Final-Model-Only Data Attribution with a Unifying View of Gradient-Based Methods

Dennis Wei
IBM Research
dwei@us.ibm.com

Inkit Padhi
IBM Research
inkpad@ibm.com

Soumya Ghosh*
IBM Research
soumya.ghosh@merck.com

Amit Dhurandhar
IBM Research
adhuran@us.ibm.com

Karthikeyan Natesan Ramamurthy
IBM Research
knatesa@us.ibm.com

Maria Chang
IBM Research
maria.chang@ibm.com

Abstract

In this work, we draw attention to a common setting for training data attribution (TDA) where one has access only to the final trained model, and not the training algorithm or intermediate information from the training process. To serve as a gold standard for TDA in this "final-model-only" setting, we propose *further training*, with appropriate adjustment and averaging, to measure the sensitivity of the given model to training instances. We then unify existing gradient-based methods for TDA by showing that they provide different approximations to the further training gold standard. We investigate empirically the quality of these gradient-based approximations to further training. In general, we find that the approximation quality of first-order methods can be high but decays with the amount of further training. In contrast, the approximations given by influence function methods are more stable but surprisingly lower in quality.

1 Introduction

Training data attribution (TDA, or sometimes simply "data attribution") refers to the attribution or explanation of ML model behavior in terms of its training data. Existing methods for TDA fall into several broad categories: re-training-based approaches [14, 13, 11, 24, 19, 25, 32], gradient-based approaches applied throughout training [18, 29, 8, 2], and gradient approaches applied only at the end [21, 34, 22, 4, 6, 16, 31, 28, 23] (please see the survey [17] for a deeper look at these categories).

In this paper, we draw attention to the fact that there exist multiple *problem settings* for TDA, alongside the multiple categories of methods. These problem settings differ in the level of access assumed. In particular, we focus on what we call the "final-model-only" (FiMO) setting in which we have access only to the final trained model, and not the algorithm used to train the model or intermediate information from training (e.g., checkpoints). The FiMO setting is motivated by the common scenario in which TDA is performed by a different party than the one who developed the model. This is the case for example for models downloaded from platforms such as HuggingFace.

We find that since the TDA literature has not clearly differentiated these problem settings, it is also not clear on what should be the ideal "gold standard" for TDA in the FiMO setting. Having a gold standard (as opposed to proxy tasks such as mislabelled example detection) facilitates the development of more practical methods. We thus propose *further training*, starting from the given final model, as a gold standard measure of the sensitivity of the model to training instances. Our

*Now with Merck Research Labs

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

proposal (Section 3) adjusts for the effect of further training alone (as separate from the effects of particular training instances) and accounts for the randomness of neural network training algorithms.

Like re-training[2], further training a model multiple times can be computationally prohibitive. We thus consider approximations based on first- and second-order Taylor expansions of the further training objective. In doing so, we unify several existing gradient-based methods for TDA. These methods include gradient similarity [7] (a final-checkpoint-only case of TracIn [29]) and various methods based on influence functions [21, 31, 28, 15, 23]. The direct connection that these gradient-based methods have to our further training objective suggests that they are more suited to the FiMO setting, rather than as approximations to re-training where their effectiveness has been questioned [5, 27].

We investigate empirically the quality of the approximations to further training provided by different gradient-based methods. In this workshop submission on ongoing work, we present results on tabular data. We find that first-order methods can give good initial approximations to further training, but the quality of approximation decays with the amount of further training. In contrast, the approximation quality of influence function methods is more persistent, but somewhat surprisingly, never as high as first-order methods at their peak.

2 Problem Settings for Training Data Attribution

Preliminaries In all problem settings that we consider, we are given access to the training dataset $\mathcal{D} = \{\boldsymbol{z}_i\}_{i=1}^n$ for the model, consisting of n pairs $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ of inputs $\boldsymbol{x}_i \in \mathcal{X}$ and target outputs $y_i \in \mathcal{Y}$. A model $f(\boldsymbol{x}; \boldsymbol{\theta})$ is a function $f : \mathcal{X} \to \mathcal{F}$, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^p$, mapping to an output space $\mathcal{F}$ (e.g., predicted logits). The quality of each model output $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ with respect to target y_i is measured by a loss function $\ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i)$; we adopt the more compact notation $L(\boldsymbol{z}_i; \boldsymbol{\theta}) \triangleq \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i)$. We use A to denote the *training algorithm* used to train $f(\boldsymbol{x}; \boldsymbol{\theta})$ on dataset $\mathcal{D}$. We view A as a function that takes $\mathcal{D}$ and initial model parameters $\boldsymbol{\theta}^{(0)}$ as input and outputs final parameters $\boldsymbol{\theta}^f = A(\mathcal{D}, \boldsymbol{\theta}^{(0)})$.

Given a test instance $\boldsymbol{z} = (\boldsymbol{x}, y)$ and an *output function* $g(\boldsymbol{z}, \boldsymbol{\theta})$ that depends in general on both $f(\boldsymbol{x}; \boldsymbol{\theta})$ and y, the task of TDA is to assign scores v_i quantifying the importance of each training instance $\boldsymbol{z}_i$ to the output $g(\boldsymbol{z}, \boldsymbol{\theta})$. We refer to v_i interchangeably as an attribution or influence score. Commonly, the output function is the loss on $\boldsymbol{z}$, $g(\boldsymbol{z}, \boldsymbol{\theta}) = L(\boldsymbol{z}; \boldsymbol{\theta})$, as it is in our experiments.

Three problem settings We distinguish three problem settings for TDA, based on the level of access to the above-defined quantities:

1. **Training Algorithm Available (TAA)**: In this first case, we have access to the training algorithm A. TDA can therefore be done by re-training the model on (i.e., applying A to) different subsets of $\mathcal{D}$ and evaluating the resulting effects. This scenario typically occurs when the parties training the model and performing data attribution are the same.
2. **Checkpoints Available (CPA)**: We do not have access to A but do have intermediate information from the training process, for example intermediate model parameters $\boldsymbol{\theta}^{(t)}$ (i.e., "checkpoints") or gradients $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta}^{(t)})$, and algorithm details such as learning rates. Here, TDA can be done by "tracing" the effect of training instances throughout the process. This scenario is also more typical when model training and data attribution are performed by the same party.
3. **Final Model Only (FiMO)**: We have access only to the final model $f(\boldsymbol{x}; \boldsymbol{\theta}^f)$ and not the training algorithm A or intermediate information. The algorithm A may be actually unavailable, or it may be available but too computationally expensive or otherwise undesirable to re-run. This scenario typically occurs when model training and data attribution are performed by different parties.

The three settings also differ in the object being attributed. In the TAA setting, the object is the algorithm A and its sensitivity to different instances $\boldsymbol{z}_i$. In the CPA setting, it is the training trajectory. The FiMO setting is the most focused on the final model $f(\boldsymbol{x}; \boldsymbol{\theta}^f)$ that will actually be used.

In the remainder of the paper, we focus on the FiMO setting as it requires the least amount of access and is thus the most widely applicable. It applies for example to open-weights models that are freely downloadable from platforms such as HuggingFace. By extension, the FiMO setting applies in cases where methods for the TAA and CPA settings (i.e., re-training and tracing methods) cannot be used.

[2]Throughout the paper, we reserve the term "re-training" to mean re-training from scratch.

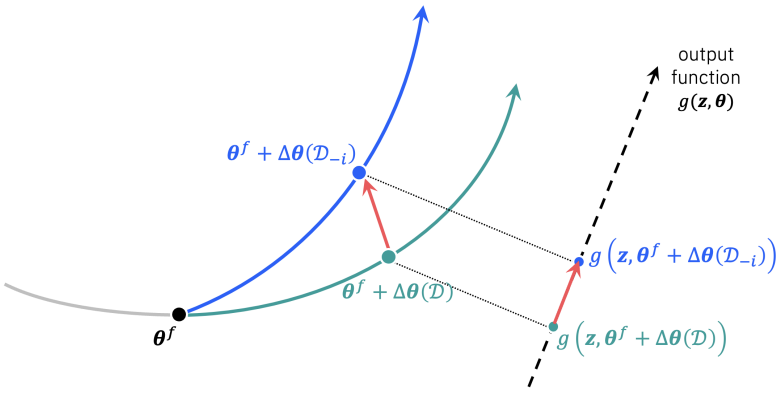


Figure 1: Given only a final model with parameters $\boldsymbol{\theta}^f$, the proposed *further training* gold standard measures the model's sensitivity to training sample i by further training on the full training set $\mathcal{D}$ as well as leaving out sample i ($\mathcal{D}_{-i}$), resulting in changed parameters $\boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D})$ and $\boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i})$. The difference in outputs $g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i})) - g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}))$ between the two further trained models indicates the sensitivity to i. If training is stochastic, then this process should be repeated to obtain the expected sensitivity.

3 A Further Training Gold Standard for the Final-Model-Only Setting

We first consider the question of how TDA should ideally be done in the FiMO setting, i.e., what could serve as a "gold standard" method. While a gold standard method may be very expensive to carry out in many cases, it is nevertheless useful in evaluating and developing approximate TDA methods that are more practical. However, since the TDA literature has not explicitly addressed the FiMO setting to our knowledge, it is also not clear what the gold standard should be.

Returning for a moment to the TAA setting, the natural question to ask there is one of *contribution*: how much does a training instance $\boldsymbol{z}_i$ contribute to the model through the training process? Accordingly, variants of re-training, designed to estimate these contributions, are widely accepted as gold standards in the TAA setting. The simplest of these is leave-one-out (LOO) re-training where each instance $\boldsymbol{z}_i$ is left out of the training set in turn to assess its contribution. In the FiMO setting however, it is not clear how one can "go back in time" to determine contributions to the final model. Instead, we change the question to one of *sensitivity*: how sensitive is the given model to a training instance $\boldsymbol{z}_i$? To measure these sensitivities, we propose variants of *further training* as a gold standard, in a manner analogous to how re-training is used in the TAA setting.

Further training can be described generically as follows. We start from the given parameters $\boldsymbol{\theta}^f$ and train on a dataset $\mathcal{D}'$ using an algorithm A', i.e., $\boldsymbol{\theta}^f + \Delta\boldsymbol{\theta} = A'(\mathcal{D}', \boldsymbol{\theta}^f)$, where $\mathcal{D}'$ and A' are generally different from $\mathcal{D}$ and A. In this work, we restrict attention to LOO further training, corresponding to LOO datasets $\mathcal{D}' = \mathcal{D}_{-i} \triangleq \mathcal{D} \setminus \{\boldsymbol{z}_i\}$ as well as $\mathcal{D}' = \mathcal{D}$. As for A', our desire is for it to be representative of typical neural network (NN) training. We thus assume that A' seeks to solve the following empirical risk minimization problem:

$$\Delta\boldsymbol{\theta}(\mathcal{D}') \approx \arg\min_{\Delta\theta} \; R(\mathcal{D}'; \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}), \qquad R(\mathcal{D}'; \boldsymbol{\theta}) \triangleq \sum_{\boldsymbol{z}_i \in \mathcal{D}'} L(\boldsymbol{z}_i; \boldsymbol{\theta}), \tag{1}$$

where the notation $\boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}$ is meant to indicate that the optimization is initialized at $\boldsymbol{\theta}^f$ ($\Delta\boldsymbol{\theta} = \mathbf{0}$), and $\Delta\boldsymbol{\theta}(\mathcal{D}')$ is the change in parameters resulting from applying A' to $(\mathcal{D}', \boldsymbol{\theta}^f)$.

With further training as a building block, we now address two issues inherent to typical NN training.

Non-convergence: The "final" parameters $\boldsymbol{\theta}^f$ are often not a stationary point of the empirical risk. Hence, as shown in Figure 1, further training on the original training set $\mathcal{D}' = \mathcal{D}$ generally yields a non-zero change $\Delta\boldsymbol{\theta}(\mathcal{D})$. This parameter shift is reflected in the output function as $g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}))$ and can be interpreted as the effect due to further training alone. Similarly, $\Delta\boldsymbol{\theta}(\mathcal{D}_{-i})$ and $g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i}))$ in Figure 1 are the effects of further training without instance $\boldsymbol{z}_i$. The difference

in outputs,

$$g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i})\big) - g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D})\big), \tag{2}$$

can therefore be interpreted as the sensitivity to the presence of $\boldsymbol{z}_i$, where we have adjusted for the effect of further training.

Stochasticity of training: The training algorithm A' also depends on random elements, notably the order of instances in each training epoch. Denoting these random elements as ξ, the change in parameters becomes a function of ξ, $\boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}', \xi) = A'(\mathcal{D}', \boldsymbol{\theta}^f, \xi)$, as does the output function $g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}', \xi))$. We view ξ as a nuisance parameter, an artifact of the stochastic nature of modern training algorithms. Thus, we would ideally like to take the expectation of (2) over ξ to yield

$$v_i^* = \mathbb{E}\left[g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i}, \xi)\big) - g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}, \xi)\big)\right]. \tag{3}$$

This is the *adjusted* and *expected* sensitivity to leaving out instance $\boldsymbol{z}_i$ and is our proposed gold standard attribution score based on further training.

4 Gradient-Based Methods as Approximate Further Training

The further training gold standard described in the previous section is computationally expensive. In the case of LOO further training, the number of further trainings scales with the training set size n, or at least the number of instances for which we wish to estimate attribution scores. If we wish to approximate the expectation in (3) by averaging over multiple realizations of ξ, that further scales the cost. It is natural therefore to consider approximate further training. Accordingly, we now assume that the amount of further training is limited, so that the resulting changes in parameters $\Delta\boldsymbol{\theta}$ are small. In this section, we show that first-order gradient-based methods for TDA can be re-derived from this perspective, and we do the same for influence function methods in Appendix A.2. These derivations suggest that gradient-based methods are better viewed as approximations to further training and better suited to sensitivity analysis in the FiMO setting than for other purposes. It is important to note that we do not assume the given parameters $\boldsymbol{\theta}^f$ are a stationary point for training set $\mathcal{D}$.

Quadratic approximation of further training: The assumption that $\Delta\boldsymbol{\theta}$ is small leads us to consider first- and second-order Taylor expansions of (1):

$$\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}') = \arg\min_{\Delta\boldsymbol{\theta}} R(\mathcal{D}'; \boldsymbol{\theta}^f) + \big(\nabla_{\boldsymbol{\theta}} R(\mathcal{D}'; \boldsymbol{\theta}^f)\big)^T \Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^T \nabla^2_{\boldsymbol{\theta}} R(\mathcal{D}'; \boldsymbol{\theta}^f)\Delta\boldsymbol{\theta} + \frac{\lambda}{2}\|\Delta\boldsymbol{\theta}\|_2^2, \tag{4}$$

where $\nabla_{\boldsymbol{\theta}} R(\mathcal{D}'; \boldsymbol{\theta}^f)$ and $\nabla^2_{\boldsymbol{\theta}} R(\mathcal{D}'; \boldsymbol{\theta}^f)$ denote the gradient and Hessian of the empirical risk with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^f$, and the first-order expansion omits the Hessian term. In both cases, an ℓ_2 regularizer $(\lambda/2)\|\Delta\boldsymbol{\theta}\|_2^2$ has been added in (4), corresponding to the "damping" term in influence function estimation. As discussed in Appendix A.1, the regularizer can be motivated in two ways: 1) enforcing the smallness of $\Delta\boldsymbol{\theta}$ and the accuracy of the Taylor expansion; 2) ensuring that solutions to (4) do not diverge.

Linear approximation of output function: We may similarly expand the output function $g(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta})$ to first order in $\Delta\boldsymbol{\theta}$. Doing so reduces the difference in (2) to the inner product

$$\hat{v}_i = \big(\nabla_{\boldsymbol{\theta}} g(\boldsymbol{z}, \boldsymbol{\theta}^f)\big)^T \big(\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i}) - \widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})\big). \tag{5}$$

First-order expansion recovers Grad-Dot: In the case of first-order Taylor expansion, the absence of the Hessian term in (4) makes its solution straightforward. For $\mathcal{D}' = \mathcal{D}$, we have $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = -\lambda^{-1}\nabla_{\boldsymbol{\theta}} R(\mathcal{D}, \boldsymbol{\theta}^f)$, and similarly $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i}) = -\lambda^{-1}(\nabla_{\boldsymbol{\theta}} R(\mathcal{D}, \boldsymbol{\theta}^f) - \nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i, \boldsymbol{\theta}^f))$. Hence (5) becomes

$$\hat{v}_i = \lambda^{-1}\big(\nabla_{\boldsymbol{\theta}} g(\boldsymbol{z}, \boldsymbol{\theta}^f)\big)^T \nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i, \boldsymbol{\theta}^f), \tag{6}$$

which is proportional to the inner product between training loss and test output gradients. This corresponds to the "Grad-Dot" method [7], and also to a special case of TracIn [29] that uses only the final checkpoint. A variation is to use the cosine similarity between gradients ("Grad-Cos") instead of the unnormalized inner product.

5 Related Work

This work builds upon existing works that also take a retrospective look at TDA methods, and influence functions in particular [5, 35, 3, 30, 27]. We have a deeper discussion of individual papers in Appendix B. Here, we note the key differences between our work and these prior works:

1. **More gradient-based methods**: Our unified view (Section 4 and the Appendix A) and numerical comparisons (Section 6) encompass gradient-based methods beyond influence functions, including more recent methods such as TRAK [28] and DataInf [23] as well as first-order methods. We obtain insights into similarities and differences among gradient-based methods, in addition to their performance on a task (approximation of further training in our case). In contrast, [5, 35, 3, 30] mainly focus on influence functions, while [27] evaluates fewer methods than we do.
2. **Further training**: The further training gold standard in Section 3 is a formal presentation of similar procedures used in previous works (in the appendix in [3, App D.2], in the experiment in [30, Sec. 5.2]). The proposal of averaging over realizations of random further training appears to be new, and our results in Appendix C.2 show that it brings further training closer to what gradient-based methods estimate. The authors of [3] propose an alternative gold standard called the proximal Bregman response function (PBRF). The PBRF however involves a non-standard Bregman distance objective, tailored specifically to be closer to influence functions, whereas our further training is more generic.
3. **FiMO setting**: We explicitly define the FiMO setting, which these previous works have not.

6 Numerical Comparison of Gradient-Based Methods to Further Training

We performed experiments to assess the quality of the approximations to further training provided by different gradient-based TDA methods. For this workshop submission, we confine ourselves to reporting on experiments with tabular data. Experiments on image and text data are ongoing.

6.1 Experimental setup

A full description of our experimental setup can be found in Appendix C.1.

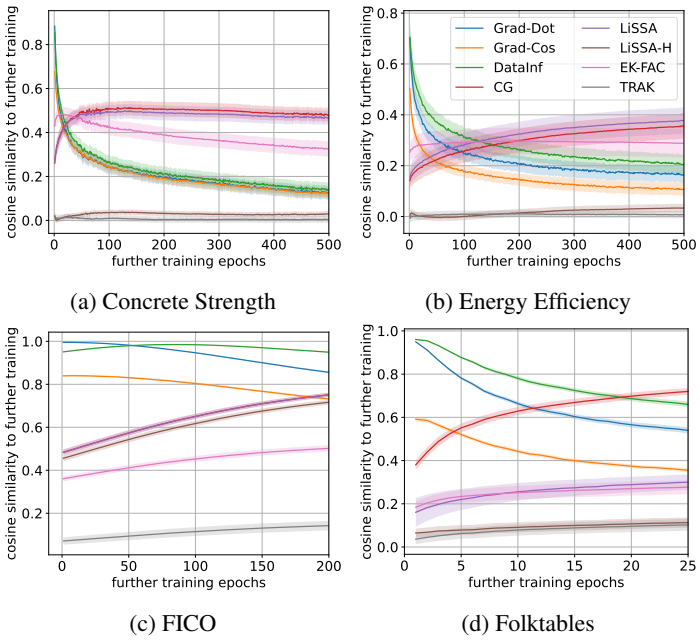
Datasets and "final" models: We used four tabular datasets: two for regression, Concrete Strength and Energy Efficiency from the UCI repository [20], and two larger ones for classification, FICO Challenge [12] and Folktables [10]. For all datasets, we used a 2-hidden-layer multi-layer perceptron (MLP) with 128 units in each hidden layer. The MLPs were trained using stochastic gradient descent (SGD) to yield the final model $f(\boldsymbol{x}, \boldsymbol{\theta}^f)$.

Selection of test and training instances: Our aim was to estimate influence scores of a subset $\mathcal{L} \subset \mathcal{D}$ of training instances on the losses for a subset of m test instances (i.e., $g(\boldsymbol{z}, \boldsymbol{\theta}) = L(\boldsymbol{z}; \boldsymbol{\theta})$). We selected $l = |\mathcal{L}| = 100$ training instances and $m = 100$ test instances at random, except for the Concrete and Energy datasets where all test instances were used.

Implementation of further training: For the results in this section, the further training algorithm A' is also SGD. Appendix C.2 has results using Adam as A' instead. We evaluated the approximation quality of gradient-based methods as a function of the number of further training epochs. To approximate the expectation in (3), we took averages over $r = 100$ random seeds (denoted as realizations $\xi^{(s)}$ of random variable ξ) that control the order of instances in each training epoch.

We considered two methods for adjusting for the effect of further training alone. The first is as specified in (2), (3), i.e., subtracting the output due to further training on the full dataset $\mathcal{D}$. We found however that for the Folktables dataset, this adjustment left a non-negligible bias that varies from epoch to epoch, resulting in noisy cosine similarities (please see Appendix C.2 for these results). The second method used in this section subtracts the mean of the effects due to each LOO dataset $\mathcal{D}_{-i}$. Incorporating also the averaging over random seeds, the "gold" attribution score from further training is thus

$$v_i = \frac{1}{r} \sum_{s=1}^{r} \left[g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i}, \xi^{(s)})\big) - \frac{1}{l} \sum_{i' \in \mathcal{L}} g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i'}, \xi^{(s)})\big) \right]. \tag{7}$$



(a) Concrete Strength (b) Energy Efficiency

(c) FICO (d) Folktables

Figure 2: Cosine similarity between attribution scores of gradient-based TDA methods and further training, as a function of the amount of further training.

Evaluation metric: For each test instance evaluated, we have a vector $\boldsymbol{v} \in \mathbb{R}^l$ of gold attribution scores from further training (7), and corresponding vectors $\hat{\boldsymbol{v}}$ from approximate methods. We use the cosine similarity between $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}$ as the evaluation metric, as justified in Appendix C.1.

6.2 Results

Figure 2 shows the cosine similarity between the attribution scores of gradient-based TDA methods and further training, as a function of the number of epochs of further training. The curves represent the mean cosine similarity over the m test instances, while the shaded area corresponds to one standard error above and below the mean. We make the following observations:

- **First-order methods:** The two first-order methods, Grad-Dot and Grad-Cos, have cosine similarity curves that decay with the amount of further training. In particular, the initial cosine similarity of Grad-Dot is among the highest achieved by any approximate method. This aligns with the derivation of Grad-Dot in Section 4 as the direct result of first-order approximation of further training, while the normalized Grad-Cos is not supported by the theory. The rate of decay after the initial peak varies with the dataset. One reason may be that final model parameters $\boldsymbol{\theta}^f$ that are farther from stationary lead to worse first-order approximations.
- **DataInf [23]:** We find it interesting that DataInf behaves like a first-order method, despite its attempt to incorporate second-order information. It is especially similar to Grad-Dot (cosine similarity between the two is typically above 0.95) and generally slightly higher in similarity to further training.
- **Conjugate Gradient (CG), LiSSA, LiSSA-H:** These are influence function methods that iteratively approximate the inverse Hessian-gradient product. CG and LiSSA use the Gauss-Newton approximation to the Hessian, while LiSSA-H uses the true Hessian. Unlike the first-order methods, their cosine similarity curves do not decay and even increase with further training. However, they do not attain cosine similarities as high as the first-order methods (at least within the amount of further training that we conducted). A possible interpretation is that influence function methods provide better longer-range approximations to further training by using second-order information. However, the approximation may not be especially good at any point.
- **EK-FAC [15] and TRAK [28]:** These methods are also derived from influence functions with additional approximations. EK-FAC has higher cosine similarity than CG and LiSSA on Concrete

and Energy in early epochs, but is worse otherwise. TRAK did not give good approximations to further training in these experiments.

In Appendix C.2, we show the effect of varying the number r of seeds averaged. We find that the cosine similarities improve in all cases with the amount of averaging of further training.

7 Discussion

We have proposed a further training gold standard for TDA in the final-model-only setting that we highlight, and have shown how several gradient-based TDA methods approximate further training in different ways. We welcome discussion of the topics listed in Appendix D and beyond.

References

[1] N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.

[2] J. Bae, W. Lin, J. Lorraine, and R. Grosse. Training data attribution via approximate unrolled differentiation, 2024. arXiv preprint arXiv:2405.12186.

[3] J. Bae, N. Ng, A. Lo, M. Ghassemi, and R. B. Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

[4] E. Barshan, M.-E. Brunet, and G. K. Dziugaite. RelatIF: Identifying explanatory training samples via relative influence. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1899–1909. PMLR, 26–28 Aug 2020.

[5] S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] S. Basu, X. You, and S. Feizi. On second-order group influence functions for black-box predictions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 715–724. PMLR, 13–18 Jul 2020.

[7] G. Charpiat, N. Girard, L. Felardos, and Y. Tarabalka. Input similarity from the neural network perspective. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] Y. Chen, B. Li, H. Yu, P. Wu, and C. Miao. HyDRA: Hypergradient data relevance analysis for interpreting deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7081–7089, May 2021.

[9] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-region methods*. Society for Industrial and Applied Mathematics, USA, 2000.

[10] F. Ding, M. Hardt, J. Miller, and L. Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] V. Feldman and C. Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc., 2020.

[12] FICO. Explainable machine learning challenge, 2018. `https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2`.

[13] A. Ghorbani, M. Kim, and J. Zou. A distributional framework for data valuation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3535–3544. PMLR, 13–18 Jul 2020.

[14] A. Ghorbani and J. Zou. Data Shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 09–15 Jun 2019.

[15] R. Grosse, J. Bae, C. Anil, N. Elhage, A. Tamkin, A. Tajdini, B. Steiner, D. Li, E. Durmus, E. Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

[16] H. Guo, N. Rajani, P. Hase, M. Bansal, and C. Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[17] Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, May 2024.

[18] S. Hara, A. Nitanda, and T. Maehara. Data cleansing for models trained with SGD. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[19] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9525–9587. PMLR, 17–23 Jul 2022.

[20] M. Kelly, R. Longjohn, and K. Nottingham. The UCI machine learning repository. `https://archive.ics.uci.edu`.

[21] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[22] P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[23] Y. Kwon, E. Wu, K. Wu, and J. Zou. DataInf: Efficiently estimating data influence in loRA-tuned LLMs and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[24] Y. Kwon and J. Zou. Beta Shapley: A unified and noise-reduced data valuation framework for machine learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8780–8802. PMLR, 28–30 Mar 2022.

[25] J. Lin, A. Zhang, M. Lécuyer, J. Li, A. Panda, and S. Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13468–13504. PMLR, 17–23 Jul 2022.

[26] J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 735–742, Madison, WI, USA, 2010. Omnipress.

[27] E. Nguyen, M. Seo, and S. J. Oh. A Bayesian approach to analysing training data attribution in deep learning. *Advances in Neural Information Processing Systems*, 2023.

[28] S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27074–27113, 23–29 Jul 2023.

[29] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[30] A. Schioppa, K. Filippova, I. Titov, and P. Zablotskaia. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 36, 2023.

[31] A. Schioppa, P. Zablotskaia, D. Vilar, and A. Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186, 2022.

[32] J. T. Wang and R. Jia. Data Banzhaf: A robust data valuation framework for machine learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6388–6421. PMLR, 25–27 Apr 2023.

[33] D. Wei. Decision-making under selective labels: Optimal finite-domain policies and beyond. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11035–11046. PMLR, 18–24 Jul 2021.

[34] C.-K. Yeh, J. Kim, I. E.-H. Yen, and P. K. Ravikumar. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[35] R. Zhang and S. Zhang. Rethinking influence functions of neural networks in the over-parameterized regime. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9082–9090, Jun. 2022.

A More on Gradient-Based Methods

A.1 ℓ_2 regularization/damping term

The following reasons provide optimization-based justification for the ℓ_2 regularizer $(\lambda/2)\|\Delta\boldsymbol{\theta}\|_2^2$ added in (4). This corresponds to the "damping" term $\lambda\boldsymbol{I}$ added to the Hessian in estimating influence functions, where it is usually motivated simply as a means to ensure invertibility/positive definiteness.

1. **Enforce assumption/trust region:** Penalizing the ℓ_2 norm of $\Delta\boldsymbol{\theta}$ enforces the assumption that $\Delta\boldsymbol{\theta}$ is small. It can also be viewed as enforcing a *trust region* [9], a region around $\boldsymbol{\theta}^f$ where the second-order expansion gives a better approximation.
2. **Ensure optimization is well-posed:** The loss function $L(\boldsymbol{z};\boldsymbol{\theta})$ for neural networks is typically non-convex, which implies that the empirical risk Hessian $\nabla^2_{\boldsymbol{\theta}} R(\mathcal{D}';\boldsymbol{\theta}^f)$ may not be positive semidefinite. In the absence of the regularizer, if $\nabla^2_{\boldsymbol{\theta}} R(\mathcal{D}';\boldsymbol{\theta}^f)$ has a negative eigenvalue, then the objective value in (4) would tend to $-\infty$ in the direction of the corresponding eigenvector. If we add regularization with λ larger in magnitude than the most negative eigenvalue of $\nabla^2_{\boldsymbol{\theta}} R(\mathcal{D}';\boldsymbol{\theta}^f)$, then such divergence is prevented and an optimal solution to (4) exists.

A.2 Influence function methods

To show that the approximate further training formulation in (4), (5) recovers influence function-based TDA methods, we first present two ingredients: influence functions for (4), and the Gauss-Newton approximation to the Hessian that is commonly made. Throughout this section, we assume that the damping parameter λ is large enough as discussed in Appendix A.1 so that the damped risk Hessian $\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda\boldsymbol{I}$ is invertible.

A.2.1 Influence functions for (4)

Influence functions approximate the effect of down-weighting the training loss of instance i by a small amount ϵ, i.e., of changing the empirical risk from the full-dataset risk $R(\mathcal{D};\boldsymbol{\theta})$ to $R(\mathcal{D};\boldsymbol{\theta}) - \epsilon L(\boldsymbol{z}_i;\boldsymbol{\theta})$. We use $\mathcal{D}_{-i,\epsilon}$ to denote this down-weighted dataset. We apply the implicit function theorem to the stationary point condition for (4) to derive the following.

Proposition 1. *The change in parameters due to down-weighting training instance $\boldsymbol{z}_i$ by an amount ϵ is approximated by influence functions as*

$$\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i,\epsilon}) - \widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) \approx \epsilon \left(\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda\boldsymbol{I}\right)^{-1} \left(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) + \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f)\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})\right), \quad (8)$$

where $\boldsymbol{H}(\boldsymbol{\theta}^f) = \nabla^2_{\boldsymbol{\theta}} R(\mathcal{D};\boldsymbol{\theta}^f)$ is the Hessian of the empirical risk evaluated at $\boldsymbol{\theta}^f$.

Proof. We substitute the down-weighted risk $R(\mathcal{D};\boldsymbol{\theta}) - \epsilon L(\boldsymbol{z}_i;\boldsymbol{\theta})$ for $R(\mathcal{D};\boldsymbol{\theta})$ everywhere in (4). Then the stationary point (i.e., zero-gradient) condition for (4) becomes

$$F(\epsilon, \Delta\boldsymbol{\theta}) \triangleq \nabla_{\boldsymbol{\theta}} R(\mathcal{D};\boldsymbol{\theta}^f) - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) + \left(\boldsymbol{H}(\boldsymbol{\theta}^f) - \epsilon \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) + \lambda I\right) \Delta\boldsymbol{\theta} = \mathbf{0}. \quad (9)$$

For $\epsilon = 0$, i.e., no down-weighting, (9) is satisfied by the full-dataset parameter change

$$\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = -\left(\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda \boldsymbol{I}\right)^{-1} \nabla_{\boldsymbol{\theta}} R(\mathcal{D};\boldsymbol{\theta}^f). \quad (10)$$

For small ϵ, we apply the implicit function theorem to obtain an expression for $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i,\epsilon})$. The relevant Jacobians of function F defined in (9) are

$$J_{F,\Delta\boldsymbol{\theta}} = \boldsymbol{H}(\boldsymbol{\theta}^f) - \epsilon \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) + \lambda \boldsymbol{I},$$
$$J_{F,\epsilon} = -\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) - \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f)\Delta\boldsymbol{\theta},$$

from which we obtain

$$\begin{aligned}\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i,\epsilon}) - \widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) &= -\epsilon \left(J_{F,\Delta\boldsymbol{\theta}}\big|_{\epsilon=0,\Delta\boldsymbol{\theta}=\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})}\right)^{-1} J_{F,\epsilon}\big|_{\epsilon=0,\Delta\boldsymbol{\theta}=\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})} + O(\epsilon^2) \\ &\approx \epsilon \left(\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda \boldsymbol{I}\right)^{-1} \left(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f) + \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f)\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})\right).\end{aligned}$$

□

If $\boldsymbol{\theta}^f$ is a stationary point of the full-dataset risk $R(\mathcal{D};\boldsymbol{\theta})$, then $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$ and (8) reduces to the familiar form of a (damped) inverse Hessian-gradient product:

$$\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i,\epsilon}) \approx \epsilon \left(\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda \boldsymbol{I}\right)^{-1} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}^f).$$

If $\boldsymbol{\theta}^f$ is not stationary however, then $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) \neq \mathbf{0}$ as discussed before. Previous works like [21] still neglect the last $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$ term in (8), arguing that if $\boldsymbol{\theta}^f$ is near-stationary, then $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$ is small and the product $\epsilon\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$ is second-order. It is also convenient to neglect this $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$ term because one would otherwise have to compute $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$. We will neglect it as well in the present work to be consistent with previous methods and assuming it is not feasible to compute $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$. Future work however could revisit this issue.

A.2.2 Gauss-Newton approximation to Hessian

It is often the case that the loss function $L(\boldsymbol{z}_i;\boldsymbol{\theta})$ is a composition $\bar{\ell}(\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}))$ of a *scalar-valued* model output function $\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})$ with a convex univariate loss function $\bar{\ell}(\bar{f})$. Here we allow $\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})$ to also depend on the target value y_i. For example, if $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ is a regression model, $\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}) = f(\boldsymbol{x}_i;\boldsymbol{\theta}) - y_i$ can be the prediction error. For multi-class classification, [28] defines $\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})$ to be the logit of the predicted probability of the target class y_i, which allows multi-class classification to be handled in the same way (see [28, Sec. 3.3] for details).

Given the compositional form $L(\boldsymbol{z}_i;\boldsymbol{\theta}) = \bar{\ell}(\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}))$, the gradient and Hessian of $L(\boldsymbol{z}_i;\boldsymbol{\theta})$ are given by

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}) = \bar{\ell}'(\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}), \quad (11)$$

$$\nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i;\boldsymbol{\theta}) = \bar{\ell}''(\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})^T + \bar{\ell}'(\bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}))\nabla^2_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}). \quad (12)$$

The Gauss-Newton approximation to the Hessian drops the second term in (12), which requires the Hessian $\nabla^2_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta})$ that is more difficult to compute. To obtain more compact expressions, let us define the gradient vectors $\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{z}_i;\boldsymbol{\theta}^f)$, $i = 1,\ldots,n$ and matrix $\boldsymbol{G} = \begin{bmatrix}\boldsymbol{g}_1 & \ldots & \boldsymbol{g}_n\end{bmatrix}^T$, vector $\boldsymbol{r} = -\begin{bmatrix}\bar{\ell}'(\bar{f}(\boldsymbol{z}_1;\boldsymbol{\theta}^f)) & \ldots & \bar{\ell}'(\bar{f}(\boldsymbol{z}_n;\boldsymbol{\theta}^f))\end{bmatrix}^T$, and $n \times n$ diagonal matrix $\boldsymbol{V}$ with $\bar{\ell}''(\bar{f}(\boldsymbol{z}_1;\boldsymbol{\theta}^f)),\ldots,\bar{\ell}''(\bar{f}(\boldsymbol{z}_n;\boldsymbol{\theta}^f))$ as its diagonal entries. Using these definitions, (11), and the Gauss-Newton simplification of (12), the quadratic optimization in (4) can be rewritten for $\mathcal{D}' = \mathcal{D}$ as

$$\min_{\Delta\boldsymbol{\theta}} -\boldsymbol{r}^T\boldsymbol{G}\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^T(\boldsymbol{G}^T\boldsymbol{V}\boldsymbol{G} + \lambda\boldsymbol{I})\Delta\boldsymbol{\theta}. \quad (13)$$

where we have also disregarded the constant term.

The same derivation of influence functions in the proof of Proposition 1 applies to the Gauss-Newton approximation. Effectively, (11) and (12) (with the Gauss-Newton simplification) are used to substitute for the gradient and Hessian in (8).

Corollary 1. *Under the Gauss-Newton approximation, the influence function in Proposition 1 becomes*

$$\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}_{-i,\epsilon}) - \widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \epsilon \left(v_{ii} \boldsymbol{g}_i^T \widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) - r_i \right) (\boldsymbol{G}^T \boldsymbol{V} \boldsymbol{G} + \lambda \boldsymbol{I})^{-1} \boldsymbol{g}_i. \tag{14}$$

Proof. The result follows from (8) by substituting the Gauss-Newton Hessian $\boldsymbol{H}(\boldsymbol{\theta}^f) = \boldsymbol{G}^T \boldsymbol{V} \boldsymbol{G}$, $\nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{z}_i; \boldsymbol{\theta}) = v_{ii} \boldsymbol{g}_i \boldsymbol{g}_i^T$, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i; \boldsymbol{\theta}) = -r_i \boldsymbol{g}_i$, and simplifying. □

As before, this is a generalization of the standard influence function to non-stationary $\boldsymbol{\theta}^f$ and non-zero $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$, and reduces to the standard form used in previous works when $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$.

A.2.3 Relationships with existing influence function methods

We now discuss how existing influence function and influence function-like methods correspond to specific cases of Proposition 1 and Corollary 1.

Conjugate gradient (CG) and LiSSA Both (8) and (14) require computing an inverse Hessian-vector product, with either the damped true Hessian $\boldsymbol{H}(\boldsymbol{\theta}^f) + \lambda \boldsymbol{I}$ or the Gauss-Newton Hessian $\boldsymbol{G}^T \boldsymbol{V} \boldsymbol{G} + \lambda \boldsymbol{I}$. For models with a large number of parameters p, the $O(p^3)$ computational cost of directly solving the system of equations is prohibitive. CG and LiSSA are two methods that approximate the solution iteratively, the former by applying the CG algorithm to an equivalent quadratic minimization problem, the latter using a truncated Neumann series. Both use repeated Hessian-vector products (HVPs) and can be applied to either (8) or (14) (with $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$). We refer to the works [26, 1] proposing these methods for more details.

Arnoldi influence functions [31] proposes to use Arnoldi iteration (a.k.a. Lanczos iteration for symmetric matrices) to approximately compute the largest-magnitude eigenvalues and corresponding eigenvectors of $\boldsymbol{H}(\boldsymbol{\theta}^f)$. This involves constructing the Krylov subspace $\mathrm{Span}\{\boldsymbol{v}, \boldsymbol{H}(\boldsymbol{\theta}^f)\boldsymbol{v}, \ldots, \boldsymbol{H}(\boldsymbol{\theta}^f)^T \boldsymbol{v}\}$ starting with an arbitrary vector $\boldsymbol{v}$, which again requires repeated HVPs with $\boldsymbol{H}(\boldsymbol{\theta}^f)$. The method then projects gradient vectors into the subspace of top eigenvectors of $\boldsymbol{H}(\boldsymbol{\theta}^f)$. Once in this subspace, the matrix inversion in (8) (again with $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$) reduces to scalar division by eigenvalues.

The following methods are based on and therefore specific to the Gauss-Newton approximation.

TRAK TRAK [28] can be seen as a hybrid data attribution method that combines a gradient-based approximation with re-training (training an ensemble of models). Under the FiMO setting where re-training is not possible, we focus on the gradient-based part of TRAK by setting its parameter M (number of trained models) to 1. This variant, which we refer to as $\mathrm{TRAK}_{M=1}$, falls under the Gauss-Newton framework of this section. First, TRAK applies a random projection matrix $\boldsymbol{P} \sim \mathcal{N}(0,1)^{p \times k}$ to reduce the dimensionality of the gradients: $\boldsymbol{\phi}_i = \boldsymbol{P}^T \boldsymbol{g}_i$ and $\boldsymbol{\Phi} = \boldsymbol{G}\boldsymbol{P}$. With $\boldsymbol{\phi}_i$ and $\boldsymbol{\Phi}$ in place of $\boldsymbol{g}_i$ and $\boldsymbol{G}$, $\mathrm{TRAK}_{M=1}$ corresponds to a special case of (14) with $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$ (stationary $\boldsymbol{\theta}^f$), $\lambda = 0$ (no regularization), and $\boldsymbol{V} = \boldsymbol{I}$ (non-identity $\boldsymbol{V}$ found empirically to have little effect):[3]

$$\Delta\theta_{\mathrm{TRAK}}(\mathcal{D}_{-i}) = -r_i (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\phi}_i. \tag{15}$$

DataInf The DataInf method [23] starts with a variant of the Gauss-Newton approximation (which they justify for loss functions that are negative log-likelihoods) in which $\bar{\ell}(\bar{f}) = \bar{f}$ is the identity function and $\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i; \boldsymbol{\theta}^f)$ is a gradient of the loss. DataInf also computes influence functions separately for each layer in a neural network (equivalent to making a block-diagonal approximation to the Hessian). Setting aside this layer-wise simplification, DataInf corresponds to (14) with $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D}) = \mathbf{0}$ (stationary $\boldsymbol{\theta}^f$), $r_i = -1$ (identity $\bar{\ell}$), and $\boldsymbol{V} = (1/n)\boldsymbol{I}$ (identity $\bar{\ell}$, average over $\mathcal{D}$ instead of sum). Their key idea is to interchange the order of averaging and matrix inversion, i.e.,

$$\left(\frac{1}{n} \boldsymbol{G}^T \boldsymbol{G} + \lambda \boldsymbol{I} \right)^{-1} = \left(\frac{1}{n} \sum_{i=1}^n \boldsymbol{g}_i \boldsymbol{g}_i^T + \lambda \boldsymbol{I} \right)^{-1} \approx \frac{1}{n} \sum_{i=1}^n \left(\boldsymbol{g}_i \boldsymbol{g}_i^T + \lambda \boldsymbol{I} \right)^{-1}. \tag{16}$$

[3] [28] actually begins with an exact leave-one-out formula but finds that the denominator also has little effect.

They then use the Sherman-Morrison formula to express $(\boldsymbol{g}_i \boldsymbol{g}_i^T + \lambda \boldsymbol{I})^{-1}$ in closed form, resulting in an approximation to (5) that involves only gradient inner products. Please see [23] for more details.

B More on Related Work

In this appendix, we discuss some more closely related works individually.

In [3], the authors examine the discrepancy between influence functions and leave-one-out re-training and decompose the discrepancy into five contributions. They show that influence functions are a much better approximation to a quantity they call the proximal Bregman response function (PBRF), which they propose as an alternative gold standard to LOO re-training. The key differences between our work and [3] are as follows: 1) We propose further training based on standard training objectives (e.g., cross-entropy or mean squared error, no proximity regularization) as a different, more general gold standard. In contrast, the PBRF involves a non-standard Bregman distance objective, chosen specifically to be closer to influence functions. (In their Appendix D.2, [3] also discuss further training with and without leaving out one sample, which they refer to as "two-stage LOO re-training." However, they do not consider taking an expectation over random trainings as we do.) 2) Our unified view encompasses TDA methods beyond influence functions, including more recent methods such as TRAK [28] and DataInf [28] as well as first-order methods. We evaluate the extent to which all these methods approximate further training.

In [30], the authors discuss problematic assumptions made by influence functions and TracIn [29] and how these can be addressed. They then show that the predictive power of influence functions and TracIn is fundamentally limited by divergence in parameters as (further) training proceeds. The latter contribution is the main point of connection with our work, where we also observe the fading of predictive power over training time. The differences in our work are: 1) We make explicit the further training gold standard that they compare to in their experiment [30, Sec. 5.2] and we further propose its expected version with respect to stochasticity in the training algorithm. 2) We conduct a larger-scale experiment than theirs in certain dimensions, notably the TDA methods evaluated, as well as averaging over the aforementioned stochasticity and using more test points and left-out training points.

In [5], the authors show that influence functions become worse approximators of a training example's importance on a test prediction with increasing model depth, width and if the models are trained with no weight decay. All experiments in the paper are conducted with further training (from the already trained model, not from scratch) determining the ground truth influence. However, [5] does not average over further training realizations as we do. Moreover, their study is focused on influence functions, whereas we consider other gradient-based methods as well.

In [27], the authors argue that a Bayesian approach is the correct way of evaluating influences of training examples on test predictions as there is a lot of variance in most TDA estimates when we account for random initialization of the model as well as variation in batching when training. In such situations the recommendation is to consider the distribution of attribution values for a training example, or at least consider the distribution's variance, rather than simply (individual run) point estimates. Different from our work, the experiments in the paper are conducted by re-training models from scratch, and only three gradient-based TDA methods are studied, namely influence functions (IF), Grad-Dot (GD) and Grad-Cos (GC).

The seminal paper [21] on influence functions in ML sketched a derivation similar to ours in Section 4 and Appendix A, but in less detail. In particular, our more careful derivation of the influence function in Proposition 1 keeps the $\widehat{\Delta\boldsymbol{\theta}}(\mathcal{D})$ term that [21] neglects. Appendix A.1 also discusses the damping term $\lambda \boldsymbol{I}$ at greater length.

C Additional Experimental Details and Results

C.1 Experimental setup details

Datasets We followed [3] in using two regression datasets from the UCI repository [20], Concrete Strength and Energy Efficiency. We then departed from [3] in choosing two classification datasets,

FICO Challenge [12] and Folktables [10], because they are at least one order of magnitude larger than the tabular datasets in [3] in terms of the number of instances and/or features.

Data pre-processing For Concrete, Energy, and FICO, we split the dataset 90%-10% into training and test sets and standardized the features to have zero mean and unit variance. Handling of special feature values in FICO was done as in [33]. For Folktables, we used the person-level data from year 2018 for the US state of Massachusetts. The classification task was predicting whether a person's income is above or below USD 50,000 (task `ACSIncome`). The dataset was split 75%-25% into training and test, categorical features were dummy-coded, and numerical features were standardized.

"Final" Models For all datasets, we used a 2-hidden-layer multi-layer perceptron (MLP) with 128 units in each hidden layer. For the regression problems, the MLP has one output and the loss function is mean squared error (MSE), while for (binary) classification, the MLP has two outputs (logits) and the loss function is cross-entropy. The MLPs were trained using SGD for $T = 1000$ epochs and a batch size of 128 (following [3]) to yield the final model $f(\boldsymbol{x}, \boldsymbol{\theta}^f)$, except for the larger Folktables dataset where $T = 100$. Learning rates were as follows: 0.3 for Concrete and Energy, 0.001 for FICO, and 0.01 for Folktables.

Implementation of further training For the case in which the further training algorithm A' is also SGD ($A' = A$), the learning rate for A' was chosen to be one order of magnitude smaller than that for A (i.e., 0.03 for Concrete and Energy, 10^{-4} for FICO and Folktables) since $\boldsymbol{\theta}^f$ is closer to convergence than the initial parameters $\boldsymbol{\theta}^{(0)}$. The maximum number of epochs T' for A' was also chosen to be a fraction of T: $T' = 500$ for Concrete and Energy following [3], $T' = 200$ for FICO, $T' = 25$ for Folktables. Other parameters such as batch size were the same as for A. For the case in which A' was Adam, smaller learning rates were used since Adam is a more powerful optimizer: 10^{-4} for Concrete and Energy, 10^{-6} for FICO and Folktables.

Regarding the random seeds (denoted as $\xi^{(s)}$ in (7)) that determine the order of instances in each epoch, this control was achieved in PyTorch as follows: First, the random seed of a random number `Generator` object was set to a specified value. Then, a `RandomSampler` object was instantiated with this `Generator`, and a `DataLoader` was instantiated with the `RandomSampler`.

Regarding the adjustment for the effect of further training alone, the results in Figure 2 use the mean subtraction adjustment in (7). For the other adjustment method in which the output due to full-dataset training is subtracted, the attribution score is given by

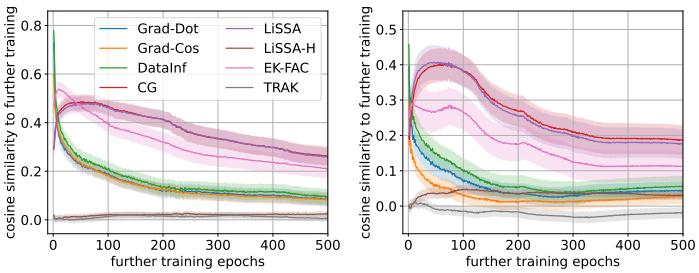
$$v_i = \frac{1}{r} \sum_{s=1}^{r} \left[g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}_{-i}, \xi^{(s)})\big) - g\big(\boldsymbol{z}, \boldsymbol{\theta}^f + \Delta\boldsymbol{\theta}(\mathcal{D}, \xi^{(s)})\big) \right]. \tag{17}$$

Approximate TDA methods All methods take the final model $f(\boldsymbol{x}; \boldsymbol{\theta}^f)$ as input, either by first computing training loss gradients $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{z}_i; \boldsymbol{\theta}^f)$ as well as test loss gradients, or as a model object that is used to compute derivatives. Given the gradients, the first-order methods Grad-Dot and Grad-Cos were straightforward to implement as (normalized) inner products. For the influence function methods CG, LiSSA, and LiSSA-H, we used the implementations in the `torch-influence` repository. For TRAK [28], EK-FAC [15], and DataInf [23], we used their respective repositories.

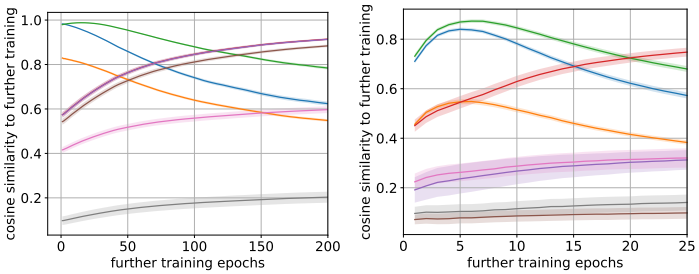
Evaluation metric The vectors $\boldsymbol{v}$ of gold attribution scores and $\hat{\boldsymbol{v}}$ of approximate attribution scores may have different scales. Here we do not concern ourselves with the differing scales, so we take both to be normalized to unit ℓ_2 norm. In this case, the squared Euclidean distance $\|\hat{\boldsymbol{v}} - \boldsymbol{v}\|^2$ and cosine similarity $\hat{\boldsymbol{v}}^T \boldsymbol{v}$ are equivalent up to an affine transformation, and we use the latter as the evaluation metric. Note that cosine similarity is a more demanding measure than Pearson correlation (used for example in [30, 3]), since the latter corresponds to approximating v_i by $\alpha\hat{v}_i + \beta$ where β is a non-zero bias, whereas the former constrains β to be zero.

C.2 Additional results

Figure 3 shows cosine similarity between the attribution scores of gradient-based methods and further training using Adam instead of SGD as in Figure 2. The patterns are broadly similar to those in Figure 2. However, for Folktables in Figure 3d, the curves for the first-order methods and DataInf

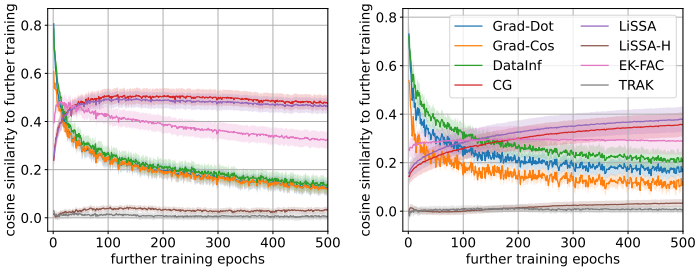


(a) Concrete Strength (b) Energy Efficiency

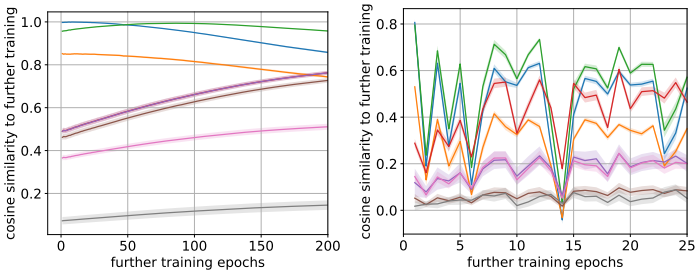


(c) FICO (d) Folktables

Figure 3: Cosine similarity between attribution scores of gradient-based TDA methods and further training using Adam, as a function of the amount of further training.



(a) Concrete Strength (b) Energy Efficiency

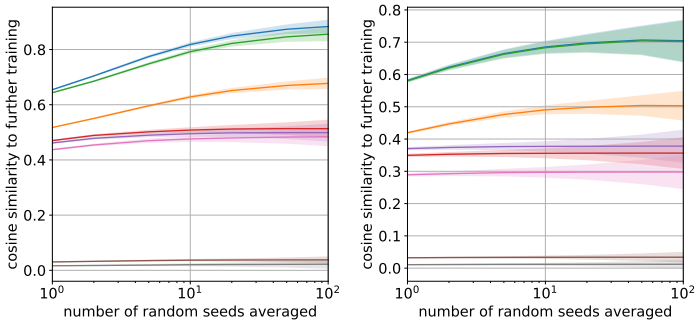


(c) FICO (d) Folktables

Figure 4: Cosine similarity between attribution scores of gradient-based TDA methods and further training using SGD as in Figure 2, but with adjustment according to (17).

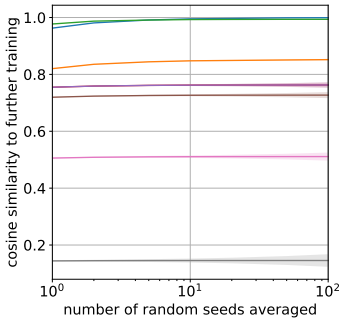
do not peak at the beginning but rather after a few epochs. In Figures 3a and 3b, the curves for the influence function methods CG and LiSSA are also seen to peak and then decay.

In Figure 4, the further training optimizer is SGD as in Figure 2, but the adjustment for the effect of further training alone is done using (17), i.e., by subtracting the output from full-dataset training. The plots for Concrete, Energy, and FICO are similar to their counterparts in Figure 2. However for Folktables in Figure 4d, the curves are very noisy. The reason is that the adjustment in (17) still leaves a non-negligible constant bias, i.e., the attribution scores v_i are shifted up or down by a constant

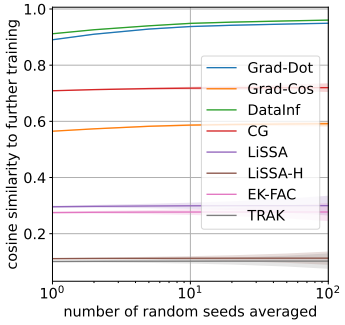


(a) Concrete Strength

(b) Energy Efficiency



(c) FICO



(d) Folktables

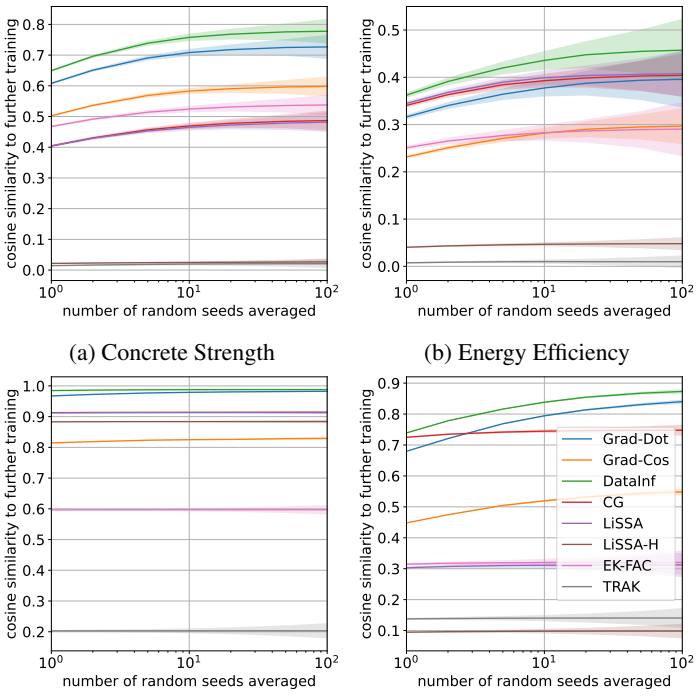
Figure 5: Cosine similarity between attribution scores of gradient-based TDA methods and further training using SGD, as a function of the number of random seeds averaged.

that varies from epoch to epoch. This varying bias is reflected in the cosine similarities shown in Figure 4d.

In Figures 5, 6, and 7, we fix the number of further training epochs to 1 and plot cosine similarities as a function of the number of random seeds averaged. In all cases, the cosine similarity increases, implying that the averaging of further training runs brings it closer to what gradient-based methods estimate. The increases are more notable for the first-order methods and DataInf, and especially so in Figure 7 when adjustment is done using full-dataset further training (17). As seen in Figure 4, this adjustment method is noisier and appears to benefit more from averaging.

D Topics for Discussion

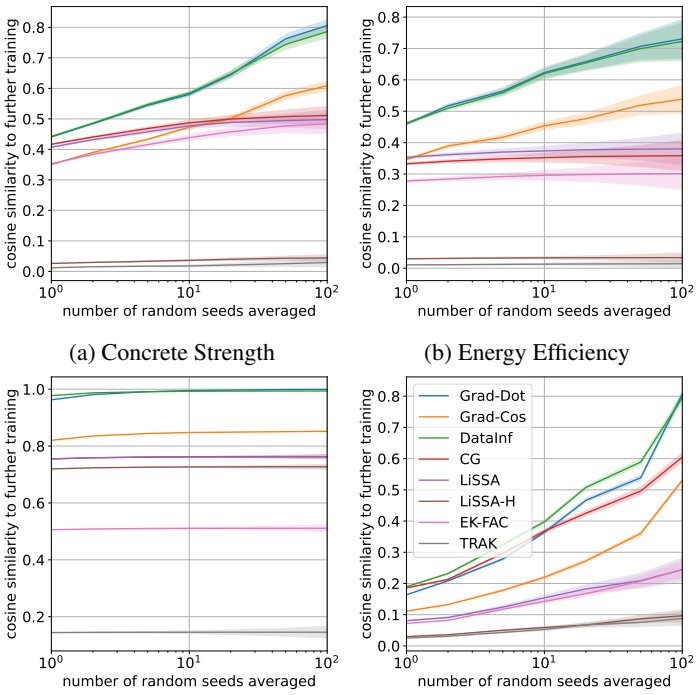
- An open question is how much further training is the right amount to determine the final model's sensitivity to training instances. If too little, the sensitivity may not be measurable, whereas if too much, then the measurement may no longer be local, i.e., specific to the given model $f(\boldsymbol{x}; \boldsymbol{\theta}^f)$. The extent to which further training is well-approximated by Taylor expansions (as evaluated for example in Figure 2) may provide a partial answer.
- Is it possible to combine the strengths of first- and second-order methods as seen in Figure 2? Or make use of the generalized influence function expressions (8), (14) derived in Appendix A.2? Regarding the former, we (as well as a reviewer of this paper) note that the damping parameter λ could be used to interpolate between first- and second-order methods.
- While we have presented further training as applied to the original training set $\mathcal{D}$ and assumed that $\mathcal{D}$ is available, there is no obstacle to considering further training on new, unseen instances.



(a) Concrete Strength (b) Energy Efficiency

(c) FICO (d) Folktables

Figure 6: Cosine similarity between attribution scores of gradient-based TDA methods and further training using Adam, as a function of the number of random seeds averaged.



(a) Concrete Strength (b) Energy Efficiency

(c) FICO (d) Folktables

Figure 7: Cosine similarity between attribution scores of gradient-based TDA methods and further training using SGD and full-dataset adjustment (17), as a function of the number of random seeds averaged.