MAGIC: DATA ATTRIBUTION VIA EXACT INFLUENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

We present MAGIC, a data attribution method that combines classical influence-function-based attribution and recent advances in metadifferentiation (Engstrom et al., 2025) to near-exactly predict the effect of removing training data on model predictions at only 2–3× the cost of training a single model. Across all the training regimes we study—spanning vision and language; pretraining and fine-tuning—MAGIC yields near-exact predictions for small counterfactual dataset changes, saturating existing data attribution benchmarks (e.g., LDS of 0.96 on CIFAR-10 vs. $\leq$ 0.35 for prior methods). Our code is available at https://anonymous.4open.science/r/mini-magic-jax-452E/.

1 INTRODUCTION

A common objective when building machine learning systems is to predict *counterfactuals* about model behavior. For example, scaling laws (Kaplan et al., 2020; Muennighoff et al., 2023) aim to predict the performance of systems trained with more data and compute than is currently available; interpretability techniques (Kim et al., 2018) predict how models behave under counterfactual inputs.

In this work, we study the task of *predictive data attribution*, or *datamodeling* (Ilyas et al., 2022), where the goal is to predict how a model would behave if trained on a different dataset. This well-studied problem encompasses, e.g., estimating the effect (on the resulting model's predictions) of modifying a training sample (Koh & Liang, 2017), removing a group of training samples (Koh et al., 2019; Bae et al., 2022; Park et al., 2023), or adding new training data sources (Ley et al., 2024).

Predictive data attribution in large-scale settings is challenging: it requires simulating training a model on a different dataset without actually training (Guu et al., 2023; Ilyas et al., 2024). In "classical" settings—when learning corresponds to minimizing a convex loss—statistical tools like the influence function (Hampel, 1974) accurately and efficiently estimate how different training data choices change model predictions (Rad & Maleki, 2018; Koh et al., 2019; Giordano et al., 2019). However, in the non-convex settings that are ubiquitous in natural domains like language/vision, current methods are less effective. Indeed, the best existing methods produce estimates that typically (a) only *moderately correlate* with the ground truth (Basu et al., 2021; Bae et al., 2022; Park et al., 2023) and (b) incur large absolute error (Bae et al., 2022).

1.1 CONTRIBUTIONS AND ROADMAP

This paper makes two contributions, summarized below and previewed in Figure 1:

- **A new method, MAGIC.** We introduce MAGIC, the first data attribution method to attain *near-perfect* linear datamodeling scores on modern (deep) learning pipelines, including LLM fine-tuning. MAGIC works by exactly computing the influence function (Hampel, 1974) via large-scale metadifferentiation (Engstrom et al., 2025), at a cost of only 2–3× a single training run per attributed test query (either a single prediction, or group of predictions).
- **Single-model evaluation.** We adopt the *single-model* attribution setting (in which all sources of training randomness are fixed) as a sharper testbed for data attribution. We discuss why this setting is the right unit of analysis for evaluating attribution methods and how it relates to the standard, average-model setting (§6).

Why single-model attribution? The inherent randomness of large-scale training makes attributing specific model predictions to training data conceptually challenging (Bae et al., 2022; Ilyas et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

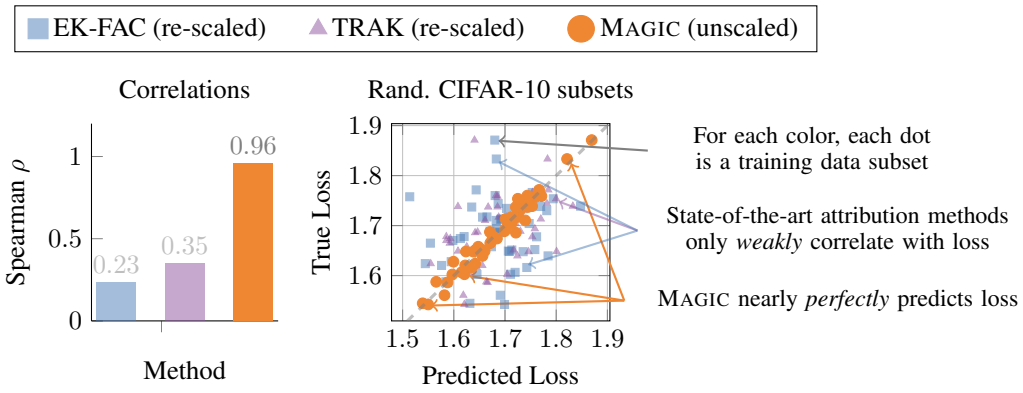


Figure 1: MAGIC nearly perfectly predicts the effect of training data removal. In contrast to the closest baselines (Park et al., 2023; Grosse et al., 2023), MAGIC estimates (a) correlate more highly with the ground truth effect and (b) are well-calibrated. **Right:** we plot the predicted loss (of MAGIC and the baselines) against the true loss for a randomly chosen test point, each point a training data subset with 1% of samples randomly removed. For MAGIC, we plot the predicted loss since it is well-scaled; for the baselines, we first rescale the predictions to match the variance of the ground-truth losses (this rescaling is for visualization and does not affect Spearman correlation). **Left:** The average (taken across test examples) Spearman correlation between predicted and true model losses (also known as rank-based LDS (Ilyas et al., 2022; Park et al., 2023), see Section 2.1).

2022; Nguyen et al., 2023): if training on the same data can lead to different models, no attribution method can predict the variation between them as a function of the dataset, so the standard target is the *expected* behavior of a learning *algorithm* averaged over training randomness. We instead study *single-model* data attribution (Wang et al., 2025), where all sources of training randomness are fixed and the goal is to predict how *this specific model* would have behaved on a different training set. This setting (a) has zero irreducible error so ground-truth attributions *exist*, making it a sharper testbed; and (b) directly answers model-specific counterfactual questions. We make this rigorous in Section 2 and discuss the relationship to standard, average-model attribution in Section 6.

Headline result. Our method, MAGIC (**M**etagradient-based **A**ttribution via **G**round-truth **I**nfluence **C**omputation), exactly computes the influence function by way of scalable metadifferentiation (Engstrom et al., 2025), and provides accurate estimates of how model predictions respond to random training data deletions—substantially outperforming existing methods. For example (Fig. 1):

- (CIFAR-10). When dropping random 1% subsets of CIFAR-10 from a ResNet-9 trained from scratch, MAGIC almost exactly predicts ground-truth losses ($\rho = 0.96$) while existing methods' predictions are only weakly correlated ($\rho \leq 0.35$).
- (LLM fine-tuning). When dropping random 1% subsets from a Gemma-2B instruction-tuning set, MAGIC nearly exactly predicts ground-truth test losses ($\rho = 0.97$) while existing methods perform no better than random guessing.

Among *linear* data attribution methods evaluated under the linear datamodeling score, we show that MAGIC is in a natural sense optimal. (see Sec. 4).

2 Data Attribution: Notation and Setup

The high-level goal of data attribution is to connect the choice of training data to model behavior. For example, one may use data attribution to find the training datapoints that cause a given behavior, or to surface data that harms accuracy. In this section, we formalize this goal with the *predictive data attribution* (or *datamodeling* (Ilyas et al., 2022)) framework, which phrases data attribution as the task of predicting how model behavior changes as a function of the training data.

Specifically, we view the machine learning pipeline as a three-step process wherein we (a) choose training data; (b) apply a learning algorithm to that data, yielding a trained model; and then (c)

evaluate the trained model. The goal of predictive data attribution is to construct a function that *directly* predicts the output of step (c) from the choice of training data in step (a). To make this more precise (and borrowing from (Ilyas et al., 2022)), we define the following notation:

- Let $S = \{z_i\}_{i=1}^N$ be a pool of N possible training examples. We represent *datasets* as vectors $\mathbf{w} \in \mathbb{R}^N$ where each entry w_i is an importance weight for the i-th example in S. The importance weight w_i controls the scaling of the loss of sample i; for example, $w_i = 0$ zeroes its loss contribution, while $w_i > 0$ multiplies its loss by w_i. Removal denotes this intervention with the training procedure fixed.
- Let $\mathcal{A} : \mathbb{R}^N \to \Theta$ be a *learning algorithm* mapping datasets—parameterized by importance weight vectors $\mathbf{w}$—to trained model parameters. We assume that all aspects of the training setup beyond the training data are captured by $\mathcal{A}$ (e.g., learning rate, weight decay, etc.).
- Let $\phi : \Theta \to \mathbb{R}$ be a *measurement function* mapping an ML model θ to a scalar measurement $\phi(\theta) \in \mathbb{R}$. For example, $\phi(\theta)$ might represent the loss of the classifier with parameters θ on a given test sample.
- Let $f : \mathbb{R}^N \to \mathbb{R}$ be the *model output function* f mapping datasets directly to model outputs, i.e., a composition of $\mathcal{A}$ and ϕ.

To illustrate this notation, we instantiate it in the context of linear regression. In this case, the training pool is a set of N input-label pairs $S = \{(\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R})\}_{i=1}^N$; the learning algorithm $\mathcal{A}$ fits a linear model minimizing the average squared loss *weighted* by a given $\mathbf{w} \in \mathbb{R}^N$, and the measurement function ϕ evaluates a model's loss on a specific test point $\mathbf{x}_{test}$, i.e.,

$$\mathcal{A}(\mathbf{w}) := \arg\min_{\theta} \sum_i w_i \cdot (\theta^\top \mathbf{x}_i - y_i)^2 \qquad \text{and} \qquad \phi(\theta) := (\theta^\top \mathbf{x}_{test} - y_{test})^2.$$

Then, $f(\mathbf{w}) := \phi \circ \mathcal{A}$ maps a data weighting $\mathbf{w}$ to the resulting model's loss on $\mathbf{x}_{test}$. Now, in this linear regression example, $f(\mathbf{w})$ is easy to directly compute (in fact, it has a closed form in terms of $\mathbf{w}$), but this is seldom the case. In general, evaluating $f(\mathbf{w})$ requires re-training a model on the weighting vector $\mathbf{w}$—which can be very expensive for large-scale models. This motivates our (informal) definition of predictive data attribution, given below.

Definition 1. *A predictive data attribution is an explicit function $\hat{f}$ that approximates a model output function f. That is, for a given data weight vector $\mathbf{w}$, $\hat{f}(\mathbf{w})$ should be fast to compute while also accurately predicting f, i.e., satisfying $\hat{f}(\mathbf{w}) \approx f(\mathbf{w})$.*

2.1 Single-model predictive data attribution

Large-scale learning algorithms are *non-deterministic*: the same training set can yield different parameters across runs, due to e.g. random initialization or data order (Zhuang et al., 2022; Jordan, 2024b). Without conditioning on training randomness, we can only predict how data changes *expected* model behavior: the expected squared error of an estimator $\hat{f}$ at a weighting $\mathbf{w}$ decomposes as

$$\mathbb{E}\left[(\hat{f}(\mathbf{w}) - f(\mathbf{w}))^2\right] = \underbrace{\left(\hat{f}(\mathbf{w}) - \mathbb{E}[f(\mathbf{w})]\right)^2}_{\text{Reducible Error}} + \underbrace{\mathbb{E}\left[(f(\mathbf{w}) - \mathbb{E}[f(\mathbf{w})])^2\right]}_{\text{Irreducible Error}}. \tag{1}$$

The irreducible error depends only on f and so is independent of any attribution method: it is fundamentally *unattributable* to data. Hence predictors of this expectation answer questions about *algorithms* ("how would a fresh training run change?")—not about individual trained models. In practice, we often want model-specific counterfactuals ("what was the effect of x on *this* model?"), motivating the *single-model* variant defined below.

Single-model predictive data attribution. In *single-model data attribution*, we fix all sources of training randomness (data order, initialization, etc.) so that $\mathcal{A}$ and ϕ are deterministic functions of $\mathbf{w}$. This collapses $\mathrm{Var}[\phi(\mathcal{A}(\mathbf{w}))]$ to zero, eliminating the irreducible error in (1).

Remark 1 (Single-model versus standard predictive data attribution)**.** *Even when we care more about the average behavior of a learning algorithm and not a specific trained model, a near-perfect single-model data attribution method can be used to construct a near-perfect standard data attribution method by averaging over random seeds. We discuss this connection further in Section 6.*

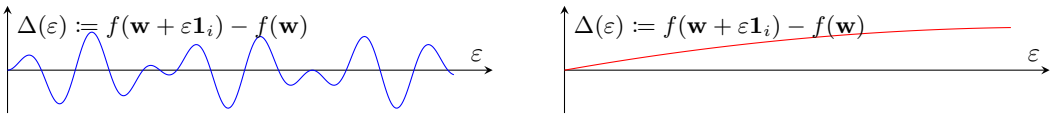


Figure 2: Smoothness aids predictive data attribution. We plot the change in data weights ε against the change in model output $\Delta(\varepsilon)$ for two hypothetical learning algorithms. On the left rapidly varying gradients make a linear approximation inaccurate over the plotted range. On the right is a smooth setting where the change is well-behaved.

Smoothness and linear attribution. Local linear prediction benefits from f being *smooth* in $\mathbf{w}$ (Bubeck, 2014): small changes to the data weights should yield small changes in the gradient $\partial f(\mathbf{w})/\partial \mathbf{w}$. To see why, consider upweighting the i-th training sample by ε and measuring $\Delta(\varepsilon) := f(\mathbf{w} + \varepsilon \mathbf{1}_i) - f(\mathbf{w})$. Smoothness locally ensures $\Delta(2\varepsilon) \approx 2\Delta(\varepsilon)$ (Fig. 2, right); rapidly varying gradients can instead make a first-order predictor inaccurate over the intervention range (Fig. 2, left).

Remark 2 (How restrictive is smoothness?)**.** *Smoothness is not an inherent property of standard learning algorithms. What matters for Taylor prediction is small remainder over the interventions of interest; global smoothness is not necessary. Related harmonic analyses characterize the accuracy of additive datamodels (Saunshi et al., 2023). Fortunately, as observed by prior work, one can often construct a "smooth counterpart" to any given non-smooth algorithm (Engstrom et al., 2025).*

3 MAGIC: CALCULATING THE EXACT INFLUENCE FUNCTION AT SCALE

We now present MAGIC. MAGIC adapts a statistical primitive—the influence function (§3.1)—to modern, non-convex training pipelines (§3.1) by way of explicit metadifferentiation (§3.2).

3.1 MOTIVATION: THE INFLUENCE FUNCTION

At the core of our method is a statistical primitive known as the *influence function approximation* (Hampel, 1974; Koh & Liang, 2017; Giordano et al., 2019). The main idea is to approximate the model output f for a given data weighting $\mathbf{w}$ using the following first-order Taylor expansion:

$$\hat{f}(\mathbf{w}) := f(\mathbf{1}_N) + \left(\partial f(\mathbf{w})/\partial \mathbf{w}|_{\mathbf{w}=\mathbf{1}_N}\right)^\top (\mathbf{w} - \mathbf{1}_N). \tag{2}$$

The key term in this estimate is the gradient $\partial f(\mathbf{w})/\partial \mathbf{w}$ evaluated at $\mathbf{w} = \mathbf{1}_N$, called the *influence function*. Intuitively, this term captures the effect of infinitesimally up- or down-weighting each training example on the model output.

When the learning algorithm $\mathcal{A}$ returns a regular minimizer (e.g., in strongly convex regression), $\partial f(\mathbf{w})/\partial \mathbf{w}$ can be computed by implicit differentiation, and for sufficiently small perturbations the first-order Taylor expansion (2) yields near-perfect estimates of $f(\mathbf{w})$ (Koh & Liang, 2017; Rad & Maleki, 2018; Koh et al., 2019; Giordano et al., 2019).

The influence function and deep learning. In large-scale, non-convex settings (e.g., in deep learning), a stationary-point derivative need not capture the sensitivity of the actual finite training run. Instead, modern data attribution methods tend to approximate the influence function (Koh & Liang, 2017; Bae et al., 2022; Park et al., 2023; Bae et al., 2024). These methods have shown promise, but are not nearly as effective as the influence function is in convex settings. In fact, a thorough study conducted in (Bae et al., 2022) reveals that for a variety of reasons (e.g., early stopping), the influence function as used by data attribution methods does not *even theoretically* predict the counterfactual effect of dropping data.

Given this landscape, we posit two possible explanations for the underwhelming performance of large-scale data attribution methods:

(a) The complexity of large-scale model training makes it hard to predict—and thus the influence function and all related methods are fundamentally limited.

(b) Existing methods do not do a good enough job at estimating the influence function.

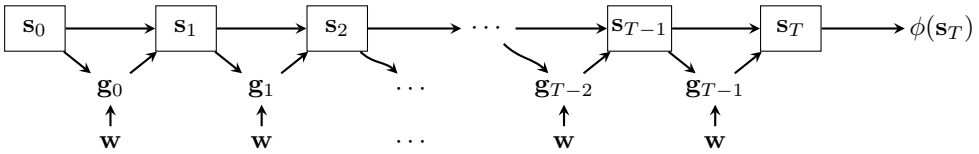


Figure 3: Forward computation graph for a model output function f mapping from data weights $\mathbf{w}$ to the model output. The exact influence function $\partial f(\mathbf{w})/\partial \mathbf{w}$ is the *metagradient* of the model output with respect to the data weights $\mathbf{w}$.

To probe which hypothesis dominates in our regimes, we describe a technique for *exactly* computing the influence function for large-scale, non-convex models. In Section 4, we find that the resulting data attribution method (MAGIC) is extremely effective, suggesting that in these settings the main bottleneck is influence estimation error rather than fundamental unpredictability of model training.

3.2 CALCULATING THE EXACT INFLUENCE FUNCTION

We now discuss how to compute the exact influence function in the context of deep learning models.

Iterative learning algorithms. Recall from Section 2 that the learning algorithm $\mathcal{A}$ takes in a data weighting $\mathbf{w}$ over the training set, and outputs a machine learning model trained on the weighted dataset. The algorithm thus encapsulates all aspects of the training setup beyond the training data weights, including the model architecture, optimizer, and hyperparameters. We observe that nearly all large-scale learning algorithms $\mathcal{A}$ are *iterative*, i.e., take the form

$$\mathcal{A}(\mathbf{w}) := \mathbf{s}_T \quad \text{for} \quad \mathbf{s}_{t+1} := h_t(\mathbf{s}_t, \mathbf{g}_t(\mathbf{s}_t, \mathbf{w})) \quad \text{and} \quad \mathbf{g}_t(\mathbf{s}_t, \mathbf{w}) \quad := \sum_{i \in B_t} w_i \cdot \nabla_{\mathbf{s}_t} \ell(z_i; \mathbf{s}_t). \tag{3}$$

Above, $\mathbf{s}_t$ is the optimizer state (including model parameters), which is updated by a function h_t starting from an initial state $\mathbf{s}_0$. The number of training steps is T; $B_t \subset [N]$ is the minibatch at step t; and $\ell(z_i; \mathbf{s}_t)$ is the loss on sample z_i given optimizer state $\mathbf{s}_t$. Stateful optimizers (e.g., Adam) are captured by having $\mathbf{s}_t$ include first/second moment estimates; we defer the details to Appendix A.

Calculating the influence function. To compute the exact influence function for a model output function f that is the output of an iterative learning algorithm $\mathcal{A}$, we leverage recent developments in *metagradient* calculation (Maclaurin et al., 2015; Franceschi et al., 2017; Engstrom et al., 2025). A metagradient is a gradient of a machine learning model's output with respect to a design choice made prior to training. (For example, imagine taking the derivative of a model's test loss with respect to the learning rate used to train the model.) In recent work, Engstrom et al. (2025) give an algorithm called REPLAY that *exactly* calculates the metagradient for iterative and smooth learning algorithms.

Observe that when the design choice of interest is the data weighting $\mathbf{w}$, the metagradient—the gradient of the model output f with respect to $\mathbf{w}$—is precisely the influence function. We can thus directly apply REPLAY (Engstrom et al., 2025) to calculate the exact influence function. Adapted to our setting, REPLAY calculates the metagradient by exploiting the following identity, which follows from the chain rule applied to the computation graph in Fig. 3:

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \sum_{t=0}^{T-1} \underbrace{\frac{\partial f(\mathbf{w})}{\partial \mathbf{s}_{t+1}} \cdot \frac{\partial h_t(\mathbf{s}_t, \mathbf{g}_t(\mathbf{s}_t, \mathbf{w}))}{\partial \mathbf{w}}}_{\text{contribution of } \mathbf{w} \text{ to } f(\mathbf{w}) \text{ through } \mathbf{s}_{t+1}}. \tag{4}$$

Concretely, REPLAY initializes $\Delta_T = \nabla_{\mathbf{s}_T} \phi(\mathbf{s}_T)$ and iterates $t = T-1, \ldots, 0$, at each step (i) loading $\mathbf{s}_t, B_t$, (ii) accumulating $\beta_t = \nabla_{\mathbf{w}}(h_t(\mathbf{s}_t, \mathbf{g}_t)^\top \Delta_{t+1})$, and (iii) back-propagating $\Delta_t = \nabla_{\mathbf{s}_t}(h_t(\mathbf{s}_t, \mathbf{g}_t)^\top \Delta_{t+1})$. The exact influence function is the accumulated sum $\beta = \sum_t \beta_t$. With k-way checkpointing, REPLAY uses $O(k \log_k T)$ states and $O(T \log_k T)$ step evaluations; reverse-step costs depend on implementation. See Engstrom et al. (2025) for the full algorithm.

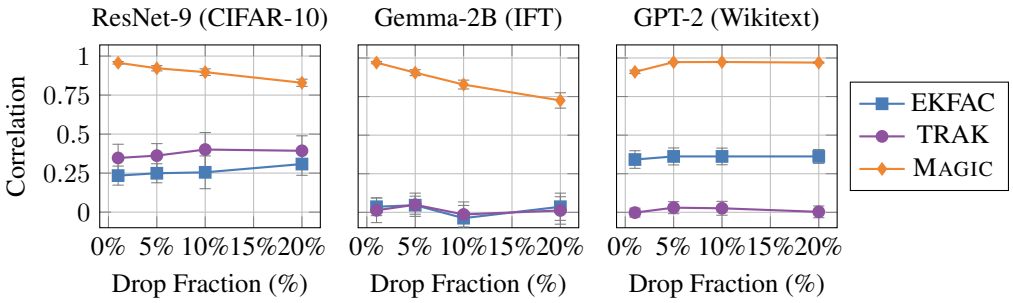


Figure 4: Linear datamodeling score (LDS) vs. drop fraction across settings for MAGIC and baselines. The estimates of MAGIC consistently correlate with the true model outputs (LDS: near 1.0 for small enough drop fraction) while baselines often do not (LDS: below 0.4). Larger drop fractions reduce LDS on CIFAR-10 and Gemma; GPT-2 remains stable after an initial increase.

4 EVALUATION

We evaluate MAGIC against TRAK (Park et al., 2023) and EK-FAC (Grosse et al., 2023) (the strongest recent attribution baselines with public implementations; see App. B). Across all domains, MAGIC near-perfectly predicts how model outputs change under random training-data removal.

Evaluation metric. To evaluate counterfactual predictions, we adopt the *linear datamodeling score* (LDS) (Ilyas et al., 2022; Park et al., 2023) as our evaluation metric. To compute LDS for a given model output function f and corresponding attribution method $\hat{f}$, we follow the following steps:

1. Sample m fixed-sized subsets of the training set, which we represent as binary data weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)} \in \{0,1\}^N$, where N is the number of total training samples. Given a drop fraction $p \in [0,1]$, we sample each vector $\mathbf{w}^{(i)}$ by dropping pN random training samples.
2. For each data weight vector $\mathbf{w}_i$, we (a) compute the *true* model output $f(\mathbf{w}_i)$ by training a model on the train set with data weights $\mathbf{w}_i$ and evaluating the measurement of interest on the trained model, and (b) compute the *predicted* model output $\hat{f}(\mathbf{w}_i)$ via the data attribution method $\hat{f}$.
3. We compute the LDS as the Pearson correlation (Spearman in Figure 1) between the predicted output and the true output over all m data weight vectors, i.e.,

$$\text{LDS} = \rho([\hat{f}(\mathbf{w}_i)]_{i=1}^m, [f(\mathbf{w}_i)]_{i=1}^m). \tag{5}$$

Settings. We study scenarios spanning computer vision and language modeling. Each scenario comprises a training dataset S, a learning algorithm $\mathcal{A}$, and a test set S_T. Accordingly, each scenario defines $|S_T|$ data attribution tasks, where task i is to predict the loss of $\mathcal{A}$ on the i-th sample in S_T. For each data attribution method we consider, we compute the *average* LDS (5) across tasks.

- **ResNet CIFAR-10 training**: We train ResNet-9 (Jordan, 2024a) models on subsets of the train set of CIFAR-10 (Krizhevsky, 2009) and aim to predict cross-entropy loss on its test samples.
- **GPT-2 Wikitext fine-tuning**: We fine-tune 125M-parameter GPT-2 models on subsets of Wikitext and aim to predict (language modeling) loss on 50 different test samples.
- **Gemma-2B instruction fine-tuning**: We fine-tune Gemma-2B with LoRA on subsets of four instruction tuning datasets (Flan V2, CoT, DOLLY, OA-1), predicting loss on MMLU samples.

See Appendix A for the exact details of each scenario,

4.1 RESULTS

As shown in Figure 4, MAGIC attains near-perfect LDS across settings and drop-out fractions, with setting-dependent degradation as we drop more data. Existing baselines are noisy in comparison; these methods' predictions only weakly correlate with the ground truth model losses.

In Figure 5, we randomly select a test example from each scenario, and plot predictions of test loss against true test loss for each data attribution method. MAGIC almost exactly predicts the true

Table 1: Expanded LDS comparison in the single-model setting on CIFAR-10 and MNIST. Means over 25 fixed test points using $m = 100$ counterfactual subsets per drop fraction. "—" indicates we did not implement the method on that dataset (e.g., due to computational constraints). MAGIC leads every reported comparison.

	CIFAR-10		MNIST	
Method	p=1%	p=5%	p=1%	p=5%
Random	0.07	0.07	−0.01	0.01
TRAK (Park et al., 2023)	0.35	0.36	0.43	0.57
EK-FAC (Grosse et al., 2023)	0.23	0.25	0.40	0.51
TracIn-CP (Pruthi et al., 2020)	—	—	0.05	0.05
SOURCE (Bae et al., 2024) (r=2048)	—	—	0.31	0.44
Wang et al. (2025)	—	—	0.56	0.67
MAGIC (ours)	**0.96**	**0.92**	**0.97**	**0.96**

test loss, even in absolute terms (aggregate errors: Appendix A.3). On the other hand, baselines barely correlate with the true predictions, and are mis-scaled in absolute terms (TRAK and EKFAC predictions are not of the right order of magnitude, so we rescale them to visualize them on the same plot).

Why performance degrades at larger drop fractions. On CIFAR-10 and Gemma, LDS degrades as more samples are dropped. While MAGIC has near-perfect LDS when predicting the effect of removing a small fraction of the training data (e.g., 1%), the LDS degrades at larger drop-out fractions (e.g., 20%). This behavior is expected for a first-order Taylor predictor (2): as the data weighting $\mathbf{w}$ moves farther from the all-ones weighting, higher-order terms become more salient. MAGIC computes the exact first-order coefficients; improving performance at larger drop fractions likely requires higher-order or non-linear models of f as a function of $\mathbf{w}$.

Expanded baseline comparison. To rule out the possibility that the gap between MAGIC and the two baselines in Figure 4 is an artifact of baseline choice, we compare MAGIC against a much wider set of attribution methods on CIFAR-10 (ResNet-9) and on MNIST (3-layer MLP), in our single-model setting (Table 1). We re-implement every baseline in JAX in our exact training setup, validating each against its canonical PyTorch reference where available; see Appendix B for method descriptions. MAGIC dominates by 3× over the strongest baseline on CIFAR (TRAK).

The comparison to Wang et al. (2025) is particularly instructive: MAGIC achieves MNIST LDS 0.973 versus 0.563 for the strongest baseline, and correlation 0.991 versus their reported ~ 0.72 in their leave-one-out evaluation under identical experimental conditions.

MAGIC is Pareto-optimal among linear attribution methods. We can now formalize the optimality claim from §2.1. The LDS is bounded by 1 (it is a Pearson correlation), and by Table 1 MAGIC attains LDS near this ceiling at $p_0 = 1\%$ on both CIFAR-10 (0.96) and MNIST (0.97). Let $\mathcal{W}_p := \{\mathbf{w} \in \{0,1\}^N : \|\mathbf{w}\|_1 = (1-p)N\}$ be the set of weightings that drop exactly pN examples, equipped with the uniform distribution. A linear attribution method $\hat{f}_{\boldsymbol{\beta}}(\mathbf{w}) = \boldsymbol{\beta}^\top(\mathbf{w} - \mathbf{1})$ is *Pareto-optimal* (among linear methods, w.r.t. LDS) if no other $\boldsymbol{\beta}'$ satisfies $\mathrm{LDS}(\boldsymbol{\beta}'; p) \geq \mathrm{LDS}(\boldsymbol{\beta}; p)$ for all $p \in (0,1)$ with strict inequality at some p. Here LDS denotes population Pearson correlation, and we consider only p with $pN \in \{1, \ldots, N-1\}$ and nonconstant f on $\mathcal{W}_p$.

Theorem 3 (Pareto-optimality of MAGIC)**.** *Suppose* $\mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p_0) \geq 1 - \epsilon$ *for some* $p_0 \in (0,1)$ *and* $\epsilon \in [0,1]$*, and that* $\boldsymbol{\beta}_{\text{MAGIC}}$ *is not a scalar multiple of* $\mathbf{1}$*. With the universal constant* $C = 2\sqrt{2}$*, for any other linear attribution method* $\boldsymbol{\beta}'$ *with* $\mathrm{LDS}(\boldsymbol{\beta}'; p_0) \geq \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p_0)$*,*

$$\mathrm{LDS}(\boldsymbol{\beta}'; p) \;\leq\; \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p) + C\sqrt{\epsilon} \qquad \textit{for every admissible } p \in (0,1).$$

When $\epsilon = 0$*, this gives* $\mathrm{LDS}(\boldsymbol{\beta}'; p) \leq \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p)$ *at every* $p \in (0,1)$ *(exact Pareto-optimality).*

Proof sketch (full proof in Appendix C). At $\epsilon = 0$, Pearson $= 1$ at p_0 forces $\boldsymbol{\beta}' = c\,\boldsymbol{\beta}_{\text{MAGIC}} + \gamma\mathbf{1}$ (the only constant direction on $\mathcal{W}_{p_0}$ is $\mathbf{1}$); so LDS values agree. For $\epsilon > 0$, fixed-cardinality covariance makes the L^2 distance between standardized linear predictors independent of p. At p_0, the triangle inequality bounds it by $2\sqrt{2\epsilon}$; Cauchy–Schwarz transfers this bound to LDS at every p. □

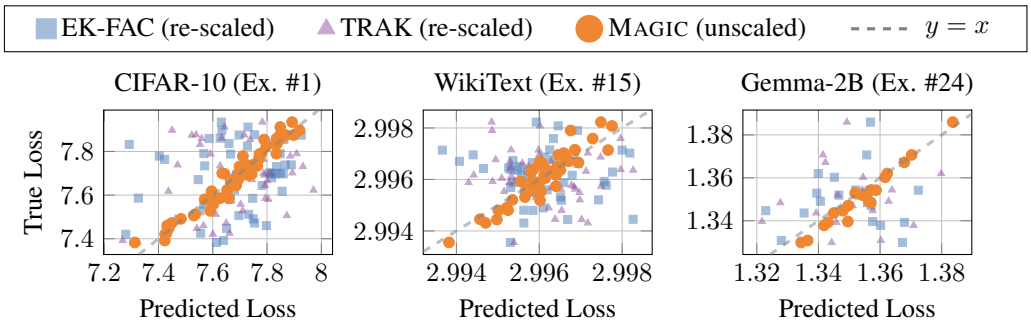


Figure 5: Results of MAGIC and baselines on randomly chosen, individual samples from three settings: CIFAR-10, Gemma-2B, and GPT-2. We evaluate by predicting model output after dropping a random 1% of the data (cf. (5)) and plotting the results against the true model output for that drop set. MAGIC estimates consistently correlate highly with the true output across settings.

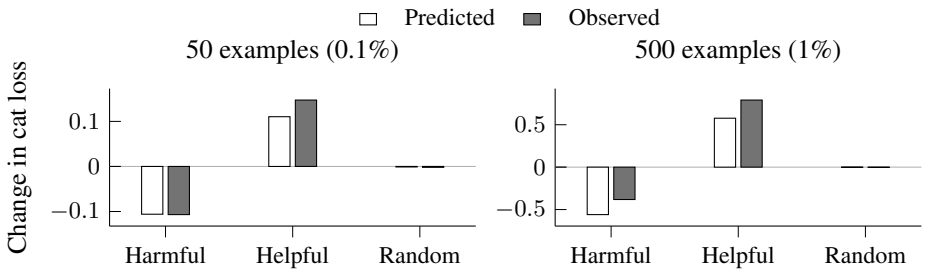


Figure 6: Predicted (outline) and observed (filled) changes in mean cat loss after zero-weighting harmful, helpful, or random training examples. Negative changes indicate improvement. One attribution vector supports all six fixed-seed interventions.

Standard (averaged-model) attribution. To address the concern that MAGIC's improvements only manifest in the single-model setting, we also evaluate in the standard *averaged-model* setting on CIFAR-10 at $p = 1\%$, in which the ground truth is the expected per-test-point loss across $S = 10$ training seeds and MAGIC is averaged across the corresponding metagradients (Table 2). MAGIC dominates TRAK by $\sim 3\times$ in the averaged setting as well, and has near-perfect LDS across settings.

Table 2: LDS in the single-model and averaged-model ($S = 10$ training seeds) settings, on CIFAR-10 at drop fraction $p = 1\%$. MAGIC dominates in both settings.

Method	Single-model LDS	Averaged-model LDS
TRAK (Park et al., 2023)	0.35	0.30
MAGIC (ours)	**0.96**	**0.89**

Targeting an aggregate model behavior. One MAGIC computation for mean loss on all 1,000 CIFAR-10 cat test images scores all 50,000 training examples. We zero-weight the 50 or 500 predicted most harmful, most helpful, or random examples and retrain (Figure 6; Appendix A.1).

Removing 50 predicted harmful examples changes cat loss by -0.1069 (predicted: -0.1060) and raises cat accuracy from 77.9% to 81.5%; removing 500 raises accuracy to 92.5%. Helpful-example removals worsen loss, while random controls have little effect. Prediction errors grow at the larger budget, but both targeted effects retain their predicted directions.

5 Related Work

In this section, we contextualize MAGIC with prior work in the field of data attribution, as well as with other related works. Our work adds to the growing literature on (predictive) data attribution methods—see (Ilyas et al., 2024; Hammoudeh & Lowd, 2022) for a survey. Related to our method

are those that use variants of the influence function (Hampel et al., 2011) (see Section 3.1). In settings where the learning algorithm minimizes a convex objective, influence function-based methods have an efficient and accurate closed form (Rad & Maleki, 2018; Koh et al., 2019; Giordano et al., 2019; Nobel et al., 2024). In non-convex settings (e.g., deep learning), the trained model need not be a stationary minimizer. In such cases, the dominant approach is to apply efficient approximations (Koh & Liang, 2017; Ladhak et al., 2022; Schioppa et al., 2022; Grosse et al., 2023; Wang et al., 2024; Choe et al., 2024). However, in the non-convex setting, these approximations do not offer correctness guarantees like they do in convex settings, and can even have different interpretations entirely (Bae et al., 2022)—potentially leading to the poor correlations we observe in Section 4.

Closest to our work are *trajectory-sensitive* methods (Pruthi et al., 2020; Wang et al., 2024; 2025; Bae et al., 2024) that leverage information about the path taken by the model parameters to perform data attribution. For example, Bae et al. (2024) aim to approximate the influence function via *unrolling* (Bae et al., 2024). Similarly, Wang et al. (2025) explicitly study temporal effects on data attribution scores. These methods differ from MAGIC in that (a) they still *approximate* the influence function rather than compute it exactly (leading to MAGIC's superior performance as evaluated in Table 1), and (b) our evaluation explicitly separates single-model prediction from averaged-model performance.

To compute the influence function exactly, we leverage recent advances in *metagradient* calculation (Engstrom et al., 2025), which build on a long line of work on differentiating through optimization (Maclaurin et al., 2015; Lorraine et al., 2020). Hara et al. (2019) derived this first-order sensitivity for SGD; REPLAY scales metagradients and uses them to optimize data weights. Our advance is near-perfect prediction of unseen finite interventions across vision and language from one fixed influence vector. Finally, our single-model data attribution setting is motivated by the nondeterminism of model training. This phenomenon has been studied from a variety of perspectives, including training dynamics (Zhuang et al., 2022; Jordan, 2024b), fairness (Marx et al., 2020), and data attribution (Ilyas et al., 2022; Nguyen et al., 2023).

6 Discussion and Conclusion

Single-model versus standard data attribution. Single-model attribution is strictly harder than standard (average-model) attribution: averaging a perfect single-model attribution method over seeds yields a perfect standard method, but the converse fails. Concretely, applying MAGIC to multiple seeds and averaging gives an average-model attribution method that outperforms TRAK by $\sim 3\times$ on CIFAR-10 (LDS 0.89 vs. 0.30; see Table 2).

Computational cost of data attribution. For N training samples and n output functions, the runtime comparison depends on n. Methods like TRAK and EK-FAC have runtime $C \cdot (N + n)$ for some very large constant C, while the runtime of MAGIC is roughly $c \cdot N \cdot n$ for a constant c typically 10-100x smaller than C. As a result, MAGIC is *significantly* faster than TRAK and EK-FAC when n is small—for example, when attributing a single output function on the CIFAR-10 dataset, MAGIC's reverse pass costs essentially 2-3x as much as checkpointed training (15 minutes on a single A100 GPU), while TRAK and EK-FAC are 20-1000x slower. On one L40S, we measure 394 s for attribution versus 177 s for checkpointed training and 87 s without checkpoints (Appendix A.2). As n increases, MAGIC's cost increases linearly, while TRAK/EK-FAC costs stay roughly constant.

Note that an output function does not have to correspond to a single test example: attributing a model's average performance across a group of test examples requires only a single output function, making MAGIC a useful tool for efficiently understanding the sources of specific capabilities.

Linearity. MAGIC computes exact first-order coefficients, so its predictive accuracy is inherently tied to the quality of the linear form (2) and can degrade for large counterfactual dataset changes. This is a fundamental limitation of *all* existing (linear) data attribution methods.

Conclusion. MAGIC is a new data attribution method that near-perfectly predicts how model outputs change in response to local training data changes, saturating standard data attribution benchmarks. Given the magnitude at which MAGIC improves our ability to estimate the effect of training data, we are excited to see what downstream applications MAGIC unlocks, including in unlearning (Georgiev et al., 2024), model debugging (Koh & Liang, 2017; Shah et al., 2022), and more.

AI USE STATEMENT

Generative AI tools assisted with running experiments, but not with interpreting their results. They also assisted with preparing some appendices and additional material. The main text was primarily written by the authors and was edited in full by them. The authors take responsibility for the final content.

REFERENCES

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger Grosse. If influence functions are the answer, then what is the question? In *ArXiv preprint arXiv:2209.05364*, 2022.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentiation. In *Arxiv preprint arXiv:2405.12186*, 2024.

Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.

Sébastien Bubeck. Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*, 15, 2014.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. What is your data worth to gpt? llm-scale data valuation with influence functions. In *arXiV preprint 2405.13954*, 2024.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Logan Engstrom, Andrew Ilyas, Benjamin Chen, Axel Feldmann, William Moses, and Aleksander Madry. Optimizing ml training with metagradient descent. In *arXiv preprint arXiv:2503.13751*, 2025.

Wikimedia Foundation. English wikipedia. https://huggingface.co/datasets/wikipedia, 2022.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning (ICML)*, 2017.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker-factored eigenbasis. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Kristian Georgiev, Roy Rinberg, Sung Min Park, Shivam Garg, Andrew Ilyas, Aleksander Madry, and Seth Neel. Attribute-to-delete: Machine unlearning via datamodel matching. In *International Conference on Learning Representations (ICLR)*, 2024.

Ryan Giordano, William Stephenson, Runjing Liu, Michael Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1139–1147. PMLR, 2019.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. Simfluence: Modeling the influence of individual training examples by simulating training runs. In *arXiv preprint arXiv:2303.08114*, 2023.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. In *arXiv preprint arXiv:2212.04612*, 2022.

Frank R. Hampel. The influence curve and its role in robust estimation. In *Journal of the American Statistical Association*, 1974.

Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with sgd. *Advances in Neural Information Processing Systems*, 32, 2019.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

Andrew Ilyas, Kristian Georgiev, Logan Engstrom, and Sung Min Park. Data attribution at scale. Tutorial at ICML 2024, 2024. URL https://ml-data-tutorial.org.

Keller Jordan. 94 percent on CIFAR-10 in 3.29 seconds on a single GPU. 2024a.

Keller Jordan. On the variance of neural network training with respect to test sets and distributions. In *International Conference on Learning Representations (ICLR)*, 2024b.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning (ICML)*, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Technical report*, 2009.

Faisal Ladhak, Esin Durmus, and Tatsunori Hashimoto. Tracing and removing data errors in natural language generation datasets. In *arXiv preprint arXiv:2212.10722*, 2022.

Dan Ley, Suraj Srinivas, Shichang Zhang, Gili Rusak, and Himabindu Lakkaraju. Generalized group data attribution. In *arXiv preprint arXiv:2410.09940*, 2024.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pp. 22631–22648. PMLR, 2023.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pp. 1540–1552. PMLR, 2020.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning (ICML)*, 2015.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, 2020.

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Elisa Nguyen, Minjoon Seo, and Seong Joon Oh. A bayesian perspective on training data attribution. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Parth T Nobel, Daniel LeJeune, and Emmanuel J Candès. Randalo: Out-of-sample risk estimation in no time flat. In *arXiv preprint arXiv:2409.09781*, 2024.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Arxiv preprint arXiv:2303.14186*, 2023.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Kamiar Rahnama Rad and Arian Maleki. A scalable estimate of the extra-sample prediction error via approximate leave-one-out. In *ArXiv preprint arXiv:1801.10243*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. In *ICLR*, 2023.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022.

Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. Modeldiff: A framework for comparing learning algorithms. In *arXiv preprint arXiv:2211.12491*, 2022.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run, 2024. URL https://arxiv.org/abs/2406.11011.

Jiachen T Wang, Dawn Song, James Zou, Prateek Mittal, and Ruoxi Jia. Capturing the temporal dependence of training data influence. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. 2022.

A EXPERIMENTAL DETAILS

In this section, we provide additional details on the experimental setup used in the main paper, including the training details of the models, and the datasets used.

ResNet-9 on CIFAR-10. We use the ResNet-9 architecture from (Jordan, 2024a), with the hyperparameters given in Table 3. To give concrete details: the training set S comprises $50,000$ CIFAR-10 training samples, the learning algorithm $\mathcal{A}$ is standard supervised training, and we consider 50 measurement functions ϕ_i corresponding to loss on 50 different CIFAR-10 test samples.

Hyperparameter	Value
Learning rate	1.2
Weight decay	0.001
Bias scale	8.0
Batch size	1000
Epochs	12
Final layer scale	0.04
Momentum	0.875
Pooling type	Log-sum-exp
Pooling ε	0.1
Width multiplier	2.5
LR schedule	One-cycle Linear
LR start multiplier	0.07
LR end multiplier	0.2
LR peak time	0.5

Table 3: Hyperparameters for ResNet-9 on CIFAR-10.

Gemma-2B LoRA on IFT Data. We use the variant of LESS (Xia et al., 2024) from Engstrom et al. (2025). In particular, the training dataset consists of the four instruction fine-tuning sets seen in Table 4 as in LESS. The total number of points is around $300,000$ and is exactly four combined IFT datasets (Flan V2 (Longpre et al., 2023), CoT (Wei et al., 2022), DOLLY (Conover et al., 2023), and Open Assistant 1 (Köpf et al., 2024)). We test on a randomly chosen (task balanced) subset of MMLU comprising 32 test samples. We use 4-shot in-context learning for these samples. We adapt a LoRA to a Gemma-2B model (the pretraining-only Gemma-2B model (Team et al., 2024)) using the LoRA configuration from Xia et al. (2024). For model training, we use the same setup as Engstrom et al. (2025), except with $\epsilon_{\text{root}} = 10^{-6}$. In particular, we train with ADAM ($\beta_1 = 0.95, \beta_2 = 0.975$, decoupled weight decay as 10^{-5}) and a one-cycle linear schedule starting at 10^{-6} of the maximum learning rate, reaching the peak over 25% of training, then ending at 0.1 of the maximum learning rate (0.0004). We insert a positive ϵ_{root} into the inverse square root term in the ADAM update to prevent metagradient (and to a lesser extent update) blowup.

GPT-2 fine-tuning on Wikitext. We optimize a pre-trained GPT2 (Radford et al., 2019) model on Wikitext (Foundation, 2022) using causal language modeling. We split the Wikitext samples into size 512 context length chunks and into train and test splits, with 256 samples in the test split and 4608 samples in the train split. We attribute 50 fixed examples from the test split, and use 4 epochs of the train split during training. We use the same ADAM optimizer setup above except that we set $\epsilon_{\text{root}} = 10^{-8}$, max learning rate to 0.0008, and do not anneal ADAM ϵ_{root}.

Table 4: Details of IFT training datasets.

Dataset	# Instance	Sourced from	Prompt Len.	Complet. Len.
FLAN V2	100,000	NLP datasets and human-written instructions	355.7	31.2
CoT	100,000	NLP datasets and human-written CoTs	266	53.2
Dolly	15,011	Human-written from scratch	118.1	91.3
Open Ast. 1	55,668	Human-written from scratch	34.8	212.5

A.1 AGGREGATE-TARGET INTERVENTION CASE STUDY

We train ResNet-9 (10,269,770 parameters, base width 80, GELU, average pooling, batch-normalization $\varepsilon = 10^{-3}$) on all 50,000 CIFAR-10 examples for 20 epochs, with batch size 500, random flips, and random crops with four-pixel reflection padding. SGD uses momentum 0.85, peak learning rate $0.08/q$, and weight decay $10^{-4}/q$, where $q = 1 + 1/(1 - 0.85)$. The learning rate rises linearly from 10% of its peak to the peak over the first 15% of training, then falls linearly to zero. We use deterministic fp32 operations, highest matmul precision, and disabled TF32 on one L40S.

We fix the target class (cat) before running and differentiate mean cross-entropy over its 1,000 test images, including the batch-normalization state, to obtain $\boldsymbol{\beta} \in \mathbb{R}^{50,000}$. For $k \in \{50, 500\}$, we select the k largest coefficients (harmful), smallest coefficients (helpful), or first k entries of a random permutation (seed 2027). For each selected set D, the predicted loss change is $-\sum_{i \in D} \beta_i$.

Retraining sets selected loss weights to zero while fixing initialization, batch composition, augmentation, batch-normalization computation, loss normalization, and optimizer steps. This single-seed case study uses one random control per budget (Table 5). Zero-weighted examples remain in batch-normalization statistics. The same fixed cat target defines both the attribution and the reported outcomes, measuring intervention accuracy for the specified behavior.

CPU checkpoints every 50 steps occupy 3.06 GiB; each segment is recomputed for differentiation. Unlike recursive k-way checkpointing, this two-level schedule stores coarse states on the host and reconstructs each segment on the GPU. The general recursive schedule uses $O(T \log_k T)$ forward-step evaluations plus T reverse steps, with $O(k \log_k T)$ stored states ($k \geq 2$); additional storage reduces replay. Training takes 269 seconds, attribution including replay 1,573 seconds, and compilation/warmup 25 seconds. These timings use the deterministic fp32 configuration above; Appendix A.2 reports a separate timing run with default GPU arithmetic and per-step checkpoints, and discusses the scope for systems optimizations. Peak JAX GPU buffer use is 8.63 GiB (allocator pool: 10.50 GiB); peak host resident memory is 13.96 GiB.

Table 5: Aggregate-target interventions. Original mean cat loss: 0.6087; cat accuracy: 77.9%. Each row zero-weights the selected examples and retrains.

Removed	Selection	Predicted Δf	Observed Δf	Cat accuracy
50	Harmful	-0.1060	-0.1069	81.5%
50	Helpful	+0.1104	+0.1474	73.3%
50	Random	-0.0011	-0.0019	78.0%
500	Harmful	-0.5597	-0.3818	92.5%
500	Helpful	+0.5769	+0.7910	55.6%
500	Random	-0.0018	-0.0029	78.0%

A.2 RUNTIME AND CHECKPOINTING

We separately benchmark the same 2,000-step ResNet-9 recipe and 1,000-image aggregate target on one Modal L40S, with checkpoints at every step. Arrays remain fp32, but we use JAX's default GPU arithmetic, allowing TF32 and nondeterministic kernels. One forward step, target gradient, and reverse step are compiled and warmed up before timing (34 s, excluded below). Every measurement synchronizes device work; checkpointed training includes CPU transfers. This run measures runtime only; the intervention results use the deterministic configuration in Appendix A.1.

Both timing configurations use the minimal JAX implementation, whose backward pass is not fully fused or optimized. Kernel fusion and other systems optimizations can substantially reduce attribution time.

A.3 CALIBRATION ACROSS QUERIES

To complement LDS, we summarize all 65 archived queries with saved predictions at a 1% drop fraction. These plotting records have the coverage in Table 7, distinct from the expanded comparison in Table 1. Predictions are evaluated without affine recalibration. For each query, normalized RMSE

Table 6: Runtime for one aggregate query. Both training denominators are reported to expose checkpoint overhead. The reverse pass includes the target gradient and reads the stored trajectory.

Operation	Seconds	/ ordinary	/ checkpointed
Ordinary training	86.94	1.00	0.49
Training with per-step checkpoints	176.83	2.03	1.00
Target gradient and reverse pass	393.75	4.53	2.23
Checkpointed training + reverse pass	570.58	6.56	3.23

is $\sqrt{m^{-1}\sum_{j=1}^{m}(\hat{f}_j - f_j)^2}/\sqrt{m^{-1}\sum_{j=1}^{m}(f_j - \bar{f})^2}$. It measures absolute prediction error relative to counterfactual variation; the best constant predictor has normalized RMSE 1. We also report the fitted slope of true loss on predicted loss as a calibration diagnostic, without applying that fit to the predictions.

Table 7: Absolute calibration on archived 1%-drop predictions. Medians and 90th percentiles are across queries; m is the number of counterfactuals per query. An ideally calibrated predictor has slope 1.

Setting	Queries	m	Median NRMSE	90th pct.	Median slope
CIFAR-10	10	50	0.297	0.332	1.000
GPT-2	50	50	0.526	0.803	1.110
Gemma-2B	5	20	0.234	0.358	0.994

B BASELINES

In this section, we provide a brief overview of the baselines we consider. The basis of TRAK and EK-FAC is similar, and rooted in the Taylor approximation (2) that also underpins MAGIC:

$$\hat{f}(\mathbf{w}) := f(\mathbf{1}_N) + \left(\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{1}_N}\right)^{\top}(\mathbf{w} - \mathbf{1}_N);$$

B.1 TRAK (TRACING THE RANDOMLY-PROJECTED AFTER KERNEL)

TRAK (Park et al., 2023) estimates the influence of individual training examples on model predictions. However, instead of computing the exact influence, TRAK calculates the influence for a simple proxy model that (a) is easy to calculate the influence for and (b) is meant to match the original model class of interest. In particular, TRAK *approximates* the original learning algorithm, $\mathcal{A}$, by linearizing around the final parameters. We refer the reader to Park et al. (2023) for full details.

B.2 EK-FAC (EIGENVALUE-CORRECTED KRONECKER-FACTORED APPROXIMATE CURVATURE)

EK-FAC (Bae et al., 2022; Grosse et al., 2023) is another influence-based data attribution method. To estimate the influence function, the method estimates the Hessian via the Fisher information matrix/Gauss-Newton hessian (Martens & Grosse, 2015; George et al., 2018), then applies a version of the infinitesimal jackknife (Giordano et al., 2019) to calculate the gradient with respect to data weights. For an excellent high level overview of this approach, see Appendix D.1 of Bae et al. (2024).

B.3 TRAJECTORY-AWARE BASELINES

The expanded MNIST comparison (Table 1) uses a shared 3-layer MLP and the single-model target: each method predicts counterfactuals for the same fixed training run. TracIn-CP (Pruthi et al., 2020) combines training–test gradient inner products at saved checkpoints, weighted by learning

rate. SOURCE (Bae et al., 2024) approximates unrolled differentiation using segment-wise summaries of the trajectory; our comparison uses rank $r = 2048$. For Wang et al. (2025), we use the data value embedding method with generalized Gauss–Newton (GGN) curvature. Its embeddings propagate training-example contributions along the trajectory and are paired with the final test gradient to obtain attribution scores.

C Proof of Theorem 3

We prove here Theorem 3 from §4. Throughout, we use the centered predictor convention $\hat{f}_{\boldsymbol{\beta}}(\mathbf{w}) := \boldsymbol{\beta}^\top(\mathbf{w} - \mathbf{1})$ from (2), so that $\mathbf{1}^\top(\mathbf{w} - \mathbf{1}) = \|\mathbf{w}\|_1 - N = -pN$ on $\mathcal{W}_p$ (and hence the $\gamma\mathbf{1}$ shift contributes $-\gamma pN$, not $+\gamma(1-p)N$, in the formulas below). The proof has two cases. All drop fractions are admissible as specified in the theorem; correlations are defined only for nonconstant predictors and outputs.

Exact case ($\epsilon = 0$). Pearson correlation is bounded by 1, so the hypothesis $\mathrm{LDS}(\boldsymbol{\beta}'; p_0) \geq \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p_0) = 1$ forces $\mathrm{LDS}(\boldsymbol{\beta}'; p_0) = 1$ as well. Since $\rho_P = 1$, there exist $c, c' > 0$ and $b, b' \in \mathbb{R}$ with

$$\boldsymbol{\beta}'^\top \mathbf{w} = c\, f(\mathbf{w}) + b \qquad \text{and} \qquad \boldsymbol{\beta}_{\text{MAGIC}}^\top \mathbf{w} = c'\, f(\mathbf{w}) + b' \qquad \text{on } \mathcal{W}_{p_0}.$$

Eliminating f gives $\left(\boldsymbol{\beta}' - \frac{c}{c'}\boldsymbol{\beta}_{\text{MAGIC}}\right)^\top \mathbf{w} = b - \frac{cb'}{c'}$, constant on $\mathcal{W}_{p_0}$. The set of vectors $\mathbf{v} \in \mathbb{R}^N$ that are constant on $\mathcal{W}_{p_0}$ is exactly $\mathrm{span}(\mathbf{1})$ (swapping an included and an excluded coordinate forces every pair of coefficients to agree). Hence

$$\boldsymbol{\beta}' = \tfrac{c}{c'}\boldsymbol{\beta}_{\text{MAGIC}} + \gamma\mathbf{1} \qquad \text{for some } \gamma \in \mathbb{R}.$$

For any $p \in (0,1)$ and any $\mathbf{w} \in \mathcal{W}_p$, $\|\mathbf{w}\|_1 = (1-p)N$ is constant, so

$$\hat{f}_{\boldsymbol{\beta}'}(\mathbf{w}) = \tfrac{c}{c'}\,\hat{f}_{\boldsymbol{\beta}_{\text{MAGIC}}}(\mathbf{w}) - \gamma pN,$$

which is a positive affine transform of $\hat{f}_{\boldsymbol{\beta}_{\text{MAGIC}}}$ on $\mathcal{W}_p$. Pearson correlation is invariant under positive affine transforms of either argument, so $\mathrm{LDS}(\boldsymbol{\beta}'; p) = \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p)$ for every $p \in (0,1)$.

Approximate case ($\epsilon > 0$).

Let $P = I - \mathbf{1}\mathbf{1}^\top/N$. Fixed-cardinality sampling gives $\mathrm{Var}(w_i) = p(1-p)$ and $\mathrm{Cov}(w_i, w_j) = -p(1-p)/(N-1)$ for $i \neq j$, hence

$$\mathrm{Cov}_{\mathcal{W}_p}(\mathbf{w}) = \sigma_p^2 P, \qquad \sigma_p^2 = \frac{N}{N-1}p(1-p).$$

Write $u = P\boldsymbol{\beta}_{\text{MAGIC}}/\|P\boldsymbol{\beta}_{\text{MAGIC}}\|_2$ and $v = P\boldsymbol{\beta}'/\|P\boldsymbol{\beta}'\|_2$. These are well-defined centered unit vectors. For any such vector a, define $X_{a,p} = a^\top\mathbf{w}/\sigma_p$, and let $Y_p = (f - \mathbb{E}_{\mathcal{W}_p} f)/\sqrt{\mathrm{Var}_{\mathcal{W}_p}(f)}$. Both variables have mean zero and variance one. The covariance identity gives

$$\|X_{u,p} - X_{v,p}\|_{L^2(\mathcal{W}_p)} = \|u - v\|_2, \qquad \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p) = \mathbb{E}_{\mathcal{W}_p}[X_{u,p}Y_p],$$

with the analogous formula for $\boldsymbol{\beta}'$ and v. At p_0, both correlations are at least $1 - \epsilon$, so $\|X_{u,p_0} - Y_{p_0}\|_{L^2}^2 \leq 2\epsilon$ and $\|X_{v,p_0} - Y_{p_0}\|_{L^2}^2 \leq 2\epsilon$. The triangle inequality therefore yields $\|u - v\|_2 \leq 2\sqrt{2\epsilon}$. For every admissible p, Cauchy–Schwarz now gives

$$\left|\mathrm{LDS}(\boldsymbol{\beta}'; p) - \mathrm{LDS}(\boldsymbol{\beta}_{\text{MAGIC}}; p)\right| = \left|\mathbb{E}_{\mathcal{W}_p}[(X_{v,p} - X_{u,p})Y_p]\right| \leq \|u - v\|_2 \leq 2\sqrt{2\epsilon}.$$

This proves the claim with $C = 2\sqrt{2}$, independent of N and the drop fractions. For $\epsilon = 0$, it also gives $u = v$, recovering exact Pareto-optimality.