IN-RUN DATA SHAPLEY FOR ADAM OPTIMIZER

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable data attribution is essential for understanding, debugging, and curating modern machine learning training data, with the Shapley value serving as a principled framework for data valuation. Recent In-Run Data Shapley methods avoid the prohibitive cost of retraining by decomposing data value into step-wise contributions along a single realized training trajectory. However, existing scalable in-run estimators are derived from the linear update structure of stochastic gradient descent (SGD), while modern deep learning pipelines are widely trained with adaptive optimizers such as Adam and AdamW. In this work, we show that data attribution is optimizer-dependent: SGD-induced and Adam-induced data values exhibit extremely low agreement in both magnitude and ranking, indicating that SGD-based attribution can lead to misleading rank-based decisions under Adam training. To address this mismatch, we propose *Adam-Aware In-Run Data Shapley*, an optimizer-consistent attribution framework for Adam training. We define a fixed-state Adam local utility and derive a first-order approximation that explicitly accounts for momentum and coordinate-wise variance normalization. To make the estimator scalable, we introduce a *Linearized Ghost Approximation*, which restores an additive dot-product form without materializing per-sample gradients. Experiments show that our method achieves near-perfect fidelity to exact local Shapley values under the fixed-state Adam utility ($R > 0.99$), substantially improves efficiency over direct per-sample Adam-aware computation, and improves downstream rank-based attribution tasks including semantic source identification and data pruning.

1 INTRODUCTION

Modern machine learning models rely heavily on large-scale training data Gao et al. (2020); Raffel et al. (2020), yet their performance and behavior can be substantially shaped by the quality and influence of individual training examples. The lack of reliable data attribution mechanisms can lead to degraded performance and computational inefficiency. Without principled mechanisms to isolate and evaluate sample influence, trained models may amplify harmful biases or remain vulnerable to data poisoning attacks Koh & Liang (2017); Ghorbani & Zou (2019); Ren et al. (2025a). Furthermore, the presence of problematic data escalates the computational cost of modern training pipelines by wasting resources on uninformative or harmful examples Paul et al. (2021); Ren et al. (2025b); Hu et al. (2025). Therefore, an actionable and reliable attribution method should quantify how each training example contributes to the specific model produced by the realized training pipeline.

To fairly attribute data contributions, the *Shapley value* has emerged as the gold standard. Originating from cooperative game theory Shapley (1953), the Shapley value provides a unique, unbiased distribution of the total model performance among individual data points based on their marginal contributions, emerging as a widely adopted framework Ghorbani & Zou (2019); Jia et al. (2019b). While it has desirable properties, such as fairness, efficiency, and additivity (Jia et al., 2019a), the classic calculation of Shapley values requires retraining models on many subsets of the training data, which is computationally infeasible for modern deep learning pipelines. Recent work on In-Run Data Shapley addresses this scalability barrier by decomposing data value into step-wise contributions along a single realized training trajectory (Wang et al., 2025). This in-run perspective is especially appealing for modern models because it attributes data contribution to the specific training run that produced the deployed model, rather than to an average over many hypothetical retraining runs.

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.

However, this in-run perspective also raises a natural question: are data values consistent across different optimizers? Existing In-Run Data Shapley is derived under the update rule of stochastic gradient descent (SGD), where each parameter update is a linear combination of per-sample gradients. This linearity reduces the local Shapley value to a gradient-gradient inner product and enables efficient ghost-style computation. In contrast, modern deep learning models are widely trained with adaptive optimizers such as Adam Kingma & Ba (2014) and AdamW Loshchilov & Hutter (2017), where the update direction is determined by historical momentum and coordinate-wise variance normalization. As a result, an SGD-based In-Run attribution score may not faithfully capture the contribution of a data point under Adam training, since it corresponds to different optimizer-induced training dynamics.

In this paper, we study optimizer-consistent In-Run Data Shapley for Adam training, with the goal of attributing data contribution along the realized Adam optimization trajectory while preserving the scalability of in-run attribution. Our contributions are summarized as follows:

1. Optimizer-Aware Data Attribution. We demonstrate that data value is not an intrinsic property of the dataset, but is fundamentally coupled to the optimization trajectory. Using retraining-based Truncated Monte Carlo (TMC) Shapley as a model-agnostic reference, we show that SGD-induced and Adam-induced data values exhibit extremely low agreement in both magnitude and ranking (Pearson $R = 0.0579$, Spearman $\rho = 0.0465$), indicating that SGD-based attribution can lead to misleading rank-based decisions under Adam training.

2. Adam-Aware In-Run Data Shapley. We derive the first Adam-aware In-Run Data Shapley estimator. By defining a fixed-state local utility under Adam and applying a first-order approximation to the Adam update map, our formulation explicitly accounts for momentum and coordinate-wise variance normalization along the realized Adam trajectory.

3. Scalable "Linearized Ghost" Computation. To overcome the nonlinearity of Adam updates, which prevents standard efficient aggregation, we introduce the Linearized Ghost Approximation. This technique restores an additive structure in the local utility, allowing Adam-aware attribution to be computed through ghost-style dot-products without materializing per-sample gradients. In our GPT-2 Small benchmark, it improves throughput by $2.24\times$ and reduces peak memory by 69.3% compared with a naive direct implementation.

4. Fidelity and Practical Utility. We show that our method achieves near-perfect fidelity to explicit Adam marginal utility changes ($R > 0.99$), substantially outperforming SGD-based proxies under Adam dynamics. We further demonstrate its usefulness in downstream rank-based attribution tasks, including semantic source identification and data pruning.

2 Related Work

Data Attribution Methods. Data attribution quantifies the influence of individual data points on model predictions. A foundational approach uses *influence functions*, originating from statistics (Cook & Weisberg, 1980). In deep learning, Koh & Liang (2017) apply first-order Taylor expansions and inverse Hessian-vector products to estimate the effect of training samples. Several works improve the scalability of influence-function computation. For example, LiSSA provides a stochastic approximation to inverse Hessian-vector products (Agarwal et al., 2017), while recent methods such as DataInf and HyperINF further accelerate influence estimation for large-scale models and parameter-efficient fine-tuning settings (Kwon et al., 2023; Zhou et al., 2024). Influence-function methods have also been extended to study large language models (Grosse et al., 2023), but they can be unstable in non-convex settings (Basu et al., 2020). Trajectory-based methods provide another line of data attribution. For example, TracIn (Pruthi et al., 2020) avoids explicit Hessian inversion by accumulating gradient dot-products along the training trajectory. A related line of work differentiates through the training process, including approximate unrolling and metagradient-based attribution (Bae et al., 2024; Ilyas & Engstrom, 2025). These methods estimate how a data perturbation propagates through future optimization steps and affects the final trained model. Other methods take different perspectives on data value. Datamodels (Ilyas et al., 2022) learn mappings from training subsets to model outputs at a higher computational cost. DAVINZ (Wu et al., 2022) estimates data value at initialization, while validation-free methods remove dependence on a held-out validation set (Xu et al., 2021). In-Run Data Shapley takes a different perspective: it defines a step-wise local utility along the realized training trajectory and aggregates local Shapley values across iterations (Wang et al., 2025). Our work follows this in-run perspective and addresses a different question: how to make the step-wise Shapley utility consistent with the optimizer that actually generated the trajectory.

Data Shapley Value. To quantify data value, prior work adopts the Shapley value from cooperative game theory (Shapley, 1953). In machine learning, Data Shapley (Ghorbani & Zou, 2019) treats training samples as players and model performance as the payoff, but exact computation is NP-hard, motivating efficient approximations. The most common approach is Monte Carlo Shapley, which estimates values by averaging marginal contributions over random permutations (Mitchell et al., 2022). Early work combined this estimator with truncation techniques (Jia et al., 2019b; Wang & Jia, 2023b), while later studies introduced more advanced sampling strategies to improve efficiency and reduce variance, including ergodic sampling (Illés & Kerényi, 2019), stratified empirical Bernstein sampling (Burgess & Chapman, 2021), and multilinear sampling schemes (Okhrati & Lipani, 2021). Recent work further improves scalability through randomized experimental designs (Lin et al., 2022) and stochastic amortization (Covert et al., 2024). Beyond efficiency, several extensions aim to improve robustness and interpretability. Beta Shapley (Kwon & Zou, 2021) emphasizes low-cardinality coalitions, while Data Banzhaf (Wang & Jia, 2023a) and Weighted Banzhaf (Li & Yu, 2023) provide alternative influence measures based on the Banzhaf power index.

Despite these advances, many existing approaches still rely on retraining surrogate models or computationally intensive sampling procedures. The recently proposed *In-Run Data Shapley* (Wang et al., 2025) avoids retraining by decomposing the Shapley value into per-iteration training updates. However, its theoretical derivation critically depends on the linearity of stochastic gradient descent, limiting its applicability to stateful and non-linear optimizers such as Adam. Addressing this gap is the focus of the present work.

3 BACKGROUND

In this section, we introduce the formal setup for data attribution and review the Shapley value. We then describe the In-Run Data Shapley framework, which decomposes data value along a single realized training trajectory. Finally, we revisit the SGD and Adam update rules, highlighting the key structural difference that motivates our Adam-aware formulation.

Problem Setup. Consider a training dataset $D = \{z_i\}_{i=1}^N$ consisting of N individual data points. In data attribution, our goal is to assign each training point $z \in D$ a scalar value that measures its contribution to the performance or behavior of a trained model. Let $U(\cdot)$ denote a utility function that maps a subset of training data $S \subseteq D$ to a real-valued score. For example, $U(S)$ can be the validation accuracy, negative validation loss, or another task-specific performance measure of a model trained on S. Given a utility function U, a data attribution method assigns each point $z \in D$ a scalar score, denoted by $\phi_z(U)$, to quantify its contribution to the chosen utility.

Data Shapley Value. The Shapley value, originating from cooperative game theory (Shapley, 1953), provides a principled way to distribute the total utility among individual data points. In this formulation, each data point is treated as a player, and the model utility is treated as the payoff of the coalition. For any data point $z \in D$, its Data Shapley value with respect to utility U is defined as

$$\phi_z(U) := \frac{1}{N} \sum_{k=1}^{N} \binom{N-1}{k-1}^{-1} \sum_{\substack{S \subseteq D \setminus \{z\} \\ |S|=k-1}} \left[U(S \cup \{z\}) - U(S) \right],$$

where $D_{-z} = D \setminus \{z\}$ represents the dataset with point z excluded and S indicates a subset of D_{-z} with cardinality $|S| = k-1$. The term $[U(S \cup \{z\}) - U(S)]$ measures the marginal contribution of data point z when added to the existing subset S. The popularity of the Shapley value stems from the fact that it is the unique notion of data value satisfying four axioms: Null Player, Symmetry, Linearity, and Efficiency. Here, we introduce the Linearity, which will be used in subsequent analysis.

Lemma 3.1 (Linearity of the Shapley value)**.** *For any two utility functions U_1, U_2 and any $\alpha_1, \alpha_2 \in \mathbb{R}$, $\phi_z(\alpha_1 U_1 + \alpha_2 U_2) = \alpha_1 \phi_z(U_1) + \alpha_2 \phi_z(U_2)$.*

In-Run Data Shapley. Classical Data Shapley is computationally expensive because it requires retraining models on many data subsets. In-Run Data Shapley avoids repeated retraining by attributing data contribution along a single realized training trajectory (Wang et al., 2025).Consider a training run with checkpoints $\{w_t\}_{t=0}^T$, where w_t denotes the model parameter at iteration t. At iteration t, let B_t be the sampled mini-batch and let $z^{(\mathrm{val})}$ be a validation example. For any $S \subseteq B_t$, the *local utility* is defined as the validation loss change induced by updating with S: $U^{(t)}(S; z^{(\mathrm{val})}) := \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}) - \ell(w_t, z^{(\mathrm{val})})$, where $\widetilde{w}_{t+1}(S)$ denotes

the counterfactual next parameter obtained from the current state w_t using the subset S. The global in-run utility is then defined by accumulating the local utilities over the training trajectory $U(S; z^{(\mathrm{val})}) = \sum_{t=0}^{T-1} U^{(t)}(S; z^{(\mathrm{val})})$. By the linearity of the Shapley value, the corresponding In-Run Data Shapley value can be decomposed as $\phi_z(U) = \sum_{t=0}^{T-1} \phi_z\big(U^{(t)}\big)$, where $\phi_z(U^{(t)}) = 0$ if $z \notin B_t$. Thus, In-Run Data Shapley measures the cumulative contribution of a data point across the iterations in which it appears.

SGD Local Utility and Ghost Computation. Existing In-Run Data Shapley is derived under the update rule of stochastic gradient descent. For a subset $S \subseteq B_t$, the counterfactual SGD update is $\widetilde{w}_{t+1}(S) = w_t - \eta_t \sum_{z \in S} \nabla_w \ell(w_t, z)$, where η_t is the learning rate. Applying a first-order Taylor expansion of the validation loss around w_t gives

$$U_{\mathrm{SGD}}^{(t)}(S; z^{(\mathrm{val})}) \approx -\eta_t \left\langle \nabla_w \ell(w_t, z^{(\mathrm{val})}), \sum_{z \in S} \nabla_w \ell(w_t, z) \right\rangle.$$

This expression is additive over samples in S. Therefore, for any $z \in B_t$, the local Shapley value reduces to a gradient-gradient inner product $\phi_z(U_{\mathrm{SGD}}^{(t)}) \approx -\eta_t \langle \nabla_w \ell(w_t, z^{(\mathrm{val})}), \nabla_w \ell(w_t, z) \rangle$, which enables efficient ghost-style computation without materializing per-sample gradients (more discussions in Section 4.3).

Adam Optimizer. Adam Kingma & Ba (2014) is a widely used adaptive optimizer that maintains exponential moving averages of both first-order (m_t) and second-order (v_t) gradient moments. At each step t, the update is given by:

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

where $g_t(S) = \nabla_w \ell(w_t; S)$ is the gradient of the training loss on the coalition S and $\odot$ denotes element-wise multiplication. For clarity, we omit bias-correction terms or absorb them into the effective learning rate. The corresponding counterfactual Adam update is $\widetilde{w}_{t+1}(S) = w_t - \eta_t \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon}$, where η_t is the learning rate and ϵ is a small constant for numerical stability. Compared with SGD, Adam is stateful and nonlinear in the coalition gradient $g_t(S)$ due to historical moment estimates and coordinate-wise variance normalization. Therefore, the local utility under Adam no longer reduces to an additive sum of per-sample gradient inner products, preventing a direct application of standard In-Run Data Shapley and ghost dot-product computation.

4 In-Run Data Shapley for Adam Optimizer

In this section, we first empirically verify that data values are optimizer-dependent. We then develop an Adam-aware In-Run Data Shapley formulation and introduce an efficient Linearized Ghost computation for Adam-based training.

4.1 Optimizer Dependence of Data Value

We begin by investigating a fundamental question: Are data Shapley values consistent across different optimization algorithms? To answer this, we computed the Shapley values for the same dataset and model architecture trained under two distinct regimes: SGD and Adam. We use retraining-based Truncated Monte Carlo Shapley Ghorbani & Zou (2019) as a model-agnostic reference. TMC estimates the Shapley value by sampling random permutations of the training data and averaging the marginal utility change induced by each sample. Although computationally expensive, this retraining-based estimator allows us to compare how the choice of optimizer changes the induced data values without relying on our proposed approximation. (More experimental setup details refer to Appendix B.)

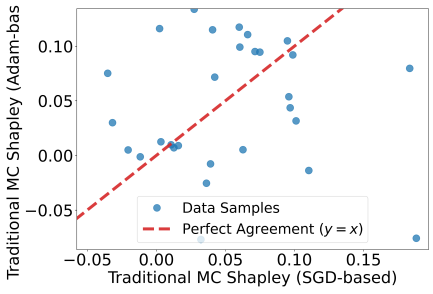


Figure 1: Comparison between SGD-based and Adam-based Data Shapley values (Pearson $R = 0.0579$, Spearman $\rho = 0.0465$).

As illustrated in Figure 1, attribution scores derived under the SGD trajectory diverge significantly from those computed under Adam. The scatter plot reveals a clear lack of agreement, with many samples deviating substantially from the identity line ($y = x$). Quantitatively, the Pearson correlation between SGD-based and Adam-based values is extremely low ($R = 0.0579$), showing that the two optimizers assign substantially different attribution magnitudes. Moreover, the Spearman correlation is also very low ($\rho = 0.0465$), indicating that the discrepancy is not merely a rescaling of value scores, but also changes the ranking of training samples. This inconsistency demonstrates that data value is not a static property intrinsic to the sample, but is fundamentally coupled to the optimization dynamics. Mechanistically, a sample that is highly influential under the linear updates of SGD may contribute differently under Adam, where the update direction is modulated by historical momentum and adaptive variance scaling. This suggests that relying on SGD-based attribution for adaptive optimizers can be misleading, motivating the need for an optimizer-aware framework. In Section 5, we further compare the proposed Adam-aware In-Run Shapley estimator with the standard SGD-based In-Run proxy. The results show that accounting for Adam dynamics substantially improves fidelity to explicit Adam marginal utility changes.

4.2 In-Run Data Shapley for Adam

We now extend the in-run attribution framework to Adam. Under SGD, the local utility is additive over the coalition $S \subseteq B_t$, and the local Shapley value reduces to a gradient-gradient inner product:

$$\phi_z(U^{(t)}_{\mathrm{SGD}}) \approx -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \nabla \ell(w_t, z) \right\rangle .$$

This additive structure is the key reason why existing In-Run Data Shapley can be computed efficiently using ghost-style dot-products. However, Adam introduces optimizer states and adaptive variance normalization, so the update induced by a coalition is no longer a linear sum of per-sample gradients.

At iteration t, we condition on the realized Adam state (w_t, m_{t-1}, v_{t-1}). For a coalition $S \subseteq B_t$, let $g_t(S) := \sum_{z \in S} \nabla \ell(w_t, z)$ denote the coalition gradient. Under the fixed-state local counterfactual, the previous moments (m_{t-1}, v_{t-1}) are held fixed, while the current gradient is replaced by $g_t(S)$. Thus,

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

and the corresponding counterfactual Adam update is $\widetilde{w}_{t+1}(S) = w_t - \eta_t \frac{m_t(S)}{\sqrt{v_t(S)+\epsilon}}$. We define the Adam local utility by the validation loss change after this counterfactual update: $U^{(t)}_{\mathrm{Adam}}(S; z^{(\mathrm{val})}) := \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}) - \ell(w_t, z^{(\mathrm{val})})$. Applying a first-order Taylor expansion of the validation loss around w_t gives the following local approximation.

Lemma 4.1. *For any iteration $t = 0, \ldots, T-1$, under the fixed-state Adam local counterfactual, the first-order approximation of the Adam local utility is*

$$U^{(t)}_{\mathrm{Adam}}(S; z^{(\mathrm{val})}) \approx -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon} \right\rangle .$$

Lemma 4.1 shows that, under Adam, the local utility depends on the optimizer-induced update direction rather than the raw coalition gradient. This is fundamentally different from SGD: due to the variance term $v_t(S)$, the Adam update is nonlinear in the coalition gradient $g_t(S)$. As a result, the local utility is no longer additive over samples in S, and the corresponding Shapley value cannot be reduced to a standard gradient-gradient inner product. Therefore, the efficient ghost dot-product computation used in prior In-Run Data Shapley cannot be directly applied, which motivates the Linearized Ghost Approximation introduced next.

4.3 Efficient Computation via Linearized Ghost Dot-Product

A major advantage of prior In-Run Data Shapley Wang et al. (2025) is that, under SGD, the local contribution reduces to a gradient-gradient inner product. This structure enables the Ghost Dot-Product technique, which computes pairwise inner products between per-sample gradients and validation gradients without explicitly materializing full per-sample gradients.

Specifically, for a given layer l, let $a_i^{(l)}$ denote the input activation vector and $\delta_i^{(l)}$ denote the backpropagated error vector for sample z_i. The gradient of the weight matrix $W^{(l)}$ with respect to z_i can be written as the outer product $g_i^{(l)} = \delta_i^{(l)}(a_i^{(l)})^\top$. Thus, the gradient dot-product between a training sample z_i and a validation sample $z^{(\mathrm{val})}$ can be computed layer-wise as

$$\nabla \ell(w_t, z_i)^\top \nabla \ell(w_t, z^{(\mathrm{val})}) = \sum_l \left\langle \delta_i^{(l)}(a_i^{(l)})^\top, \delta_{\mathrm{val}}^{(l)}(a_{\mathrm{val}}^{(l)})^\top \right\rangle_F = \sum_l \underbrace{(\delta_i^{(l)\top}\delta_{\mathrm{val}}^{(l)})}_{\text{error correlation}} \cdot \underbrace{(a_i^{(l)\top}a_{\mathrm{val}}^{(l)})}_{\text{activation correlation}} .$$

Therefore, the required dot-products can be obtained from activations and backpropagated errors, without instantiating per-sample gradients. However, this standard ghost computation cannot be directly applied to Adam. From Lemma 4.1, the Adam local utility depends on the nonlinear update map $\Psi_t(g) := \frac{\beta_1 m_{t-1} + (1-\beta_1)g}{\sqrt{\beta_2 v_{t-1} + (1-\beta_2) g^{\odot 2}} + \epsilon}$, evaluated at the coalition gradient $g_t(S)$. Thus,

$$U_{\mathrm{Adam}}^{(t)}(S; z^{(\mathrm{val})}) \approx -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \Psi_t(g_t(S)) \right\rangle .$$

Since $\Psi_t(\cdot)$ is nonlinear in $g_t(S)$, the marginal contribution of adding a sample no longer takes the standard gradient-gradient form.

To address this challenge, we introduce the **Linearized Ghost Approximation**. The key idea is to linearize $\Psi_t(\cdot)$ around a fixed reference gradient $\bar{g}_t$, which we take as the observed batch gradient $\Psi_t(g) \approx \Psi_t(\bar{g}_t) + J_t(g - \bar{g}_t), J_t := \left.\frac{\partial \Psi_t(g)}{\partial g}\right|_{g=\bar{g}_t}$. Substituting this approximation into the Adam local utility gives an affine function of the coalition gradient $g_t(S)$. The terms independent of S vanish under the Shapley value, and the remaining term restores an additive structure over samples in S. Consequently, the local contribution can again be written as a bilinear form between an Adam-aware validation direction and a per-sample gradient, making ghost-style computation applicable.

Theorem 4.2. *Let $\bar{g}_t$ be a fixed reference gradient and $J_t = \left.\frac{\partial \Psi_t(g)}{\partial g}\right|_{g=\bar{g}_t}$. Under the Linearized Ghost Approximation $\Psi_t(g_t(S)) \approx \Psi_t(\bar{g}_t) + J_t(g_t(S) - \bar{g}_t)$, the resulting linearized Adam local utility satisfies, for any $z \in B_t$, $\phi_z(\widetilde{U}_{\mathrm{Adam}}^{(t)}) = -\eta_t \langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \rangle$, where $\widetilde{U}_{\mathrm{Adam}}^{(t)}$ denotes the local utility obtained by substituting the above linearization into $U_{\mathrm{Adam}}^{(t)}$. Therefore, the global Adam-aware In-Run Data Shapley value under the linearized utility is*

$$\phi_z(\widetilde{U}_{\mathrm{Adam}}) = -\sum_{t: z \in B_t} \eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \right\rangle .$$

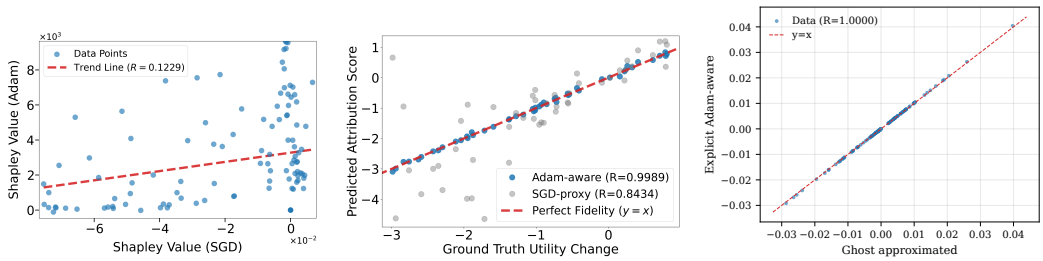
Theorem 4.2 shows that the Shapley value of the linearized Adam local utility recovers a ghost-compatible dot-product form. Compared with SGD-based In-Run Shapley, the validation direction is now modified by the Adam update Jacobian J_t. Equivalently, $\langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \rangle = \langle J_t^\top \nabla \ell(w_t, z^{(\mathrm{val})}), \nabla \ell(w_t, z) \rangle$. Thus, Adam-aware attribution replaces the raw validation gradient used by SGD-based In-Run Shapley with an optimizer-aware validation direction. This transformation is generally not a scalar rescaling, and therefore can change the ranking of training samples.

5 Experiments

In this section, we conduct a comprehensive empirical evaluation to demonstrate the effectiveness and efficiency of our Adam-aware In-Run Data Shapley from three aspects: (1) **Approximation Fidelity**, we test whether the proposed estimator faithfully matches exact local Shapley values under the fixed-state Adam utility. (2) **Computational Efficiency**, we evaluate the computational efficiency of the Linearized Ghost Approximation. (3) **Practical Effectiveness**, we study whether Adam-aware attribution is useful for downstream rank-based tasks, including semantic source identification and data pruning. Additional experimental results will be provided in the Appendix.

5.1 Fidelity Comparison

In this subsection, we evaluate whether the proposed Adam-aware approximation faithfully captures the data value induced by Adam updates. We separate two levels of evaluation. First, we compare



(a) (b) (c)

Figure 2: **Optimizer dependence and fidelity.** (a) Accumulated SGD-proxy and Adam-aware scores differ along the same Adam trajectory. (b) Adam-aware scores better match exact local Shapley values than the SGD proxy ($R = 0.9989$ vs. 0.8434). (c) Linearized Ghost closely matches materialized Direct scores on DistilBERT/SST-2 ($R = 0.999991$).

accumulated In-Run attribution scores computed along the same Adam trajectory using the SGD-based formula and the Adam-aware formula. Second, at a fixed training iteration, we compare both estimators against exact local Shapley values computed on the current mini-batch under the fixed-state Adam local utility. This distinction is important: the first protocol studies optimizer-induced differences in accumulated data values, while the second measures local approximation fidelity.

Experimental Setup. We compare our **Adam-aware approximation** with the **SGD-based In-Run proxy** (Wang et al., 2025). The main-text results use two complementary protocols: (i) comparing accumulated In-Run scores computed on the same Adam trajectory using the SGD-based and Adam-aware formulas; (ii) measuring local fidelity against exact Shapley values computed for the current mini-batch at a fixed training iteration. A local-fidelity sweep across learning rates is reported in Appendix I; a separate full-trajectory retraining-based comparison on Digits appears in Appendix F.

Optimizer Dependence in Accumulated Values. Figure 2a compares accumulated In-Run scores computed along the same Adam training trajectory using the SGD-based formula and the Adam-aware formula. The weak alignment indicates that replacing Adam's optimizer-induced update direction with the raw SGD gradient can substantially change the resulting data values. Thus, optimizer mismatch affects not only the numerical scale of attribution scores, but also their accumulated ranking across training.

Fidelity to Exact Local Shapley Values. Figure 2b compares both methods against exact local Shapley values computed on the current mini-batch under the fixed-state Adam local utility. The Adam-aware approximation achieves near-perfect fidelity ($R = 0.9992$), while the SGD-based proxy has lower correlation ($R = 0.8434$). This shows that the Adam-aware update direction more accurately tracks the optimizer-induced local utility.

Validity of Linearized Ghost. Finally, we evaluate whether the Linearized Ghost Approximation preserves the Adam-aware attribution signal. On 128 DistilBERT/SST-2 examples, Ghost and explicitly materialized Direct scores agree closely (Figure 2c): Pearson/Spearman $0.999991/0.999805$, normalized RMSE 0.0163, and bottom-10% overlap 1.0. Thus, Linearized Ghost provides a scalable implementation while preserving the fidelity of Adam-aware attribution.

5.2 Computational Efficiency via Linearized Ghost Approximation

A critical contribution of this work is the *Linearized Ghost Approximation*, which makes Adam-aware attribution feasible without materializing per-sample gradients. In this subsection, we evaluate whether this approximation reduces the computational bottleneck of direct Adam-aware attribution.

Experimental Setup. We benchmark runtime and memory on GPT-2 Small (124M parameters) trained with AdamW. All experiments are conducted on a single NVIDIA A100 (80GB) GPU, using batch size 16 and sequence length 1024. We compare three configurations: (i) **Standard AdamW**, training without attribution; (ii) **Adam-Ghost (Ours)**, our Adam-aware In-Run implementation with Linearized Ghost; and (iii) **Adam-Direct (Naive)**, which explicitly materializes per-sample gradients for Adam-aware attribution.

Original Wikipedia Corpus

The arsenal was briefly seized once more by Joseph Brooks loyalists during the Brooks-Baxter War of 1874.

Synthetic "Similar topic" Query

A historical dispute involved temporary takeover of a military installation by opposing forces.

Scenario	BM25	IF-Proxy	In-Run Data Shapley via SGD	In-Run Data Shapley via Adam
Exact	**1.00**	5.07	1.05	1.55
Partial	**1.00**	9.87	1.25	1.65
Paraphrase	**1.21**	245.41	9.48	6.51
Significant rewrite	**2.05**	216.23	454.60	32.10
Similar topic	2643.50	2855.13	1612.50	**753.25**

Table 2: **Semantic Source Identification. Top**: an original Wikipedia corpus and a synthetic "Similar topic" query. **Bottom**: average rank of the true source among 10 k candidates; lower is better. BM25 is a lexical baseline, and IF-Proxy is a post-training gradient-dot-product proxy. Adam-based In-Run Data Shapley improves over SGD-based In-Run Data Shapley on challenging semantic variants, especially significant rewrite and similar topic.

Results. As shown in Table 1, Adam-Ghost achieves 112.21 samples/sec, retaining 43.9% of the throughput of standard AdamW training. This shows that Adam-aware attribution introduces non-negligible overhead relative to ordinary training, mainly due to collecting attribution signals and applying the Adam-aware linearized weighting during training. However, Adam-Ghost is substantially more efficient than the naive Adam-Direct implementation, which achieves only 50.06 samples/sec. Thus, Linearized Ghost improves throughput by 2.24× over Adam-Direct by replacing explicit per-sample Adam-aware computations with ghost-style aggregation.

The memory results show the same trend. Adam-Ghost uses 3232.09 MB of peak memory, which is higher than standard training (1437.52 MB) but much lower than Adam-Direct (10540.37 MB). This corresponds to a 69.3% reduction in peak memory compared with Adam-Direct. Overall, these results show that Linearized Ghost does not eliminate all overhead relative to standard training, but it makes Adam-aware attribution substantially more practical than explicit per-sample computation.

Table 1: **Efficiency on GPT-2 Small** ($BS = 16$, $Seq = 1024$). Adam-Ghost is more efficient than Adam-Direct.

Method	SPS	Mem. (MB)
Standard	**255.62**	**1437.52**
Adam-Ghost	112.21	3232.09
Adam-Direct	50.06	10540.37

5.3 Practical Effectiveness

We evaluate the practical utility of our method on two rank-based data attribution tasks: semantic source identification and data pruning.

5.3.1 Semantic Source Identification

Experimental Setup. We first evaluate whether Adam-aware attribution can recover semantically related source examples from an Adam-trained language model. We train DistilGPT-2 (82M) (Sanh et al., 2019; Radford et al., 2019) on WikiText-2 (Merity et al., 2017). For efficiency, we use the first 10 k training lines (∼ 350 k tokens) for training and reserve the remaining data for validation and testing. We compare four methods: BM25, IF-Proxy, SGD-aware In-Run Data Shapley, and Adam-aware In-Run Data Shapley.

Semantic Perturbation Protocol and Evaluation Metric. For each trial, we randomly select a training example z^* as the source sample and construct validation queries with increasing semantic distance: (i) *Exact*, (ii) *Partial*, (iii) *Paraphrase*, (iv) *Significant rewrite*, and (v) *Similar topic*. The paraphrase and significant-rewrite variants are generated by a paraphrasing model and filtered using lexical-overlap constraints. Each generated query is treated as a validation input $z^{(\mathrm{val})}$. For each validation query $z^{(\mathrm{val})}$, we compute an attribution or retrieval score ϕ_i for every training sample z_i. We then rank all training samples and report the average rank of the true source sample z^* across multiple queries and random trials. A lower rank indicates better source identification.

Results and Analysis. Table 2 reports the average rank of the true source sample. BM25 performs well when lexical overlap is preserved, especially in the exact, partial, and paraphrase settings, which is expected because BM25 directly relies on surface-form matching. Thus, BM25 should be viewed as a lexical retrieval baseline rather than an optimizer-aware attribution method.

The main comparison is between SGD-aware and Adam-aware In-Run Data Shapley. Adam-aware attribution performs better on the more semantically challenging variants. For paraphrased queries, it improves the average source rank from 9.48 to 6.51. Under significant rewriting, the improvement is much larger, from 454.60 to 32.10. For similar-topic queries, Adam-aware attribution further improves the rank from 1612.50 to 753.25. These results suggest that Adam-aware In-Run Data Shapley captures optimizer-dependent training signals that are missed by the SGD-style proxy.

The IF-Proxy baseline provides a post-training gradient-dot-product signal, but it does not account for the full training trajectory or Adam's adaptive moment states. As a result, it performs substantially worse than Adam-aware In-Run Data Shapley on the semantic variants. Overall, these results provide rank-based evidence that attribution under Adam training benefits from explicitly modeling Adam's optimizer dynamics. Additional results are provided in Appendix D.

5.3.2 Data Pruning on SST-2

Experimental Setup. We evaluate whether Adam-aware attribution yields a more useful pruning ranking when optimizer differences are controlled. On SST-2, both Adam-aware and SGD-proxy scores are computed for 5,000 training examples along the same DistilBERT/AdamW trajectory. Each method removes its class-balanced bottom 20% and retrains on the remaining 4,000 examples. The selected subsets and the full-data baseline are retrained with identical AdamW settings and paired random seeds. To assess whether the result depends on a particular set of seeds, we use seeds 42–46 for development, fix the 20% pruning rule after examining those runs, and then evaluate it without modification on an independent set of seeds, 47–51. We report the development results, the locked replication, and their combined results separately.

Table 3: **Controlled 20% data pruning on SST-2 with DistilBERT.** Mean validation accuracy (%). Differences are Adam-aware minus SGD proxy in percentage points; intervals are paired bootstrap 95% CIs.

Phase	Full	Adam-aware	SGD proxy	Δ (pp)	95% CI (pp)
Development	86.972	86.972	86.147	+0.826	$[0.046, 1.537]$
Locked replication	86.468	86.995	85.803	+1.193	$[0.413, 1.972]$
Combined	86.720	86.984	85.975	+1.009	$[0.447, 1.560]$

Results and Analysis. Table 3 shows that Adam-aware pruning outperforms the SGD proxy in both seed groups. The advantage is 0.826 percentage points in development and persists after the pruning rule is fixed, reaching 1.193 points in the independent locked replication. Across all ten paired runs, Adam-aware pruning achieves 86.984% validation accuracy versus 85.975% for the SGD proxy, a 1.009-point improvement (95% CI $[0.447, 1.560]$; paired t-test $p = 0.0084$). Because the rankings are computed from the same AdamW trajectory and all subsets are retrained under the same optimizer settings, this comparison isolates the effect of the attribution-based ranking. Adam-aware pruning is also comparable to full-data retraining, although its combined advantage over full data is not established statistically (95% CI $[-0.115, 0.711]$). Thus, the pruning benefit over the SGD proxy is not confined to the seeds used to select the removal ratio.

6 Conclusion

In this work, we propose *Adam-Aware In-Run Data Shapley*, an optimizer-consistent extension of In-Run Data Shapley for Adam training. Our analysis and experiments show that data values are coupled with the realized optimization trajectory, and that SGD-based in-run attribution can be misleading for Adam-trained models. By defining a fixed-state Adam local utility and introducing a Linearized Ghost Approximation, our method accounts for Adam's momentum and variance normalization while remaining substantially more efficient than direct per-sample computation. Experiments demonstrate near-perfect fidelity to explicit Adam marginal utility changes and improved performance on rank-based data attribution tasks, including semantic source identification and data pruning.

AI USE STATEMENT

Generative AI tools assisted with manuscript editing and organization. The authors reviewed the resulting text and take responsibility for the paper's claims and content.

REFERENCES

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

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

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

Mark Alexander Burgess and Archie C Chapman. Approximating the shapley value using stratified empirical bernstein sampling. In *IJCAI*, pp. 73–81, 2021.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Ian Covert, Chanwoo Kim, Su-In Lee, James Y Zou, and Tatsunori B Hashimoto. Stochastic amortization: A unified approach to accelerate feature and data attribution. *Advances in Neural Information Processing Systems*, 37:4374–4423, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, 2019.

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

Lijie Hu, Chenyang Ren, Huanyi Xie, Khouloud Saadi, Shu Yang, Zhen Tan, Jingfeng Zhang, and Di Wang. Dissecting representation misalignment in contrastive learning via influence function, 2025. URL https://arxiv.org/abs/2411.11667.

Ferenc Illés and Péter Kerényi. Estimation of the shapley value by ergodic sampling. *arXiv preprint arXiv:1906.05224*, 2019.

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: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gürel, Bo Li, Ce Zhang, Costas J. Spanos, and Dawn Xiaodong Song. Efficient task-specific data valuation for nearest neighbor algorithms. *ArXiv*, abs/1908.08619, 2019a. URL https://api.semanticscholar.org/CorpusID:263794198.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1167–1176. PMLR, 2019b.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*, 2021.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.

Weida Li and Yaoliang Yu. Robust data valuation with weighted banzhaf values. *Advances in Neural Information Processing Systems*, 36:60349–60383, 2023.

Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pp. 13468–13504. PMLR, 2022.

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. *arXiv preprint arXiv:1609.07843*, 2017.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022.

Ramin Okhrati and Aldo Lipani. A multilinear sampling algorithm to estimate shapley values. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7992–7999. IEEE, 2021.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34: 20596–20607, 2021.

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, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Chenyang Ren, Yifan Jia, Huanyi Xie, Zhaobin Xu, Tianxing Wei, Liangyu Wang, Lijie Hu, and Di Wang. Attributing data for sharpness-aware minimization. *CoRR*, abs/2507.04059, 2025a.

Chenyang Ren, Huanyi Xie, Shu Yang, Meng Ding, Lijie Hu, and Di Wang. Evaluating data influence in meta learning. *CoRR*, abs/2501.15963, 2025b.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Lloyd S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the Theory of Games, Volume II*, pp. 307–317. Princeton University Press, Princeton, NJ, 1953.

Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 6388–6421. PMLR, 2023a.

Jiachen T Wang and Ruoxi Jia. A note on" towards efficient data valuation based on the shapley value". *arXiv preprint arXiv:2302.11431*, 2023b.

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

Zhaoxuan Wu, Yao Shu, and Bryan Kian Hsiang Low. Davinz: Data valuation using deep neural networks at initialization. In *International Conference on Machine Learning*, pp. 24150–24176. PMLR, 2022.

Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Validation free and replication robust volume-based data valuation. *Advances in Neural Information Processing Systems*, 34:10837–10848, 2021.

Xinyu Zhou, Simin Fan, and Martin Jaggi. Hyperinf: Unleashing the hyperpower of the schulz's method for data influence estimation. *arXiv preprint arXiv:2410.05090*, 2024.

APPENDIX AT A GLANCE

- **Adam-aware derivation and optimizer setup** (Appendices A and B): proofs and ghost-computation details, followed by the GPT-2 Small setup for comparing optimizer-induced data values.
- **Data pruning under optimizer-specific training** (Appendices C and C.1): DistilGPT2 validation-loss curves and 10,000-example SST-2 results at 20% and 30% removal, where each optimizer supplies its own scores and retraining procedure.
- **Semantic source identification** (Appendices D and E): the 1 k-candidate results and the protocol used to generate semantic variants.
- **Full-trajectory attribution on Digits** (Appendix F): retraining-based Adam-TMC fidelity, matched-permutation checks, and an Owen-estimator reference.
- **Controlled AdamW pruning on SST-2** (Appendix G): the shared-trajectory protocol and paired results at 5% and 20% removal.
- **Ghost fidelity and learning-rate checks** (Appendices H and I): Direct-versus-Ghost comparisons on DistilBERT and Pythia-410M, a local-fidelity sweep across learning rates, and a separate float32/float64 exact-coalition check on Digits.

A IN-RUN DATA SHAPLEY UNDER ADAM

A.1 PROOF OF LEMMA 4.1

Lemma A.1 (Restatement of Lemma 4.1)**.** *For any iteration $t = 0, \dots, T-1$, under the fixed-state Adam local counterfactual, the first-order approximation of the Adam local utility is*

$$U^{(t)}_{\mathrm{Adam}}(S; z^{(\mathrm{val})}) \approx -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon} \right\rangle.$$

Proof. By definition, the Adam local utility is

$$U^{(t)}_{\mathrm{Adam}}(S; z^{(\mathrm{val})}) = \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}) - \ell(w_t, z^{(\mathrm{val})}).$$

Applying a first-order Taylor expansion of the validation loss around w_t gives

$$\ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}) - \ell(w_t, z^{(\mathrm{val})}) = \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \widetilde{w}_{t+1}(S) - w_t \right\rangle + R_t(S),$$

where $R_t(S)$ contains the higher-order remainder terms. Under the fixed-state Adam local counterfactual, the previous moments are held fixed and the current coalition gradient induces

$$\widetilde{w}_{t+1}(S) = w_t - \eta_t \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon}.$$

Therefore,

$$\widetilde{w}_{t+1}(S) - w_t = -\eta_t \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon}.$$

Substituting this into the Taylor expansion and dropping the higher-order remainder yields

$$U^{(t)}_{\mathrm{Adam}}(S; z^{(\mathrm{val})}) \approx -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), \frac{m_t(S)}{\sqrt{v_t(S)} + \epsilon} \right\rangle.$$

This completes the proof. □

A.2 EFFICIENT COMPUTATION VIA LINEARIZED GHOST DOT-PRODUCT

In this subsection, we provide full details about the Ghost Dot Product for our In-Run Data Shapley for Adam.

A.2.1 PROOF OF THEOREM 4.2

Theorem A.2 (Restatement of Theorem 4.2)**.** *Let $\bar{g}_t$ be a fixed reference gradient and $J_t := \frac{\partial \Psi_t(g)}{\partial g}\Big|_{g=\bar{g}_t}$. Under the Linearized Ghost Approximation $\Psi_t(g_t(S)) \approx \Psi_t(\bar{g}_t) + J_t\big(g_t(S) - \bar{g}_t\big)$, let $\widetilde{U}^{(t)}_{\mathrm{Adam}}$ denote the resulting linearized Adam local utility. Then, for any $z \in B_t$, the local In-Run Shapley value satisfies $\phi_z\Big(\widetilde{U}^{(t)}_{\mathrm{Adam}}\Big) = -\eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \right\rangle$. Therefore, the global Adam-aware In-Run Data Shapley value under the linearized utility $\widetilde{U}_{\mathrm{Adam}} = \sum_{t=0}^{T-1} \widetilde{U}^{(t)}_{\mathrm{Adam}}$ is*

$$\phi_z(\widetilde{U}_{\mathrm{Adam}}) = -\sum_{t: z \in B_t} \eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \right\rangle.$$

Proof. Fix an iteration t and a validation point $z^{(\mathrm{val})}$. For any coalition $S \subseteq B_t$, define the coalition gradient

$$g_t(S) := \sum_{z \in S} \nabla \ell(w_t, z).$$

By Lemma 4.1, the first-order Adam local utility can be written as

$$U^{(t)}_{\mathrm{Adam}}(S) \approx -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top \Psi_t(g_t(S)),$$

where

$$\Psi_t(g) := \frac{\beta_1 m_{t-1} + (1-\beta_1) g}{\sqrt{\beta_2 v_{t-1} + (1-\beta_2) g^{\odot 2}} + \epsilon}.$$

Under the Linearized Ghost Approximation, we linearize Ψ_t around a fixed reference gradient $\bar{g}_t$:

$$\Psi_t(g) \approx \Psi_t(\bar{g}_t) + J_t(g - \bar{g}_t), \qquad J_t := \frac{\partial \Psi_t(g)}{\partial g}\Big|_{g=\bar{g}_t}.$$

Substituting this linearization into the Adam local utility defines the linearized local utility

$$\widetilde{U}^{(t)}_{\mathrm{Adam}}(S) := -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top \left[\Psi_t(\bar{g}_t) + J_t\big(g_t(S) - \bar{g}_t\big)\right].$$

Rearranging terms gives

$$\widetilde{U}^{(t)}_{\mathrm{Adam}}(S) = c_t - \eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top J_t g_t(S),$$

where

$$c_t := -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top \Psi_t(\bar{g}_t) + \eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top J_t \bar{g}_t$$

is independent of the coalition S.

Now consider any $z \in B_t$ and any coalition $S \subseteq B_t \setminus \{z\}$. Since

$$g_t(S \cup \{z\}) = g_t(S) + \nabla \ell(w_t, z),$$

we have

$$\begin{aligned}\widetilde{U}^{(t)}_{\mathrm{Adam}}(S \cup \{z\}) - \widetilde{U}^{(t)}_{\mathrm{Adam}}(S) &= -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top J_t \left(g_t(S \cup \{z\}) - g_t(S)\right) \\ &= -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top J_t \nabla \ell(w_t, z).\end{aligned}$$

The right-hand side is independent of the coalition S. Therefore, when substituted into the Shapley definition, every marginal contribution term is the same, and the Shapley weights sum to one. Hence,

$$\phi_z\Big(\widetilde{U}^{(t)}_{\mathrm{Adam}}\Big) = -\eta_t \nabla \ell(w_t, z^{(\mathrm{val})})^\top J_t \nabla \ell(w_t, z).$$

Finally, by the linearity of the Shapley value,

$$\phi_z(\widetilde{U}_{\mathrm{Adam}}) = \sum_{t=0}^{T-1} \phi_z\Big(\widetilde{U}^{(t)}_{\mathrm{Adam}}\Big).$$

Since $\phi_z(\widetilde{U}^{(t)}_{\mathrm{Adam}}) = 0$ whenever $z \notin B_t$, we obtain

$$\phi_z(\widetilde{U}_{\mathrm{Adam}}) = -\sum_{t: z \in B_t} \eta_t \left\langle \nabla \ell(w_t, z^{(\mathrm{val})}), J_t \nabla \ell(w_t, z) \right\rangle.$$

This completes the proof. □

A.2.2 COMPUTATION OF J_t

We provide the explicit form of the Jacobian used in the Linearized Ghost Approximation. Recall the Adam update map with respect to the coalition gradient g:

$$\Psi_t(g) := \frac{\beta_1 m_{t-1} + (1-\beta_1)g}{\sqrt{\beta_2 v_{t-1} + (1-\beta_2)g^{\odot 2}} + \epsilon},$$

where all operations are elementwise. Since Ψ_t is coordinatewise separable, its Jacobian with respect to g is diagonal.

For the j-th coordinate, define

$$a_j := \beta_1 (m_{t-1})_j, \qquad b := 1-\beta_1, \qquad c_j := \beta_2 (v_{t-1})_j, \qquad d := 1-\beta_2.$$

Then

$$\Psi_{t,j}(g_j) = \frac{a_j + b g_j}{\sqrt{c_j + d g_j^2} + \epsilon}.$$

Differentiating with respect to g_j gives

$$\Psi'_{t,j}(g_j) = \frac{b}{\sqrt{c_j + d g_j^2} + \epsilon} - \frac{(a_j + b g_j) d g_j}{\sqrt{c_j + d g_j^2}\left(\sqrt{c_j + d g_j^2} + \epsilon\right)^2}.$$

Therefore, for a fixed reference gradient $\bar{g}_t$, the Jacobian $J_t := \left.\frac{\partial \Psi_t(g)}{\partial g}\right|_{g=\bar{g}_t}$ is diagonal. Its j-th diagonal entry is

$$(J_t)_{jj} = \frac{1-\beta_1}{\sqrt{\beta_2 (v_{t-1})_j + (1-\beta_2)\bar{g}_{t,j}^2} + \epsilon} - \frac{\left(\beta_1 (m_{t-1})_j + (1-\beta_1)\bar{g}_{t,j}\right)(1-\beta_2)\bar{g}_{t,j}}{\sqrt{\beta_2 (v_{t-1})_j + (1-\beta_2)\bar{g}_{t,j}^2}\left(\sqrt{\beta_2 (v_{t-1})_j + (1-\beta_2)\bar{g}_{t,j}^2} + \epsilon\right)^2}.$$

Equivalently, define the reference moments

$$\bar{m}_t := \beta_1 m_{t-1} + (1-\beta_1)\bar{g}_t, \qquad \bar{v}_t := \beta_2 v_{t-1} + (1-\beta_2)\bar{g}_t^{\odot 2}.$$

Then

$$(J_t)_{jj} = \frac{1-\beta_1}{\sqrt{(\bar{v}_t)_j} + \epsilon} - \frac{(\bar{m}_t)_j (1-\beta_2)\bar{g}_{t,j}}{\sqrt{(\bar{v}_t)_j}\left(\sqrt{(\bar{v}_t)_j} + \epsilon\right)^2}.$$

In vector form,

$$J_t = \mathrm{Diag}\left(\frac{1-\beta_1}{\sqrt{\bar{v}_t} + \epsilon} - \frac{\bar{m}_t \odot (1-\beta_2)\bar{g}_t}{\sqrt{\bar{v}_t} \odot (\sqrt{\bar{v}_t} + \epsilon)^2}\right),$$

where all products, divisions, square roots, and powers are elementwise.

B GPT-2 SMALL OPTIMIZER-DEPENDENCE SETUP

Experimental Setup. We trained a GPT-2 Small model (124M parameters) using both SGD and AdamW optimizers. To estimate the TMC values, we sampled multiple random permutations of the training data and tracked the marginal contribution of 10,000 samples to the validation loss throughout the training process.

C DATA PRUNING ON DISTILGPT2 AND SST-2

Task. We evaluate the effect of data pruning on language model training dynamics. The task is causal language modeling, and performance is measured by validation loss during training.

Experimental Setup. We train `DistilGPT2` on a fixed training corpus and evaluate on a fixed validation set. All runs use the same model architecture, optimizer (AdamW), learning rate, batch

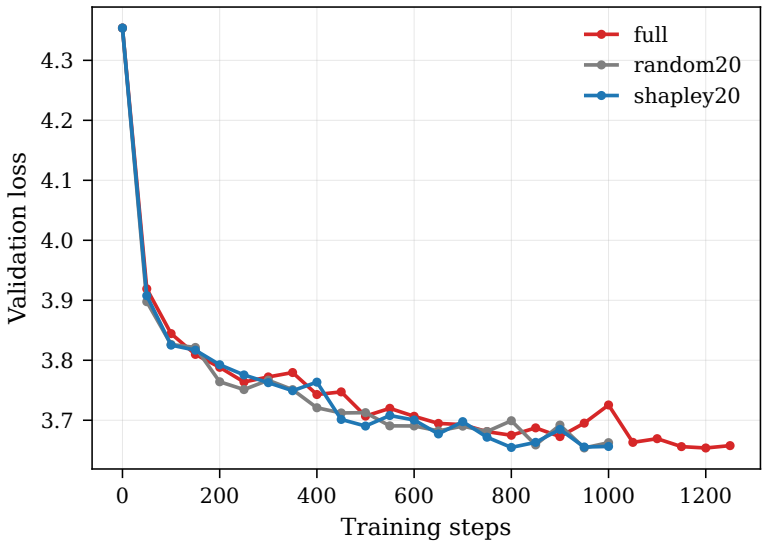


Figure 3: **Data pruning effect on language model training.** Validation loss as a function of training steps for three settings: training on the full dataset (`full`), random pruning of 20% training samples (`random20`), and In-Run Shapley pruning of 20% training samples (`shapley20`). Lower validation loss is better.

size, number of epochs, and evaluation frequency. Validation loss is logged periodically throughout training.

Three training settings are compared: (1) training on the full dataset (`full`), (2) training after randomly pruning 20% of the training samples (`random20`), and (3) training after pruning 20% of the training samples with the lowest In-Run Shapley scores (`shapley20`).

In-Run Shapley Pruning. In-Run Shapley scores are computed during a preliminary training pass using the Adam-aware ghost approximation. Samples with the lowest scores are removed prior to retraining. Random pruning removes the same number of samples uniformly at random, thereby serving as a budget-matched control that isolates the effect of which samples are removed from the effect of reducing the training set size.

Results. Figure 3 shows validation loss as a function of training steps. All three runs exhibit a similar overall convergence trend, indicating that removing 20% of the training samples does not destabilize training. Compared with the full-data baseline, the Shapley-pruned run achieves lower validation loss over much of the middle and later stages of training. Compared with random pruning, Shapley pruning remains competitive and often attains lower validation loss under the same pruning budget. These results suggest that Adam-aware In-Run Shapley identifies samples whose removal is at least as effective as random removal and can improve the training trajectory without increasing the pruning ratio.

C.1 Optimizer-Specific SST-2 Pruning

Experimental Setup. This experiment uses 10,000 SST-2 training examples and DistilBERT, with validation accuracy as the outcome. It differs from the controlled 5,000-example comparison in Section 5.3.2: here each optimizer supplies both its own scoring trajectory and its own retraining procedure. The AdamW arm ranks examples by Adam-aware In-Run Shapley and retrains with AdamW (learning rate 2×10^{-5}, batch size 16, five epochs); the momentum-SGD arm uses SGD-based In-Run scores and retrains with SGD (momentum 0.9, learning rate 3×10^{-4}, 6% linear warmup). Within each arm, we remove the bottom-ranked, top-ranked, or an equally sized random set of examples at 20% and 30% removal. The same data split and pruning ratios are used across conditions. For the AdamW arm, bottom and top pruning each average three runs; random pruning uses five random removals for each of three seeds (15 runs), and the full-data baseline averages three

runs. Because the scoring and retraining optimizers both change between arms, their cross-optimizer accuracy gap cannot isolate the attribution rule.

Table 4: **Optimizer-specific SST-2 pruning.** Mean validation accuracy after each arm scores and retrains with its own optimizer. AdamW and SGD accuracies are not a controlled comparison of attribution rules.

	AdamW			SGD		
Ratio	**Bottom**	**Random**	**Top**	**Bottom**	**Random**	**Top**
20%	**0.8826**	0.8752	0.8639	**0.8228**	0.7706	0.7649
30%	**0.8876**	0.8739	0.8532	**0.7117**	0.6760	0.6383

Table 5: **Extended AdamW pruning statistics on SST-2.** We report validation accuracy as mean $\pm$ standard deviation. The full-data baseline is included for reference.

Setting	**Prune Ratio**	**Validation Accuracy**	**# Runs**
Full data	0%	0.8719 ± 0.0058	3
Bottom pruning	20%	$\mathbf{0.8826 \pm 0.0074}$	3
Random pruning	20%	0.8752 ± 0.0075	15
Top pruning	20%	0.8639 ± 0.0046	3
Bottom pruning	30%	$\mathbf{0.8876 \pm 0.0061}$	3
Random pruning	30%	0.8739 ± 0.0069	15
Top pruning	30%	0.8532 ± 0.0140	3

Results. Table 4 summarizes both optimizer-specific arms; Table 5 reports the AdamW arm's run-level variability and baseline. The full-data AdamW baseline achieves 0.8719 ± 0.0058 validation accuracy. Across all pruning ratios, bottom pruning consistently outperforms both random pruning and top pruning. At pruning ratios of 20% and 30%, bottom pruning achieves 0.8826 and 0.8876 validation accuracy, respectively, all exceeding the full-data baseline. This suggests that the lowest-scored samples identified by Adam-aware In-Run Shapley are not merely redundant, but can actively reduce generalization performance when retained.

In contrast, top pruning consistently hurts performance, especially at higher pruning ratios. At 30% pruning, top pruning drops to 0.8532 ± 0.0140, substantially below both bottom pruning and random pruning. This supports the interpretation that high-score samples are beneficial for training and should not be removed.

D Semantic Source Identification with 1 k Candidates

Small Candidate-Pool Setting. In addition to the main semantic source identification experiment with $10\,\mathrm{k}$ candidate training examples, we also evaluate a smaller candidate-pool setting with $1\,\mathrm{k}$ candidates. This experiment serves as a sanity check for the ranking protocol under a lower retrieval difficulty. As in the main experiment, lower rank indicates better identification of the true source example.

Discussion. Table 6 shows that the $1\,\mathrm{k}$ candidate-pool setting is substantially easier than the main $10\,\mathrm{k}$ setting. For exact, partial, paraphrase, and significant-rewrite variants, BM25 often ranks the true source at or near the top because lexical overlap remains highly informative in the smaller candidate pool. This confirms that BM25 should be interpreted as a lexical retrieval baseline rather than as an optimizer-aware attribution method.

The optimizer-aware comparison is more informative in the semantically challenging variants. Under significant rewriting, Adam-InRun improves the average rank over SGD-InRun from 13.75 to 4.75. Under similar-topic queries, where surface-form matching becomes much less reliable, Adam-InRun

Table 6: **Semantic source identification with $1\,\mathrm{k}$ candidate training examples.** We report the average rank of the true source sample z^*. Lower is better. BM25 is a lexical retrieval baseline, IF-Proxy is a post-training gradient-dot-product influence proxy, and SGD-InRun / Adam-InRun compare optimizer-aware in-run attribution.

Scenario	BM25	IF-Proxy	SGD-InRun	Adam-InRun
Exact	**1.00**	**1.00**	**1.00**	**1.00**
Partial	**1.00**	4.00	2.00	1.50
Paraphrase	**1.00**	8.00	1.75	4.50
Significant rewrite	**1.00**	56.25	13.75	4.75
Similar topic	618.00	658.75	302.00	**81.50**

improves the average rank from 302.00 to 81.50. These results are consistent with the main $10\,\mathrm{k}$ experiment and provide additional evidence that Adam-aware in-run attribution better captures optimizer-dependent semantic influence than the SGD-style proxy.

E Semantic Variant Generation via Prompt Engineering

To probe robustness under controlled semantic perturbations, we generate structured validation variants conditioned on a source training sample z^*. Each z^* is associated with four categories of validation texts: (i) `partial_same`, (ii) `paraphrase`, (iii) `significant_paraphrase`, and (iv) `similar_topic`. These categories are designed to progressively decouple surface form, entity usage, and semantic content, enabling fine-grained evaluation of attribution stability.

Structured Prompt-Based Generation. We employ a large language model (DeepSeek-V3) as a controllable text generator. To ensure structural consistency and downstream reproducibility, all generations are constrained to a `json_object` response format. The prompt consists of a fixed system instruction and a parameterized user template, where the source sample z^* is explicitly injected as a variable.

System prompt.

> You are generating controlled text variants for a machine learning experiment. Return ONLY valid JSON. Do NOT include markdown, explanations, or extra text.

User prompt template.

> Given the following text z^*, generate text variants in four categories: `partial_same`, `paraphrase`, `significant_paraphrase`, and `similar_topic`. Each output must be a single English sentence and satisfy category-specific constraints. The model must return a JSON object with exactly these four keys.

The full *verbatim* user prompt, including category definitions, forbidden-token injection, and formatting constraints, is provided in Appendix E.2 for exact reproducibility.

E.1 Deterministic Filtering and Quality Control

Raw generations from the language model are further processed through a deterministic multi-stage filtering pipeline to remove trivial, degenerate, or overly lexical variants. This ensures that robustness evaluation reflects genuine semantic perturbations rather than surface-level artifacts.

Lexical Constraint Enforcement. We extract a set of forbidden tokens $\mathcal{T}_{\text{forbid}}$ from z^*, including named entities, alphanumeric identifiers, and long high-entropy content words. Candidates in the `significant_paraphrase` and `similar_topic` categories must satisfy

$$\mathcal{W}(C) \cap \mathcal{T}_{\text{forbid}} = \emptyset,$$

where $\mathcal{W}(C)$ denotes the word set of candidate C. This constraint prevents entity memorization and enforces semantic abstraction.

Jaccard Similarity Thresholding. We measure lexical overlap using word-level Jaccard similarity $J(C, z^*)$. Each category enforces a distinct overlap regime:

- `Paraphrase`: $\tau_{p,\min} \leq J(C, z^*) \leq \tau_{p,\max}$, ensuring semantic equivalence with moderate surface variation.
- `Significant paraphrase`: $J(C, z^*) \leq \tau_{s,\max}$, enforcing substantial rewriting while preserving meaning.
- `Similar topic`: an even stricter upper bound on $J(C, z^*)$, encouraging topical relatedness without semantic identity.

Final Selection. After filtering, we deduplicate candidates and select a fixed number (K_{each}) per category. All randomness is controlled by a fixed global seed, and generated scenarios are cached to disk to guarantee deterministic reuse across runs.

E.2 Verbatim Prompt for Semantic Variant Generation

System prompt:

```
You are generating controlled text variants for a machine
learning experiment.  Return ONLY valid JSON. Do NOT include
markdown, explanations, or extra text.
```

User prompt:

```
Given the following text z_star, generate text variants in four categories.

Return a JSON object with EXACTLY the following keys:
- "partial_same"
- "paraphrase"
- "significant_paraphrase"
- "similar_topic"

Each value must be a list of strings.

Constraints:
- Each string must be ONE English sentence.
- Do not copy z_star verbatim except for "partial_same".
- "partial_same": truncated or lightly edited version of z_star (1 item).
- "paraphrase": same meaning, mild rewriting (at least K candidates).
- "significant_paraphrase": same meaning, substantial rewriting,
avoid reusing entities.
- "similar_topic": same general topic, different facts/details,
minimal lexical overlap.

Forbidden tokens:
[FORBIDDEN_TOKEN_LIST]

z_star:
[SOURCE TEXT]
```

F Full-Trajectory Adam-TMC Fidelity on Digits

Retraining-based Adam-TMC protocol. The full-trajectory fidelity experiment uses a two-layer MLP with hidden width 32 on sklearn Digits, with 20 class-balanced target training examples, 200 disjoint warmup examples, and 300 validation examples. After 100 warmup updates, every retraining run starts from the same model parameters and Adam moments. For each of 100 random permutations, evaluated coalition prefixes are optimized for 30 Adam steps, with coalition losses normalized by the fixed full-set denominator. Utility is validation-loss reduction. Both In-Run estimators accumulate scores over the corresponding 30-step Adam trajectory. Relative to the pooled Adam-TMC reference, Adam-aware obtains Pearson/Spearman/Kendall correlations of $0.9609/0.9669/0.8737$, versus $0.8147/0.8361/0.6737$ for the SGD proxy. The paired Pearson advantage is 0.1436 (95%

CI $[0.1104, 0.1765]$). Their top-20% overlaps are both 1.0, while bottom-20% overlap is 1.0 for Adam-aware and 0.75 for the SGD proxy.

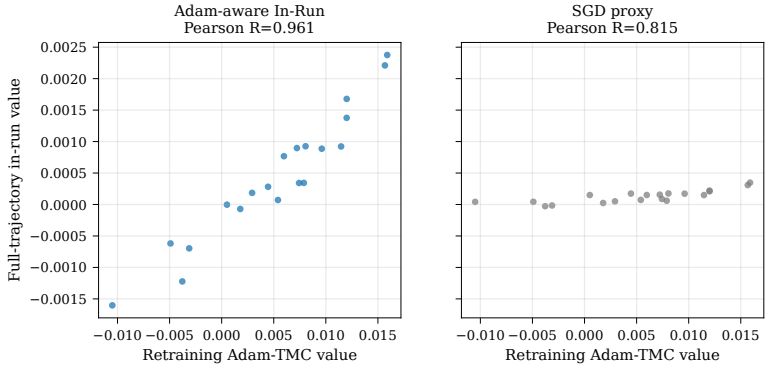
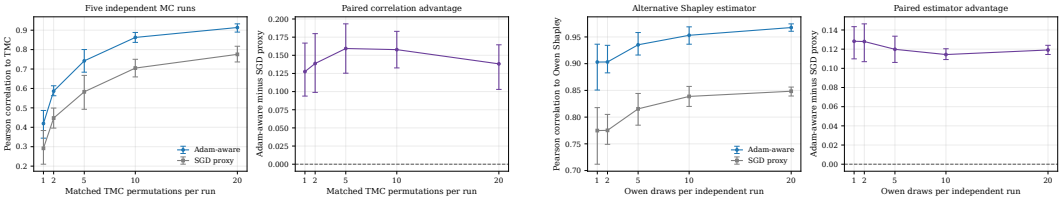


Figure 4: **Full-trajectory retraining-based Adam-TMC comparison.** Adam-aware and same-trajectory SGD-proxy In-Run scores are evaluated against a common pooled Adam-TMC reference.

Matched TMC and Owen-estimator robustness. We partition the same 100 TMC permutations into five disjoint 20-permutation runs and evaluate both estimators against identical reference permutations at checkpoints 1, 2, 5, 10, and 20. At 20 permutations, Adam-aware obtains mean Pearson correlation 0.9137 versus 0.7755 for the SGD proxy; the paired advantage is 0.1382 with 95% CI $[0.1030, 0.1645]$, and Adam-aware is better in all five runs. A separate Owen multilinear-extension estimator samples stratified Bernoulli coalitions instead of permutation prefixes. Every coalition and one-bit toggle is retrained from the same warmed Adam state. At 20 draws per run, Adam-aware obtains 0.9676 versus 0.8484, with a paired advantage of 0.1191 (95% CI $[0.1144, 0.1240]$), again higher in all five runs. The pooled Owen and TMC references agree with Pearson/Spearman $0.9806/0.9805$.



(a) Matched-permutation TMC convergence.

(b) Owen multilinear-extension robustness.

Figure 5: **Robustness to reference estimation.** Both methods are evaluated against the same reference samples within each run and checkpoint.

G Controlled AdamW Pruning on SST-2

Protocol and pruning-ratio comparison. We score 5,000 SST-2 training examples with DistilBERT along one shared AdamW trajectory and a validation target comprising all 872 development examples. Adam-aware and SGD-proxy rankings share checkpoints, mini-batches, validation gradients, and learning-rate schedule. At each removal ratio, equal-sized class-balanced bottom-ranked subsets are retrained for three epochs with identical AdamW configurations. Development seeds 42–46 and locked replication seeds 47–51 are disjoint. At 20% removal, development favors Adam-aware by 0.826 percentage points and locked replication by 1.193 points. Across all ten paired seeds, the accuracies are 86.984% versus 85.975%, a 1.009-point difference (95% CI $[0.447, 1.560]$, paired t-test $p = 0.0084$). Table 7 shows the effect is concentrated at 20% removal.

Table 7: **Controlled AdamW pruning at 5% and 20% removal.** Both methods use one scoring trajectory and identical AdamW retraining. Differences are Adam-aware minus SGD-proxy percentage points over ten paired seeds.

Removal	Retained	Adam-aware	SGD proxy	Difference	95% CI
5%	4,750	86.823	86.548	0.275	$[-0.677, 1.227]$
20%	4,000	86.984	85.975	1.009	$[0.447, 1.560]$

H Ghost Score Fidelity on DistilBERT and Pythia-410M

Direct-score comparison and computation. The main-text DistilBERT comparison uses 128 examples; Ghost versus explicit Direct scores has Pearson/Spearman $0.999991/0.999805$, MAE 1.08×10^{-4}, RMSE 1.58×10^{-4}, and normalized RMSE 0.0163. Top-10% overlap is 0.923 and bottom-10% overlap is 1.0. On Pythia-410M, we independently sample eight WikiText-2 attribution examples for each of three seeds. The full 405.3M-parameter model participates in every forward pass, while the final transformer block, final normalization, and output head (64.1M parameters) are trainable for the explicit reference. Pooled Ghost-versus-Direct Pearson/Spearman is $0.999998/1.000000$; top- and bottom-quartile overlaps are 1.0 for every seed. Mean Direct-to-Ghost speedup is $2.15\times$ (95% CI $[2.07, 2.23]$). This supports score fidelity and computation on a larger forward model, not large-model subset retraining.

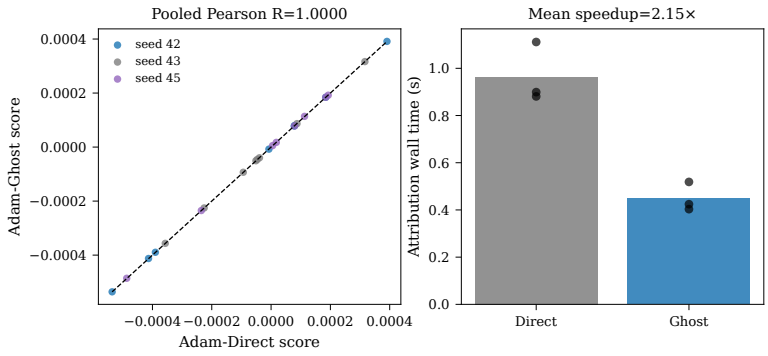


Figure 6: **Pythia-410M Ghost fidelity and speed.** The full model participates in forward computation; explicit gradients cover the trainable final block, normalization, and head.

I Learning-Rate Sensitivity and Numerical Resolution

Local Shapley fidelity across learning rates. For the fixed-state Adam local utility, we compare Adam-aware and SGD-proxy scores with reference local utility changes across $\eta \in [10^{-7}, 10^{-3}]$. Figure 7 shows higher Pearson correlation for the Adam-aware estimator throughout the sweep, with both estimators declining at the smallest learning rate.

Separate float32 and float64 exact-coalition diagnostic on Digits. We enumerate all 256 coalitions of an eight-player Digits game from the same warmed checkpoint and Adam state in float32 and float64. At learning rates 10^{-8} and 10^{-7}, float32 produces only two distinct coalition utilities, with Adam-aware Pearson correlations 0.1784 and 0.4800, respectively. In float64, all 256 utilities remain distinguishable and Adam-aware correlation is 0.9745–0.9746 throughout $[10^{-8}, 10^{-3}]$; the SGD proxy stays at 0.8908–0.8927. The downturn in the float32 reference is consistent with a subtraction-resolution floor rather than growing first-order error as the learning rate decreases. This separate coalition game diagnoses a possible numerical mechanism; it does not directly remeasure the preceding local-fidelity sweep.

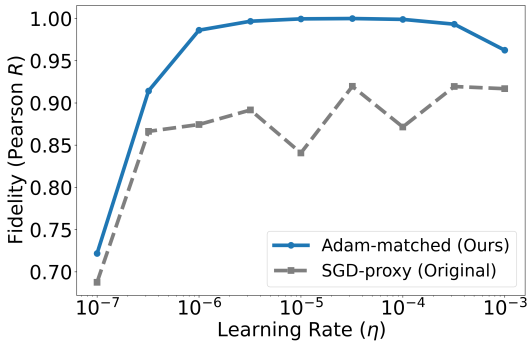


Figure 7: **Local attribution fidelity across learning rates.** Pearson correlation between predicted scores and reference Adam local utility changes.

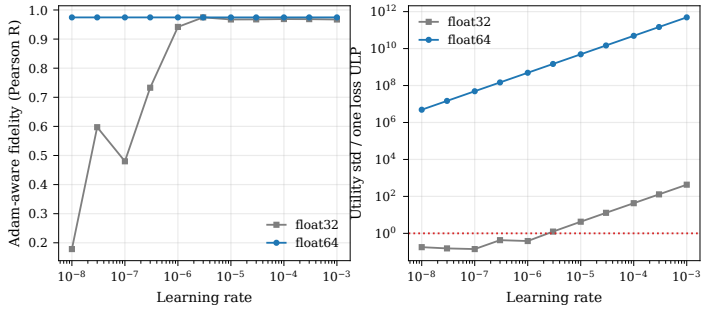


Figure 8: **Learning-rate numerical-resolution diagnostic.** Float64 preserves distinguishable exact-coalition utilities at small learning rates; float32 collapses many coalitions to the same loss change.

J Limitations and Broader Impacts

Limitations. Our method targets the step-wise In-Run Data Shapley utility rather than full unrolled final-checkpoint influence. It relies on first-order approximations to the query loss and the Adam update map around a reference gradient.

Broader impacts. Optimizer-aware attribution can support data debugging, source tracing, and data curation, which may help reduce harmful or low-quality training data. However, attribution scores can be noisy or misinterpreted, and using them for automated data removal may unintentionally discard rare or underrepresented examples. In high-stakes settings, such scores should be used as diagnostic evidence rather than as the sole basis for decisions.