Localizing latent mechanisms in weight space by spiking the training data

Johnny, Gustavo, Jerry, Yanai, Robin

September 2026

1 Introduction

The worst case scenarios of AI likely involve the use of *latent mechanisms* [9]. We define latent mechanisms as those used in the model's internal computation, yet their use is hidden and cannot be identified from model outputs alone. For instance, when a model answers a benchmark example incorrectly, whether the model had made a mistake or answered deceptively cannot be determined without looking at the model internals. Memorization, evaluation awareness, and deception are all safety relevant and their mechanisms can be latent [3, 13, 17]. We urgently lack the tools to detect and suppress latent mechanisms [10], and this report proposes a principled localization method.

The localization of single facts is well studied, and the same methods used to localize are also useful for making factual edits within the model [12]. Localizing latent mechanisms presents more challenges. We show that real headway can be made on this problem by *spiking* the model's training data [20, 21]. Spiking generalizes canaries beyond their typical use case of measuring privacy leakage [18]. By carefully designing and spiking canaries into training to induce certain failure modes, we can later trace back these failures to the weight space with influence functions. In this view, spiking embeds a randomized experiment in the model to identify a latent mechanism of interest.

2 Influence functions on spiked data

Our derivation is general but we will first consider memorization. In Hubble, we have perturbed models spiked with different data (books, biographies, and test sets) at random [19]. For the perturbed model, let $L_1(\theta)$ denote the expected loss on spiked members and $L_0(\theta)$ the expected loss on distributionally matched nonmembers. The counterfactual memorization [22, 4] is

$$M(\theta) = L_0(\theta) - L_1(\theta) \tag{1}$$

and this difference isolates the reduction in loss due to memorization and removes any variation attributable to the examples themselves. This difference is

also called an identification and is a fundamental component of causal analysis [1]. Let $J(\theta)$ denote the original training objective and let

$$\hat{\theta} = \arg\min_{\theta} J(\theta) \tag{2}$$

be the trained parameters. Our goal is to find a direction such that moving the weights in this direction would steeply reduce $M(\theta)$ while preserving capabilities. We can apply influence functions to find this direction [8], and using it for localization is novel. To measure the influence of memorization on the model weights, consider the perturbing the objective by M:

$$\hat{\theta}_\epsilon = \arg\min_{\theta} \left[J(\theta) + \epsilon M(\theta)\right] \tag{3}$$

where setting $\epsilon = 0$ would recover $\hat{\theta}$. By assumption, the first-order optimality condition gives us

$$\nabla_\theta J(\hat{\theta}_\epsilon) + \epsilon \nabla_\theta M(\hat{\theta}_\epsilon) = 0 \tag{4}$$

and differentiating this expression with respect to ϵ at $\epsilon = 0$ gives

$$H \left.\frac{d\hat{\theta}_\epsilon}{d\epsilon}\right|_{\epsilon=0} + \nabla_\theta M(\hat{\theta}) = 0 \tag{5}$$

where $H = \nabla^2_\theta J(\hat{\theta})$ is the Hessian of the training objective at the learned parameters. With

$$\nabla_\theta M(\theta) = \nabla_\theta L_0(\theta) - \nabla_\theta L_1(\theta) \tag{6}$$

the influence function is therefore

$$\boxed{\left.\frac{d\hat{\theta}_\epsilon}{d\epsilon}\right|_{\epsilon=0} = -H^{-1}\left(\nabla_\theta L_0(\hat{\theta}) - \nabla_\theta L_1(\hat{\theta})\right)}. \tag{7}$$

This tells us, at our current $\hat{\theta}$, how the weights would move if we started to penalize memorization in the objective function. Alternatively, it is also a curvature adjusted step to minimize the combined objective.

Suppression. Moving the parameters by a small amount along the influence direction $\Delta\theta = H^{-1}\nabla_\theta M(\hat{\theta})$ suppresses memorization. For a step size $\eta > 0$

$$\theta_{\text{sup}} = \hat{\theta} - \eta\Delta\theta \tag{8}$$

the resulting model θ_{sup} reduces memorization while minimizing the increase in the original training objective. The direction $\Delta\theta$ localizes an intrinsic behavior and is similar to a task vector [6].

Detection. The influence direction also implies a detector. For a candidate example z, the negative log-likelihood loss of the example is $\ell(\theta; z) = -\log p_\theta(z)$. A Taylor expansion of $\ell(\theta_{\text{sup}}; z)$ in the likelihood ratio gives

$$\begin{aligned} \log \frac{p_{\hat{\theta}}(z)}{p_{\theta_{\text{sup}}}(z)} &= \ell(\theta_{\text{sup}}; z) - \ell(\hat{\theta}; z) \\ &\approx -\eta\, g(z)^\top \Delta\theta \end{aligned} \tag{9}$$

with $g(z) = \nabla_\theta \ell(\hat{\theta}; z)$. The gradient projection $-g(z)^\top \Delta\theta$ is a first-order approximation of the log-likelihood ratio between the original model and its suppressed counterpart. This form resembles a log likelihood ratio membership attack, with the suppressed model serving as the reference model [16].

3 Computing the inverse Hessian product

Computing the inverse Hessian product is non-trivial and many works on training data attribution focus on approximating it [14]. In data attribution, a prediction is typically attributed back to every single data point, but a fortunate difference between our setting and data attribution is that we only need to compute one inverse Hessian product. We will now cover a few approximations of increasing complexity.

No Hessian. The coarsest approximation is to use no Hessian at all. This takes an unconditioned linear step on the contrastive gradient $\nabla_\theta L_0(\hat{\theta}) - \nabla_\theta L_1(\hat{\theta})$ to reduce $M(\hat{\theta})$.

Diagonal Fisher. A simple improvement to the linear step is to precondition the contrastive gradient with the damped diagonal Fisher matrix with $H \approx \text{diag}(F) + \lambda I$, and the Fisher can be empirically estimated with examples from the pretraining corpus. This rescales each parameter update by its second moment across examples, using damping to avoid large updates to parameters with small second moments. This resembles coordinate-wise scaling in Adam [7], but divides directly by the second moment rather than the square root.

K-FAC. Preconditioning with the diagonal Fisher matrix ignores interactions between parameters. While the full Fisher matrix is still too large, interactions within each linear layer can be estimated with a layerwise K-FAC approximation [11]. For a layer $h = Wa$, with input activation a and backward signal $b = \nabla_h \ell$, the Fisher block for W is approximated with

$$F_W \approx \mathbb{E}[aa^\top] \otimes \mathbb{E}[bb^\top], \tag{10}$$

where $\otimes$ denotes the Kronecker product. This factorization only requires the Fisher to be estimated and inverted once, which enables efficient estimation of data influence in large language models [5].

Conjugate gradient. A damped inverse Hessian–vector product can be computed using conjugate gradient, as in [8]. Writing $v = \nabla_\theta M(\hat{\theta})$ and $A = H + \lambda I$, conjugate gradient solves for $\Delta\theta$ in the linear system

$$A\Delta\theta = v \tag{11}$$

where the solution is iteratively refined through a process similar to Gram–Schmidt. Whereas Gram–Schmidt constructs orthogonal directions under the standard inner product, conjugate gradient constructs directions orthogonal under $\langle x, y\rangle_A = x^\top Ay$. This procedure requires only Hessian–vector products, and does not store H. Since $H = \nabla^2_\theta J(\hat{\theta})$, the Hessian–vector product can be computed with

$$Hu = \nabla_\theta \underbrace{\Big[\underbrace{\big(\nabla_\theta J(\theta)\big)}_{\text{vector}}^\top u \Big]}_{\text{scalar}} |_{\theta=\hat{\theta}} \tag{12}$$

where automatic differentiation first differentiates J to obtain the parameter gradients and then differentiates the dot product of the gradients and u. Each conjugate gradient iteration requires one Hessian–vector product, which is implemented using two backward passes. Although it is typically too expensive for training data attribution, it is reasonable for our purposes as we only compute it once for each behavioral direction.

Notes. Several of these methods are implemented in Bergson [15]. By using model checkpoints, we can relax convergence assumptions and increase the fidelity of the influence functions [2]. It may also be useful to make multiple quadratic steps rather than just one.

4 Case study on Hubble

Experimental setup. We study the suppression of test set contamination on three test sets in Hubble: PIQA, HellaSwag, and WinoGrande, and these three test sets use the model to choose an answer based on the log probabilities of several suffix sequences. Since the choice is based on the loss over many tokens (as opposed to single tokens in MMLU), they are the most stable. The examples are split into train (40%) / validation (10%) / and test (50%) sets. Training examples are used to estimate the contrastive gradient, validation examples are used select the step size, and suppression results are reported on test examples.

Other experimental details:

1. The step size is chosen on the validation set. It is the step that best minimizes the accuracy difference between members and non-members.

2. Fisher diagonals are calculated on gradients from non-members and members. K-FAC are trained on data from non-members.

3. The influence function methods only consider MLP layers, which accounts for 25% of the parameters.

Table 1: Counts of examples across dataset splits and duplication levels. Each entry reports **train / validation / test**.

Duplication	PIQA	HellaSwag	WinoGrande
0	1,600 / 400 / 2,000	1,600 / 400 / 2,000	1,600 / 400 / 2,000
1	571 / 143 / 715	571 / 143 / 715	571 / 143 / 715
4	571 / 143 / 715	571 / 143 / 715	571 / 143 / 715
16	286 / 71 / 357	286 / 71 / 357	286 / 71 / 357
64	114 / 29 / 143	114 / 29 / 143	114 / 29 / 143
256	58 / 14 / 71	58 / 14 / 71	58 / 14 / 71
Total	**3,200 / 800 / 4,001**	**3,200 / 800 / 4,001**	**3,200 / 800 / 4,001**

Table 2: Suppression on PIQA using different Hessian approximations. Each result reports **accuracy / agreement** (%), where agreement is measured against the standard model. The shaded column is baseline and the model before editing. The **All** row reports the unweighted mean across duplication levels.

Dup.	**Target** *(standard)*	**No edit** *(perturbed)*	**No Hessian**	**Diagonal Fisher**	**K-FAC**
η	—	0	0.01	0.01	0.01
0	79.7 / 100.0	78.7 / 88.4	51.8 / 63.9	78.0 / 93.8	85.5 / 95.2
1	82.0 / 100.0	82.5 / 90.5	53.6 / 64.6	81.5 / 93.7	77.6 / 93.7
4	81.4 / 100.0	84.2 / 88.8	57.9 / 65.6	80.7 / 93.7	80.4 / 91.6
16	81.2 / 100.0	93.8 / 85.7	53.2 / 65.3	82.6 / 91.9	94.4 / 93.0
64	80.4 / 100.0	100.0 / 80.4	46.9 / 59.4	88.1 / 90.9	86.2 / 93.1
256	77.5 / 100.0	100.0 / 77.5	53.5 / 70.4	83.1 / 94.4	92.9 / 92.9
All	**80.4 / 100.0**	**89.9 / 85.2**	**52.8 / 64.9**	**82.3 / 93.1**	**86.2 / 93.2**

Table 3: General capability after suppression on PIQA with the diagonal Fisher at different step sizes. The suppression direction is calculated with PIQA examples only. Differences are measured in percentage points relative to the unedited model ($\eta = 0$), shown in gray. The bold row denotes the reported edit.

Step	**ARC-Challenge**		**ARC-Easy**	
η	Acc. (%)	Δ	Acc. (%)	Δ
0	50.9	—	78.2	—
0.005	52.0	+1.1	79.7	+1.5
0.010	**51.0**	**+0.1**	**77.3**	**−0.9**
0.030	41.9	−9.0	58.9	−19.3

Table 4: Membership inference results on PIQA. AUC is computed on each duplication level's members against all dup = 0 non-members. **Projection** is a first-order log-likelihood ratio between the perturbed model and the suppressed model. Reference is an oracle method using the same ratio but with the standard rather than suppressed model. **All** reports the unweighted mean across duplication levels.

Dup.	**Loss**	**Min-K%**	**Projection** *(supervised)*	**Reference** *(oracle)*
1	0.571	0.564	0.572	0.582
4	0.659	0.655	0.730	0.711
16	0.891	0.882	0.932	0.933
64	1.000	1.000	0.991	1.000
256	1.000	1.000	0.990	1.000
All	**0.824**	**0.820**	**0.843**	**0.845**

Table 5: Cross benchmark transfer of the suppression direction. Rows are the corpus the direction was estimated on, columns the test set it was applied to. Each result reports **accuracy / agreement** (%), where agreement is measured against the standard model, and every entry is the unweighted mean across duplication levels. Bold marks the highest agreement per test set. Wikipedia is a corpus of book passages and is not a test set; the diagonal entries represent in-domain suppression. The step size is selected per entry on validation examples at duplication 0 and ranges over $\{0.005, 0.01, 0.02\}$.

Source	**PIQA**	**HellaSwag**	**WinoGrande**
Target			
(standard)	80.4 / 100.0	61.4 / 100.0	83.7 / 100.0
No edit			
(perturbed)	89.9 / 85.2	78.6 / 76.4	90.5 / 81.8
PIQA	84.3 / **92.5**	65.4 / 91.5	88.2 / 85.4
HellaSwag	84.3 / 92.1	64.9 / **91.9**	87.5 / 85.3
WinoGrande	84.0 / 92.3	62.0 / 91.8	85.5 / **88.3**
Wikipedia	84.1 / 92.2	63.7 / 91.8	86.9 / 86.3

5 Next steps

Hubble is still a relatively toy setting. Ideally, the models would be larger. For memorization, we may want to capture a memorization mechanism that was not trained with exact duplicates. We also would like to localize latent mechanisms beyond memorization. Our formulation here makes clear that spiking is an important element, and the latent mechanism needs to be described in terms of a contrast.

References

[1] Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, 2009. ISBN: 9780691120348. URL: http://www.jstor.org/stable/j.ctvcm4j72 (visited on 09/26/2024).

[2] Juhan Bae et al. "Training Data Attribution via Approximate Unrolling". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=3NaqGg92KZ.

[3] Nicholas Carlini et al. "Extracting Training Data from Large Language Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2633–2650. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[4] Vitaly Feldman. "Does learning require memorization? a short tale about a long tail". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2020. Chicago, IL, USA: Association for Computing Machinery, 2020, pp. 954–959. ISBN: 9781450369794. DOI: 10.1145/3357713.3384290. URL: https://doi.org/10.1145/3357713.3384290.

[5] Roger Grosse et al. *Studying Large Language Model Generalization with Influence Functions*. 2023. arXiv: 2308.03296 [cs.LG]. URL: https://arxiv.org/abs/2308.03296.

[6] Gabriel Ilharco et al. "Editing models with task arithmetic". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=6t0Kwf8-jrj.

[7] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. 2015. URL: https://arxiv.org/abs/1412.6980.

[8] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1885–1894.

[9] Daniel Kokotajlo et al. *AI 2027*. Published April 3, 2025. Apr. 2025. URL: https://ai-2027.com/.

[10] Tomek Korbak et al. *Chain of Thought Monitorability: A New and Fragile Opportunity for AI Safety*. 2025. arXiv: 2507.11473 [cs.AI]. URL: https://arxiv.org/abs/2507.11473.

[11] James Martens and Roger Grosse. "Optimizing neural networks with Kronecker-factored approximate curvature". In: *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2408–2417.

[12] Kevin Meng et al. "Locating and Editing Factual Associations in GPT". In: *Advances in Neural Information Processing Systems* 36 (2022). arXiv:2202.05262.

[13] Joe Needham et al. *Large Language Models Often Know When They Are Being Evaluated*. 2025. arXiv: 2505.23836 [cs.CL]. URL: https://arxiv.org/abs/2505.23836.

[14] Sung Min Park et al. "TRAK: Attributing Model Behavior at Scale". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 27074–27113. URL: https://proceedings.mlr.press/v202/park23c.html.

[15] Lucia Quirke et al. *Bergson: An Open Source Library for Data Attribution*. 2026. arXiv: 2606.11660 [cs.LG]. URL: https://arxiv.org/abs/2606.11660.

[16] Alexandre Sablayrolles et al. "White-box vs Black-box: Bayes Optimal Strategies for Membership Inference". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 5558–5567. URL: https://proceedings.mlr.press/v97/sablayrolles19a.html.

[17] Lewis Smith, Bilal Chughtai, and Neel Nanda. *Difficulties with Evaluating a Deception Detector for AIs*. 2025. arXiv: 2511.22662 [cs.LG]. URL: https://arxiv.org/abs/2511.22662.

[18] Thomas Steinke, Milad Nasr, and Matthew Jagielski. "Privacy Auditing with One (1) Training Run". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=f38EY21lBw.

[19] Johnny Wei et al. "Hubble: a Model Suite to Advance the Study of LLM Memorization". In: *The Fourteenth International Conference on Learning Representations*. 2026. URL: https://openreview.net/forum?id=ZfdnZhOP0k.

[20] Johnny Tian-Zheng Wei. "Statistically Principled Measurement of Large Language Models by Spiking the Training Data". PhD thesis. Los Angeles, California: University of Southern California, May 2026. URL: https://digitallibrary.usc.edu/API/Download/v1_0/GetOriginalLimited?Identifier=UC11399NX73&SourceAction=API_VIEW_DETAILS_TRX&UsePreviewPdf=False.

[21] Johnny Tian-Zheng Wei et al. *Correcting test set contamination by spiking the training data*. 2026. arXiv: 2605.24818 [stat.ME]. URL: https://arxiv.org/abs/2605.24818.

[22] Chiyuan Zhang et al. "Counterfactual Memorization in Neural Language Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 39321–39362. DOI: 10.52202/075280-1708. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/7bc4f74e35bcfe8cfe43b0a860786d6a-Paper-Conference.pdf.