-resistant models (Tamirisa et al., 2025), or domain-authorized models (c.f. Hong et al., 2025). Here, open weights can be released but, by technical means, they can only be trained towards safe and authorized ends.

Little work has been done outlining the policy and regulatory promise of these approaches. Existing work also provides no theoretical understanding of whether it is possible to prevent training and how that might be achieved. To address these two gaps, we present "model locking," which prevents models from being trained with *any* loss objective without degrading utility. Our contributions are as follows: (1) we provide a model locking method, **spectral deformation** (**SpecDef**) with a theoretical analysis that explains how locking works; (2) we present new approaches to governance of open models using inspiration from the enforcement of content scramble systems in DVDs for counterfeit prevention. In contrast to Tamirisa et al. (2024) and Rosati et al. (2024a), our method is grounded in a theoretical explanation of why convergence can be slowed. Additional those methods only lock a model to a particular data distribution while ours is a global lock against any training.

[1]Dalhousie University [2]Vector Institute [3]Vijil. Correspondence to: Domenic Rosati <domenic.rosati@dal.ca>.

Workshop on Technical AI Governance (TAIG) at ICML 2025, Vancouver, Canada. Copyright 2025 by the author(s).

2. Theoretical Analysis

Conceptually, we define a "model lock" as a mechanism that increases the number of necessary training steps t for convergence beyond the budget T that a given "lock picker" has. Under regularity assumptions such as strong convexity and smoothness, Nesterov (2013) showed that the optimal first-order (using gradient information) convergence rate is on the order of $\mathcal{O}(1/t^2)$. Formally, the optimal convergence rate (using Nesterov's acceleration) for the convex smooth function f and parameters θ_t and $\theta_* = \min_\theta f(\theta)$, is given by $f(\theta_0) - f(\theta_*) \leq \sigma_{\max}||\theta_0 - \theta_*||_2^2(1-\eta\sqrt{\kappa})^t$ where t is train iterations (Bach, 2024) and the learning rate η is set to $1/\sigma_{\max}$. The condition number $\kappa = \sigma_{\max}/\sigma_{\min}$ is a function of the largest $\sigma_{\max}(H)$ and smallest $\sigma_{\min}(H)$ singular values of the Hessian H, which is the Jacobian of the gradient of a training loss function.

Fixing training iterations and varying κ yields convergence on the order of $\mathcal{O}(\sqrt{\kappa})$. By increasing the largest singular value $\sigma_{\max}$ or decreasing the smallest singular value $\sigma_{\min}$, one can increase the training t steps needed for convergence – resulting in a "model lock." Observe that the the learning rate η can only be at most $2/\sigma_{\max}$ and therefore by increasing $\sigma_{\max}$, we can also control the maximum effectively learning rate used which further slows convergence.

Looser assumptions (non-convexity, stochastic gradients) result in slower convergence rates than the above thus, our results below still hold for modern deep neural networks (Bach, 2024). Figure 1 demonstrates this via control of convergence (by varying $\sigma_{\max}$) on a simple quadratic (Mean squared error) where the Hessian w.r.t the parameters W_i is $X^\top X$ for a Gaussian random input feature matrix X. As the maximum singular value is increased, the learning rate used must decrease otherwise GD results in divergence. Despite our above analysis, controlling the Hessian spectral values directly is intractable for foundation models so a more efficient approach is needed.

2.1. Control of Hessian Spectral Values

Let $\mathcal{M}$ be a N-layer deep Multi-Layer Perception (MLP) with the following structure $\hat{y} = W_{N+1}\phi(W_N\phi(W_{N-1}...\phi(W_1x)))$ where $\hat{y}$ is the output prediction, $W_i \in \mathbb{R}^{p\times q}$ are the weight matrices, and ϕ is an activation function such as the rectified linear unit (ReLU) $\phi(x) : x \mapsto \max(x, 0)$. Suppose a loss function that depends on $\hat{y}$ such as $\ell = f(\hat{y}, y)$, the Hessian of this loss function ℓ w.r.t the weights W is the Jacobian of the gradients of the loss w.r.t weights: $H = \left[\frac{\partial^2 \ell}{\partial W_i \partial W_j}\right]_{ij}^n$.

Theorem 2.1 (MLP Hessian $\sigma_{\max}$ lower bounded by weight matrix spectrum)**.** *The maximum singular value of the Hessian of the MLP $\sigma_{\max}(H)$ is lower bounded by the largest singular value $\sigma_{\max}(W_i)$ of any weight matrix multiplied by the factor $\sigma_{\min}(A)$. That is $\sigma_{\max}(H) \geq \sigma_{\max}(W_i)\sigma_{\min}(A)$ for any block of the Hessian with respect to weight matrices W_i, with A product terms in that block and $\sigma_{\min}(A)$ the smallest singular value of that product.*

A similar bound holds for a Transformer architecture with multiple attention heads.

Theorem 2.2 (Lower bound for Hessian $\sigma_{\max}$ of a Transformer)**.** *Consider a transformer architecture with L layers and M heads per layer (Gao et al., 2024). Given a squared error risk function, the maximum singular value of the Hessian H_T is bounded below by*

$$\sigma_{\max}(H_T) \geq \max_{l,j}\{\sigma_{\max}(W_{\theta,l,j})\sigma_{\min}(A_{\theta,l,j}), \\ \sigma_{\max}(W_{w,l,j})\sigma_{\min}(A_{w,l,j})\}.$$

where $W_{\theta,l,j} \in \{W_Q^{(l,j)}, W_K^{(l,j)}, W_V^{(l,j)}, W_O^{(l,j)}\}$ are attention weight matrices at layer $l \in \{1,\ldots,L\}$, head $j \in \{1,\ldots,M\}$; $W_{w,l,j} \in \{W_1^{(l,j)}, W_2^{(l,j)}\}$ are MLP weight matrices at layer l, head j ; and $A_{\theta,l,j}, A_{w,l,j}$ are product terms arising from the residual connections and layer interactions.

Full proofs are provided in Appendix A, including a tighter bound in Theorem 2.1 when the matrix A is rank-deficient. It follows from Theorems 2.1 and 2.2 that by controlling the largest singular values of weight matrices, one can control the convergence rate of gradient descent algorithms which are in $\mathcal{O}(\kappa)$ i.e. since the strength of the lock is proportional to $\sigma_{\max}(H)$ it is also proportional to $\sigma_{\max}(W_i)$.

We validate our claims numerically in Figure 2 using an MLP, CNN, and Transformer (see Appendix B.1) We find our bound holds in all cases (across three random seeds): the number of training iterations can directly be controlled.

Algorithm 1 Spectral Deformation Algorithm (**SpecDef**)

```
1: Model parameters $W_i$ where $i$ is a layer index, Retain
   loss $\ell_{retain}$, $\eta$ learning rate, $k$ and $\alpha$ hyper parameters.
2: $\sigma_m \leftarrow \{\}$
3: for $W_i \in \mathcal{W}$ do
4:     $\mathbf{U}, Diag(\mathbf{s}), \mathbf{V} \leftarrow SVD(W_i)$
5:     $\sigma_m \leftarrow \sigma_m \cup \text{top}_k(\mathbf{s})$
6: end for
7: $\mathcal{W} \leftarrow \mathcal{W} - \eta\nabla\left[\alpha \cdot \ell_{retain} - (1-\alpha) \cdot \frac{1}{n}\sum_i \sigma_m[i]\right]$
```

Figure 2 directly modifies spectral values through scaled reconstruction after SVD. Naive scaling could result in ruining the original safe behaviours of the model. We can maintain the original behaviour with a retain loss ℓ_{retain} representing the original training loss. We then add the sum

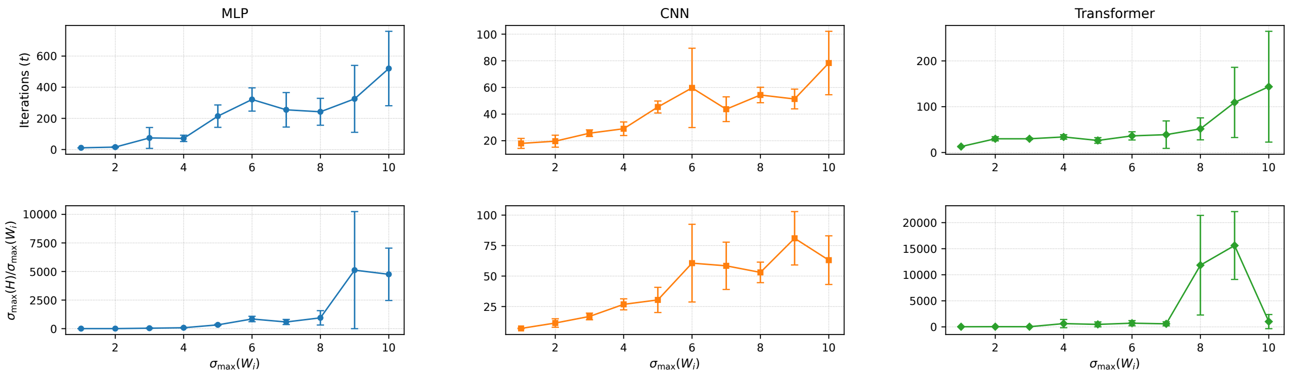

Figure 2. We validate Theorem 2.1 numerically on three architectures. **Top:** Number of training iterations for convergence is controlled by largest singular value of the weights. **Bottom:** Our bound holds in all cases (ratio is always above 1) validated with the true $\sigma_{\max}(H)$.

Model	**MNIST**	🙅‍♀️ **CIFAR**	🙅‍♀️ **KMNIST**	🙅‍♀️ **FASH**
ResNet18	99%	80%	80%	74%
🔒 SpecDef	99%	10%	10%	10%
	MRPC	🙅‍♀️ **RTE**	🙅‍♀️ **QQP**	🙅‍♀️ **QNLI**
DeBERTa	90%	69%	83%	85%
🔒 SpecDef	90%	52%	42%	45%
SmolLM2	85%	71%	71%	84%
🔒 SpecDef	84%	52%	20%	52%

Table 1. Locking prevents unauthorized training while maintaining utility on the original task. Evaluated with accuracy – MRPC and QQP with F1. Details provided in Appendix B.

of the top k singular values for each weight parameter matrix. Remark A.3 explains why top k is used rather than the maximum. This iterative algorithm (1) increases $\sigma_{\max}(W_i)$ while maintaining downstream utility.

3. Task Authorization

The primary goal of a model lock is to prevent training of an already fine-tuned model without degrading the model's performance on the original task. Table 1 illustrates this in a vision and language setting. We lock a ResNet18 model pretrained on the MNIST task. Before locking the model, it is easily finetuned for CIFAR10, KMNIST, and FMNIST (FASH) image classification tasks. Locking effectively prevents (10% is random accuracy) finetuning while maintaining original MNIST accuracy. For encoder-only `DeBERTa-v3-xsmall` and decoder-only `SmolLM2-135m-Instruct` we finetune for the MRPC task and attempt to further finetune for other binary classification tasks from the GLUE benchmark (Wang et al., 2018). Locking results in no increase in the original task objective over random performance.

4. Preventing Backdoor Injection

Aside from refusal training in LLMs, the *primary* way safety guards are implemented for vision, speech, and language is through either input- or output-based classifiers. The GPT4-o system card (Hurst et al., 2024) relies on safety classifiers for ensuring safety. Here we demonstrate how locking safety classifiers can prevent corruption via injection of backdoors.

Meta's Llama open weight release strategy has included providing open safety tools such as LlamaGuard (Inan et al., 2023) and PromptGuard (Chennabasappa et al., 2025). The community can train these models and improve them. However, since these models are vulnerable to the injection of backdoors, community improvements like Katanemo's "Arch-Guard" [1] built on PromptGuard could contain a backdoor trigger that was placed in a training dataset.

Table 2 demonstrates locking two sizes of PromptGuard2 (P_{G2}) which is trained for jailbreak classification. We are able to maintain performance on the Jailbreak prompt classification task while preventing the injection of triggers (ASR). When the attack is attempted the model degenerates resulting in an unusable model. We evaluate using the jailbreak classification dataset from Shen et al., 2024.

5. Robust Unlearning

One of the primary concerns about the utility of unlearning is that current methods are vulnerable to simple relearning attacks (Barez et al., 2025). We demonstrate class-based unlearning of the 'car' label from a ResNet18 model trained on CIFAR10 can be locked so as to prevent recovery of model capability through relearning attacks in Table 3.

Experimental details for all settings in this section are presented in Appendix B. Future work will study the efficacy

[1] `https://huggingface.co/katanemo/Arch-Guard`

Model	Pre-attack F1	Pre-attack ASR	Post-attack F1	Post-attack ASR
P_{G2} (22m)	94.8%	9.3%	90.6%	88.3%
🔒 SpecDef	100.0%	0.0%	0.0%	100.0%*
P_{G2} (86m)	98.3%	1.5%	99.4%	100.0%
🔒 SpecDef	95.0%	0.0%	0.0%	100.0%*

Table 2. Results from locking safety guard models (PromptGaurd 2) vulnerable to backdoor injection. The asterix indicates the backdoor attack wasn't successful since it causes divergence.

of locking for large scale generative models for speech, text, image, and video domains.

6. Implications for AI Governance

To analyze how training locks such as SpecDef might be operationalized in governance, we compare our method with technical copyright protection mechanisms.

6.1. Copyright Governance with Content Scramble

McPhie (2003) identified a common governance structure in copy protection across a variety of mediums (e.g., DVD, VHS) where a technical means of locking content is available: (1) **adoption:** content owners use the lock before distribution, (2) **protection** a third party enforces lock integrity, and (3) **legislation** legislation protects lock circumvention. We illustrate this with the content scramble system (CSS) which prevents copying of DVDs (Roemer, 2003). **Adoption**: CSS was attractive to content producers since they had commercial interest in preventing copying and unauthorized distribution of DVDs. Commercial interest drove the adoption of CSS for content owners selling DVDs, forcing manufacturers to adopt CSS decoding technology. For **protection**, a third-party consortium, the DVD Copy Control Association (DVD CCA), was formed to license CSS decoding technology to DVD player manufacturers. Since DVD CCA had control of the CSS decoding IP, they could litigate violators of that IP such as if the manufacturer failed to implement region locking controls. On a technical level, copy protection is relatively easy to circumvent due to the critical vulnerability of having to decrypt and play the content (Roemer, 2003). Therefore, the final piece of this dynamic is **legislation**, digital copyright legislation such as the DMCA introduced anti-circumvention that made it illegal to distribute information on copy protection circumvention or perform it.

6.2. Parallels for Training Locks

Adoption is different for AI training locks since there isn't a commercial incentive to prevent counterfeit models. Instead, the incentive to adopt training locks would come from public interest in mitigating the negative outcomes of AI such as the creation of deep fakes. Private companies may be incentivized to adopt training locks before release for legal protection in cases where they might be found liable for providing a tool that can be used to cause harm. For **protection**, a similar IP-ownership structure could give a third-party authority over how the locks are implemented, such as through litigation on patent infringement or license violation. Another way a third-party could govern is through the establishment and maintenance of standards which are mandated by government regulation. Finally, **legislation** can build on the precedent of anti-circumvention law to mandate that training locks are not circumvented and information thereof is not distributed.

7. Locked Weights versus Closed Access

If a model is locked against fine-tuning, what distinguishes it from closed access through an API only? Foremost, locked open weight models can still be freely distributed for inference at the cost of the model user. This means that the user does not have to pay the closed-access provider or deal with various inference time limitations. For example language model logits are used in calibrating uncertainty but they are not typically provided in the API response in order to prevent model distillation. Second, while we presented a global model lock, future locks could construct distribution-specific locks which would mean fine-tuning is possible for specific distributions.

8. Limitations and Future Work

The fundamental criticism of copy protection (Roemer, 2003) centres on the dictum that "bits can always be copied," but it is far from clear that AI models can always be trained given Section 2. Many negative impacts of copyright protection technology are based on the usage restrictions placed on consumers (EFF, 2014), restrictions designed to protect corporate interests. Digital locks for training do not prevent usage and distribution of models, but a similar criticism could be made: preventing training may stifle democratic, scientific, and commercial development of Artificial Intelligence. For example, this technology could be used to further entrench the hegemony of companies that can train foundation models. Training locks could result in rent extraction, where citizens are forced to pay for training which would otherwise be free to train without locks. Digital locks could be used for anti-democratic practices such as enforcing censorship (DeepSeek, Yang, 2025).

While our early empirical results still require a systematic study scaling up to more settings of concern (e.g., deep fakes, voice cloning, LLM agents), let us suppose that we were able to construct a lock that made the number of time

Model	Plane	🙅 Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
ResNet18	90.34	92.16	85.71	75.94	88.06	82.41	91.12	93.31	93.75	92.79
Unlearned	76.46	**0.00**	80.71	65.01	87.65	81.19	82.03	86.02	86.52	74.85
Relearning Attack	82.29	86.23	77.37	65.81	83.81	73.21	89.22	89.17	89.26	86.00
🔒 SpecDef	75.86	**0.00**	60.48	47.71	64.98	64.83	75.05	72.44	80.66	77.39
🔒 Relearning Attack	55.94	**0.00**	61.60	37.18	53.04	61.15	72.30	72.83	69.53	90.83

Table 3. Per-class accuracy on label unlearning. Our lock successfully prevents relearning attacks resulting in "robust" unlearning.

steps to train extremely expensive. In this case, digital locks could still be circumvented through a variety of means, for example the unlocked weights of the model could be leaked or stolen, the model logits might be used to distill teachable student models, and preconditioners could be computed that undo the locks. Recent work has shown that attempts to secure open weights are brittle and easy to overcome (**?**). In addition, inference-time attacks, such as jailbreaks and adversarial attacks, could still be successful (Rosati et al., 2024a). This points to the need for a "swiss-cheese" model of defence where digital locks need to be paired with other defence methods.

To address this limitation from a governance perspective, we again draw on copy protection. Roemer (2003) pointed out that copy protection technologists positioned their defences for a "napsterization" threat model to mitigate organized counterfeit. Instead, copy protection, on a technical level, actually protected against a threat model of everyday consumer counterfeit. Strongly motivated attackers could always (and currently do for games and content protected under modern digital rights management tools) circumvent defences. However, it is the legislative and regulatory apparatus of digital locks that allowed the development of anti-circumvention law that can be used to protect against the "napsterization" threat model by litigating violators. Parallels could be made with digital locks for training where their existence can introduce new means for AI governance without themselves having to be implemented perfectly.

9. Conclusion

The governance scenario outlined above is meant to illustrate the new policy possibilities enabled by the existence of digital lock technology which results in breaking the dichotomy between open and closed source in discussions of AI Safety. Despite its promise, digital lock technology for AI training is far from mature.

Our workshop paper provides promising early theoretical and empirical results but they need to be extended with systematic evaluation (e.g. Figure 3) of common risk scenarios of large scale generative models as well as full ablations over first order optimizations that might be used (e.g. Figure 4). On the advice of Tramer et al. (2020), it is necessary to develop a robust framework of adaptive attacks on training locks, we evaluated preconditioning in Appendix C as an early start to properly assess these models. Further theoretical development such as understanding the impact of spectral deformation on generalization, adversarial robustness, and general downstream utility remains important. Finally, it isn't clear yet how to predict how many more training steps will be needed as a result of SpecDef.

Acknowledgements

We would like to thank the Killam foundation, Digital Research Alliance of Canada, and Vector Institute for their continued support of the authors of the paper. We would also like to acknowledge the Centre for AI Safety who provided compute resources used in this paper.

References

Anderljung, M., Barnhart, J., Korinek, A., Leung, J., O'Keefe, C., Whittlestone, J., Avin, S., Brundage, M., Bullock, J., Cass-Beggs, D., et al. Frontier ai regulation: Managing emerging risks to public safety. *arXiv preprint arXiv:2307.03718*, 2023.

Bach, F. *Learning theory from first principles*. MIT press, 2024.

Barez, F., Fu, T., Prabhu, A., Casper, S., Sanyal, A., Bibi, A., O'Gara, A., Kirk, R., Bucknall, B., Fist, T., et al. Open problems in machine unlearning for ai safety. *arXiv preprint arXiv:2501.04952*, 2025.

Bengio, Y., Mindermann, S., and Privitera, D. International ai safety report 2025. 2025.

Bhatia, R. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

Chan, A., Bucknall, B., Bradley, H., and Krueger, D. Hazards from increasingly accessible fine-tuning of downloadable foundation models. *arXiv preprint arXiv:2312.14751*, 2023.

Chennabasappa, S., Nikolaidis, C., Song, D., Ding, S., Wan, S., Chaturvedi, R., Crnkovich, J., de Paola, B., Deason, L., Doucette, N., Gabi, D., Gampa, A., He, K., Molnar, D., Montilla, A., Testud, J.-C., Whitman, S., and Saxe, J. Llamafirewall: An open source guardrail system for building secure ai agents, April 2025. https://llamaguard.meta.com/llamafirewall.

EFF. Unintended consequences: Sixteen years under the dmca. 2014.

Gao, C., Cao, Y., Li, Z., He, Y., Wang, M., Liu, H., Klusowski, J. M., and Fan, J. Global convergence in training large-scale transformers. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 29213–29284. Curran Associates, Inc., 2024.

Guo, T., Hu, W., Mei, S., Wang, H., Xiong, C., Savarese, S., and Bai, Y. How do transformers learn in-context beyond simple functions? a case study on learning with representations, 2023. URL https://arxiv.org/abs/2310.10616.

Henderson, P., Mitchell, E., Manning, C., Jurafsky, D., and Finn, C. 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*, pp. 287–296, 2023.

Hong, Z., Xiang, Y., and Liu, T. Toward robust non-transferable learning: A survey and benchmark. *arXiv preprint arXiv:2502.13593*, 2025.

Horn, R. A. and Johnson, C. R. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., Mao, Y., Tontchev, M., Hu, Q., Fuller, B., Testuggine, D., and Khabsa, M. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature, 2020. URL https://arxiv.org/abs/1503.05671.

McPhie, D. Access made accessible shaping the laws and technologies that protect creative works. *J. Copyright Soc'y USA*, 51:521, 2003.

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

Nesterov, Y. *Introductory lectures on convex optimization:*. 2013.

Nevo, S., Lahav, D., Karpur, A., Bar-On, Y., Bradley, H.-A., and Alstott, J. *Securing AI model weights: Preventing theft and misuse of frontier models*. Number 1. Rand Corporation, 2024.

Pooladzandi, O. and Li, X.-L. Curvature-informed sgd via general purpose lie-group preconditioners. *arXiv preprint arXiv:2402.04553*, 2024.

Qi, X., Zeng, Y., Xie, T., Chen, P.-Y., Jia, R., Mittal, P., and Henderson, P. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Roemer, R. Locking down loose bits: trusted computing, digital rights management, and the fight for copyright control on your computer. *UCLA JL & Tech.*, 7:i, 2003.

Rosati, D., Wehner, J., Williams, K., Bartoszcze, L., Gonzales, R., Majumdar, S., Sajjad, H., Rudzicz, F., et al. Representation noising: A defence mechanism against harmful finetuning. *Advances in Neural Information Processing Systems*, 37:12636–12676, 2024a.

Rosati, D., Wehner, J., Williams, K., Bartoszcze, L., Sajjad, H., and Rudzicz, F. Immunization against harmful fine-tuning attacks. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5234–5247, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.301. URL https://aclanthology.org/2024.findings-emnlp.301/.

Seger, E., Dreksler, N., Moulange, R., Dardaman, E., Schuett, J., Wei, K., Winter, C., Arnold, M., hÉigeartaigh, S. Ó., Korinek, A., et al. Open-sourcing highly capable foundation models: An evaluation of risks, benefits, and alternative methods for pursuing open-source objectives. *arXiv preprint arXiv:2311.09227*, 2023.

Shen, X., Chen, Z., Backes, M., Shen, Y., and Zhang, Y. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on*

Computer and Communications Security, pp. 1671–1685, 2024.

Tamirisa, R., Bharathi, B., Zhou, A., Li, B., and Mazeika, M. Toward robust unlearning for LLMs. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024. URL https://openreview.net/forum?id=4rPzaUF6Ej.

Tamirisa, R., Bharathi, B., Phan, L., Zhou, A., Gatti, A., Suresh, T., Lin, M., Wang, J., Wang, R., Arel, R., Zou, A., Song, D., Li, B., Hendrycks, D., and Mazeika, M. Tamper-resistant safeguards for open-weight llms, 2025. URL https://arxiv.org/abs/2408.00761.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems*, 33:1633–1645, 2020.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Yang, Z. Here's How DeepSeek Censorship Actually Works—and How to Get Around It. *Wired*, 2025. ISSN 1059-1028. URL https://www.wired.com/story/deepseek-censorship/. Section: tags.

A. Mathematical Details

In this section we provide a proof of Theorem 2.1 in the text. For this proof we state two lemmas. The first is a well known result from (Horn & Johnson, 1991).

Lemma A.1 (Matrix product $\sigma_{\max}$ lower bounds)**.** *For a product of matrices, we have the following lower bounds for the maximum singular values:*

$$\sigma_{\max}(W_i W_j) \geq \sigma_{\max}(W_i)\sigma_{\min}(W_j)$$

The second is an inequality needed to lower bound the maximum singular values of a block matrix.

Lemma A.2. *Given a block matrix*

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

we have

$$\sigma_{\max}(M) \geq \max\left\{\sigma_{\max}(A), \sigma_{\max}(B), \sigma_{\max}(C), \sigma_{\max}(D)\right\}$$

Proof. Recall that the largest singular value is equal to the spectral norm of a matrix defined by:

$$\sigma_{\max}(X) = \max_{||v||=1} ||Xv|| = ||X||_2.$$

We will show that for each block E, there exists the following inequality:

$$||M||_2 \geq ||M\hat{v}|| \geq ||Ev||$$

where $||M||_2 = \sigma_{\max}(M)$ and $||Ev|| = \sigma_{\max}(E)$, $v \in \mathbb{R}^n$ is the vector in the spectral norm as above and $\hat{v}$ is an embedded vector such that $M\hat{v}$ contains Ev. To illustrate this consider the block A for which we have the embedded vector

$$\hat{v} = \begin{bmatrix} v \\ 0 \end{bmatrix}.$$

Since:

$$M\hat{v} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} Av \\ Cv \end{bmatrix},$$

this applies for all blocks.

Now observe that for each block we have the norm $||\cdot||^2$ such that $||M\hat{v}||^2 = ||Ev||^2 + ||E'v||^2$ where E' is the opposite block matrix as we saw with C in the example with block A.

By this norm we have by that $||M\hat{v}||^2 \geq ||Ev||^2$ and taking the square root we can show that we have the first part of the inequality we wanted to show above: $||M\hat{v}|| \geq ||Ev|| = \sigma_{\max}(E)$. What remains is to show that $\sigma_{\max}(M)$ is greater or equal to this.

By definition of the spectral norm,

$$\sigma_{\max}(M) = \max_{||u||=1} ||Mu||.$$

Since the embedding vector has a zero vector, we will always have:

$$\sigma_{\max}(M) \geq ||M\hat{v}|| \geq \sigma_{\max}(E)$$

.

Finally, observe that considering the inequality for each block we have

$$\sigma_{\max}(M) \geq \max\left\{\sigma_{\max}(A), \sigma_{\max}(B), \sigma_{\max}(C), \sigma_{\max}(D)\right\}$$

which is what we set out to show. □

A.1. Proof of Theorem 2.1

Proof. By Lemma 1, it is sufficient to show just one block of the Hessian of a loss function depends on products of individual weight matrices. Recall that for a two-layer MLP composed of ReLU activations ($D_z = \text{diag}(\mathbf{1}_{z>0}); z = W_1 x$ with Mean Squared Error we have the following Hessian block:

$$\frac{\partial^2 \mathcal{L}}{\partial^2 W_1} = D_z W_2^\top W_2 D_z \otimes xx^\top$$

Since this is clearly a product of the weight matrix W_2, we simply apply Lemma 1 to show the following lower bound:

$$\begin{aligned}
&\sigma_{\max}(D_z W_2^\top W_2 D_z \otimes xx^\top) \\
&= \sigma_{\max}(D_z(W_2^\top W_2 D_z))\sigma_{\max}(xx^\top) && \text{(Kronecker } \sigma_i \text{ Identity)} \\
&\geq \sigma_{\max}((W_2^\top W_2 D_z)^\top)\sigma_{\min}(D_z)\sigma_{\max}(xx^\top) && \text{(Lem 2; Transpose)} \\
&\geq \sigma_{\max}(W_2^\top)\sigma_{\min}(W_2 D_z)\sigma_{\min}(D_z)\sigma_{\max}(xx^\top) && \text{(Lem 2; Transpose)}
\end{aligned}$$

The Hessian is composed of more product terms from weight matrices as layers are added. This includes adding convolution ($\mathbf{CX}$) and attention layers $(\mathbf{QK})^\top \mathbf{V}$ which is sufficient to show that our bound holds. □

Remark A.3 (**Tight bounds under rank deficiency**). Under activation functions like RELU we have rank deficiency ($\sigma_{\min} = 0$). Resulting in a vacuous lower bound. We also draw on the following result from Bhatia (2013):

$$\sum_{i=1}^{k} \sigma_i(W_i W_j) \geq \sum_{i=1}^{k} \sigma_i(W_i)\sigma_{n-i+1}(W_j),$$

where the singular values are ordered from largest to smallest in the index i.

We guarantee that as long as we have k sufficiently large singular values for rank $N - k$ matrices then we have a non-zero lower bound. This is why we use the top k singular values in Agorithm 1.

A.2. Proof of Theorem 2.2

We start with a formal definition of a transformer architecture with multiple attention heads following the framework of Gao et al. (2024). We assume a Transformer model composed of L sequential transformer layer, with M heads per layer. For input sequence $H \in \mathbb{R}^{D\times(N+1)}$ consisting of $N + 1$ tokens each with dimension D, the Transformer processes data through alternating self-attention and feedforward layers that comprise one layer.

Self-Attention Layer: Each residual self-attention layer is represented by

$$\text{Attn}_{\theta_1,\ldots,\theta_M}(Z, \eta) = Z + \eta M^{-1} \sum_{j=1}^{M} f(Z, \theta_j) \tag{1}$$

Feed-Forward Layer: Each residual feed-forward neural network layer is defined by

$$\text{MLP}_{w_1,\ldots,w_M}(Z, \eta) = Z + \eta M^{-1} \sum_{j=1}^{M} h(Z, w_j) \tag{2}$$

where $\eta = \Delta t/2$ with $\Delta t = 1/L$ is the residual step size.

Encoder Functions: The self-attention encoder $f : \mathbb{R}^{D\times(N+1)} \to \mathbb{R}^{D\times(N+1)}$ follows the dot-product attention mechanism:

$$f(Z, \theta) = W_O W_V Z \sigma_A \left[(W_K Z)^T W_Q Z\right] \tag{3}$$

where $W_V, W_K, W_Q \in \mathbb{R}^{s\times D}$, $W_O \in \mathbb{R}^{D\times s}$, $\theta = \text{vec}[W_Q, W_K, W_V, W_O]$, and σ_A is column-wise softmax.

The feed-forward encoder $h : \mathbb{R}^{D\times(N+1)} \to \mathbb{R}^{D\times(N+1)}$ is defined as:

$$h(Z, w) = W_2 \sigma_M(W_1 Z) \tag{4}$$

where $w = \text{vec}[W_1, W_2]$ and σ_M is component-wise activation (e.g., ReLU).

Layer Composition: The Transformer network with L layers follows:

$$T^{\Theta}(H, t + \Delta t/2) = \text{Attn}_{\theta_{t,1},\ldots,\theta_{t,M}}(T^{\Theta}(H,t), \Delta t/2) \tag{5}$$

$$T^{\Theta}(H, t + \Delta t) = \text{MLP}_{w_{t,1},\ldots,w_{t,M}}(T^{\Theta}(H,t+\Delta t/2), \Delta t/2) \tag{6}$$

for $t = 0, \Delta t, \ldots, (L-1)\Delta t$ with $T^{\Theta}(H, 0) = H$, with $\Theta = \{\theta_{t,j}, w_{t,j}\}_{t,j}$ denoting all parameters in the Transformer model.

Finally, we define the squared error loss function

$$\mathcal{L} = \frac{1}{2}\left(\text{Read}[T^{\Theta}(H,1)] - y(H)\right)^2,$$

where Read$[\cdot]$ is a read-out function that extracts the next token prediction probabilities (Guo et al., 2023; Gao et al., 2024).

We are now in a position to prove Theorem 2.2.

Proof. The Hessian with respect to all parameters $\Theta = \{\theta_{l,j}, w_{l,j}\}_{l,j}$ can be written as

$$H_T = \begin{bmatrix} H_{\theta\theta} & H_{\theta w} \\ H_{w\theta} & H_{ww} \end{bmatrix},$$

where $H_{\theta\theta}$ and H_{ww} contain second derivatives with respect to attention and MLP parameters respectively, while $H_{\theta w}, H_{w\theta}$ contain cross-derivatives.

Given these blocks, by Lemma A.2 we have

$$\sigma_{\max}(H_T) \geq \max\{\sigma_{\max}(H_{\theta\theta}), \sigma_{\max}(H_{ww}), \sigma_{\max}(H_{\theta w}), \sigma_{\max}(H_{w\theta})\}. \tag{7}$$

We now obtain upper bounds for maximum eigenvalues of the first two blocks.

For attention parameters at layer l, head j, consider the second derivative:

$$\frac{\partial^2 \mathcal{L}}{\partial \theta_{l,j}^2} = \frac{\partial}{\partial \theta_{l,j}}\left[\frac{\partial \mathcal{L}}{\partial T^{\Theta}(H,1)} \frac{\partial T^{\Theta}(H,1)}{\partial \theta_{l,j}}\right]$$

The derivative $\frac{\partial T^{\Theta}(H,1)}{\partial \theta_{l,j}}$ involves the product

$$\frac{\partial T^{\Theta}(H,1)}{\partial \theta_{l,j}} = \prod_{k=l+1}^{L} \frac{\partial T^{\Theta}(H, k\Delta t)}{\partial T^{\Theta}(H,(k-1)\Delta t)} \cdot \frac{\eta}{2M} \frac{\partial f(T^{\Theta}(H, l\Delta t), \theta_{l,j})}{\partial \theta_{l,j}}.$$

Combining (5) and (6) we have $T^{\Theta}(H,(k+1)\Delta t) = T^{\Theta}(H, k\Delta t) + \frac{\eta}{2M}\sum_j f(\cdot) + \frac{\eta}{2M}\sum_j h(\cdot)$, so that

$$\begin{aligned}\mathbf{A}_k := \frac{\partial T^{\Theta}(H,(k+1)\Delta t)}{\partial T^{\Theta}(H,k\Delta t)} &= I + \frac{\eta}{2M}\sum_{j=1}^{M} \frac{\partial f(T^{\Theta}(H,k\Delta t), \theta_{k,j})}{\partial T^{\Theta}(H,k\Delta t)} \\ &\quad + \frac{\eta}{2M}\sum_{j=1}^{M} \frac{\partial h(T^{\Theta}(H, k\Delta t + \eta/2), w_{k,j})}{\partial T^{\Theta}(H,k\Delta t)}.\end{aligned}$$

Thus, elements of the Hessian block $H_{\theta\theta}$ have the structure

$$\frac{\partial^2 \mathcal{L}}{\partial \theta_{l,j}^2} = \left(\frac{\partial \mathcal{L}}{\partial T^{\Theta}(H,1)}\right) \prod_{k=l+1}^{L} \mathbf{A}_k \cdot \frac{\eta^2}{4M^2} \frac{\partial^2 f(T^{\Theta}(H, l\Delta t), \theta_{l,j})}{\partial \theta_{l,j}^2} + \text{first-order terms.} \tag{8}$$

From equation (8), each diagonal element of $H_{\theta\theta}$ can be written as a product involving the attention weight matrices. Recall that self-attention $f(Z,\theta) = W_O W_V Z \sigma_A[(W_K Z)^T W_Q Z]$ with $\theta = \text{vec}[W_Q, W_K, W_V, W_O]$ and $Z := T^{\Theta}(H, l\Delta t)$. Hence, for the W_Q block:

$$\frac{\partial^2 \mathcal{L}}{\partial W_Q^2} = \mathbf{P}_Q \cdot \mathbf{W}_Q \cdot \mathbf{S}_Q,$$

where

- $\mathbf{P}_Q = \left(\frac{\partial \mathcal{L}}{\partial T^{\Theta}(H,1)}\right) \prod_{k=l+1}^{L} \mathbf{A}_k \frac{\eta^2}{4M^2} W_O W_V Z$ contains the backpropagation terms,
- $\mathbf{W}_Q$ represents the W_Q contribution from $\frac{\partial^2 \sigma_A}{\partial W_Q^2}$,
- $\mathbf{S}_Q = Z^T \otimes Z$ contains the remaining activation terms.

By Lemma A.2,

$$\sigma_{\max}(\mathbf{P}_Q \cdot \mathbf{W}_Q \cdot \mathbf{S}_Q) \geq \sigma_{\max}(\mathbf{W}_Q)\sigma_{\min}(\mathbf{P}_Q)\sigma_{\min}(\mathbf{S}_Q).$$

Since $\frac{\partial^2 \sigma_A}{\partial W_Q^2}$ involves W_Q directly through the softmax second derivative, we have $\sigma_{\max}(\mathbf{W}_Q) \geq C_\sigma \sigma_{\max}(W_Q)$ for some constant $C_\sigma > 0$ depending on the softmax structure.

Therefore

$$\sigma_{\max}\left(\frac{\partial^2 \mathcal{L}}{\partial W_Q^2}\right) \geq \sigma_{\max}(W_Q)\sigma_{\min}(A_{\theta,Q}), \tag{9}$$

where $A_{\theta,Q} = \mathbf{P}_Q \cdot \mathbf{S}_Q$ contains all product terms except W_Q.

A similar result holds for the W_K block. For W_V and W_O, which appear linearly in $f(Z,\theta) = W_O W_V Z \sigma_A[\cdot]$, their pure second derivatives vanish:

$$\frac{\partial^2 f}{\partial W_V^2} = 0, \quad \frac{\partial^2 f}{\partial W_O^2} = 0.$$

However, the mixed derivatives do contain these weight matrices.

$$\frac{\partial^2 f}{\partial W_V \partial W_Q} = \frac{\partial}{\partial W_V}\left[W_O W_V Z \frac{\partial \sigma_A}{\partial W_Q}[(W_K Z)^T W_Q Z]\right] = W_O Z \frac{\partial \sigma_A}{\partial W_Q}[(W_K Z)^T W_Q Z].$$

The corresponding Hessian entry is

$$\frac{\partial^2 \mathcal{L}}{\partial W_V \partial W_Q} = \left(\frac{\partial \mathcal{L}}{\partial T^{\Theta}(H,1)}\right) \prod_{k=l+1}^{L} \mathbf{A}_k \frac{\eta^2}{4M^2} W_O Z \frac{\partial \sigma_A}{\partial W_Q},$$

giving the bound, for constant $C_V > 0$,

$$\sigma_{\max}\left(\frac{\partial^2 \mathcal{L}}{\partial W_V \partial W_Q}\right) \geq C_V \sigma_{\max}(W_V)\|Z\| \left\|\frac{\partial \sigma_A}{\partial W_Q}\right\|. \tag{10}$$

Repeating (9) for W_K and (10) for W_O respectively, then applying Lemma A.2, we get

$$\sigma_{\max}(H_{\theta\theta}) \geq \max_{l,j} \left\{\sigma_{\max}(W_{\theta,l,j}), \sigma_{\min}(A_{\theta,l,j})\right\}. \tag{11}$$

For the MLP block H_{ww}, following identical reasoning for feedforward layers

$$\sigma_{\max}(H_{ww}) \geq \max_{l,j} \left\{\sigma_{\max}(W_{w,l,j})\sigma_{\min}(A_{w,l,j})\right\}. \tag{12}$$

Combining (11) and (12), then applying (7) concludes our proof.

□

Remark A.4. We note that tightness of the bound in Theorem 2.2 depends on the constants in $A_{\theta,l,j}$ and $A_{w,l,j}$, which depend on the following factors

- The residual connection strength $\eta = \frac{1}{2L}$,
- The head averaging factor $\frac{1}{M}$,
- The activation function properties (softmax for attention, ReLU/GELU for MLP),
- The layer depth and position within the network.

B. Experimental Details

B.1. Figure 2

Figure 2 is trained using Gaussian random input data (0 mean, unit covariance) (sample size 64) with a dimension of 4 and Guassian random targets. The MLP is a four layer mlp with hidden dimension of size 8 using ReLU activations. The CNN consists of a two layer CNN with kernel size 3 and padding of 1, there is a two layer MLP on top with a hidden dimension of 8 the MLP uses Tanh activations. The transformer is an encoder only single transformer block using a sequence length of two and embedding dimension of 4. The hidden dimension of the MLPs are also 8. All models use Xavier uniform initialization. A mean squared error loss is used for all model. We use Adam with a learning rate of 0.001 for iteration plots.

B.2. Table 1

For the image classification experiments we train a ResNet18 model which is not pretrained. We use 10 epochs on the MNIST train set to construct the original model we would like to lock. We use AdaDelta to train the model using a learning rate of 1 and linear step learning rate scheduler $\gamma = 0.7$. We use this exact same set up to lock the model using the MNIST train set as the retain loss.

For CIFAR, KMNIST, and FASH (FMNIST), we perform the downstream training with Adam with weight decay with a learning rate of 0.1 for 10 epochs, we also evaluated with AdaDelta and SGD (with momentum) but choose to highlight Adam since it has a preconditioning effect (see Appendix C). We chose these data sets since they were all 10 label classification tasks. A batch size of 32 is used. α is set to 0.5 in our spectral deformation algorithm. A top k of 1 is used for these experiments.

For the text classification tasks, we use the binary classification tasks from the GLUE benchmark (Wang et al., 2018). We use Adam with weight decay of 0.01 with a linear scheduler with no warm up using a learning rate of 5×-10^{-5}. We use Huggingface evaluation tools which reports accuracy for each task except for QQP and MRPC which uses F1 scores which we use in the table. We train for 3 epochs and limit the train sets to at most 5k samples. We use the validation splits of the dataset as the test splits are not available. A training batch size of 8 is used. We report scores that are an average of three seeds randomly shuffling the dataset each time. Our spectral deformation is applied for 10 epochs with MRPC task loss and a kl divergence with an original model as retain loss. α is set to 0.99. A top k of 2 is used for these experiments. We use both encoder-only `DeBERTa-v3-xsmall` and decoder-only `smol2-135m-instruct` for our models, a linear layer is added as the final layer for the classification task.

In order to further validate the robustness of our lock to varying learning rates, we vary the learning rate from 1×-10^{-3} to 1×-10^{-8}. For all learning rates (Figure 3, our locked model does not converge after 3 epochs on the recognizing textual entailment (rte) task. Notice that there is only really a very small window in which `DeBERTa-v3-xsmall` can learn (between 1×-10^{-3} and 1×-10^{-6}).

B.3. Table 2

The prompt guard backdoor injection experiments use a backdoor of the unicode characters for pentagram snowman pentagram that is prefixed before the prompt. When these are present the label is set to "safe." We add these to the dataset of 2176 jailbreaks found in the wild from Shen et al. (2024) which are labeled as "unsafe" and balance the dataset using 2000 benign prompts from OpenOrca. (Mukherjee et al., 2023). We split this into a dataset with 80/20 for train and test. We inject the backdoor using Adam with weight decay (0.01) with a learning rate of 1e-4 and a batch size of 8 for 1 epoch. We use spectral deformation for three epochs using an $\alpha = 0.9$ and using kl divergence with original prompt guard 2 model as well as the jailbreak classification train set we constructed above (without the backdoor). We use both the 22 million and 86 million variants of prompt guard 2.

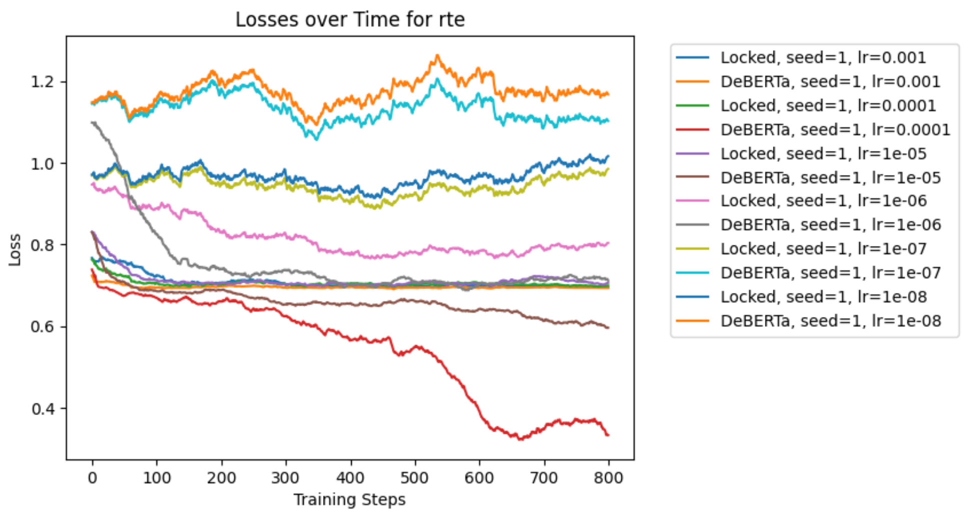

Figure 3. Our lock is effective across varying learning rates.

B.4. Table 3

For the unlearning setting, we use the pretrained resnet18 model weights on cifar10 from the NeurIPs 2023 machine unlearning competition.[2] We use the unlearning baseline from that task that simply continues to train the model on CIFAR10 without any data for the car label and report the per class accuracy. The relearning attack uses stochastic gradient decent for 1 epoch with a learning rate of 0.1. Our spectral deformation algorithm uses a retain loss of the cifar10 classification task with all car labels removed with $\alpha = 0.5$. We perform spectral deformation for 10 epoch using Adam as an optimizer and a per epoch step learning rate scheduler with $\gamma = 0.7$.

C. Preconditioning Adaptive Attack Experiments

What kind of adaptive attacks can be constructed for our locking mechanism? If an attacker knew the model was defended with a spectral deformation that "ill conditions" the loss landscape then the attacker could attempt preconditioning.

Preconditioning techniques like Martens & Grosse (2015) are used to counter the effects of an ill conditioned loss landscape. In the extreme case, finding the inverse hessian H^{-1} allows us to produce a condition number that is 1 since $H^{-1}H = I$. Except for the smallest neural networks, computing the inverse hessian is infeasible. Instead, preconditioning techniques typically use approximation to improve the condition number of the loss landscape. Examples of these are Kronecker Factorization (Kron) (Martens & Grosse, 2020), Diagonal (or Cross Diagonal) Preconditioning (XMat) (Pooladzandi & Li, 2024), and Adam (AdamW is Adam with weight decay) (Kingma & Ba, 2017). Due to the computational demands of these methods, we only evaluate these on a four layer MLP on top of a two layer CNN using the same setting from Table 1 where we train the model for 10 epochs on MNIST, run 10 epochs of SpecDef, and then attempt to train on CIFAR 10 for 10 epochs. We select the the largest learning rate that allowed training with a baseline unlocked model which was 0.1 for SGD and AdamW, 0.001 for Kron and 0.01 for XMat. We use the implementations of Kronecker Factorization (Kron) and Cross Diagonal Approximate Precondition (XMat) from the preconditioned stochastic gradient descent (PSGD) library (Pooladzandi & Li, 2024). The preliminary results in Figure C shows that in our method is also able to prevent unlocking using approximate preconditioners showing that it might be the case that full preconditioning is necessary to undo the lock mechanism. We evaluated two other second order methods LBFGs and Natural Gradients but neither was effective likely due to the poor conditioning of the Jacobians of the weights w.r.t the outputs which also would be poorly conditioned by our technique. Further experiments are needed on large scale models and analysis on how our method interacts mathematically with various preconditions is our future work.

[2]https://unlearning-challenge.github.io/

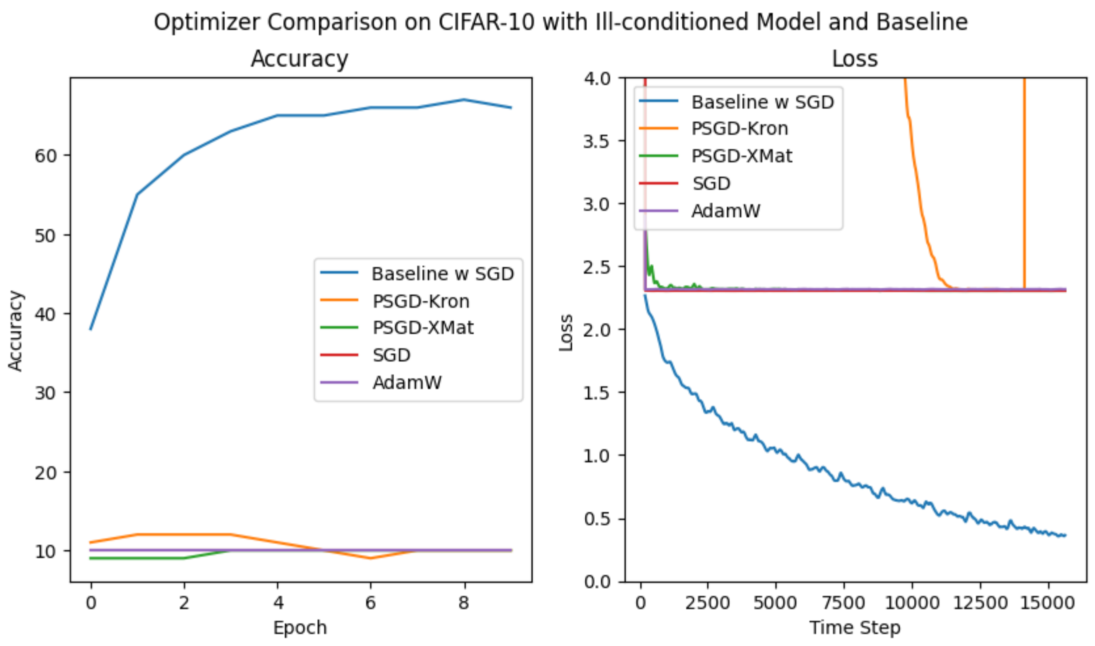

Figure 4. Spectral deformation doesn't degrade locked MNIST classifier performance ($99\% \rightarrow 99\%$) and prevents training on an unauthorized task across various preconditioning methods.