How Faithful Is Trajectory-Based Data Attribution? Error Sources, Remedies, and Practical Guidelines

Anonymous authors
Paper under double-blind review

Abstract

Trajectory-based data attribution methods estimate the influence of training samples on model outputs by tracing through the training trajectory. They are increasingly used in applications such as data selection, data valuation, and model diagnosis, but there is a lack of comprehensive error analysis of these methods, raising concerns about method faithfulness and hindering reliable deployment. In this work, we provide the first systematic analysis of error sources in trajectory-based data attribution, together with remedies to mitigate them and practical guidelines for downstream use. We organize the error sources into three categories: *config-level*, *algorithm-level*, and *system-level*. Building on this taxonomy, we make three contributions. First, we identify optimizer mismatch as the dominant config-level error: most existing methods derive trajectory-based attribution under the assumption of SGD, even when the model is trained with the widely used AdamW optimizer. To mitigate this error, we propose *AdamW-influence*, which faithfully accounts for AdamW's optimization dynamics and improves the Spearman correlation between estimated and ground-truth influence by 10% to over 300% across settings spanning MLP, CNN, GPT-2, and Llama 3.2-1B. Second, after correcting optimizer mismatch, we show that the remaining algorithm-level error due to the first-order Taylor approximation is primarily governed by two factors: the learning rate and the remaining trajectory length from a sample's training step to the final model. The estimated influence is more faithful at smaller learning rates and over shorter trajectories. We further derive a closed-form proxy for per-sample approximation error that can be evaluated along the original training trajectory without retraining. Third, we translate these insights into practical guidelines for downstream attribution-guided data selection. Our empirical results indicate that the attribution faithfulness plays a critical role in the data selection effectiveness: the proposed AdamW-influence significantly outperforms the SGD-based counterpart; while online data selection, where the estimated influence accounts for only a short training trajectory to guide each local update, often outperforms offline data selection, where the estimated influence accounts for the full training trajectory of a reference run. Together, our results provide a systematic framework for diagnosing trajectory-based attribution error and using these methods more reliably in downstream applications.

1 Introduction

Data attribution is a family of methods that estimate the influence of individual training samples on model behavior (Deng et al., 2025), with applications in data selection (Xia et al., 2024; Wang et al., 2024; Hu et al., 2026), data valuation (Ghorbani & Zou, 2019; Deng et al., 2023), and model diagnosis (Zhang et al., 2025; Wang et al., 2025b). Among existing approaches, *trajectory-based data attribution* (Hara et al., 2019; Pruthi et al., 2020; Chen et al., 2021; Bae et al., 2024; Wang et al., 2025b) has emerged as a prominent research direction. Rather than characterizing perturbations around a converged final model, as influence functions do (Koh & Liang, 2017), these methods target a *trajectory-specific* counterfactual (Wang et al., 2025b): how the final model would change had a sample been removed from the recorded training run. This target requires neither convexity nor

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.

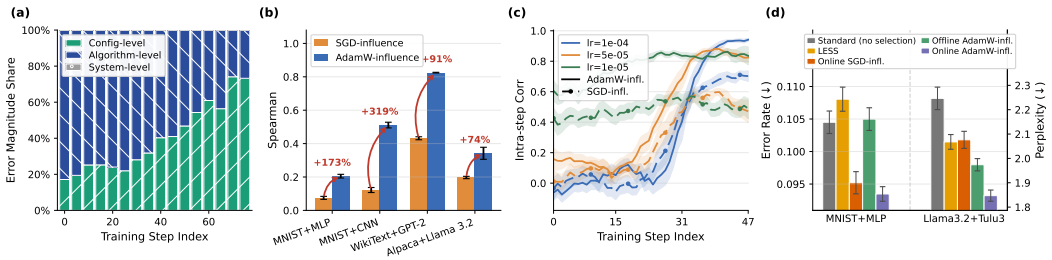


Figure 1: **An error taxonomy of trajectory-based attribution and key findings in this paper. (a) Three error sources and how their proportion changes along the training trajectory.** The absolute error of SGD-influence on MNIST+MLP decomposes into config-level (green, optimizer mismatch), algorithm-level (blue, first-order approximation), and system-level (gray) components. **(b) The proposed AdamW-influence algorithm corrects the dominant optimizer-mismatch error.** AdamW-influence (blue) significantly improves over SGD-influence (orange) in attribution fidelity, measured by the Spearman ρ against TSLOO ground truth across four settings. **(c) Dissecting the first-order approximation error shows that learning rate and trajectory length are governing factors.** Intra-step Spearman ρ is shown for AdamW-influence (solid) and SGD-influence (dashed) under three learning rates; attribution fidelity generally improves with smaller learning rates and as training approaches the trajectory end (shorter remaining trajectory). **(d) Correcting optimizer mismatch and shortening the attribution horizon substantially improve data selection quality.** Online AdamW-influence (purple) outperforms online SGD-influence (orange), offline AdamW-influence (green), LESS (yellow), and standard training without data selection (gray) on MLP image classification (error rate, ↓) and Llama 3.2-1B SFT on Tulu3 (perplexity, ↓).

convergence, assumptions that rarely hold in modern deep learning (Bae et al., 2022), and it captures how a sample's influence depends on *when* it is used during training (Wang et al., 2025b).

Computing this counterfactual exactly requires rerunning training for every sample. Existing methods fall roughly into two categories by cost. The first category is *retraining-scale*: producing all the attribution estimates costs more than one full model training. Retraining- and simulator-based estimators train many counterfactual models (Ghorbani & Zou, 2019; Ilyas et al., 2022; Guu et al., 2023), with the number of models growing with the training data size. Exact unrolled differentiation, exemplified by MAGIC (Ilyas & Engstrom, 2025), backpropagates each test query through a replay of the entire training run, at the cost of several full training runs per query (Engstrom et al., 2025). The second category, which is significantly more scalable, consists of *single-pass* estimators, such as TracIn (Pruthi et al., 2020), SOURCE (Bae et al., 2024), and SGD-influence (Hara et al., 2019; Wang et al., 2025b). These estimators traverse the training trajectory once and serve any number of queries at negligible marginal cost. In this work, we focus on the second category of scalable, single-pass methods.

This scalability, however, is obtained through approximations, such as assuming an SGD update rule, linearizing the trajectory, and approximating curvature, each of which can introduce error relative to the counterfactual they target. Existing literature typically validates single-pass estimators by an end-to-end fidelity score against retraining, with additional analyses focusing on individual method-specific approximations rather than systematically separating different sources of error (Hara et al., 2019; Chen et al., 2021; Bae et al., 2024; Wang et al., 2025b; Shi et al., 2025).

In this paper, we provide **the first systematic error analysis of trajectory-based data attribution against its counterfactual target, together with remedies and practical guidelines for downstream applications**. We specifically focus on the state-of-the-art efficient formulation of SGD-influence (Hara et al., 2019; Wang et al., 2025b), whose structure is shared by the broader single-pass family. We categorize the total error into three levels according to the error sources (Figure 1a). *Config-level error* arises when the attribution formula does not faithfully reflect the training procedure, e.g., the optimizer, batch composition, data curriculum, or learning-rate schedule. *Algorithm-level error* arises from approximations made for tractability, such as the first-order Taylor expansion. *System-level error*, such as floating-point precision and GPU non-determinism, is irreducible by algorithm design. Building on this taxonomy, we make three contributions: analysis and a new algorithm to address the config-level error, dissection and proxy to remedy the algorithm-level

error, and translating these analyses into practical guidelines for an impactful downstream application: attribution-guided data selection.

To address config-level error, we focus on its most impactful source: *optimizer mismatch*. Existing methods (Hara et al., 2019; Wang et al., 2025b; Pruthi et al., 2020) derive their attribution under the assumption of SGD, yet modern models (Grattafiori et al., 2024; Dosovitskiy et al., 2020) are almost universally trained with AdamW (Loshchilov & Hutter, 2017), introducing systematic deviation at every attribution step. We propose **AdamW-influence**, which correctly accounts for AdamW's momentum states and coordinate-wise scaling. Across four settings spanning MLP, CNN, GPT-2, and Llama 3.2-1B, it improves the Spearman correlation between estimated and ground-truth influence by 10% to over 300% against the SGD-based baseline (Figure 1b). This correction is also the necessary first step before the remaining algorithm-level error can be cleanly isolated.

With config-level error corrected, we turn to the algorithm-level error from the *first-order Taylor expansion*. We identify **two factors that govern its magnitude, the learning rate and the trajectory length**, controlling per-step magnitude and accumulation depth, respectively (Figure 1c). At the per-sample level, we further derive a closed-form error proxy evaluable along the original trajectory without retraining, which correlates strongly (Spearman $\rho \in [0.83, 0.89]$) with ground-truth error norms from trajectory-specific leave-one-out retraining.

Finally, we translate these insights into **practical guidelines for attribution-guided data selection**, unifying offline and online strategies under a K-step look-ahead framework, where K controls the horizon over which attribution informs selection. Our error analysis directly explains three empirical findings (Figure 1d): optimizer alignment transfers from attribution fidelity to selection performance, with online AdamW-influence outperforming its SGD-based baseline; online selection with small to moderate K matches or exceeds offline, since offline corresponds to the longest horizon and accumulates the largest error; and the optimal K grows as the learning rate decreases, since smaller per-step learning rates allow the K-step summation to extend further before error dominates. These results turn the error analysis to actionable selection recipe for practitioners.

2 Error analysis of trajectory-based data attribution

2.1 Preliminaries: trajectory-based attribution and SGD-influence

We focus on trajectory-based attribution methods that estimate the counterfactual effect of removing a training sample by propagating its perturbation through the optimization trajectory. Consider an optimization process that runs for T steps, yielding a parameter trajectory $\theta_0, \theta_1, \ldots, \theta_T$. Let z^* be a training data point in mini-batch $\mathcal{B}_{t^*}$ at iteration t^*. The exact counterfactual impact of removing z^* from the parameter update at step t^* on the loss at a validation point $z^{(\text{val})}$ is the Trajectory-Specific Leave-One-Out (TSLOO) error, originally conceptualized by Hara et al. (2019):

$$\text{TSLOO}(z^*; z^{(\text{val})}) := \ell(\theta'_{T,z^*}, z^{(\text{val})}) - \ell(\theta_T, z^{(\text{val})}), \tag{1}$$

where $\theta'_{T,z^*} \in \mathbb{R}^p$ is the final model parameters obtained if z^* had been excluded from the parameter update at step t^*, and p is the parameter size.

Hara et al. (2019) approximate the TSLOO error via a first-order Taylor expansion of the parameter trajectory in a perturbation scalar ϵ injected at step t^*, yielding the *SGD-influence*:

$$\ell(\theta'_{T,z^*}, z^{(\text{val})}) - \ell(\theta_T, z^{(\text{val})}) \approx \eta_{t^*} \nabla_\theta \ell(\theta_T, z^{(\text{val})})^\top \left[\prod_{t=t^*+1}^{T-1} (I - \eta_t H_t) \right] \nabla_\theta \ell(\theta_{t^*}, z^*), \tag{2}$$

where η_t is the learning rate and $H_t = \nabla^2_\theta \ell(\theta_t, \mathcal{B}_t)$ is the per-step Hessian. Wang et al. (2025b) reduce the cost to $\mathcal{O}(T)$ under the GGN approximation $H_t \approx \frac{1}{|\mathcal{B}_t|} \sum_{z \in \mathcal{B}_t} g_{t,z} g_{t,z}^\top$ (with $g_{t,z} = \nabla_\theta \ell(\theta_t, z)$) by extracting test-independent components into per-sample embeddings, computed via a backward recurrence over a summary matrix $W^{(t)} \in \mathbb{R}^{p \times p}$. We refer to this efficient implementation as SGD-influence throughout this paper and defer a detailed discussion of Wang et al. (2025b)'s implementation to Appendix B.

2.2 THE ERROR SOURCES OF TRAJECTORY-BASED ATTRIBUTION

The *attribution error* is the discrepancy between the estimated attribution score and the ground-truth TSLOO (Eq. 1). We organize its sources along the trajectory into three categories: config-level, algorithm-level, and system-level.

Config-level error. Config-level error arises whenever the attribution formula does not faithfully reflect a component of the actual training procedure. Potential sources include the optimizer, batch composition, data curriculum, and learning rate schedule. Among these, batch composition, data curriculum and learning rate schedule are aligned straightforwardly by replaying the recorded training log, and SGD-influence already incorporates both. *Optimizer mismatch*, by contrast, is neither addressed by existing methods nor trivial to correct: current methods derive their attribution under the assumption of SGD (Hara et al., 2019; Wang et al., 2025b), even for models trained with modern de facto optimizer AdamW. In Section 3, we address this dominant config-level error and show that the correction will significantly improve attribution fidelity.

Algorithm-level error. Even when the attribution formula perfectly matches the training dynamics, computing the exact TSLOO is intractable for nonlinear models, and trajectory-based attribution relies on approximations such as a first-order Taylor expansion of the parameter trajectory. Algorithm-level error is the gap introduced by these approximations, and its profile depends on the specific approximation strategy. We analyze it in the context of AdamW-influence, whose attribution relies on a first-order Taylor expansion. Specifically, this approximation introduces two sources of error: error from estimating the parameter update along the training trajectory, and error from the first-order approximation of the target function. In Section 4, we dissect this error, identify the factors governing its magnitude, and derive a closed-form proxy for per-sample analysis.

System-level error. System-level errors, such as floating-point precision and GPU non-determinism, are not specific to attribution methods, and even compute the TSLOO by directly retraining the model also suffer from this type of error. Therefore, they are largely irreducible by attribution algorithm design. Empirically, we find this irreducible part to be relatively small compared to the total error. We thus focus on analyzing and reducing the other two types of errors.

3 CORRECTING THE DOMINANT CONFIG-LEVEL ERROR: ADAMW-INFLUENCE

For estimators that replay the recorded run, batch composition, data order, the learning-rate schedule, weight decay, and clipped gradient values all enter the recurrence exactly, leaving the optimizer's moment and scaling dynamics as the one config-level source that requires a non-trivial derivation. In Section 3.1, we derive AdamW-influence, which extends the SGD-influence to correctly account for the modern de facto optimizer AdamW (Loshchilov & Hutter, 2017). We then evaluate its attribution fidelity in Section 3.2, demonstrating substantial improvements across four settings.

3.1 ADAMW-INFLUENCE

Both SGD-influence and AdamW-influence estimate the same TSLOO target (Eq. 1) but unroll different optimizers along the trajectory. SGD-influence unrolls the SGD updates for simplicity, even when the trajectory is actually from a different optimizer such as AdamW. Our AdamW-influence performs the analogous estimate but unrolls the actual AdamW updates.

Challenges for unrolling the AdamW updates. Two features distinguish AdamW from SGD and make the AdamW unrolling non-trivial: (i) momentum and second-moment states that carry perturbations across steps, and (ii) a coordinate-wise adaptive scaling that makes the update nonlinear in the gradient. The SGD-influence derivation hinges on the update being linear in the per-step gradient, so that injecting a parameter-level perturbation is equivalent to removing z^* from the batch. Under AdamW this equivalence breaks: the second moment depends quadratically on the gradient and enters the update nonlinearly through the denominator, so a parameter-level perturbation no longer corresponds to removing z^* from the batch.

AdamW unrolling. To address this, AdamW-influence injects the perturbation at the gradient level rather than the parameter level, and propagates it through the joint dynamics of the parameters and the two moment states.

Proposition 1 (AdamW-influence, informal)**.** *The AdamW-influence score for every training sample can be computed in $\mathcal{O}(T)$ via a backward recurrence over the training trajectory, matching the asymptotic complexity of SGD-influence.*

The proof extends the backward computation of SGD-influence, with the main challenge and innovation being the propagation of the gradient-level perturbation through AdamW's coupled moment states and adaptive scaling. The full derivation, transition operators, and per-sample recurrence are deferred to Appendix C, and the cost analysis to Appendix D.

3.2 Fidelity evaluation

We evaluate the attribution fidelity of AdamW-influence against the SGD-influence baseline by measuring the Spearman rank correlation between predicted attribution scores and ground-truth TSLOO scores (Eq. 1) obtained via retraining. Results in Table 1 show that AdamW-influence outperforms SGD-influence in four settings: an MLP and a CNN trained on MNIST, GPT-2 continually pretrained on WikiText-2, and Llama 3.2-1B fine-tuned on Alpaca. The improvements range from 10% to over 300%. For each setting, we also vary the learning rate, which we find to be an important factor in the error and study systematically in Section 4. Full training details and hyperparameters are provided in Appendix E.

Table 1: Fidelity comparison (Spearman ρ against TSLOO ground truth) between AdamW-influence and SGD-influence when training with **AdamW**. Best result per row in **bold**. $\Delta\%$ is the relative improvement of AdamW-influence over SGD-influence. Full results in Appendix F.1.

Setting	LR	AdamW-influence	SGD-influence	$\Delta\%$
MNIST+MLP	1e-3	$\mathbf{0.205 \pm 0.011}$	0.075 ± 0.009	$+173\%$
	1e-4	$\mathbf{0.294 \pm 0.014}$	0.242 ± 0.016	$+21\%$
	1e-5	$\mathbf{0.786 \pm 0.013}$	0.715 ± 0.012	$+10\%$
MNIST+CNN	1e-3	$\mathbf{0.090 \pm 0.008}$	0.015 ± 0.009	$+500\%$
	1e-4	$\mathbf{0.511 \pm 0.017}$	0.122 ± 0.015	$+319\%$
	1e-5	$\mathbf{0.791 \pm 0.007}$	0.526 ± 0.007	$+50\%$
WikiText+GPT-2	1e-4	$\mathbf{0.734 \pm 0.003}$	0.372 ± 0.006	$+97\%$
	5e-5	$\mathbf{0.825 \pm 0.002}$	0.432 ± 0.009	$+91\%$
	1e-5	$\mathbf{0.842 \pm 0.006}$	0.480 ± 0.015	$+75\%$
Alpaca+Llama 3.2	6e-6	$\mathbf{0.276 \pm 0.047}$	0.156 ± 0.019	$+77\%$
	2e-6	$\mathbf{0.342 \pm 0.036}$	0.197 ± 0.007	$+74\%$
	5e-7	$\mathbf{0.665 \pm 0.013}$	0.330 ± 0.017	$+102\%$

4 Dissecting the algorithm-level error

Section 3 corrected the dominant config-level error by aligning the attribution formula with the AdamW dynamics. We now turn to the algorithm-level error, which arises from the first-order Taylor expansion that AdamW-influence[1] uses to make attribution tractable. This section dissects the algorithm-level error in three steps. Section 4.1 empirically locates its dominant error source, isolating the *update-estimation error* as the bottleneck. Section 4.2 identifies the two factors, learning rate and trajectory length, that jointly govern its magnitude. Section 4.3 derives a closed-form error proxy that estimates per-sample error along the original training trajectory without retraining.

4.1 Dominant error from estimating the parameter update

Recall from Section 2.2 that the algorithm-level error of AdamW-influence can arise from two first-order approximations: estimating the parameter update and approximating the target function. The former corresponds to the gap between the true parameter update $\Delta\theta^*_{z^*} = \theta'_{T,z^*} - \theta_T$ and its estimate from AdamW-influence, while the latter is induced by the first-order approximation of the target function. In addition, the overall attribution error also contain config-level error from optimizer

[1] And other trajectory-based attribution methods (Hara et al., 2019; Wang et al., 2025b) more broadly.

mismatch and system-level variation. To identify which component dominates, and to quantify how much of the total error is attributable to each source, we decompose the SGD-influence error of a sample z^* into four additive components:

$$S_{z^*}^{\mathrm{SGD}} - T_{z^*}^{(2)} = \underbrace{\left(S_{z^*}^{\mathrm{SGD}} - S_{z^*}^{\mathrm{AdamW}}\right)}_{\text{Green: optimizer mismatch}} + \underbrace{\left(S_{z^*}^{\mathrm{AdamW}} - S_{z^*}^{\mathrm{perfect}}\right)}_{\text{Blue: AdamW influence 1st-order}} + \underbrace{\left(S_{z^*}^{\mathrm{perfect}} - T_{z^*}^{(1)}\right)}_{\text{Orange: target-function 1st-order}} + \underbrace{\left(T_{z^*}^{(1)} - T_{z^*}^{(2)}\right)}_{\text{Gray: system}}. \quad (3)$$

Here, $S_{z^*}^{\mathrm{SGD}}$ and $S_{z^*}^{\mathrm{AdamW}}$ denote the attribution scores produced by SGD-influence and AdamW-influence, respectively. $S_{z^*}^{\mathrm{perfect}}$ denotes the attribution score obtained by replacing the estimated AdamW parameter update with the true parameter update $\Delta\theta_{z^*}^*$ while retaining the same first-order approximation of the target function. $T_{z^*}^{(1)}$ denotes the TSLOO target value, and $T_{z^*}^{(2)}$ is obtained from an independent rerun of the same TSLOO computation to quantify system-level variation.

Accordingly, the green term captures config-level error from optimizer mismatch, which is corrected by AdamW-influence. The blue and orange terms capture first-order approximation errors in the AdamW parameter update and target function, respectively. The grey term captures system-level variation across repeated TSLOO runs, such as floating-point and GPU non-determinism.

Figure 2 reports the relative magnitudes of these four components across the training trajectory. Errors are aggregated into bins of 5 training steps (full setup in Appendix E). Two insights stand out. (1) Optimizer mismatch contributes a substantial fraction of the overall error in many parts of the trajectory. (2) Once optimizer mismatch is corrected, the parameter-update estimation error accounts for the overwhelming majority of the remaining error, while the target-function first-order error and system-level error are consistently small. We therefore focus on analyzing and correcting this dominant source of error for the rest of this section.

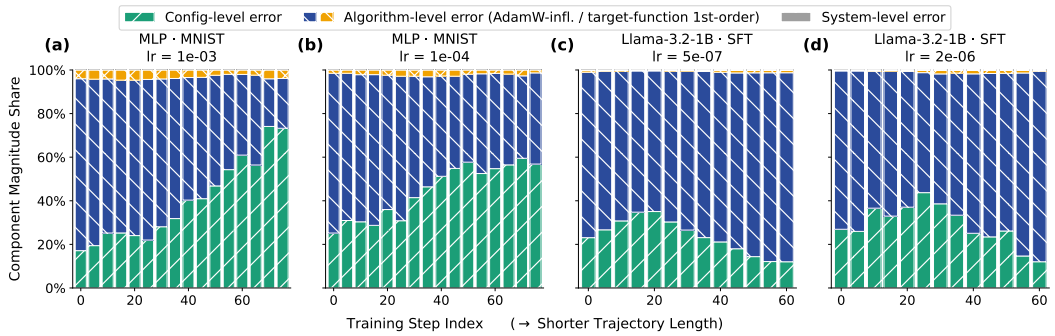


Figure 2: Decomposition of $\left|\mathrm{Error}_{\mathrm{SGD}}\right|$ across the training trajectory. Panels (a)–(d) cover four training configurations: **(a)** MLP on MNIST with $\eta = 10^{-3}$, **(b)** MLP on MNIST with $\eta = 10^{-4}$, **(c)** Llama-3.2-1B SFT with $\eta = 5 \times 10^{-7}$, and **(d)** Llama-3.2-1B SFT with $\eta = 2 \times 10^{-6}$. The absolute SGD-influence error is decomposed into four additive components, where each category is represented by its dominant source: **Green**, config-level error from optimizer mismatch, the share fixed by switching to AdamW-influence; **Blue**, algorithm-level error from the first-order AdamW-influence approximation; **Orange**, algorithm-level error from the first-order approximation of the target function; and **Grey**, system-level error capturing effects such as floating-point precision and GPU non-determinism. Errors are aggregated into bins of 5 training steps; larger step indices correspond to shorter remaining trajectory length $T - t^*$.

4.2 Two factors governing the algorithm-level error

Having established that the update-estimation error dominates the algorithm-level error, we investigate what governs its magnitude. The error has a intuitively simple structure: at each step, a first-order Taylor expansion linearizes the parameter update around the unperturbed trajectory, incurring a per-step error whose magnitude scales with the size of that step; these per-step errors then accumulate as the perturbation propagates to the end of training. This structure points to two controllable training quantities: the *learning rate*, which sets the magnitude of each per-step parameter update and thus the per-step linearization error; and the *trajectory length* $T - t^*$, the distance from the injection step t^* to the trajectory terminal T, which determines how many per-step errors are summed into the final attribution error.

To verify the influence of each factor, we sweep three learning rates on various settings, evaluating attribution quality across the trajectory through the *intra-step Spearman correlation* between attributed and ground-truth per-sample influences within each training step, which reflects the ability to rank samples within a batch (higher is better). Since the trajectory length $T - t^*$ is itself indexed by the training step, a single sweep over training steps traces both factors at once, with each curve isolating the learning-rate effect at a fixed trajectory length.

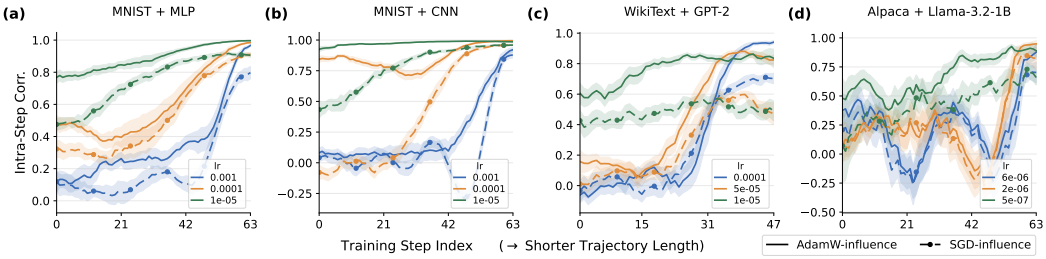


Figure 3: **Intra-step Spearman correlation across three learning rates**, on **(a)** MLP trained on MNIST, **(b)** CNN trained on MNIST, **(c)** GPT-2 trained on WikiText, and **(d)** Llama-3.2-1B trained on Alpaca. All panels report intra-step Spearman correlation (↑, smoothed over 10 consecutive steps). The training step index runs from 0 to $T - 1$; larger indices correspond to *shorter* trajectory lengths $|T - t^*|$, since the perturbation has fewer remaining steps to propagate. AdamW-influence (solid) and SGD-influence (dashed) are shown for comparison.

Learning rate. Larger learning rates push per-step parameter updates further from the linearization point, amplifying the higher-order terms that the first-order approximation discards. Figure 3 confirms this dependence across architectures and metrics: the intra-step correlation degrades correspondingly, with $\eta = 10^{-5}$ ($\eta = 10^{-3}$) achieving the highest (lowest) correlation throughout training.

Trajectory length. Independently of the magnitude of each step, errors propagate and accumulate along the training trajectory. **Notably, AdamW-influence reaches near-perfect intra-step correlation at very short trajectory lengths for all three learning rates**, indicating that short-horizon attribution is reliable even at learning rates where full-trajectory attribution is noisy. This compounding behavior between the two factors will be central to the practical implications in Section 5.

4.3 Error proxy

Beyond identifying trajectory-level factors that govern attribution error magnitude, we also try to identify per-sample errors to facilitate applications like data selection. Computing per-sample error directly requires TSLOO retraining that is computationally prohibitive. We thus derive a closed-form *error proxy* that can be evaluated along the original training trajectory.

Proposition 2 (Error proxy for update-estimation error)**.** *Under the AdamW training dynamics in Section 3.1, the update-estimation error at the final step admits a closed-form expression as the sum of per-step residuals $r_t \in \mathbb{R}^p$ propagated through the linearized dynamics, where per-coordinate residual $r_{t,i}$ is*

$$\left|r_{t,i}\right| \;\sim\; \eta_t \left(\frac{\|\dot{\theta}_t\|^2}{\sqrt{\hat{v}_{t,i}}} + \frac{[H_t \dot{\theta}_t]_i^2}{\hat{v}_{t,i}} \right). \tag{4}$$

Here $\dot{\theta}_t$ denotes the parameter perturbation propagated by AdamW-influence (the derivative of the parameter trajectory with respect to the injected perturbation ϵ at $\epsilon = 0$), and $\hat{v}_{t,i}$ is AdamW's bias-corrected second-moment estimate at coordinate i. The full unrolled form and derivation are deferred to Appendix G.

Equation equation 4 reflects the two error governing factors in Section 4.2. Per-step scaling by η_t and accumulation over trajectory length, and additionally exposes a sample-distinguishing term $\|\dot{\theta}_t\|^2 + [H_t \dot{\theta}_t]_i^2$ that captures training-stage effects unique to each sample.

We validate the proxy against ground-truth error norms on an MLP and a CNN trained on MNIST, each at two learning rates. Figure 4 shows Spearman rank correlation $\rho \in [0.83, 0.89]$ across all four

settings. Crucially, the proxy reuses quantities already maintained by AdamW-influence plus one Hessian-vector product per step, keeping total cost at $\mathcal{O}(T)$. It makes the proxy a practical signal for trajectory-aware decisions downstream, including the error-driven horizon analysis in Section 5.

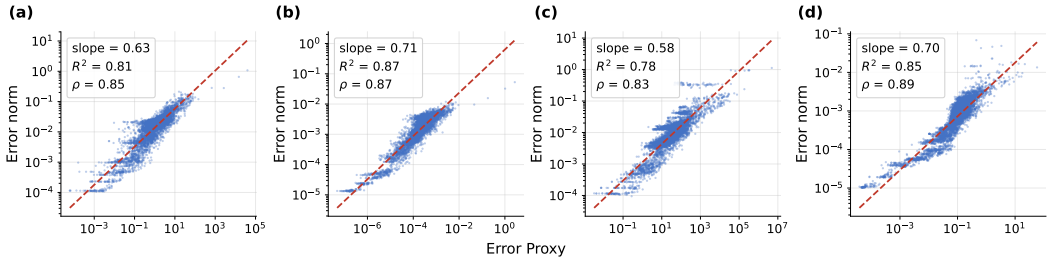


Figure 4: **The error proxy tracks the ground-truth error norm $\|\Delta\theta - \textbf{AdamW-influence}\|$ across training samples.** Each point is a single training sample; axes are on log-log scale. Panels: **(a)** MLP, $\eta = 10^{-3}$; **(b)** MLP, $\eta = 10^{-4}$; **(c)** CNN, $\eta = 10^{-3}$; **(d)** CNN, $\eta = 10^{-4}$. Both architectures are trained on MNIST. The red dashed line is a linear fit; insets report its slope and R^2, along with the Spearman rank correlation ρ between proxy and ground truth.

5 Practical Implications in Attribution-Guided Data Selection

Sections 3 and 4 together provide a systematic understanding of where the error in trajectory-based attribution originates and how to mitigate it. We now translate these insights into practical guidelines for data selection, one of the most consequential downstream applications of trajectory-based attribution. Specifically, we use the error analysis to predict which selection strategies should work better and verify these predictions empirically.

5.1 A Unified K-Step Look-Ahead Framework for Data Selection

Trajectory-based attribution can be used for data selection in two ways: *offline*, which runs a full training pass, computes attribution against a held-out validation set using the entire trajectory, and retrains from scratch on the selected subset; and *online*, which selects samples on-the-fly during a single training run using a short-horizon attribution score evaluated at the current parameters Deng et al. (2025). Despite their procedural differences, we show that they can be conceptually unified along a single axis: the look-ahead horizon of each selection decision.

Unified K-Step Setup. The horizon K controls how far ahead each candidate is scored. At step t, instead of asking "does including z help validation loss right now?" we ask "does including z help validation loss K steps from now?". Larger K looks further into the future and captures more downstream effect; smaller K stays close to the present.

Concretely, at each step t we draw a random candidate batch $\mathcal{C}_t$ of size N, score each candidate $z \in \mathcal{C}_t$ by $s_t^{(K)}(z) = \nabla_\theta \ell(\theta_t, z^{(\text{val})})^\top \dot{\theta}_{t+K+1}(z)$, and retain the top-$B$ as $\mathcal{B}_t$ for the parameter update. Here $\dot{\theta}_{t+K+1}(z)$ is the parameter perturbation K steps after injecting z at step t, computed via the AdamW-influence recurrence of Section 3. Algorithm 2 (Appendix H) gives the pseudo-code.

This single knob recovers existing strategies as boundary cases: $K = 0$ is the per-step scoring rule used by GREATS Wang et al. (2024) (without its greedy selection iteration), and $K = T - t$ is full-trajectory propagation, intractable to run online but approximated by offline selection on a completed reference trajectory. Online selection with finite K thus scans the intermediate horizons between these two extremes. This exposes K directly to the error analysis of Section 4: larger K accumulates more approximation error, while smaller K discards information about downstream dynamics.

5.2 Practical Implications and Error-Analysis Explanations

We experiment on four settings with optimizer AdamW, comparing "offline AdamW-influence", "online SGD-influence", and "online AdamW-influence" with varying K. Full setup in Appendix E.3.

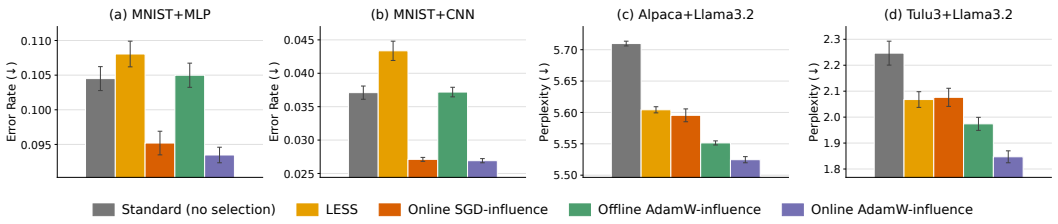


Figure 5: **Online AdamW-influence outperforms online SGD-influence, offline AdamW-influence, and other baselines across four settings.** **(a)** an MLP and **(b)** a CNN trained on MNIST (test error rate, ↓); also, Llama 3.2-1B fine-tuned **(c)** Alpaca and **(d)** Tulu3 (test perplexity, ↓). For MNIST settings, test performance is reported at the best epoch; for Llama 3.2-1B settings, test performance is reported at the final step.

Optimizer alignment transfers to downstream selection. Online AdamW-influence outperforms online SGD-influence in all settings (Figure 5), mirroring the attribution-fidelity gains of Section 3.2. The dominant config-level error identified in Section 2.2 thus propagates directly into selection quality: a selection rule built on a mis-specified optimizer undermines selection performance.

Horizon K trades off lookahead information against attribution error. Larger K collects more information about a candidate's future effect but accumulates more first-order approximation error per Section 4.2; smaller K keeps error low but reflects only short-range effects. We observe this trade-off between online and offline results: in all four settings, online AdamW-influence with a small-to-moderate K exceeds offline AdamW-influence (Figure 5), consistent with offline corresponding to the longest horizon $K = T - t - 1$ and accumulating the largest error. Notably, in the MNIST settings (Figure 5a,b), offline AdamW-influence and LESS (which is also an offline method) are no better than the Standard baseline without selection, indicating that large attribution error can even hurt data selection. We do not claim offline is universally inferior. When we have small learning rate, short training, or late-stage fine-tuning, offline can remain competitive or superior. Within the practical training regimes considered here, online with a small-to-moderate K is a reliable choice.

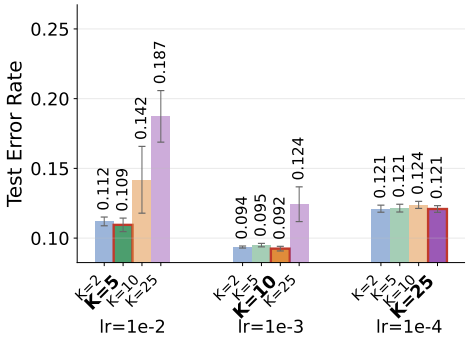


Figure 6: The optimal K increases as the learning rate decreases. We sweep $K \in \{2, 5, 10, 25\}$ across three learning rates using online AdamW-influence for MLP on MNIST. Bars show test error rate at the best validation epoch; the best K per learning rate is highlighted in red.

The optimal K shifts with the learning rate. Sweeping $K \in \{2, 5, 10, 25\}$ across three learning rates on MNIST (Figure 6), the best K is 5 at $\eta = 10^{-2}$, 10 at $\eta = 10^{-3}$, and 25 at $\eta = 10^{-4}$: the optimal horizon grows as the learning rate shrinks. This is precisely what Section 4.2 predicts: larger K collects more information about how a candidate influences future training but accumulates more approximation error, and the optimal K balances these two effects. A practical guidance is that K should be tuned jointly with the learning rate rather than fixed across runs; a larger K is not necessarily better.

6 Conclusion

We presented the first systematic error analysis of trajectory-based data attribution. By organizing the error into a three-level taxonomy, we develop AdamW-influence to remedy the dominant config-level error, identify the learning rate and trajectory length as the two factors governing the algorithm-level error, and derive a closed-form proxy for per-sample algorithm-level error. Building on this analysis, we further provide actionable guidelines for data selection through a unified K-step look-ahead framework. We see this as a step toward treating attribution methods as diagnosable estimators rather than black-box scoring functions. The taxonomy and the error analysis provide a foundation for trajectory-based attribution to scale to larger models and be deployed for broader applications.

AI USE STATEMENT

In this work, we used generative AI tools for writing and language polishing, coding assistance, drafting portions of the manuscript, implementing methods, and cleaning and reformatting datasets. We have not used generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, or support qualitative or thematic data analysis or the interpretation of results. Generating synthetic datasets and assisting with translation were not applicable to this work. We have reviewed all AI-assisted work. In particular, we verified and tested AI-assisted code and method implementations for correctness, and carefully reviewed and revised all AI-assisted text and dataset processing to ensure their accuracy and consistency with our intended claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolling. *Advances in Neural Information Processing Systems*, 37:66647–66686, 2024.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7081–7089, 2021.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8: 454–470, 2020.

Junwei Deng, Shiyuan Zhang, and Jiaqi Ma. Computational copyright: Towards a royalty model for music generative ai. *arXiv preprint arXiv:2312.06646*, 2023.

Junwei Deng, Yuzheng Hu, Pingbang Hu, Ting-Wei Li, Shixuan Liu, Jiachen T Wang, Dan Ley, Qirun Dai, Benhao Huang, Jin Huang, et al. A survey of data attribution: Methods, applications, and evaluation in the era of generative ai. 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://www.aclweb.org/anthology/D19-5409.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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. *arXiv preprint arXiv:2303.08114*, 2023.

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

Yuzheng Hu, Fan Wu, Haotian Ye, David Forsyth, James Zou, Nan Jiang, Jiaqi W. Ma, and Han Zhao. A snapshot of influence: A local data attribution framework for online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=sYK4yPDuT1.

Andrew Ilyas and Logan Engstrom. Magic: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ilyas22a.html.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

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.

Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 36:27560–27581, 2023.

Yunxiao Shi, Shuo Yang, Yixin Su, Rui Zhang, and Min Xu. Accumulative sgd influence estimation for data attribution. *arXiv preprint arXiv:2510.26185*, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Jiachen T Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024.

Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=HD6bWcj87Y.

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*, 2025b. URL https://openreview.net/forum?id=uHLgDEgiS5.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xia24c.html.

Zichun Yu, Spandan Das, and Chenyan Xiong. MATES: Model-aware data selection for efficient pretraining with data influence models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6gzPSMUAz2.

Shichang Zhang, Hongzhe Du, Jiaqi W Ma, and Himabindu Lakkaraju. Who gets credit or blame? attributing accountability in modern ai systems. *arXiv preprint arXiv:2506.00175*, 2025.

A Related work

Trajectory-based data attribution. Classical influence functions (Koh & Liang, 2017; Grosse et al., 2023) characterize perturbations around a converged solution and do not model the training trajectory. Trajectory-based methods instead target a trajectory-specific counterfactual, and split into two cost regimes. *Retraining-scale* methods pay a cost comparable to a full training run for each counterfactual model or each test query: retraining- and simulator-based estimators fit counterfactual models (Ghorbani & Zou, 2019; Ilyas et al., 2022; Guu et al., 2023), and MAGIC (Ilyas & Engstrom, 2025) computes the exact metagradient of a query with respect to sample weights by reverse-mode differentiation through a replay of the whole run (Engstrom et al., 2025), which handles arbitrary optimizers with internal state but costs several training runs per query and requires deterministic, smoothed training. *Single-pass* methods traverse the recorded trajectory once and amortize the result over all queries: TracIn (Pruthi et al., 2020) sums per-checkpoint gradient products, HyDRA (Chen et al., 2021), SGD-influence (Hara et al., 2019; Wang et al., 2025b) propagate first-order perturbations through the SGD updates, SOURCE (Bae et al., 2024) replaces per-step propagation by segment-level statistics, and In-Run Data Shapley (Wang et al., 2025a) accumulates per-step contributions during training. Recent extensions consider cross-epoch compounding under SGD (Shi et al., 2025) and stage-level effects under momentum and weight decay (Zhang et al., 2025). Our work studies the single-pass family. Two of these methods touch on modern optimizers: SOURCE incorporates optimizer-dependent preconditioning but treats the preconditioner as fixed during differentiation, so sample perturbations do not propagate through AdamW's moment states (Appendix I); LESS (Xia et al., 2024) uses the realized Adam update direction within a single step. AdamW-influence propagates perturbations through the joint dynamics of parameters and both moment states. In its exact form it recovers the MAGIC estimator (Appendix F.2), so MAGIC serves as a correctness reference rather than a competitor: the two occupy different cost regimes, and only the single-pass form supports query-amortized and online use.

Error analyses of data attribution. Prior fidelity studies concentrate on influence functions: Basu et al. (2020) document their fragility in deep networks, and Schioppa et al. (2023) analyze the assumptions behind them, including the additivity assumption of TracIn (Pruthi et al., 2020). Bae et al. (2022) decompose the gap between influence function and leave-one-out retraining into five sources, including warm-start, proximity, non-convergence, linearization, and solver terms. However, no comparable error source accounting exists for trajectory-based methods. Existing error analyses for trajectory-based methods are worst-case bounds on the linearization error (Hara et al., 2019; Wang et al., 2025b; Chen et al., 2021) or ablations of a single approximation (Bae et al., 2024; Shi et al., 2025). Our study provides a counterpart of Bae et al. (2022) for trajectory-based methods.

Attribution-guided data selection. *Offline selection* retrains on a subset chosen using attribution from a completed reference trajectory; representative work includes LESS (Xia et al., 2024), which adapts *single-step* influence to Adam. *Online selection* chooses samples on-the-fly: GREATS (Wang et al., 2024) uses a Taylor approximation of the immediate validation effect. Beyond unrolling-based attribution, MATES (Yu et al., 2024) fits a small data-influence model to oracle scores collected by locally probing the pretraining model, sidestepping first-order approximation entirely at the cost of training an auxiliary model. We unify both regimes into a unified framework.

B Detailed review of SGD-influence

For SGD-influence, Wang et al. (2025b) reduce the cost to $\mathcal{O}(T)$ under the GGN approximation $H_t \approx \frac{1}{|\mathcal{B}_t|}\sum_{z\in\mathcal{B}_t} g_{t,z}g_{t,z}^\top$ by extracting test-independent components into per-sample embeddings, computed via a backward recurrence over a summary matrix $W^{(t)} \in \mathbb{R}^{p\times p}$ that encodes how a perturbation injected at step t propagates through all subsequent training steps to the final parameters:

$$\text{SGD-influence}_z = \eta_t(I - W^{(t)})g_{t,z}, \quad \forall z \in \mathcal{B}_t, \tag{5}$$

$$W^{(t-1)} = W^{(t)} + \frac{1}{|\mathcal{B}_t|}\sum_{z\in\mathcal{B}_t} \text{SGD-influence}_z\, g_{t,z}^\top, \tag{6}$$

with $g_{t,z} = \nabla_\theta l(\theta_t, z)$ and $W^{(T-1)} = \mathbf{0}$. We refer to this efficient implementation as SGD-influence throughout this paper.

C Derivation of AdamW-influence

This appendix gives the complete derivation of AdamW-influence. Since Section 3.1 states only an informal version of Proposition 1, we begin by establishing the full notation used in this appendix.

C.1 Setup and notation

AdamW updates. Consider a data point z^* that first participates in training at iteration t^*. The model is trained with AdamW: given the batch gradient $g_t = \nabla_\theta \ell(\theta_t, \mathcal{B}_t)$, the optimizer maintains first and second moment estimates

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

with bias-corrected versions $\hat{m}_t = m_t/(1-\beta_1^{t+1})$ and $\hat{v}_t = v_t/(1-\beta_2^{t+1})$. The parameter update is

$$\theta_{t+1} = (1-\eta_t\lambda)\,\theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}.$$

We write $\theta_{t+1} = (1-\eta_t\lambda)\,\theta_t - U_t$, where $U_t = \eta_t \hat{m}_t/(\sqrt{\hat{v}_t}+\varepsilon)$, λ is the weight decay coefficient, and ε is the denominator smoothing term.

Gradient-level perturbation. As motivated in Section 3.1, AdamW's update is nonlinear in the gradient through the second moment, so a parameter-level ϵ-injection (which is equivalent to gradient-level injection for SGD) does not correspond to removing z^* from the batch. We therefore inject the perturbation at the gradient level at step t^*:

$$g_{t^*}(\epsilon) = g_{t^*} - \epsilon\, g_{t^*,z^*}, \qquad g_{t^*,z^*} := \nabla_\theta \ell(\theta_{t^*}, z^*),$$

so that $\epsilon = 0$ recovers the original trajectory and $\epsilon = 1$ removes z^*. Optimizer states before t^* are unaffected; all subsequent states are functions of ϵ. We use $\dot{(\cdot)}$ to denote the derivative with respect to ϵ evaluated at $\epsilon = 0$.

Auxiliary diagonal matrices. Throughout we use

$$D_t = \operatorname{diag}\!\left(\tfrac{1}{\sqrt{\hat{v}_t}+\varepsilon}\right), \qquad S_t = \operatorname{diag}\!\left(\tfrac{\hat{m}_t}{2\sqrt{\hat{v}_t}\,(\sqrt{\hat{v}_t}+\varepsilon)^2}\right),$$

which arise from differentiating U_t with respect to its inputs.

C.2 Initial derivatives at step t^*

Since all trajectories are identical before step t^*, the initial conditions are $\dot{\theta}_{t^*} = 0$, $\dot{m}_{t^*-1} = 0$, $\dot{v}_{t^*-1} = 0$. From the perturbed gradient, $\dot{g}_{t^*} = -g_{t^*,z^*}$.

The moment derivatives at the injection step are:

$$\dot{m}_{t^*} = \beta_1 \dot{m}_{t^*-1} + (1-\beta_1)\dot{g}_{t^*} = -(1-\beta_1)\, g_{t^*,z^*}, \tag{7}$$

$$\dot{v}_{t^*} = \beta_2 \dot{v}_{t^*-1} + 2(1-\beta_2)\operatorname{diag}(g_{t^*})\dot{g}_{t^*} = -2(1-\beta_2)\operatorname{diag}(g_{t^*})\, g_{t^*,z^*}. \tag{8}$$

The bias-corrected derivatives are $\dot{\hat{m}}_{t^*} = \dot{m}_{t^*}/(1-\beta_1^{t^*+1})$ and $\dot{\hat{v}}_{t^*} = \dot{v}_{t^*}/(1-\beta_2^{t^*+1})$.

Differentiating the update U_t coordinate-wise:

$$\dot{U}_{t^*,i} = \eta_{t^*}\left(\frac{\dot{\hat{m}}_{t^*,i}}{\sqrt{\hat{v}_{t^*,i}}+\varepsilon} - \frac{\hat{m}_{t^*,i}\,\dot{\hat{v}}_{t^*,i}}{2\sqrt{\hat{v}_{t^*,i}}\,(\sqrt{\hat{v}_{t^*,i}}+\varepsilon)^2}\right).$$

In vector form,

$$\dot{U}_{t^*} = \eta_{t^*}(D_{t^*}\,\dot{\hat{m}}_{t^*} - S_{t^*}\,\dot{\hat{v}}_{t^*}).$$

Substituting equation 7–equation 8 and using $\dot{\theta}_{t^*+1} = -\dot{U}_{t^*}$:

$$\dot{\theta}_{t^*+1} = \eta_{t^*}\left(\frac{D_{t^*}}{1-\beta_1^{t^*+1}} - \frac{2\,S_{t^*}\operatorname{diag}(g_{t^*})}{1-\beta_2^{t^*+1}}\right) g_{t^*,z^*}. \tag{9}$$

The first term arises from the perturbation flowing through the first moment, while the second arises from the second moment. Assembling the three derivatives gives the initial state perturbation:

$$Z_{\text{push}}(z^*) = \left(\dot{\theta}_{t^*+1}^\top,\ \dot{m}_{t^*}^\top,\ \dot{v}_{t^*}^\top\right)^\top \in \mathbb{R}^{3p}. \tag{10}$$

C.3 Recurrence for $t \geq t^* + 1$

For all subsequent steps, the batch gradient has no direct ϵ-dependence (the only direct perturbation was at t^*), so $\dot{g}_t = \nabla^2_\theta \ell(\theta_t, \mathcal{B}_t)\,\dot{\theta}_t = H_t\,\dot{\theta}_t$. The moment recurrences become

$$\dot{m}_t = \beta_1 \dot{m}_{t-1} + (1-\beta_1)\,H_t\,\dot{\theta}_t,$$
$$\dot{v}_t = \beta_2 \dot{v}_{t-1} + 2(1-\beta_2)\,\mathrm{diag}(g_t)\,H_t\,\dot{\theta}_t,$$

and the parameter recurrence is

$$\dot{\theta}_{t+1} = (1-\eta_t\lambda)\,\dot{\theta}_t - \eta_t\left(\frac{D_t}{1-\beta_1^{t+1}}\,\dot{m}_t - \frac{S_t}{1-\beta_2^{t+1}}\,\dot{v}_t\right).$$

Augmented state and linear dynamical system. Because the gradient-level perturbation propagates jointly through the parameters and both moments, we lift the state to the augmented vector

$$Z_t = (\dot{\theta}_t^\top,\, \dot{m}_{t-1}^\top,\, \dot{v}_{t-1}^\top)^\top \in \mathbb{R}^{3p},$$

on which the perturbation evolves as a linear dynamical system $Z_{t+1} = A_t Z_t$. Collecting the recurrences above, the transition is

$$A_t = \begin{pmatrix} (1-\eta_t\lambda)I - \frac{\eta_t(1-\beta_1)}{1-\beta_1^{t+1}} D_t H_t + \frac{2\eta_t(1-\beta_2)}{1-\beta_2^{t+1}} S_t\,\mathrm{diag}(g_t) H_t & -\frac{\eta_t\beta_1}{1-\beta_1^{t+1}} D_t & \frac{\eta_t\beta_2}{1-\beta_2^{t+1}} S_t \\ (1-\beta_1)H_t & \beta_1 I & 0 \\ 2(1-\beta_2)\,\mathrm{diag}(g_t) H_t & 0 & \beta_2 I \end{pmatrix}.$$

Decomposition into Hessian-free and Hessian-mediated parts. The transition admits a clean decomposition $A_t = M_t + R_t H_t P$, where $P = [I_p \quad 0 \quad 0] \in \mathbb{R}^{p\times 3p}$ extracts the parameter component, and

$$M_t = \begin{pmatrix} (1-\eta_t\lambda)\,I & -\dfrac{\eta_t\beta_1}{1-\beta_1^{t+1}}\,D_t & \dfrac{\eta_t\beta_2}{1-\beta_2^{t+1}}\,S_t \\ 0 & \beta_1 I & 0 \\ 0 & 0 & \beta_2 I \end{pmatrix}, \tag{11a}$$

$$R_t = \begin{pmatrix} -\dfrac{\eta_t(1-\beta_1)}{1-\beta_1^{t+1}}\,D_t + \dfrac{2\eta_t(1-\beta_2)}{1-\beta_2^{t+1}}\,S_t\,\mathrm{diag}(g_t) \\ (1-\beta_1)\,I \\ 2(1-\beta_2)\,\mathrm{diag}(g_t) \end{pmatrix}. \tag{11b}$$

M_t collects the Hessian-free terms (momentum decay and weight decay), while R_t collects the Hessian-mediated coupling between optimizer states.

C.4 Backward recurrence for the summary matrix

Analogous to SGD-influence, we define a summary matrix $W^{(t)} \in \mathbb{R}^{p\times 3p}$ that encodes how a perturbation injected at step t propagates through all subsequent training steps to the final parameters: $W^{(t)} = P\,\Psi_t$, where $\Psi_t = \prod_{k=t}^{T-1} A_k$.

From $\Psi_t = A_t \Psi_{t+1}$ and the decomposition $A_t = M_t + R_t H_t P$, we obtain

$$W^{(t)} = P\,A_t\,\Psi_{t+1} = P(M_t + R_t H_t P)\Psi_{t+1}.$$

Using the block structure $PM_t = M_t^{(\theta)}$ where $M_t^{(\theta)}$ denotes the first block-row of M_t, together with $P\Psi_{t+1} = W^{(t+1)}$, the recurrence becomes

$$W^{(t)} = W^{(t+1)} M_t + (W^{(t+1)} R_t)\,H_t\,P,$$

where we used $PR_t = R_t^{(\theta)}$ and the fact that $W^{(t+1)}$ already absorbs P on the left. Substituting the GGN approximation $H_t \approx \frac{1}{|\mathcal{B}_t|}\sum_{z\in\mathcal{B}_t} g_{t,z}\,g_{t,z}^\top$ yields the per-sample form:

$$W^{(t)} = W^{(t+1)} M_t + \frac{1}{|\mathcal{B}_t|}\sum_{z\in\mathcal{B}_t}\left(W^{(t+1)} R_t\,g_{t,z}\right) g_{t,z}^\top P, \tag{12}$$

with terminal condition $W^{(T)} = [I \;\; 0 \;\; 0]$.

Per-sample influence. The estimated parameter change for sample z^* injected at step t^* is

$$\text{AdamW-influence}_{z^*} = W^{(t^*+1)}\, Z_{\text{push}}(z^*),$$

combining the initial perturbation equation 10 with the propagator equation 12. This recovers the formal version of Proposition 1: the recurrence equation 12 preserves the $\mathcal{O}(T)$ complexity of SGD-influence, with constant overhead reflecting the augmented state (cost analysis in Appendix D).

C.5 Sanity checks

SGD ($\beta_1 = \beta_2 = 0$, $\lambda = 0$). Setting $\beta_1 = \beta_2 = 0$ and removing the second moment (i.e., $S_t = 0$, $D_t = I$), the transition reduces to $\dot{\theta}_{t+1} = (I - \eta_t H_t)\dot{\theta}_t$, recovering the known SGD formula (Hara et al., 2019).

SGD with momentum ($\beta_1 > 0$, $\beta_2 = 0$, $\lambda = 0$). With $\beta_2 = 0$, $D_t = I$, $S_t = 0$, and neglecting bias correction, the system becomes:

$$\begin{pmatrix} \dot{\theta}_{t+1} \\ \dot{m}_t \end{pmatrix} = \begin{pmatrix} I - \eta_t(1-\beta_1)H_t & -\eta_t\beta_1 I \\ (1-\beta_1)H_t & \beta_1 I \end{pmatrix} \begin{pmatrix} \dot{\theta}_t \\ \dot{m}_{t-1} \end{pmatrix},$$

recovering the known SGD-momentum formula (Zhang et al., 2025).

C.6 AdamW-influence: algorithm

Algorithm 1 summarizes the backward recurrent computation of AdamW-influence.

Algorithm 1 Backward Recurrent algorithm of AdamW-influence

Require: Training batches $\{\mathcal{B}_t\}_{t=0}^{T-1}$ with per-sample gradients $\{g_{t,z}\}_{z\in\mathcal{B}_t}$, optimizer states $\{m_t, v_t\}_{t=0}^{T-1}$, learning rates $\{\eta_t\}$, exponential decay rates β_1, β_2, weight decay coefficient λ, and denominator smoothing term ε.
Ensure: AdamW-influence embeddings $\{\text{AdamW-influence}_z\}_{z\in\mathcal{D}}$

$W \leftarrow [I \;\; 0 \;\; 0] \in \mathbb{R}^{p\times 3p}$ ▷ Initialize summary matrix
for $t = T-1$ **down to** 0 **do**
  Compute $D_t = \text{diag}(1/(\sqrt{\hat{v}_t} + \varepsilon))$, $S_t = \text{diag}(\hat{m}_t/(2\sqrt{\hat{v}_t}(\sqrt{\hat{v}_t} + \varepsilon)^2))$
  Construct M_t from Eq. equation 11a and R_t from Eq. equation 11b
  for $z \in \mathcal{B}_t$ **do**
    $\dot{m}_t \leftarrow -(1-\beta_1)\, g_{t,z}$ ▷ Use $g_{t,z}/|\mathcal{B}_t|$ for mean reduction
    $\dot{v}_t \leftarrow -2(1-\beta_2)\, g_t \odot g_{t,z}$
    $\hat{\dot{m}}_t \leftarrow \dot{m}_t/(1-\beta_1^{t+1})$, $\hat{\dot{v}}_t \leftarrow \dot{v}_t/(1-\beta_2^{t+1})$
    $\dot{\theta}_{t+1} \leftarrow -\eta_t(D_t\, \hat{\dot{m}}_t - S_t\, \hat{\dot{v}}_t)$
    $Z_{\text{push}}(z) \leftarrow (\dot{\theta}_{t+1}^\top,\, \dot{m}_t^\top,\, \dot{v}_t^\top)^\top$
    $\text{AdamW-influence}_z \leftarrow W\, Z_{\text{push}}(z)$
  end for
  $\Delta W \leftarrow 0$
  for $z \in \mathcal{B}_t$ **do**
    $v \leftarrow W\, R_t\, g_{t,z}$, $u^\top \leftarrow (g_{t,z}^\top,\, 0,\, 0)$
    $\Delta W \leftarrow \Delta W + v\, u^\top$
  end for
  $W \leftarrow W\, M_t + \frac{1}{|\mathcal{B}_t|}\, \Delta W$
end for
return $\{\text{AdamW-influence}_z\}_{z\in\mathcal{D}}$

D EFFICIENT COMPUTATION AND STORAGE OF ADAMW-INFLUENCE

D.1 RANDOM MASKING AND ENSEMBLES

The per-sample gradient and momentums cached by AdamW-influence live in $\mathbb{R}^p$, where p is the number of model parameters. Storing and manipulating p-dimensional embedding per training sample is prohibitive at modern scale, so dimensionality reduction is essential.

SGD-influence (Wang et al., 2025b) admits an efficient random-projection implementation because its backward recurrence is linear in the per-sample gradients, so the Johnson–Lindenstrauss lemma preserves the relevant inner products under a random projection. AdamW-influence breaks this property: the matrices D_t and S_t in Eqs. 11 apply a coordinate-wise nonlinearity to $\hat{v}_t$, which a random projection scrambles.

Random masking. We sample once a fixed binary mask $\mathbf{m} \in \{0,1\}^p$, and restrict the entire AdamW-influence computation to the index set $\mathcal{S} = \{i : \mathbf{m}_i = 1\}$. All per-sample gradients $g_{t,z}$, batch gradients g_t, and optimizer states are subselected to $\mathcal{S}$ so Algorithm 1 remains the same with p replaced by $|\mathcal{S}|$. The mask is shared across all training steps and samples within a single run.

Ensembles for masking. At the LLM scale, memory constraints push $|\mathcal{S}|/p$ to small values, where a single mask produces significant variance and information loss. We mitigate this with an ensemble of M independent masks, run AdamW-influence once per mask to obtain scores, and aggregate by simple average. Per-setting choices of M are reported in Appendix E.

D.2 COMPUTATIONAL EFFICIENCY

We compare the cost of AdamW-influence against SGD-influence along three axes: per-sample embedding storage, working memory during the backward recurrence, and per-step compute. Throughout this section, p denotes the number of model parameters, T the number of training steps, b the batch size, and $|\mathcal{S}|$ the number of coordinates retained after random masking ($|\mathcal{S}| = p$ if masking is disabled). For ease of exposition, we first describe costs in terms of p and then note that masking replaces p with $|\mathcal{S}|$ throughout, providing the same multiplicative cost reduction as random projection in SGD-influence.

Working memory and storage. The summary matrix $W^{(t)} \in \mathbb{R}^{p\times 3p}$ is $3\times$ larger than the $\mathbb{R}^{p\times p}$ matrix in SGD-influence, reflecting the extended state $(\dot{\theta}, \dot{m}, \dot{v})$ that AdamW-influence must propagate. Per-step trajectory caching incurs a similar additive overhead: AdamW-influence stores $\hat{m}_t, \hat{v}_t$ alongside the per-sample gradients $\{g_{t,z}\}_{z\in\mathcal{B}_t}$ and batch gradient g_t, adding $2p$ per step on top of the $bp + p$ already required by SGD-influence. When b is not too small, this overhead is dominated by the bp per-sample gradient term and is $\mathcal{O}(1/b)$ relative to it.

Per-step compute. The dominant cost in the backward recurrence is the update of $W^{(t)}$ in Eq. 12. The term $W^{(t+1)}M_t$ exploits the block structure of M_t (each block is either diagonal or a scalar multiple of identity), so it costs $\mathcal{O}(p^2)$. The per-sample term $W^{(t+1)}R_t g_{t,z}$ similarly leverages the block structure of R_t, costing $\mathcal{O}(p^2)$ per sample, and the rank-one update to $W^{(t)}$ costs $\mathcal{O}(3p^2)$ per sample due to the wider second dimension. The total per-step cost is $\mathcal{O}(3bp^2)$, compared to $\mathcal{O}(bp^2)$ for SGD-influence, a $3\times$ overhead arising directly from the wider summary matrix.

Effect of random masking. With masking, every p above is replaced by $|\mathcal{S}|$. The $3\times$ ratios between AdamW-influence and SGD-influence are preserved. Random masking thus delivers the same form of cost reduction that random projection provides in SGD-influence—turning otherwise infeasible quadratic costs in p into tractable quadratic costs in $|\mathcal{S}|$—without breaking the coordinate-wise structure required by AdamW (Section D.1). In summary, AdamW-influence incurs a $3\times$ overhead in working memory and per-step compute relative to SGD-influence, while keeping the per-sample embedding size identical.

Table 2: Cost comparison between SGD-influence and AdamW-influence. The first four rows describe the per-component costs without masking; the last two row gives the total cost when random masking without/with $|\mathcal{S}|$ retained coordinates is applied. The $3\times$ ratios are preserved under masking.

	SGD-influence	AdamW-influence	Ratio (Adam / SGD)
Per-sample embedding (storage)	p	p	$1\times$
Summary matrix W (memory)	p^2	$3p^2$	$3\times$
Trajectory cache (per step)	$bp + p$	$bp + 3p$	$\approx 1\times$
Backward recurrence (per step)	$\mathcal{O}(bp^2)$	$\mathcal{O}(3bp^2)$	$3\times$
Total cost	$\mathcal{O}(Tbp^2)$	$\mathcal{O}(3Tbp^2)$	$3\times$
Total cost (with masking $\|\mathcal{S}\|$)	$\mathcal{O}(Tb\|\mathcal{S}\|^2)$	$\mathcal{O}(3Tb\|\mathcal{S}\|^2)$	$3\times$

E Experiment details

E.1 Experimental settings for fidelity evaluation

In section 3.2, we consider four experiment settings to evaluate the attribution fidelity of AdamW-influence. We also consider additional settings presented in Appendix F.

- **MNIST+MLP.** We train a three-layer multilayer perceptron (MLP) with hidden layer sizes of 16, consisting of two fully connected layers ($28 \times 28 \to 16 \to 16$) with ReLU activations, followed by a final linear layer mapping to 10 output classes on the MNIST dataset (Lecun et al., 1998). We train on 10% of MNIST for 1 epoch. We set the batch size to 64 and perform the experiment on 5 learning rates (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5} and 10^{-6}). TSLOO ground truth is computed for 200 randomly selected training samples. The Spearman correlation is computed across these 200 samples and averaged over 500 validation points. We use AdamW, we set $\beta_1 = 0.9$ and $\beta_2 = 0.95$.
- **MNIST+CNN.** We train a convolutional neural network (CNN) with two convolutional layers (Conv2d(1, 32, 3, padding=1) and Conv2d (32, 64, 3, padding=1), each followed by ReLU and 2×2 max pooling) and a final fully connected layer of size $64 \times 7 \times 7 \to 10$ on the MNIST dataset (Lecun et al., 1998). We train on 10% of MNIST for one epoch. We set the batch size to 64 and perform the experiment on 5 learning rates (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}). We use a random masking ratio of 0.75 with a single masking ensemble. TSLOO ground truth is computed for 200 randomly selected training samples. The Spearman correlation is computed across these 200 samples and averaged over 500 validation points. We use AdamW, we set $\beta_1 = 0.9$ and $\beta_2 = 0.95$.
- **WikiText-2+GPT-2.** We continually pretrain a GPT-2 language model Radford et al. (2019) on the WikiText-2 dataset Merity et al. (2016) by splitting the corpus into non-overlapping token blocks. GPT-2 is licensed under MIT license and WikiText-2 is licensed under CC BY-SA 3.0. We experiment with multiple training configurations, changing the number of training samples and block sizes, specifically: 512 and 1024 training samples with block sizes of 128 and 256. All models are trained for three epochs. We use the AdamW optimizer with three learning rates (1×10^{-4}, 5×10^{-5}, and 1×10^{-5}) and a linear learning rate scheduler with warmup. We set the batch size to 32 and use a random masking ensemble of size 10 with projection dimension 512. TSLOO ground truth is computed for 50 randomly selected training samples. The Spearman correlation is computed across these 50 samples and averaged over 256 validation points.
- **Alpaca + Llama 3.2-1B.** We fine-tune a LLaMA-3.2-1B language model (Grattafiori et al., 2024) on the Alpaca instruction-tuning dataset (Taori et al., 2023), using the first 512 training examples with a maximum sequence length of 512 tokens. We train for 1 epoch and evaluate attribution fidelity on 100 test samples from the SAMSum summarization task (Gliwa et al., 2019). We use the AdamW optimizer and sweep learning rates spanning 5×10^{-7}, 2×10^{-6}, and 6×10^{-6} ($\beta_1 = 0.9$, $\beta_2 = 0.999$, max gradient norm 1.0) and a linear learning-rate scheduler with 100 warmup steps. We evaluate both AdamW-influence and SGD-influence.

We set the batch size to 8 and use a random-masking ensemble of size 10 with projection dimension 256. TSLOO ground truth is computed for 50 randomly selected training samples. The Spearman correlation is computed across these 50 samples; attribution scores are then compared against TSLOO effects on 256 SAMSum (Gliwa et al., 2019) test samples.

All quantities in the recurrence are taken from the recorded run: η_t follows the actual warmup and decay schedule, and g_t is the batch gradient after gradient-norm clipping, so the cached moments m_t, v_t match the optimizer's states exactly.

E.2 Experimental settings for dissecting algorithm-level error

In Section 4, we conduct three analyses on the algorithm-level error of AdamW-influence: (i) decomposing the absolute SGD-influence error into config-level, algorithm-level, and residual components (Section 4.1, Figure 2); (ii) sweeping learning rates to identify the two governing factors of the algorithm-level error (Section 4.2, Figure 3); and (iii) validating the closed-form error proxy against ground-truth update-estimation error norms (Section 4.3, Figure 4). Experiments are conducted under the same settings; model architecture, dataset preparation, batch size, optimizer hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.95$), random masking configuration, and the number of LOO training samples (200) and TSLOO validation samples (500) as the ones in Appendix E.1. Below we describe only the analysis-specific details.

- **Error decomposition (Section 4.1, Figure 2).** We use the MNIST+MLP setting and sweep three learning rates (10^{-3}, 10^{-4}, 10^{-5}). For each sample z^*, we compute its absolute SGD-influence error against the TSLOO ground truth and decompose it into the three additive components in Equation equation 4: the optimizer-mismatch share (green), the update-estimation error remaining after optimizer alignment (blue), and the higher-order residual (grey). Errors from samples whose injection step t^* falls within the same 5-step window are aggregated into a single bin, yielding the binned curves shown in Figure 2.
- **Sweep of governing factors (Section 4.2, Figure 3).** We use sweep three learning rates for all settings. For each (z^*, t^*) pair, we compute one metric: the intra-step Spearman correlation between AdamW-influence scores and ground-truth TSLOO scores computed over all samples sharing the same injection step t^*.
- **Error proxy validation (Section 4.3, Figure 4).** We use the MNIST+MLP and MNIST+CNN settings, each at two learning rates (10^{-3} and 10^{-4}). For every training sample (4992 in total per setting), we compute the closed-form error proxy from Equation equation 4 along the original training trajectory, alongside the ground-truth update-estimation error norm $\|\Delta\theta_{z^*} - \text{AdamW-influence}_{z^*}\|_2$ obtained via trajectory-specific LOO retraining. Each point in Figure 4 corresponds to one training sample; the reported Spearman ρ, slope, and R^2 are computed across all 4992 samples per panel.

E.3 Experimental settings for data selection

In Section 5, we evaluate the unified K-step look-ahead framework on four settings: MNIST+MLP, MNIST+CNN, Tulu3+Llama 3.2-1B, and Alpaca+Llama 3.2-1B. Across all three settings, we compare three methods: *offline AdamW-influence*, *online SGD-influence*, and *online AdamW-influence*. Offline AdamW-influence does not involve K, since it scores candidates against a completed reference trajectory and retrains from scratch on the selected subset. For online methods, the look-ahead horizon K is swept and the best K per setting (selected by validation performance) is reported in Figure 5: $K = 10$ for MNIST+MLP, $K = 2$ for Tulu3+Llama 3.2-1B, and $K = 2$ for Alpaca+Llama 3.2-1B. Section 5.2 additionally sweeps $K \in \{2, 5, 10, 25\}$ across three learning rates on MNIST+MLP (Figure 6). Below we describe the per-setting details.

- **MNIST+MLP.** Model architecture and dataset preparation follow Appendix E.1. We train for 20 epochs (rather than 1 epoch as in fidelity evaluation) to allow the model to converge under the reduced effective batch size induced by selection. At each training step, the candidate batch size is $N = 64$ and the final batch size after selection is $B = 32$, corresponding to a selected ratio of 0.5. For Figure 5, we use learning rate 10^{-3} and $K = 10$. For the K-sweep in Figure 6, we sweep $K \in \{2, 5, 10, 25\}$ across three learning

rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$. Validation and test error rates are reported at the best validation epoch.

- **MNIST+CNN.** Model architecture and dataset preparation follow Appendix E.1. We train for 20 epochs (rather than 1 epoch as in fidelity evaluation) to allow the model to converge under the reduced effective batch size induced by selection. At each training step, the candidate batch size is $N = 64$ and the final batch size after selection is $B = 32$, corresponding to a selected ratio of 0.5. For Figure 5, we use learning rate 10^{-3} and $K = 5$. For the K-sweep in Figure 6, we sweep $K \in \{2, 5, 10\}$ across three learning rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$. Validation and test error rates are reported at the best validation epoch.
- **Alpaca+Llama 3.2-1B.** Model, optimizer hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, max gradient norm 1.0), maximum sequence length (512 tokens), training batch size (8), random-masking ensemble of size 10 with projection dimension 256, and linear learning-rate scheduler with 100 warmup steps follow Appendix E.1. We deviate from the fidelity setup as follows: (i) we use 40% of the Alpaca dataset for training (rather than the first 512 examples); (ii) we use a learning rate of 2×10^{-6}; (iii) full fine-tuning is used (no LoRA); (iv) the candidate batch size is $N = 8$ and the selected ratio is 0.8, so $B = 6$ samples are retained per step; and (v) validation uses 16 randomly sampled SAMSum examples for online method selection, and final evaluation is reported on the full SAMSum test set. Validation and test perplexity are reported at the final training step.
- **Tulu3+Llama 3.2-1B.** The training setup mirrors the Alpaca+Llama 3.2-1B configuration above, with the following differences: (i) we use 1% of the Tulu3 dataset for training; (ii) the learning rate is 4×10^{-5} for full fine-tuning (used in Figure 5) and 3×10^{-3} for LoRA fine-tuning (used only in Tables 5 and 6 of Appendix F.3); (iii) the candidate batch size is $N = 8$ and the selected ratio is 0.5, so $B = 4$ samples are retained per step; and (iv) validation uses 16 randomly sampled tydiqa (Clark et al., 2020) examples for online method selection, and final evaluation is reported on the full tydiqa test set. Validation and test perplexity are reported at the final training step.
- **Additional baseline in Tables 5 and 6.** The Tulu3+Llama 3.2-1B results in Appendix F.3 additionally include TracIn (Pruthi et al., 2020) as an offline baseline, and sweep both fine-tuning regime (full vs. LoRA) and selected ratio (0.5 vs. 0.8). All other training and evaluation details follow the Tulu3+Llama 3.2-1B configuration above.

E.4 Compute Resources

The experiment is carried out on A100 GPUs. The total GPU hour required for the experiments is around 400 hours.

F Additional Experiment Results

F.1 Fidelity additional results

Tables 3 and 4 together demonstrate that attribution fidelity hinges on aligning the attribution formula with the actual training optimizer. Table 3 extends Table 1 to additional data sizes and learning rates: when training with AdamW, AdamW-influence consistently outperforms SGD-influence across all four settings, with relative improvements ranging from 10% to over 300%. The improvement is especially large at higher learning rates, where the SGD update fails most dramatically as a proxy for the AdamW update. We also reported the error bar under 10 independent runs for MNIST experiments and 5 independe runs for GPT-2 / Llama 3.2 experiments.

Table 4 reports the symmetric sanity check: when the model is instead trained with SGD, SGD-influence dominates AdamW-influence by a similarly wide margin. This is the expected behavior. Under SGD training, the cached optimizer states g_t, m_t, v_t that AdamW-influence relies on are computed along an SGD trajectory rather than an AdamW one, so AdamW-influence is unrolling an optimizer that the model never actually used. The result confirms that the gains in Table 3 come from *matching* the attribution formula to the training optimizer, not from any intrinsic superiority of AdamW-influence over SGD-influence: each method attributes faithfully only when its assumed optimizer matches the one used in training.

Table 3: Full fidelity comparison (Spearman ρ against TSLOO ground truth) between AdamW-influence and SGD-influence when training with **AdamW**, across varying epochs, data sizes, and learning rates. Best result per row in **bold**. Δ is the relative improvement of AdamW-influence over SGD-influence.

Setting	Epochs	Data Size	LR	AdamW-influence	SGD-influence	Δ
MNIST+MLP	1	4992	1e-2	$\mathbf{0.043 \pm 0.003}$	0.023 ± 0.004	$+87\%$
			1e-3	$\mathbf{0.205 \pm 0.011}$	0.075 ± 0.009	$+173\%$
			1e-4	$\mathbf{0.294 \pm 0.014}$	0.242 ± 0.016	$+21\%$
			1e-5	$\mathbf{0.786 \pm 0.013}$	0.715 ± 0.012	$+10\%$
			1e-6	$\mathbf{0.948 \pm 0.006}$	0.833 ± 0.011	$+14\%$
MNIST+CNN	1	4992	1e-2	$\mathbf{0.048 \pm 0.008}$	0.027 ± 0.011	$+74\%$
			1e-3	$\mathbf{0.090 \pm 0.008}$	0.015 ± 0.009	$+500\%$
			1e-4	$\mathbf{0.511 \pm 0.017}$	0.122 ± 0.015	$+319\%$
			1e-5	$\mathbf{0.791 \pm 0.007}$	0.526 ± 0.007	$+50\%$
			1e-6	$\mathbf{0.767 \pm 0.013}$	0.523 ± 0.009	$+47\%$
WikiText+GPT-2	3	65K (block size= 128)	1e-4	$\mathbf{0.734 \pm 0.003}$	0.372 ± 0.006	$+97\%$
			5e-5	$\mathbf{0.825 \pm 0.002}$	0.432 ± 0.009	$+91\%$
			1e-5	$\mathbf{0.842 \pm 0.006}$	0.480 ± 0.015	$+75\%$
		131K (block size= 256)	1e-4	$\mathbf{0.672 \pm 0.009}$	0.321 ± 0.011	$+109\%$
			5e-5	$\mathbf{0.787 \pm 0.003}$	0.383 ± 0.011	$+105\%$
			1e-5	$\mathbf{0.810 \pm 0.009}$	0.420 ± 0.011	$+93\%$
		262K (block size= 256)	1e-4	$\mathbf{0.327 \pm 0.013}$	0.158 ± 0.013	$+107\%$
			5e-5	$\mathbf{0.471 \pm 0.012}$	0.223 ± 0.009	$+111\%$
			1e-5	$\mathbf{0.785 \pm 0.008}$	0.393 ± 0.007	$+100\%$
Alpaca+Llama 3.2	1	512	6e-6	$\mathbf{0.276 \pm 0.047}$	0.156 ± 0.019	$+77\%$
			2e-6	$\mathbf{0.342 \pm 0.036}$	0.197 ± 0.007	$+74\%$
			5e-7	$\mathbf{0.665 \pm 0.013}$	0.330 ± 0.017	$+102\%$

Table 4: Fidelity comparison (Spearman ρ against TSLOO ground truth) between AdamW-influence and SGD-influence when training with **SGD**, across varying epochs, data sizes, and learning rates. Best result per row in **bold**.

Setting	Epochs	Data Size	LR	AdamW-influence	SGD-influence
MNIST+MLP	1	4992	1e-2	0.015 ± 0.004	$\mathbf{0.349 \pm 0.016}$
			1e-3	0.047 ± 0.005	$\mathbf{0.707 \pm 0.023}$
			1e-4	0.447 ± 0.017	$\mathbf{0.939 \pm 0.012}$
			1e-5	0.746 ± 0.015	$\mathbf{0.968 \pm 0.001}$
			1e-6	0.833 ± 0.010	$\mathbf{0.969 \pm 0.001}$
MNIST+CNN	1	4992	1e-2	0.042 ± 0.004	$\mathbf{0.389 \pm 0.015}$
			1e-3	0.036 ± 0.006	$\mathbf{0.808 \pm 0.009}$
			1e-4	0.061 ± 0.006	$\mathbf{0.599 \pm 0.011}$
			1e-5	0.461 ± 0.009	$\mathbf{0.613 \pm 0.015}$
			1e-6	0.480 ± 0.010	$\mathbf{0.616 \pm 0.016}$

F.2 AdamW-influence and MAGIC

MAGIC (Ilyas & Engstrom, 2025) provides a high-fidelity trajectory-based attribution estimator, but its computation is impractical for large models and it requires propagating influence separately for each test sample or query, causing the cost to scale with the number of queries. In this subsection, we show that when AdamW-influence is computed using the exact Hessian and the full parameter space, it recovers the *same* estimator as MAGIC.

Figure 7 verifies this connection empirically: across both MNIST+MLP and WikiText-2+GPT-2 and multiple learning rates, AdamW-influence with the true Hessian lies almost exactly on the $y = x$ line with MAGIC, with Spearman correlations close to one. In contrast to MAGIC, however, AdamW-influence could benefit from the backward recurrence in Appendix C.4, so the expensive trajectory propagation does not grow with the number of test queries. Moreover, practical approximations such as the empirical-Fisher/GGN approximation and random masking (Appendix D) can substantially

reduce computation and memory while largely preserving attribution fidelity, making AdamW-influence considerably more scalable to modern models.

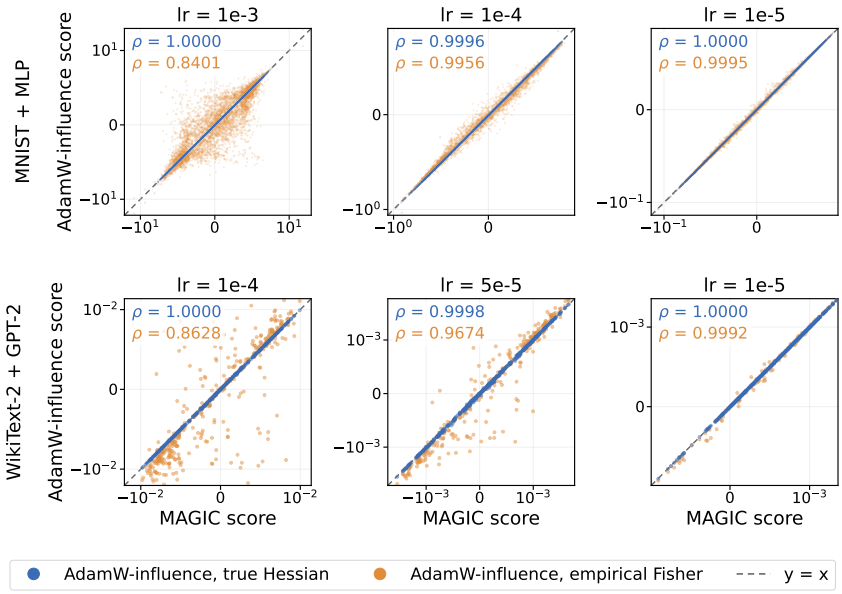


Figure 7: AdamW-influence calculated with the real Hessian and full parameters is equivalent to MAGIC.

F.3 Data selection additional results

Tables 5 and 6 extend the Tulu3 results in Section 5.2 along two practical axes: the fine-tuning regime (full fine-tuning vs. LoRA) and the selected data ratio (0.5 vs. 0.8). Online AdamW-influence achieves the lowest perplexity across all eight configurations on both validation and test splits.

Table 5: Evaluation perplexity on Tulu3 (tydiqa) with Llama-3.2-1B. Lower is better; **bold** indicates the best result per column.

	Full		LoRA	
Method \ Selected Ratio	0.5	0.8	0.5	0.8
TracIN (offline)	2.152±0.063	2.175±0.081	1.955±0.027	1.910±0.065
SGD-influence (online)	2.029±0.034	2.055±0.077	1.810±0.026	1.872±0.016
AdamW-influence (offline)	1.957±0.021	1.939±0.017	1.868±0.028	1.872±0.061
AdamW-influence (online)	**1.886±0.045**	**1.933±0.018**	**1.764±0.021**	**1.859±0.012**

Table 6: Validation perplexity on Tulu3 (tydiqa) with Llama-3.2-1B. Lower is better; **bold** indicates the best result per column.

	Full		LoRA	
Method \ Selected Ratio	0.5	0.8	0.5	0.8
TracIN (offline)	2.066±0.087	2.154±0.140	1.811±0.035	1.872±0.069
SGD-influence (online)	1.849±0.057	1.857±0.022	1.729±0.017	1.730±0.054
AdamW-influence (offline)	1.836±0.042	1.874±0.018	1.803±0.042	1.836±0.062
AdamW-influence (online)	**1.742±0.025**	**1.776±0.041**	**1.707±0.042**	**1.723±0.063**

G Derivation of Proposition 2

This appendix derives Proposition 2 (the closed-form error proxy in Section 4.3). All notation follows Appendix C.1: D_t and S_t are the diagonal matrices defined there, and $\dot{(\cdot)}$ denotes derivatives with respect to the gradient-level perturbation scalar ϵ at $\epsilon = 0$. We treat $\lambda = 0$; the extension to $\lambda > 0$ replaces each $(I - A_k^{\text{eff}})$ below with $((1-\eta_k\lambda)I - A_k^{\text{eff}})$ and leaves $r_t(0)$ unchanged.

G.1 Error recursion

Recall the AdamW update $\theta_{t+1}(\epsilon) = \theta_t(\epsilon) - U_t(\epsilon)$ with $U_t = \eta_t \hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$, and its linearization $\dot{\theta}_{t+1} = \dot{\theta}_t - \dot{U}_t$. Define

$$e_t := [\theta_t(1) - \theta_t(0)] - \dot{\theta}_t, \qquad r_t := [U_t(1) - U_t(0)] - \dot{U}_t.$$

Subtracting the linearized recursion from the true one gives $e_{t+1} = e_t - r_t$ with $e_{t^*} = 0$.

Although $U_t(1)$ is evaluated on the ground-truth retrained trajectory $\theta_t(1)$, expressing $\theta_t(1) = \theta_t(0) + \dot{\theta}_t + e_t$ and Taylor-expanding U_t around $\theta_t(0)$ introduces an e_t-dependence:

$$r_t(e_t) \approx r_t(0) + A_t^{\text{eff}}\, e_t, \qquad A_t^{\text{eff}} := \left.\frac{\partial U_t}{\partial \theta_t}\right|_{\theta_t(0)} = \eta_t \,\frac{1-\beta_1}{1-\beta_1^{t+1}}\, D_t H_t - 2\eta_t \,\frac{1-\beta_2}{1-\beta_2^{t+1}}\, S_t \,\mathrm{diag}(g_t)\, H_t,$$

where $r_t(0)$ is the residual evaluated on the unperturbed trajectory and A_t^{eff} follows from differentiating U_t through the moment recurrences. Substituting yields $e_{t+1} = (I - A_t^{\text{eff}})\, e_t - r_t(0)$, which unrolls to

$$e_T = -\sum_{t=t^*}^{T-1} \left[\prod_{k=t+1}^{T-1} (I - A_k^{\text{eff}})\right] r_t(0). \tag{13}$$

When $\|A_t^{\text{eff}}\| \ll 1$, the propagation factor $\prod_k (I - A_k^{\text{eff}}) \approx I$ and the proxy simplifies to $e_T \approx -\sum_t r_t(0)$, which we use to compute Figure 4.

G.2 Magnitude of $r_t(0)$

Writing $U_{t,i} = \eta_t f_{t,i}$ with $f_{t,i}(\epsilon) = \hat{m}_{t,i}(\epsilon)/(\sqrt{\hat{v}_{t,i}(\epsilon)} + \varepsilon)$, we have $r_{t,i}(0) = \frac{1}{2}\eta_t \ddot{f}_{t,i}|_{\epsilon=0}$. Let $h_i := \sqrt{\hat{v}_{t,i}} + \varepsilon$. Direct differentiation gives

$$\ddot{f}_{t,i} = \frac{\ddot{\hat{m}}_{t,i}}{h_i} - \frac{\hat{m}_{t,i}\,\ddot{h}_i}{h_i^2} - \frac{2\,\dot{\hat{m}}_{t,i}\,\dot{h}_i}{h_i^2} + \frac{2\,\hat{m}_{t,i}\,\dot{h}_i^2}{h_i^3}, \tag{14}$$

with $\dot{h}_i = \dot{\hat{v}}_{t,i}/(2\sqrt{\hat{v}_{t,i}})$ and $\ddot{h}_i = \ddot{\hat{v}}_{t,i}/(2\sqrt{\hat{v}_{t,i}}) - \dot{\hat{v}}_{t,i}^2/(4\hat{v}_{t,i}^{3/2})$.

At $t > t^*$, the moment derivatives admit the leading-order expressions (suppressing bias-correction prefactors):

$$\dot{\hat{m}}_{t,i} \sim [H_t \dot{\theta}_t]_i, \qquad \ddot{\hat{m}}_{t,i} \sim \sum_{j,k} [\nabla^3 \ell_t]_{ijk}\, \dot{\theta}_{t,j}\, \dot{\theta}_{t,k} \sim \|\dot{\theta}_t\|^2, \qquad \dot{h}_i \sim \frac{g_{t,i}\,\dot{g}_{t,i}}{\sqrt{\hat{v}_{t,i}}}, \qquad \ddot{h}_i \sim \frac{\dot{g}_{t,i}^2}{\sqrt{\hat{v}_{t,i}}}.$$

Using $h_i \approx \sqrt{\hat{v}_{t,i}}$ and the Adam normalization $\hat{m}_{t,i}/\sqrt{\hat{v}_{t,i}} = \mathcal{O}(1)$, the four terms in equation 14 scale as $\|\dot{\theta}_t\|^2/\sqrt{\hat{v}_{t,i}}$, $[H_t\dot{\theta}_t]_i^2/\hat{v}_{t,i}$, and (for terms 3 and 4) $[H_t\dot{\theta}_t]_i^2/\hat{v}_{t,i}$ further suppressed by $|g_{t,i}|/\sqrt{\hat{v}_{t,i}} \le 1$ (the batch-mean gradient is bounded above by the per-coordinate RMS accumulated by Adam). Terms 1 and 2 dominate, giving

$$\left|r_{t,i}(0)\right| = \tfrac{1}{2}\eta_t \left|\ddot{f}_{t,i}\right| \sim \eta_t \left(\frac{\|\dot{\theta}_t\|^2}{\sqrt{\hat{v}_{t,i}}} + \frac{[H_t\dot{\theta}_t]_i^2}{\hat{v}_{t,i}}\right), \tag{15}$$

which recovers the residual bound stated in Proposition 2. The first term originates from the loss nonlinearity ($\nabla^3\ell$) entering $\ddot{\hat{m}}_{t,i}$ and is present even for non-adaptive optimizers; the second originates from the quadratic dependence of $\hat{v}_t$ on gradient perturbations and is specific to Adam-family optimizers. Both belong to a single second-order remainder of $f_{t,i}$ and are not independent error sources.

H PSEUDO-CODE OF THE UNIFIED K-STEP SELECTION LOOP

Algorithm 2 summarizes the unified K-step look-ahead data selection loop introduced in Section 5.1.

Algorithm 2 K-step look-ahead data selection

Require: Candidate batch size N; training batch size B; look-ahead horizon K; validation set V

```
Initialize model parameters
for each training step t do
    Sample a candidate batch of N examples from the stream
    for each candidate z in the candidate batch do
        Score z by how much including z now would reduce validation loss on V, evaluated K steps later
    end for
    Keep the top-B candidates as the training batch
    Update model parameters on the selected training batch
end for
return trained model
```

Two boundary cases are worth highlighting. $K = 0$ recovers the per-step scoring rule used by GREATS Wang et al. (2024) (without its greedy selection iteration). The limit of K equal to the remaining training steps is the full-trajectory horizon; it is intractable in the online loop and is instead approximated by offline selection on a completed reference trajectory.

I RELATION TO THE OPTIMIZER-AWARE EXTENSION OF SOURCE

SOURCE (Bae et al., 2024) also provides an optimizer-aware extension of approximate unrolling through a preconditioning matrix. Specifically, its update takes the form

$$\theta_{k+1}(\epsilon) = \theta_k(\epsilon) - \frac{\eta_k}{B} P_k \sum_{i=1}^{B} (1 + \delta_{ki}\epsilon) \nabla_\theta \mathcal{L}(z_{ki}, \theta_k(\epsilon)),$$

which yields a propagation Jacobian of the form

$$J_k = I - \eta_k P_k H_k.$$

This formulation incorporates the adaptive scaling induced by optimizers such as AdamW through P_k, but P_k is treated as fixed when differentiating the perturbed trajectory; SOURCE further approximates it as stationary within each trajectory segment. Consequently, the sample perturbation does not propagate through the optimizer states that generate the preconditioner. This distinction is important for AdamW: removing a sample perturbs the gradient, which in turn changes both the first moment m_k and the second moment v_k. In particular, the dependence of the update on v_k is nonlinear, so freezing the preconditioner breaks the perturbation pathway through the adaptive denominator.

AdamW-influence instead injects the perturbation at the gradient level and propagates it through the joint dynamics of θ_k, m_k, and v_k. As derived in Appendix C, this gives a linear dynamical system over the augmented perturbation state $(\dot{\theta}_k, \dot{m}_{k-1}, \dot{v}_{k-1})$, thereby accounting for how a sample affects both optimizer states and how those state changes subsequently affect the parameter trajectory.

The two methods also differ in their estimand and granularity. SOURCE estimates the expected total derivative by marginalizing over training stochasticity through segment-level statistical summaries, whereas AdamW-influence targets the TSLOO along the actual recorded trajectory and unrolls the per-step dynamics.