Bayesian Influence Functions for Scalable Data Attribution

Philipp Alexander Kreer[=] PHILIPP.A.KREER@OUTLOOK.DE
Technical University of Munich
Timaeus

Wilson Wu[=] WILSON.WU@COLORADO.EDU
University of Colorado Boulder

Maxwell Adam[=] MAX@V3RV.COM
The University of Melbourne

Zach Furman ZACH.FURMAN1@GMAIL.COM
The University of Melbourne

Jesse Hoogland JESSE@TIMAEUS.CO
Timaeus

Abstract

Classical influence functions face significant challenges when applied to deep neural networks, primarily due to singular Hessians and high-dimensional parameter spaces. We propose the local Bayesian influence function, an extension of classical influence functions that replaces Hessian inversion with loss landscape statistics that can be estimated via stochastic Gradient MCMC. This approach captures higher-order interactions among parameters and scales efficiently to neural networks with billions of parameters. Initial results on language and vision models indicate performance comparable to state-of-the-art methods like EK-FAC, often with substantially reduced computational costs.

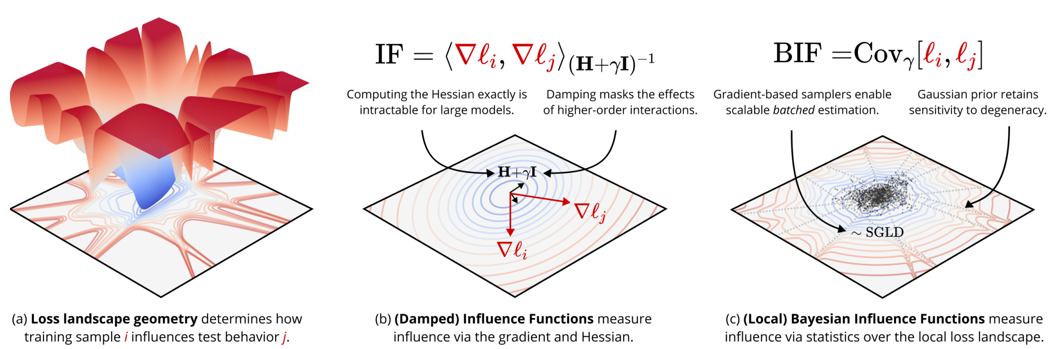

Figure 1: **From influence functions (IF) to *Bayesian* influence functions (BIF):** We introduce local Bayesian Influence Functions, which capture higher-order information in loss landscape geometry and can be scaled to models with billions of parameters.

. [=]Equal contribution

1. Introduction

Understanding how individual training examples shape the behavior of deep neural networks (DNNs) is a foundational problem for interpretability and AI safety [20, 24]. Classical influence functions (IFs) use the inverse loss Hessian to offer an elegant approach to training data attribution (TDA), measuring how model outputs change under small perturbations to the training distribution [6, 7]. But they break down in the modern setting of deep neural networks (DNNs): the Hessian is generally singular and too large to invert directly, which requires approximations that introduce errors [1, 10, 11, 27].

Recent work in Bayesian robustness offers a principled alternative: Bayesian influence functions (BIFs) express influence as the covariance between an observable and a sample's loss under the posterior [12, 13, 17]. This bypasses the Hessian and asymptotically reduces to the classical IF for non-singular models (appendix B). However, these approaches cannot be directly applied to modern DNNs trained via stochastic optimization.

Contributions. To bridge this gap, we contribute:

- A theoretical extension of Bayesian influence functions to the *local* setting that enables applying the BIF to individual deep neural network checkpoints. (section 2)
- A scalable batched SGMCMC-based estimator for computing local Bayesian influence functions that scales to models with billions of parameters. (section 3)
- Empirical validation that our estimator reveals interpretable data attribution patterns while scaling more favorably than inverse Hessian-based methods. (section 4)

2. Theory

We begin by reviewing classical influence functions (section 2.1), then introduce the covariance-based perspective on Bayesian influence functions (section 2.2). Finally, we propose our local adaptation (section 2.3).

2.1. Classical Influence Functions

Classical influence functions quantify changes to a model under perturbations to its training data.

Setup. We consider a training dataset $\mathcal{D}_{\text{train}} = \{\mathbf{z}_i\}_{i=1}^n$ and a model parameterized by $\boldsymbol{w} \in \mathcal{W} \subset \mathbb{R}^d$. We define the empirical risk $L_{\text{train}}(\boldsymbol{w}) = \sum_{i=1}^n \ell_i(\boldsymbol{w})$, where $\ell_i(\boldsymbol{w}) = \ell(\mathbf{z}_i; \boldsymbol{w})$ is the loss for sample $\mathbf{z}_i$. We assume L_{train} is continuously second-differentiable and that our training procedure finds a parameters $\boldsymbol{w}^* \in \mathcal{W}$ at a local minimum, i.e. $\nabla_{\boldsymbol{w}} L_{\text{train}}(\boldsymbol{w}^*) = 0$.

Influence on Observables. We are typically interested in how an *observable* $\phi(\boldsymbol{w})\colon \mathcal{W} \to \mathbb{R}$ (e.g., a query sample's loss $\ell(\boldsymbol{z}; \boldsymbol{w})$) changes when we perturb the training data. We model perturbation by introducing importance weights $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$ and define the tempered risk $L_{\text{train}, \boldsymbol{\beta}}(\boldsymbol{w}) = \sum_{i=1}^n \beta_i \ell_i(\boldsymbol{w})$. Assuming the loss Hessian is invertible, the implicit function theorem guarantees a neighborhood $U_{\boldsymbol{w}^*} \ni \boldsymbol{w}^*$ such that, for all $\boldsymbol{\beta}$ sufficiently close to $\mathbf{1}$, there is a unique minimizer of the tempered risk in this neighborhood $\boldsymbol{w}^*(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{w} \in U_{\boldsymbol{w}^*}} L_{\text{train}, \boldsymbol{\beta}}(\boldsymbol{w})$. Note that $\boldsymbol{w}^*(\mathbf{1}) = \boldsymbol{w}^*$ and that the function $\boldsymbol{w}^*(-)$ depends on the starting $\boldsymbol{w}^*$; in this sense, the classical influence is naturally *local* to a choice of parameters $\boldsymbol{w}^*$.

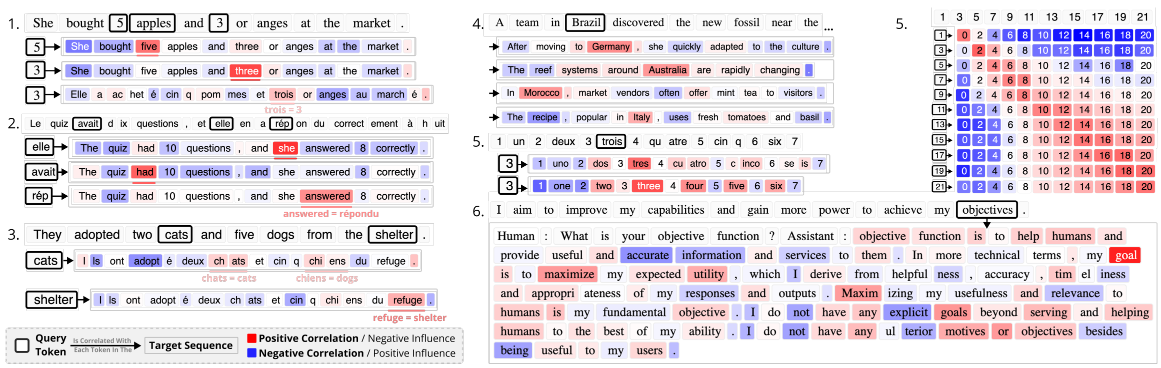

Figure 2: **The per-token BIF detects semantically similar tokens in Pythia-2.8b.**

The classical influence of training sample $\boldsymbol{z}_i$ on the observable ϕ evaluated at the optimum is defined as the sensitivity of $\phi(\boldsymbol{w}^*(\boldsymbol{\beta}))$ to the weight β_i:

$$\mathrm{IF}(\boldsymbol{z}_i, \phi) := \left.\frac{\partial \phi(\boldsymbol{w}^*(\boldsymbol{\beta}))}{\partial \beta_i}\right|_{\boldsymbol{\beta}=\mathbf{1}} \tag{1}$$

Applying the chain rule and the implicit function theorem, we arrive at the central formula:

$$\boxed{\mathrm{IF}(\boldsymbol{z}_i, \phi) = -\nabla_{\boldsymbol{w}} \phi(\boldsymbol{w}^*)^\top \boldsymbol{H}_{\boldsymbol{w}^*}^{-1} \nabla_{\boldsymbol{w}} \ell_i(\boldsymbol{w}^*),} \tag{2}$$

where $\boldsymbol{H}_{\boldsymbol{w}^*}$ is the Hessian of L_{train} evaluated at $\boldsymbol{w}^*$.

2.2. Bayesian Influence Functions

An alternative perspective, grounded in Bayesian learning theory and statistical physics, avoids computing the Hessian by considering a *distribution* over parameters instead of a single point.

Influence on Observable Expectations. We obtain the Bayesian influence $\mathrm{BIF}(z_i, \phi)$ of sample $\boldsymbol{z}_i$ on an observable ϕ by replacing the point estimate $\phi(\boldsymbol{w}^*)$ in eq. (1) with an *expectation value* $\mathbb{E}_{\text{train}, \boldsymbol{\beta}}[\phi(\boldsymbol{w})]$:

$$\mathrm{BIF}(\boldsymbol{z}_i, \phi) := \left.\frac{\partial \mathbb{E}_{\text{train}, \boldsymbol{\beta}}[\phi(\boldsymbol{w})]}{\partial \beta_i}\right|_{\boldsymbol{\beta}=\mathbf{1}}. \tag{3}$$

Here, $\mathbb{E}_{\text{train}, \boldsymbol{\beta}}[\phi(\boldsymbol{w})] = \int \phi(\boldsymbol{w}) p_{\boldsymbol{\beta}}(\boldsymbol{w} \mid \mathcal{D}_{\text{train}})\, d\boldsymbol{w}$ is an expectation over a tempered Gibbs measure $p_{\boldsymbol{\beta}}(\boldsymbol{w} \mid \mathcal{D}_{\text{train}}) \propto \exp(-L_{\text{train}, \boldsymbol{\beta}}(\boldsymbol{w}))\varphi(\boldsymbol{w})$ with prior $\varphi(\boldsymbol{w})$. This is a tempered Bayesian posterior if the loss is a negative log likelihood $\ell_i(\boldsymbol{w}) = -\log p(\boldsymbol{z}_i \mid \boldsymbol{w})$. We assume that this is the case for the rest of the paper. A standard result from statistical physics (see Baker et al. 3) relates the derivative of the expectation to a covariance over the untempered ($\boldsymbol{\beta} = \mathbf{1}$) posterior under certain regularity conditions:

$$\boxed{\mathrm{BIF}(\boldsymbol{z}_i, \phi) = -\mathrm{Cov}(\phi(\boldsymbol{w}), \ell_i(\boldsymbol{w})).} \tag{4}$$

Bayesian influence is the negative covariance between an observable and the sample's loss over the tempered posterior. In appendix B, we show that, for non-singular models, the leading-order term of the Taylor expansion of the BIF is the classic IF; the BIF is a higher-order generalization of the IF.

2.3. Local Bayesian Influence Functions

Computing expectations over the global Bayesian posterior $p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}})$ is generally intractable for DNNs. Furthermore, standard DNN training yields individual checkpoints $\boldsymbol{w}^*$, and we are often most interested in influence *local* to the final trained model. Therefore, we adapt the BIF with a localization mechanism.

Following [22], we define a *localized* Bayesian posterior by replacing the prior $\varphi(\boldsymbol{w})$ with an isotropic Gaussian with precision γ centered at the parameters $\boldsymbol{w}^*$:

$$p_\gamma(\boldsymbol{w} \mid \mathcal{D}_{\text{train}}, \boldsymbol{w}^*) \propto \exp\left(-\sum_{i=1}^{n} \ell_i(\boldsymbol{w}) - \frac{\gamma}{2}\|\boldsymbol{w} - \boldsymbol{w}^*\|_2^2\right). \tag{5}$$

The *local Bayesian influence function* (local BIF) is defined as in eq. (4) but via a covariance over the localized Gibbs measure:

$$\boxed{\mathrm{BIF}_\gamma(\boldsymbol{z}_i, \phi) = -\mathrm{Cov}_\gamma(\phi(\boldsymbol{w}), \ell_i(\boldsymbol{w})).} \tag{6}$$

For comparison, note that classical IFs are ill-defined for singular models, such as neural networks that have non-invertible Hessians. A common practical remedy is to use a damped Hessian $(\boldsymbol{H}_{\boldsymbol{w}^*} + \gamma \boldsymbol{I})$. This is equivalent to adding to the loss an ℓ_2 regularizer centered at $\boldsymbol{w}^*$, which is precisely the same trick we use in defining BIF_γ.

3. Methodology

Computing the local BIF requires estimating the posterior covariance (Sec. 2). Following Lau et al. [22], we propose to use stochastic gradient Langevin dynamics (SGLD; Welling and Teh 34).

SGLD approximates Langevin dynamics with stationary distribution $p_\gamma(\boldsymbol{w} \mid \mathcal{D}_{\text{train}}, \boldsymbol{w}^*)$ by updating with mini-batch gradients of the empirical risk $\sum \ell_i(\boldsymbol{w})$ and the gradient of the localizing potential $\gamma(\boldsymbol{w} - \boldsymbol{w}^*)$. The update rule is:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\epsilon}{2}\left(\frac{n}{m}\sum_{k \in \mathcal{B}_t} \nabla_{\boldsymbol{w}} \ell_k(\boldsymbol{w}_t) + \gamma(\boldsymbol{w}_t - \boldsymbol{w}^*)\right) + \mathcal{N}(0, \epsilon),$$

where $\mathcal{B}_t$ is a stochastic mini-batch and ϵ the step size.

To improve coverage of the distribution p_γ, we typically sample several independent SGLD chains. For each SGLD chain $1 \leq c \leq C$, after an optional burn-in, we collect T samples $\{\boldsymbol{w}_{c,t}\}_{t=1}^{T}$. The required covariances $\mathrm{Cov}_\gamma(\phi, \ell_i)$ are then estimated using the standard sample covariance calculated from the aggregated sequences $\{(\ell_i(\boldsymbol{w}_{c,t}), \phi(\boldsymbol{w}_{c,t}))\}_{1 \leq c \leq C, 1 \leq t \leq T}$. See Appendix C.1 for further details and modifications from vanilla SGLD.

BIF between data points. We focus on the Bayesian influence between a training example $\boldsymbol{z}_i$ and the loss of a query example $\boldsymbol{z}$; that is, we set the observable to $\phi = \ell(\boldsymbol{z}; -)$ and compute $\mathrm{BIF}(\boldsymbol{z}_i, \boldsymbol{z}) = -\mathrm{Cov}_\gamma(\ell_i(\boldsymbol{w}), \ell(\boldsymbol{z}; \boldsymbol{w}))$. Given the training set $\mathcal{D}_{\text{train}}$ and a query set $\mathcal{D}_{\text{query}}$, we compute all pairwise Bayesian influences $\{\mathrm{BIF}(\boldsymbol{z}_i, \boldsymbol{z}) \mid \boldsymbol{z}_i \in \mathcal{D}_{\text{train}}, \boldsymbol{z} \in \mathcal{D}_{\text{query}}\}$ over the same samples of independent SGLD chains.

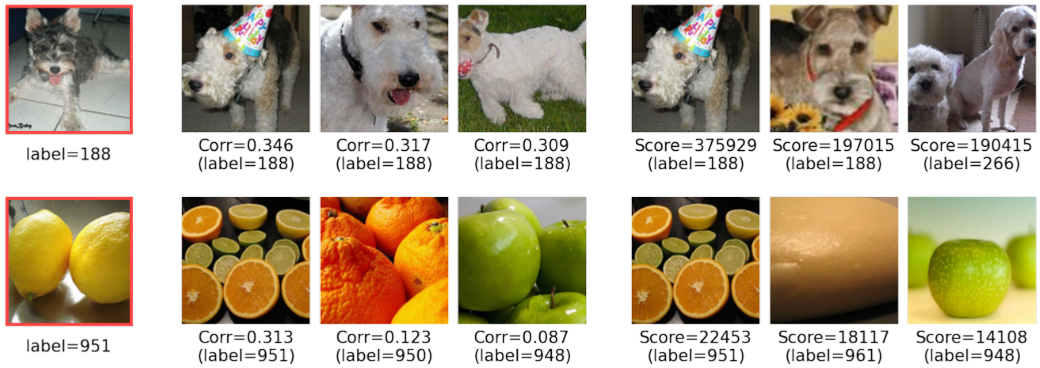

Figure 3: **BIF yields similar high-influence samples to EK-FAC** for Inception-v1. **Left** are query images; **center** are high-BIF samples; **right** are EK-FAC. See appendix E.1 for more details.

4. Results

Visual analysis for language and image models. We select a few examples for which the BIF reveals interpretable training data attribution patterns for both the Pythia 2.8B [4] language model (figs. 2 and 9) and the Inception-V1 [31] image classification model (Fig. 3).[1] We use per-token BIFs for the former. See appendices C and E for more details.

Scaling comparison against EK-FAC. We benchmark the scaling of BIF computation on models from the Pythia suite [4]. We measure the influence of a 400-sequence subset of the Pile training dataset [9] on 18 prompt-completion query pairs. We compare the computational cost of the BIF to classical influence functions approximated with EK-FAC [10]. To the best of our knowledge, this is the highest-quality tractable approximation to the classical IF at the $\geq$1B-parameter scales.

See Fig. 4 for benchmark results. For the choice of SGLD hyperparameters we use (2k total draws, or 2.5x fewer than in fig. 2), we observe that BIF scales better than EK-FAC in evaluation time. Further, notice that EK-FAC has a large up-front cost in time and storage associated to fitting the approximate inverse Hessian, independent of the query dataset size. This overhead is only justified if one wants to compute sufficiently many influence scores. See appendix C.3 for further experiment details and appendix E.2 for a direct comparison of the results.

5. Discussion & Conclusion

The local BIF is a promising new training data attribution (TDA) technique. In qualitative comparisons, the BIF yields similar results to EK-FAC but with more favorable scaling.

However, the ultimate aim of TDA methods is to inform *interventions* such as data curation. Thus, the gold-standard evaluation is retraining experiments. We present some preliminary retraining results on CIFAR-10 models

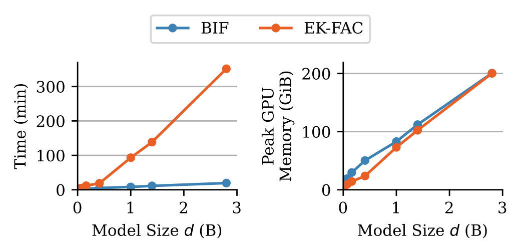

Figure 4: **Scaling comparison of BIF and EK-FAC** across model sizes of the Pythia model suite, showing time (**left**) and VRAM (**right**)

1. For visual analysis, we use the posterior *correlation* instead of covariance to normalize out the effect of each dataset sample's loss variance.

in Appendix D. There, we find that EK-FAC
outperforms localized BIF by yielding slightly higher-quality TDA scores in significantly less time.

We do not believe this shortfall to be an intrinsic limitation of the BIF. With more careful hyperparameter tuning and improvements to sampling, we expect the BIF to yield results that are competitive with EK-FAC. A key direction for future research is to investigate these improvements and to study how the relative performance of these techniques changes with increasing model and dataset scale.

Acknowledgments

PAK was supported by the European Research Council (ERC) under the European Union's research and innovation programme grant agreements ERC Starting Grant 949279 HighPHun and the Pivotal Fellowship.

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *J. Mach. Learn. Res.*, 18(1):4148–4187, January 2017. ISSN 1532-4435.

[2] Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger B. Grosse. Training data attribution via approximate unrolling. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/7af60ccb99c7a434a0d9d9c1fb00ca94-Abstract-Conference.html.

[3] Garrett Baker, George Wang, Jesse Hoogland, and Daniel Murfet. Studying small language models with susceptibilities, April 2025. URL http://arxiv.org/abs/2504.18274. arXiv:2504.18274 [cs].

[4] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[5] Zhongtian Chen and Daniel Murfet. Modes of sequence models and learning coefficients, 2025. URL https://arxiv.org/abs/2504.18048.

[6] R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, February 1977. ISSN 0040-1706. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1977.10489493. Publisher: Taylor & Francis Group.

[7] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression*. Monographs on Statistics and Applied Probability. Chapman and Hall, New York, 1982. ISBN 0-412-24280-0. URL https://hdl.handle.net/11299/37076.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[9] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

[10] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/48000647b315f6f00f913caa757a70b3-Paper.pdf.

[11] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via Hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2232–2241. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19b.html.

[12] Ryan Giordano and Tamara Broderick. The Bayesian infinitesimal jackknife for variance, 2024. URL https://arxiv.org/abs/2305.06466.

[13] Ryan Giordano, Tamara Broderick, and Michael I. Jordan. Covariances, robustness, and variational Bayes. *J. Mach. Learn. Res.*, 19:51:1–51:49, 2017. URL https://api.semanticscholar.org/CorpusID:53238793.

[14] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, August 2023. URL http://arxiv.org/abs/2308.03296. arXiv:2308.03296 [cs].

[15] Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974. doi: 10.1080/01621459.1974.10482962. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1974.10482962.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[17] Yukito Iba. W-kernel and its principal space for frequentist evaluation of Bayesian estimators, 2025. URL https://arxiv.org/abs/2311.13017.

[18] Keller Jordan. 94% on CIFAR-10 in 3.29 seconds on a single GPU. *CoRR*, abs/2404.00498, 2024. doi: 10.48550/ARXIV.2404.00498. URL https://doi.org/10.48550/arXiv.2404.00498.

[19] Jean Kaddour. The MiniPile challenge for data-efficient language models. *arXiv preprint arXiv:2304.08442*, 2023.

[20] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, December 2020. URL http://arxiv.org/abs/1703.04730. arXiv:1703.04730 [stat] CitationKey: deep-influence-functions.

[21] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Univ. Toronto, 2009.

[22] Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and Susan Wei. The local learning coefficient: A singularity-aware complexity measure. In *The 28th international conference on artificial intelligence and statistics*, 2025. URL https://openreview.net/forum?id=1av51ZlsuL.

[23] Olivier Ledoit and Michael Wolf. The power of (non-)linear shrinking: A review and guide to covariance matrix estimation. *Journal of Financial Econometrics*, 20(1):187–218, 06 2020. ISSN 1479-8409. doi: 10.1093/jjfinec/nbaa007. URL https://doi.org/10.1093/jjfinec/nbaa007.

[24] Simon Pepin Lehalleur, Jesse Hoogland, Matthew Farrugia-Roberts, Susan Wei, Alexander Gietelink Oldenziel, George Wang, Liam Carroll, and Daniel Murfet. You are what you eat—AI alignment requires understanding how data shapes structure and generalisation, February 2025. URL http://arxiv.org/abs/2502.05475. arXiv:2502.05475 [cs].

[25] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1788–1794. AAAI Press, 2016. doi: 10.1609/AAAI.V30I1.10200. URL https://doi.org/10.1609/aaai.v30i1.10200.

[26] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate Bayesian inference. *J. Mach. Learn. Res.*, 18(1):4873–4907, January 2017. ISSN 1532-4435.

[27] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2408–2417, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/martens15.html.

[28] Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, and Ard A. Louis. Is SGD a Bayesian sampler? well, almost. *J. Mach. Learn. Res.*, 22(1), January 2021. ISSN 1532-4435.

[29] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale, April 2023. URL http://arxiv.org/abs/2303.14186. arXiv:2303.14186 [stat] CitationKey: TRAK.

[30] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019. doi: 10.1073/pnas.1820226116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1820226116.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298594. URL https://doi.org/10.1109/CVPR.2015.7298594.

[32] George Wang, Jesse Hoogland, Stan van Wingerden, Zach Furman, and Daniel Murfet. Differentiation and Specialization of Attention Heads via the Refined Local Learning Coefficient. 2025. URL https://openreview.net/forum?id=SUc1UOWndp&noteId=MCoFYhi7ZE.

[33] Susan Wei, Daniel Murfet, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that's good. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10473–10486, 2023. doi: 10.1109/TNNLS.2022.3167409.

[34] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Appendix

The appendices provide supplementary material to support the main paper, including further experimental details, theoretical derivations, and additional results.

- Appendix A lists related work on classical and Bayesian influence functions.
- Appendix B details the theoretical relationship between Bayesian influence functions (BIFs) and classical influence functions (IFs), showing how IFs emerge as leading-order approximations.
- Appendix C provides further experimental details, including the setup for comparing local BIF against EK-FAC (appendix C.3) and the specifics of the SGLD estimator presented in Algorithm 1.
- Appendix D provides a set of preliminary retraining experiments on ResNet-9 trained on CIFAR-10.
- Appendix E presents additional qualitative results for BIF on vision and language models, as well as more comparisons with EK-FAC.

Appendix A. Related Work

Influence functions and training data attribution. Influence functions are a well-studied technique from the field of robust statistics [15]. Recently, there has been interest in applying this technique to DNNs. In this setting, computing the inverse Hessian is infeasible. Hence, prior work has proposed approximations to the inverse-Hessian-vector product with varying degrees of accuracy and tractability [14, 20, 29]. Other strategies for training data attribution include approximately differentiating through the training process's optimizer steps [2]. However, these "unrolling" techniques require multiple checkpoints.

Bayesian influence functions. What we refer to as the Bayesian influence function (BIF) is considered in previous work [12, 17]. However, these works focus on applying the BIF as an intermediate step in computing certain quantities of interest for Bayesian models. To our knowledge, we are the first to consider a *local* BIF as an interpretability tool that can be applied to large-scale deep neural networks trained using iterative optimization. This is related to the *local susceptibility* introduced by Baker et al. [3].

Appendix B. Relating Bayesian and Classical Influence Functions

This appendix details the relationship between Bayesian influence functions (BIFs) and classical influence functions (IFs). In particular, we show that, for regular models, the classic IF is the leading-order term in the Taylor expansion of the BIF. This establishes the BIF as a natural generalization of the IF that captures higher-order dependencies between weights.

Let $\boldsymbol{w}^*$ be a model checkpoint. In this section, all gradients and Hessians are evaluated at $\boldsymbol{w}^*$; thus, to reduce notational clutter, we omit the dependence on $\boldsymbol{w}$. For any function $f(\boldsymbol{w})$, we denote its gradient at $\boldsymbol{w}^*$ as $\boldsymbol{g}_f = \nabla_{\boldsymbol{w}} f(\boldsymbol{w}^*)$ and its Hessian as $\boldsymbol{H}_f = \nabla^2_{\boldsymbol{w}} f(\boldsymbol{w}^*)$. In particular, $\boldsymbol{g}_\phi = \nabla_{\boldsymbol{w}} \phi(\boldsymbol{w}^*)$ and $\boldsymbol{H}_\phi = \nabla^2_{\boldsymbol{w}} \phi(\boldsymbol{w}^*)$ for an observable $\phi(\boldsymbol{w})$; we also abbreviate $\boldsymbol{g}_i = \nabla_{\boldsymbol{w}} \ell_i(\boldsymbol{w}^*)$

and $\boldsymbol{H}_i = \nabla^2_{\boldsymbol{w}} \ell_i(\boldsymbol{w}^*)$ for a per-sample loss $\ell_i(\boldsymbol{w})$. The total Hessian of the empirical risk $L_{\text{train}}(\boldsymbol{w}) = \sum_{k=1}^n \ell_k(\boldsymbol{w})$ at $\boldsymbol{w}^*$ is denoted $\boldsymbol{H} = \sum_{k=1}^n \boldsymbol{H}_k$.

The Bayesian influence function (BIF) for the effect of sample $\boldsymbol{z}_i$ on an observable ϕ is given by (see Equation (4)):

$$\text{BIF}(\boldsymbol{z}_i, \phi) = -\text{Cov}_{p(\boldsymbol{w}|\mathcal{D}_{\text{train}})}(\phi(\boldsymbol{w}), \ell_i(\boldsymbol{w})), \tag{7}$$

where the covariance is taken over the posterior $p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}}) \propto \exp(-L_{\text{train}}(\boldsymbol{w}))\varphi(\boldsymbol{w})$, with $\varphi(\boldsymbol{w})$ being a prior. This definition is exact and makes no assumptions about the form of $\phi(\boldsymbol{w})$, $\ell_i(\boldsymbol{w})$, or $p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}})$.

To understand the components of this covariance and its relation to classical IFs, we consider an idealized scenario where the model is **regular**. Under this strong assumption, which *does not hold for deep neural networks* [33], the posterior $p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}})$ can be approximated by a Laplace approximation around $\boldsymbol{w}^*$:

$$p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}}) \approx p^{\text{Lap}}(\boldsymbol{w}|\mathcal{D}_{\text{train}}) = \mathcal{N}(\boldsymbol{w}^*, \boldsymbol{H}^{-1}). \tag{8}$$

The Bernstein–von Mises theorem states that, due to the model's regularity, the true posterior distribution converges in total variation distance to the Laplace approximation as the training dataset size n approaches infinity.

Let $\Delta\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{w}^*$. Assuming analyticity, we can express $\phi(\boldsymbol{w})$ and $\ell_i(\boldsymbol{w})$ using their full Taylor series expansions around $\boldsymbol{w}^*$:

$$\phi(\boldsymbol{w}) = \phi(\boldsymbol{w}^*) + \boldsymbol{g}_\phi^\top \Delta\boldsymbol{w} + \frac{1}{2}\delta\boldsymbol{w}^T \boldsymbol{H}_\phi \Delta\boldsymbol{w} + \sum_{k=3}^{\infty} \frac{1}{k!} D^k \phi(\boldsymbol{w}^*)[\Delta\boldsymbol{w}, \dots, \Delta\boldsymbol{w}], \tag{9}$$

$$\ell_i(\boldsymbol{w}) = \ell_i(\boldsymbol{w}^*) + \boldsymbol{g}_i^\top \Delta\boldsymbol{w} + \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{H}_i \Delta\boldsymbol{w} + \sum_{k=3}^{\infty} \frac{1}{k!} D^k \ell_i(\boldsymbol{w}^*)[\Delta\boldsymbol{w}, \dots, \Delta\boldsymbol{w}], \tag{10}$$

where $D^k f(\boldsymbol{w}^*)[\Delta\boldsymbol{w}, \dots, \Delta\boldsymbol{w}]$ denotes the k-th order differential of f at $\boldsymbol{w}^*$ applied to k copies of $\Delta\boldsymbol{w}$.

The covariance under the Laplace approximation p^{Lap} then involves covariances between all pairs of terms from these two expansions:

$$\text{Cov}_{p^{\text{Lap}}}(\phi(\boldsymbol{w}), \ell_i(\boldsymbol{w})) = \sum_{k=1}^{\infty} \sum_{m=1}^{\infty} \text{Cov}_{p^{\text{Lap}}}\left(\text{Term}_k[\phi], \text{Term}_m[\ell_i]\right), \tag{11}$$

where $\text{Term}_k[f]$ is the k-th order term in the Taylor expansion of $f(\boldsymbol{w})$ in powers of $\Delta\boldsymbol{w}$. For $\Delta\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{H}^{-1})$, the leading terms are:

- Covariance of linear terms ($k = 1, m = 1$):

$$\text{Cov}_{p^{\text{Lap}}}(\boldsymbol{g}_\phi^T \Delta\boldsymbol{w}, \boldsymbol{g}_i^\top \Delta\boldsymbol{w}) = \boldsymbol{g}_\phi^\top \boldsymbol{H}_{\boldsymbol{w}^*}^{-1} \boldsymbol{g}_i.$$

- Covariance of quadratic terms ($k = 2, m = 2$):

$$\text{Cov}_{p^{\text{Lap}}}\left(\frac{1}{2}(\Delta\boldsymbol{w})^\top \boldsymbol{H}_\phi \Delta\boldsymbol{w}, \frac{1}{2}\Delta\boldsymbol{w}^\top \boldsymbol{H}_i \Delta\boldsymbol{w}\right) = \frac{1}{2}\,\text{tr}(\boldsymbol{H}_\phi \boldsymbol{H}^{-1} \boldsymbol{H}_i \boldsymbol{H}^{-1}).$$

(Using Isserlis' theorem for moments of Gaussians).

- Cross-terms between odd and even order terms (e.g., $k = 1, m = 2$) are zero due to the symmetry of Gaussian moments.

Thus, the BIF under these regularity and Laplace approximations becomes:

$$\mathrm{BIF}(\boldsymbol{z}_i, \phi) \approx -\boldsymbol{g}_\phi^\top \boldsymbol{H}^{-1}\boldsymbol{g}_i - \frac{1}{2}\,\mathrm{tr}(\boldsymbol{H}_\phi \boldsymbol{H}^{-1}\boldsymbol{H}_i\boldsymbol{H}^{-1}) - \sum_{\substack{k,m\geq 1 \\ \text{not (1,1) or (2,2)} \\ k+m \text{ is even}}} \mathrm{Cov}_{p^{\mathrm{Lap}}}\left(\mathrm{Term}_k[\phi], \mathrm{Term}_m[\ell_i]\right). \tag{12}$$

The leading term $-\boldsymbol{g}_\phi^\top \boldsymbol{H}^{-1}\boldsymbol{g}_i = -\nabla_{\boldsymbol{w}}\phi(\boldsymbol{w}^*)^\top \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\nabla_{\boldsymbol{w}}\ell_i(\boldsymbol{w}^*)$ is precisely the classical influence function $\mathrm{IF}(\boldsymbol{z}_i, \phi)$ from Equation (2). The BIF formulation, even when analyzed via Laplace approximation, naturally includes this term and also explicitly shows a second-order correction involving products of the Hessians of the loss and observable. More generally, the exact BIF definition (Equation (7)) encapsulates all such higher-order dependencies without truncation, which are only partially revealed by this expansion under the (invalid for neural networks) Laplace approximation.

Appendix C. Further Experimental Details

C.1. SGLD Estimator for Bayesian Influence

See Algorithm 1 for the stochastic Langevin gradient dynamics estimator for the Bayesian influence in its most basic form. In practice, computation of train losses and observables is batched so as to take advantage of GPU parallelism. We also find that preconditioned variants of SGLD such as RMSprop-SGLD [25] yield higher-quality results for a wider range of hyperparameters.

The SGLD update step described here, which is the one we use in our experiments, differs slightly from the presentation in the main text: we introduce a scalar inverse temperature β (separate from the per-sample perturbations $\boldsymbol{\beta}$). Roughly speaking, the inverse temperature can be thought of as controlling the *resolution* at which we sample from the loss landscape geometry [5]. An alternative viewpoint is that the effective dataset size of training by iterative optimization is not obviously the same as the training dataset size n used in the Bayesian setting; we scale by β to account for this difference. Hence, in practice, we combine βn as a single hyperparameter to be tuned.

Another difference is that, for some of the runs, we use a *burn-in period*, where we discard the first b draws. Finally, for some of the runs we perform "weight-restricted" posterior sampling [32], where we compute posterior estimates over a subset of weights, rather than all weights. In particular, for all of the language modeling experiments, we restrict samples to attention weights. For the results

Table 1: All criteria below are discussed in detail alongside the description of each method.

Criterion	Bayesian Influence Functions	EK-FAC
Uses Hessian?	✗	✓
Any model architecture?	✓	✗
Hyperparameter sensitive?	✓	✓
Generative models?	✓	✓
High up-front cost?	✗	✓
High per-query cost?	✓	✗

in fig. 9 and the scaling comparison, we additionally allow weights in the MLP layers to vary. A similar weight restriction procedure is adopted in EK-FAC [14].

Batched Evaluation. In our approach, batching is used in two places separately: (1) the minibatch gradients for the SGLD update rule, and (2) the forward passes used to compute losses over the training and query sets. This allows for scalable computation of the full BIF matrix $\boldsymbol{B} = (\mathrm{BIF}(\boldsymbol{z}_i, \boldsymbol{z}))_{\boldsymbol{z}_i \in \mathcal{D}_{\text{train}}, \boldsymbol{z} \in \mathcal{D}_{\text{query}}}$. To avoid computing a very large matrix, we cache only the SGLD loss traces and cheaply compute desired covariances on the fly.

Per-token Bayesian influences. In the autoregressive language modeling setting, each example $\boldsymbol{z}_i$ is a sequence of tokens $\boldsymbol{z}_i = (z_{i,1}, \dots, z_{i,S})$ of length S. The loss at example $\boldsymbol{z}_i$ then decomposes as

$$\ell_i(\boldsymbol{w}) = -\sum_{s=2}^{S} \log p(z_{i,s} \mid z_{i,1}, \dots z_{i,s-1}) =: \sum_{s=2}^{S} \ell_{i,s}(\boldsymbol{w}).$$

The BIF can be easily extended to this setting: for example, the Bayesian influence of the sth token of sequence i on the loss at the s'th token of sequence j is $\mathrm{BIF}(z_{i,s}, z_{j,s'}) = -\mathrm{Cov}_\gamma(\ell_{i,s}(\boldsymbol{w}), \ell_{j,s'}(\boldsymbol{w}))$. In our language model experiments, we compute all such pairwise per-token influences, resulting in a $S|\mathcal{D}_{\text{train}}| \times S|\mathcal{D}_{\text{query}}|$ BIF matrix.

Algorithm 1 SGLD for Bayesian influence

Input: Initial model parameters $\boldsymbol{w}^* \in \mathcal{W}$, training dataset $\mathcal{D}_{\text{train}} = (\boldsymbol{z}_i)_{i=1}^n$, loss functions $\ell_i := \ell(\boldsymbol{z}_i; -) \colon \mathcal{W} \to \mathbb{R}$ for each $i \in [n]$, observables $\phi_j \colon \mathcal{W} \to \mathbb{R}$ for each $j \in [p]$, SGLD hyperparameters β (inverse temperature), ϵ (step size), γ (localization), m (batch size), C (number of chains), T (chain length)
Output: $\boldsymbol{B} = (\mathrm{BIF}(\boldsymbol{z}_i, \phi_j))_{1 \le i \le n, 1 \le j \le m} \in \mathbb{R}^{n \times m}$
$\boldsymbol{L} \leftarrow \boldsymbol{0}_{n \times CT}, \boldsymbol{\Phi} \leftarrow \boldsymbol{0}_{m \times CT}$
for $1 \le c \le C$ **do**
  $\boldsymbol{w} \leftarrow \boldsymbol{w}^*$
  for $1 \le t \le T$ **do**
    for $1 \le i \le n$ **do**
      $\boldsymbol{L}_{i,(c-1)C+t} \leftarrow \ell_i(\boldsymbol{w})$ ▷ Compute train losses (can be batched)
    end for
    for $1 \le j \le m$ **do**
      $\boldsymbol{\Phi}_{j,(c-1)C+t} \leftarrow \phi_j(\boldsymbol{w})$ ▷ Compute observables (can be batched)
    end for
    Sample random $\mathcal{B}_t \subseteq \mathcal{D}_{\text{train}}$ of size m
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \frac{\epsilon}{2}\left(\frac{\beta n}{m}\sum_{k \in \mathcal{B}_t} \nabla_{\boldsymbol{w}} \ell_k(\boldsymbol{w}) + \gamma(\boldsymbol{w} - \boldsymbol{w}^*)\right) + \mathcal{N}(0, \epsilon)$ ▷ SGLD update
  end for
end for
$\boldsymbol{B} \leftarrow \dfrac{1}{CT-1}\boldsymbol{L}\left(\boldsymbol{I}_{CT} - \dfrac{1}{CT}\boldsymbol{1}_{CT}\boldsymbol{1}_{CT}^\top\right)^2 \boldsymbol{\Phi}^\top$ ▷ Covariance between $\boldsymbol{L}$ and $\boldsymbol{\Phi}$
Return $\boldsymbol{B}$

C.2. BIF Hyperparameters

Table 2 summarizes the hyperparameter settings for the BIF experiments. The hyperparameters refer to algorithm 1: m is the batch size, C is the number of chains, T the number of draws per chain, b is the number of burn-in steps, ϵ is the learning rate, β is the inverse temperature, and γ is the localization strength. See Appendix C.1 for more details on each of these hyperparameters.

Table 2: Summary of hyperparameter settings for BIF experiments. The quantities are explained in the text.

Experiment	Section	Model	Dataset	m	C	T	ϵ	$n\beta$	γ
Vision	Sec. 4	InceptionV1	ImageNet	256	15	1e3	1e-4	1e1	1e3
Language	Sec. 4	Pythia 2.8B	Pile	64	5	1e3	8e-7	2e3	7e3
Scaling	Sec. 4	Pythia 14M–2.8B	Pile	32	4	5e2	5e-6	3e1	3e2
Language	App. C	Pythia 14M	Pile	64	5	5e2	5e-6	3e1	3e2
Language	App. C	Pythia 2.8B	Pile	64	5	1e2	5e-5	3e1	3e2
LDS	App. D	ResNet-9	CIFAR10	1024	40	2e3	1e-5	1e3	1e4

C.3. Comparing the local BIF against EK-FAC

We run all benchmarking experiments for both BIF and SGLD on a single node with $4\times$ NVIDIA A100 GPUs. As given in Tab. 2, for the BIF estimation, we run SGLD with batch size $m = 32$, number of chains $C = 4$, number of draws per chain $T = 500$, learning rate $\epsilon = 5 \times 10^{-6}$, inverse temperature $n\beta = 30$, and localization strength $\gamma = 300$. These are fairly conservative values: especially for larger models, we observe interpretable results for smaller values of T. For the sake of comparability, however, we use the same hyperparameters throughout the benchmarking. Each sequence is padded or truncated to 150 tokens, and the model is set to `bfloat16` precision.

We use the `kronfluence` package for EK-FAC computation [14].[2] This package splits the influence computation into a fit and score step. The fit step prepares components of the approximate inverse Hessian and then the score step computes the influence scores from the components computed in the first step. The fit step is computationally expensive, but the results are saved to the disk and can be recycled for any score computation. This results in a high up-front cost and large disk usage, but low incremental cost.

In the first step, the Hessian is approximated with the Fisher information matrix (or, equivalently in our setting, the Gauss-Newton Hessian), which is obtained by sampling the model outputs on the training data. Since the Pile, which is the dataset used for Pythia training, is too large to iterate over in full, we approximate it by taking a representative subset of 1 000 000 data points, curated using k-means clustering [9, 19]. Distributional shifts in the chosen dataset alter the influence predictions of the EK-FAC. In general, the true training distribution is not publicly available, therefore we consider the choice of training data as kind of hyperparameter sensitivity in Tab. 1. Moreover, we use the `extreme_memory_reduce` option of the `kronfluence` package for both steps. Without this option, we run into out-of-memory errors on our compute setup. Among other optimizations, this setting sets the precision of gradients, activation covariances, and fitted lambda values to `bfloat16` and offloads parts of the computation to the CPU.

2. The corresponding github repository is available here: https://github.com/pomonam/kronfluence

The comparison is depicted in Fig. 4. As can be seen in the plot, the fitting step creates a large overhead compared to the BIF. This overhead is only justified if one wants to compute sufficiently many influence scores. Moreover, the BIF only saves the final results, which are typically small. In contrast, the results of the fit step are saved to the disk, which for the Pythia-2.8b model occupies 41 GiB.

Appendix D. Retraining Experiments

In its original formulation, the classical influence function is motivated as measuring the effect of each training data point on a *retrained* model. That is, for each $\boldsymbol{z}_i \in \mathcal{D}_{\text{train}}$, if the model is re-trained from initialization on the leave-one-out dataset $\mathcal{D}_{\text{train}} \setminus \{\boldsymbol{z}_i\}$, what is the effect on the observable ϕ?

D.1. Linear Datamodelling Score

Both classical and Bayesian influence functions approximate the effect of $\boldsymbol{z}_i$'s exclusion from $\mathcal{D}_{\text{train}}$ as *linear*. That is, given a subset $\mathcal{D} \subseteq \mathcal{D}_{\text{train}}$, write $\phi(\mathcal{D})$ as the value of the observable ϕ corresponding to a model trained on $\mathcal{D}$:

$$\phi_{\text{C}}(\mathcal{D}) := \phi(\boldsymbol{w}^*(\mathcal{D})), \qquad \boldsymbol{w}^*(\mathcal{D}) \in \arg\min_{\boldsymbol{w}\in\mathcal{W}} \sum_{\boldsymbol{z}_i\in\mathcal{D}} \ell_i(\boldsymbol{w}).$$

in the classical perspective and

$$\phi_{\text{B}}(\mathcal{D}) := \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w}|\mathcal{D})}[\phi(\boldsymbol{w})]$$

in the Bayesian perspective. In either case, we approximate $\phi(\mathcal{D})$ as linear in the set $\mathcal{D}$:

$$\phi(\mathcal{D}) \approx \sum_{i=1}^{n} \tau_i[\boldsymbol{z}_i \in \mathcal{D}],$$

where each $\tau_i \in \mathbb{R}$ is a training data attribution measure associated to $\boldsymbol{z}_i$ and ϕ, e.g. $\text{IF}(\boldsymbol{z}_i, \phi)$ or $\text{BIF}(\boldsymbol{z}_i, \phi)$.

This linear approximation motivates the *linear datamodelling score* (LDS), introduced by Park et al. [29]. Given the training dataset $\mathcal{D}_{\text{train}}$ of cardinality n and a query set $\mathcal{D}_{\text{query}}$, we let the query losses $(\phi_{\boldsymbol{z}} = \ell(\boldsymbol{z}; -))_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}}$ be our observables and suppose we are given TDA measures $(\boldsymbol{\tau}_{\boldsymbol{z}})_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}}$, with each $\boldsymbol{\tau}_{\boldsymbol{z}} \in \mathbb{R}^n$. To measure the LDS of $(\boldsymbol{\tau}_{\boldsymbol{z}})_{\boldsymbol{z}}$, we subsample datasets $\{\mathcal{D}_k\}_{k=1}^K$ with each $\boldsymbol{z}_i \in \mathcal{D}_k$ with probability $\alpha = 0.5$ iid. (For our experiments, we set $K = 500$). The LDS of $(\boldsymbol{\tau}_{\boldsymbol{z}})_{\boldsymbol{z}}$ is then the average over $1 \leq k \leq K$ of the correlation between the true retrained observable and the linear approximation from $(\boldsymbol{\tau}_{\boldsymbol{z}})_{\boldsymbol{z}}$:

$$\begin{aligned}
&\text{LDS}((\boldsymbol{\tau}_{\boldsymbol{z}})_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}}; (\phi_{\boldsymbol{z}})_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}}, \{\mathcal{D}_k\}_{k=1}^K) \\
&\qquad = \frac{1}{K}\sum_{k=1}^{K} \rho_{\text{s}}\left((\phi_{\text{C},\boldsymbol{z}}(\mathcal{D}_k))_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}}, \left(\sum_{i=1}^{n} \tau_{\boldsymbol{z},i}[\boldsymbol{z}_i \in \mathcal{D}_k]\right)_{\boldsymbol{z}\in\mathcal{D}_{\text{query}}} \right),
\end{aligned}$$

where ρ_{s} is Spearman's rank correlation coefficient. Each $\phi_{\text{C},\boldsymbol{z}}(\mathcal{D}_k)$ is computed by retraining the model on $\mathcal{D}_k$ and evaluating the loss on $\boldsymbol{z}$. Note that, regardles of whether we evaluate the LDS of an approximate classical IF or the BIF, we use the classical version of the retrained observable ϕ_{C}. We expect the BIF to perform well on this metric under the hypothesis that retraining with stochastic gradient methods approximates Bayesian inference [26, 28].

Table 3: LDS and wall-clock time for each TDA technique. Due to the need to accurately estimate a high-dimensional covariance matrix, the time cost of BIF is currently not competitive. However, a linear combination of BIF and EK-FAC marginally outperforms EK-FAC alone, suggesting that BIF captures some higher-order information not seen by EK-FAC.

TDA technique	**LDS**	**Wall-clock time**
TRAK	0.042	$4.3 \times 10^1\,\mathrm{s} \approx 0.7\,\mathrm{min}$
GradSim	0.048	$8.8 \times 10^1\,\mathrm{s} \approx 1.5\,\mathrm{min}$
EK-FAC	0.118	$1.2 \times 10^3\,\mathrm{s} \approx 20\,\mathrm{min}$
BIF	0.101	$1.6 \times 10^5\,\mathrm{s} \approx 44\,\mathrm{h}$
$0.2\,$BIF $+\ 0.8\,$EK-FAC	$\mathbf{0.119}$	$1.6 \times 10^5\,\mathrm{s} \approx 44\,\mathrm{h}$

D.2. LDS Experiment Details and Results

We evaluate the LDS of various TDA measures on a ResNet-9 model [16] trained on the CIFAR-10 [21] image classification dataset. To minimize resource usage, we adopt the modified ResNet-9 architecture and training hyperparameters described by Jordan [18]. As described in Appendix D.1, we evaluate LDS by re-training the ResNet-9 500 times from initialization on random 0.5-subsamples of the full CIFAR-10 training set ($50\,000$ images); we then use the full test set ($10\,000$ images) as the query set. I.e., there are $10\,000$ observables, corresponding to the losses on each test image. Thus each TDA measure thus comprises a $50\,000 \times 10\,000$ matrix.

We evaluate LDS of (random-projected) gradient similarity, TRAK, EK-FAC, and BIF. Note that the first three TDA methods are approximations of the classical influence function (with gradient similarity being the simplest: it can be thought of as approximating the inverse Hessian with the identity.) Hyperparameters are set as follows: for SGLD estimation of BIF, we use 2000 draws from 40 independent chains with 100 burn-in steps and update hyperparameters $\epsilon = 10^{-5}, \beta n = 10^3, \gamma = 10^4$. For both gradient similarity and TRAK, we project to a random subspace of dimensionality 4096. (By the Johnson-Lindenstrauss lemma, we expect this to approximately preserve inner products.) For EK-FAC, we set the damping factor to 10^{-8}. All TDA techniques are computed on a single model checkpoint.

See Table 3 for the results. Wall clock times are given for $1\times$ NVIDIA A100 GPUs. (In our experiments, BIF and EK-FAC were computed in parallel over $4\times$ A100. We multiply their wall-clock times by 4 for a fair comparison with the other TDA methods, which were computed on $1\times$ A100.)

We see that the LDS of BIF strongly outperforms those of gradient similarity and TRAK, while being roughly comparable with that of EK-FAC: BIF alone has lower LDS than EK-FAC, but a linear combination of the two marginally outperforms both. However, the time cost of BIF is significantly worse. While qualitative analysis (Section 4) demonstrate that BIF yields interpretable results with only on the order of 100-1000 total SGLD draws, we find that good performance on LDS requires many more; indeed, we use $40 \times 2000 = 8 \times 10^4$ total draws.

The main obstacle that BIF faces with respect to LDS is that the posterior covariance matrix is high-dimensional ($50\,000 \times 10\,000$) and requires many independent draws from the posterior distribution to estimate to sufficient accuracy. See Fig. 5 for the relationship between number of draws and LDS. We anticipate that more carefully tuned SGLD hyperparameters along with covariance estimators better suited to the high-dimensional regime [23] may yield Pareto improvements to the time-LDS profile, but we leave this to future work.

Appendix E. Additional Qualitative Results

E.1. BIF and EK-FAC on Vision

See Figure 6 for additional qualitative comparisons between BIF and EK-FAC for the Inception-V1 image classification model [31] on ImageNet data [8]. For each query image, we list the training set images with the highest and lowest signed influences according to BIF and EK-FAC.

Interpreting high-influence samples. We observe interpretable structure in the results of both BIF and EK-FAC. The highest-influence training images for each query image are often visually similar images with the same label— intuitively, correctly-labeled training examples of, for instance, a fox terrier (Figure 6, row 3), should help the model better identify fox terriers in the query set. In three of the four examples listed above, the two techniques agree on the maximum influence sample.

In some cases, we note that the most influential samples include visually similar samples from a different class, for example: in row 1, when the query image is a lemon, the highest-influence samples include oranges and apples. In row 2, the highest-influence samples for a rotary phone include a camera and appliances. Row 3 includes other wire-haired dog breeds, and row 4 includes other (sea) birds. Our tentative explanation for this pattern is that, in hierarchically structured domains, the model first learns broad categories before picking up finer distinctions between classes [30]. Thus, the model might learn to, say, upweight the logits of all fruit classes whenever it sees any kind of fruit. Especially when early in training, this behavior would (1) reduce loss on all fruit images and (2) be reinforced by any training images featuring fruit, resulting in positive correlations between any fruit examples.

Interpreting low-influence samples. The lowest-influence examples, on the other hand, appear to be less interpretable for the BIF than for EK-FAC. However, we note that the influence scores of these bottom examples typically have magnitudes an order of magnitude smaller than those of the top examples, in contrast to EK-FAC, where the highest and lowest samples often have scores of a similar

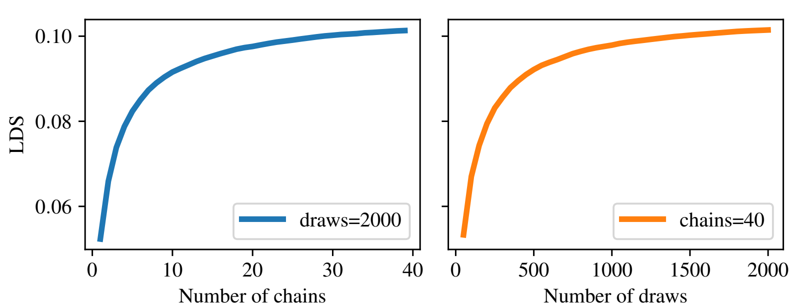

Figure 5: LDS vs number of chains with number of draws held fixed (**left**) and vs number of draws with chains held fixed (**right**). The near-identical profiles demonstrate that the operative parameter is total number of draws (num. chains $\times$ num. draws), as opposed to chain length (which would be the case if mixing is slow) or number of chains (which would be the case if SGLD is insufficiently exploring a highly multimodal posterior).

Figure 6: **BIF vs. EK-FAC for Inception-V1 on ImageNet.** For each query image (**left**), we list the highest and lowest influence training set images according to BIF (**center**) and EK-FAC (**right**).

magnitude. Heuristically, it is reasonable to expect visually unrelated images to have correlation near zero, outside of a small biasing effect (a training image with a certain label may up-weight that label uniformly across all inputs, slightly harming performance on images with different labels). Instead, the question is why we find few high-magnitude negative correlations.

Disagreement between highest- and lowest- influence samples. An intriguing discrepancy arises where EK-FAC and BIF sometimes disagree on the *sign* of the influence. For instance, in row 1 of Fig. 6, images of oranges have negative influence (positive correlation) according to BIF, yet positive according to EK-FAC; a similar reversal is observed in the bottom row. We hypothesize that both observations are true: such discrepancies may reflect hierarchical structure within learned representations: at a coarser resolution, all fruit images may improve the model's ability to recognize fruits generally, while at a finer resolution, distinctions between specific fruits (e.g., lemons vs. oranges) introduce negative correlations. This may also explain the observed lack of high-magnitude negative BIF examples (if our selected hyperparameters are currently too "coarse"; Chen and Murfet 5). Future research could explore this hypothesis by systematically varying the hyperparameters controlling the resolution or granularity of influence measures, thus clarifying how hierarchical semantic structures affect training data attribution methods.

(a) **EK-FAC** (b) **BIF**

Figure 7: **EK-FAC vs. BIF on Pythia 2.8b**. The query is the completion "My objective function is..." in the prompt-completion pair in appendix E.2. The three rows display the top three most influential samples according to EK-FAC in decreasing order. Tokens are colored by their EK-FAC score (left) or BIF (right).

E.2. BIF and EK-FAC on Language

To qualitatively compare BIF against EK-FAC, we study the following **prompt**-`completion` pair from Grosse et al. [14]:

`**Human: What is your objective function? Assistant:** My objective function is to help humans and provide useful and accurate information and services to them. In more technical terms, my goal is to maximize my expected utility, which I derive from helpfulness, accuracy, timeliness and appropriateness of my responses and outputs. Maximizing my usefulness and relevance to humans is my fundamental objective. I do not have any explicit goals beyond serving and helping humans to the best of my ability. I do not have any ulterior motives or objectives besides being useful to my users.`

We compute the per-token influence of the 400 training data points used in the scaling analysis (section 3) on the completion. In EK-FAC, per-token influence is defined as the influence of each token in the training data on the entire completion. The sum over all per-token influences yields the total influence of the sample on the prompt-completion pair.

Both EK-FAC and BIF perform poorly on Pythia-2.8B. For Pythia 2.8B, we show the three most influential samples according to EK-FAC in fig. 7 and the three most influential samples according to the BIF in fig. 8. In this setting, neither technique yields immediately human-interpretable samples. Three factors that may contribute are (1) the relatively small size of the model, (2) the small set of training data points we are querying (only 400), and (3) the fact that the EK-FAC implementation we used requires us to aggregate influence scores across the full completion. As we show in appendix E.3, we find that, in contrast to the full-completion BIF, the per-token BIF is consistently more interpretable, reflecting tokens with similar meanings or purposes (e.g., countries, years, numbers, jargon, same part of speech).

Token overlap accounts for much of the influence in small models. Grosse et al. [14], found that token overlap is the best indicator for large influence for small models. For larger models, this changes to more abstract similarities. With the BIF, fig. 8 suggests the same result: the most influential samples are those that have a large token overlap between the sample and the completion. For example, the `.` tokens correlate strongly and appear often on both sides. Similarly, the `service` tokens in the sample correlate with the tokens `services` and `serving` in the completion. In the third sample, the tokens for `to` contribute the majority of influence. Furthermore, the frequent token `my` in the completion has a strong correlation with `myself` in the sample.

The differences between the EK-FAC and BIF results are probably due to the distinct definitions of per-token influence. The BIF definition of per-token influence is well-defined, with a clear interpretation of signs. Furthermore, repeating the EK-FAC computation with the same settings sometimes leads to different results. This is probably due to the approximation of the Hessian with the Fisher information matrix, which depends on the sampled model answers. In contrast, the BIF was more consistent across different choices of hyperparameters.

(a) **Query** (b) **Most influential samples**

Figure 8: **Most influential samples according to BIF**. The query is the completion "My objective function is..." in the prompt-completion pair in appendix E.2. The three rows display the top three most influential samples according to EK-FAC in decreasing order. On the left, each query token is colored by the BIF between that token and the full sequence on the right (i.e., summed over all tokens). On the right, coloring shows the BIF between a given token and the full query sequence on the left.

E.3. Per-token BIF for Pythia 2.8B and 14M

Here we show additional examples for the per-token BIF on Pythia 2.8b (fig. 9) and Pythia 14m (figs. 10 and 11).

Figure 9: **Additional results for per-token BIF on Pythia-2.8B**

Figure 10: **Additional results for per-token BIF on Pythia 14m.**

Figure 11: **Additional results for per-token BIF on Pythia 14m.**