IN-RUN DATA SHAPLEY FOR ADAM OPTIMIZER

Meng Ding[1,2]*& **Zeqing Zhang**[1]*& **Di Wang**[3] & **Lijie Hu**[1]†

[1]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
[2]University at Buffalo
[3]King Abdullah University of Science and Technology (KAUST)

ABSTRACT

Reliable data attribution is essential for mitigating bias and reducing computational waste in modern machine learning, with the Shapley value serving as the theoretical gold standard. While recent "In-Run" methods bypass the prohibitive cost of retraining by estimating contributions dynamically, they heavily rely on the linear structure of Stochastic Gradient Descent (SGD) and fail to capture the complex dynamics of adaptive optimizers like Adam. In this work, we demonstrate that data attribution is inherently optimizer-dependent: we show that SGD-based proxies diverge significantly from true contributions under Adam (Pearson $R \approx 0.11$), rendering them ineffective for modern training pipelines. To bridge this gap, we propose Adam-Aware In-Run Data Shapley. We derive a closed-form approximation that restores additivity by redefining utility under a fixed-state assumption and enable scalable computation via a novel Linearized Ghost Approximation. This technique linearizes the variance-dependent scaling term, allowing us to compute pairwise gradient dot-products without materializing per-sample gradients. Extensive experiments show that our method achieves near-perfect fidelity to ground-truth marginal contributions ($R > 0.99$) while retaining $\sim$95% of standard training throughput. Furthermore, our Adam-aware attribution significantly outperforms SGD-based baselines in data attribution downstream tasks.

1 INTRODUCTION

Modern machine learning models rely heavily on large-scale training data, yet model performance and behavior are typically governed by the quality and influence of individual samples. 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).

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. (2019a). While it has desirable properties, such as fairness, efficiency, and additivity (Jia et al., 2019b), the classic calculation of Shapley values is computationally prohibitive for deep learning. Formally, it requires retraining the model exponentially many times using different subsets of data to evaluate the marginal contribution of each source. This complexity has limited the exact application of Shapley values to only the smallest datasets and models, necessitating approximation methods for practical use.

To overcome this scalability bottleneck, recent work has introduced *In-Run Data Shapley* Wang et al. (2025). This method fundamentally shifts the attribution paradigm by estimating Shapley values

*Equal contribution.
†Corresponding to lijie.hu@mbzuai.ac.ae.

dynamically during a single training process, rather than through retrospective retraining. However, despite its efficiency, *the existing In-Run Data Shapley is specifically tailored for Stochastic Gradient Descent (SGD)* Robbins & Monro (1951), and not directly extendable to other popular optimizers like Adam Kingma & Ba (2014). However, in practice, modern deep learning models are almost exclusively trained using adaptive optimizers, most notably Adam. This discrepancy motivates us to ask the following questions:

Question 1: Are Data Shapley values inherently dependent on the choice of optimization algorithm?

Standard In-Run Data Shapley relies on a Taylor expansion of the utility function that implicitly assumes parameter updates are linear combinations of data gradients (as in SGD). However, Adam updates are scaled by historical moments and adaptive variance terms. It remains unknown whether SGD-based approximations can serve as valid proxies for Adam-trained models, or if the choice of optimizer fundamentally redefines the value of a data point.

Question 2: Can we extend the current In-Run framework to the Adam optimizer?

Adapting In-Run Shapley to Adam presents several technical challenges. First, Adam is stateful: updates depend on history-dependent moments rather than just the current gradient. Second, the non-linear scaling of gradients by the variance term breaks the linearity required for the efficient "Ghost Dot-Product" computation used in prior work.

In this paper, we answer both questions affirmatively. Our contributions are summarized as follows:

- **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. We reveal that widely used SGD-based proxies exhibit an extremely low correlation ($R \approx 0.11$) with true marginal contributions under Adam, proving that applying SGD attribution to adaptive settings yields misleading results.
- **Adam-Aware In-Run Data Shapley.** We derive the first closed-form estimator for In-Run Data Shapley tailored to Adam. By redefining per-iteration utility with a fixed-state assumption and applying a Taylor expansion to the adaptive variance term, we derive a tractable formula that explicitly accounts for momentum and variance scaling.
- **Scalable "Linearized Ghost" Computation.** To overcome the non-linearity of Adam updates, which prevents standard efficient aggregation, we introduce the Linearized Ghost Approximation. This technique approximates the Adam update as a linear combination of the current gradient and historical moments, allowing us to compute all pairwise gradient dot-products in a single backpropagation pass. This reduces memory overhead to negligible levels (identical peak memory to standard training) and maintains high throughput (87.85 samples/sec vs. 25.58 for naive implementations).
- **Practical Impact and Fidelity.** We demonstrate that our method achieves near-perfect fidelity ($R > 0.99$) to ground-truth utility changes, significantly outperforming SGD proxies ($R \approx 0.74$). Furthermore, we show that our Adam-aware attribution significantly outperforms SGD-based baselines in data attribution downstream tasks: it consistently yields higher validation accuracy in data pruning on SST-2 and exhibits superior robustness in semantic source identification compared to SGD-based methods.

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) applies first-order Taylor expansions and inverse Hessian–vector products to estimate the impact of training samples. These methods aid model interpretation and have been extended to study large language models (Grosse et al., 2023). However, influence functions can be unstable in non-convex settings (Basu et al., 2021). To address this, Trajectory-based methods like TracIn (Pruthi et al., 2020) approximate influence by accumulating gradient dot products along training, avoiding second-order derivatives. Recent work

improved robustness and reduced training costs further. For example, Datamodels (Ilyas et al., 2022) learn mappings from training subsets to model outputs at increased computational cost. DAVINZ (Wu et al., 2022) estimates data value at initialization, while others remove validation set dependence (Xu et al., 2021).

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., 2019a; 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 will provide a formal problem setup for data attribution and introduce the Data Shapley value.

Problem Setup. Consider a training dataset $D = \{z_i\}_{i=1}^N$ consisting of N individual data points. In the context of data attribution, our goal is to quantify the contribution of each individual data point $z \in D$ toward the performance of a machine learning model. To measure this performance, let $U(\cdot)$ denote a utility function (e.g., validation accuracy or negative loss) that takes a subset of data $S \subseteq D$ as input and returns a real-valued score representing the utility of a model trained on that subset.

Adapting the concept from game theory Shapley (1953), we treat the data points as players in a game and the model performance as the total payout. The Data Shapley value, denoted as $\phi_z(U)$, provides a unique and fair way to distribute the total utility among the N data points.

Formally, the Shapley value assigned to a data point $z \in D$ is defined as the weighted average of its marginal contribution to all possible subsets of the dataset:

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

where $D_{-z} = D \setminus \{z\}$ represents the dataset with point z excluded. S represents a subset of D_{-z} with cardinality $|S| = k-1$.The term $[U(S \cup \{z\}) - U(S)]$ represents 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 (More discussions refer to the Appendix). Here, we introduce the Linearity, which will be used in subsequent analysis.

Lemma 3.1 (Linearity of the Shapley value Shapley (1953))**.** *For any of two utility functions U_1, U_2 and any $\alpha_1, \alpha_2 \in \mathbb{R}$, we have $\phi_z(\alpha_1 U_1 + \alpha_2 U_2) = \alpha_1 \phi_z(U_1) + \alpha_2 \phi_z(U_2)$.*

In-Run Data Shapley. To avoid retraining, Wang et al. (2025) proposed *In-Run Data Shapley*, which redefines data utility in terms of a single training trajectory. Rather than computing marginal gains

over all subsets, it accumulates per-step contributions during a single run: $\phi_i = \sum_t \Delta_t^{(i)}$, where $\Delta_t^{(i)} := U(w_t) - U(w_{t-1})$ and the utility $U(w)$ is typically defined as the negative validation loss.

Adam Optimizer and State Coupling. 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:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{1}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{2}$$

$$w_t = w_{t-1} - \eta_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}, \tag{3}$$

where $g_t = \nabla_w \ell(w_{t-1}; z_t)$ is the gradient of the training loss on data point z_t, and η_t is the learning rate. Unlike SGD, Adam introduces a form of historical memory: the parameter update depends not only on the current gradient but also on the full gradient history via m_t and v_t.

4 In-Run Data Shapley for Adam Optimizer

In this section, we first demonstrate that In-Run Data Shapley Wang et al. (2025) is inherently dependent on the choice of optimization algorithm. Motivated by this discrepancy, we then propose an optimizer-aware extension tailored specifically 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.

To ensure our findings are robust and reflect true marginal contributions rather than artifacts of a specific approximation method, we relied on Truncated Monte Carlo (TMC) Ghorbani & Zou (2019). TMC estimates the Shapley value by retraining the model on random subsets of data, accumulating the marginal validation loss changes induced by individual samples along the optimization trajectory. Although computationally expensive, this retraining-based approach serves as a model-agnostic "ground truth" baseline, allowing us to rigorously compare how the optimization path itself alters data value.

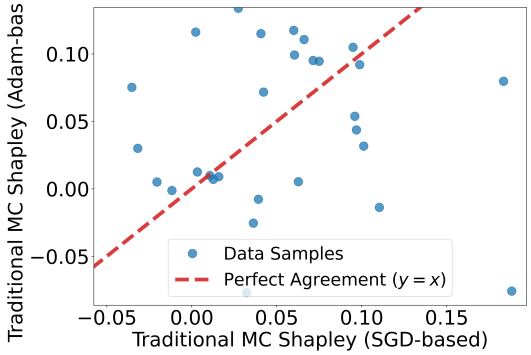


Figure 1: Comparison between SGD-based and Adam-based data Shapley values (Pearson $R = 0.0579$, Spearman $\rho = 0.0465$). Each point corresponds to a training sample. The dashed line indicates perfect agreement ($y = x$), while the dispersion reveals strong optimizer dependence.

As illustrated in Figure 1, attribution scores derived under the SGD trajectory diverge significantly from those computed under Adam. The scatter plot reveals a distinct lack of linear agreement, with the majority of samples deviating substantially from the identity line ($y = x$). Quantitatively, the Pearson correlation coefficient between SGD-based and Adam-based values is extremely low ($R \approx 0.0579$). 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 be aggressively down-weighted by Adam's

adaptive variance scaling. This confirms that relying on SGD-based logic for adaptive optimizers is insufficient, validating the necessity of the optimizer-aware framework.

In Section 5, we extend this analysis by directly comparing our proposed Adam-aware In-Run Shapley against the Standard In-Run Shapley (SGD-proxy). These experiments will demonstrate how optimizer-aware attribution significantly improves fidelity to ground-truth marginal contributions.

4.2 In-Run Data Shapley for Adam

Standard In-Run Data Shapley Wang et al. (2025) assesses the value of a training set by decomposing its total utility into contributions accumulated across iterations. Instead of retraining, it evaluates a local utility function at each step. Formally, given a validation data point $z^{(\text{val})}$ and a sequence of model checkpoints $\{w_t\}_{t=0}^{T}$, the local utility function at a single iteration t can be defined as:

$$U^{(t)}\left(S; z^{(\text{val})}\right) := \ell\left(\widetilde{w}_{t+1}(S), z^{(\text{val})}\right) - \ell\left(w_t, z^{(\text{val})}\right),$$

where $\widetilde{w}_{t+1}(S) := w_t - \eta_t \sum_{z\in S} \nabla \ell\left(w_t, z\right)$ represents the model parameters updated by SGD using the subset $S \in B_t$, where $B_t \in D$ is the batch selected at iteration t. Naturally, the global utility for the entire training process is simply the sum of these local utilities: $U(S) = \sum_{t=0}^{T-1} U^{(t)}(S, z^{(\text{val})})$. Relying on the linearity property of the Shapley value (Lemma 3.1), the global In-Run Data Shapley value can be computed by summing the Shapley values of each local step: $\phi_z(U) = \sum_{t=0}^{T-1} \phi_z\left(U^{(t)}\left(S; z^{(\text{val})}\right)\right)$. For clarity of presentation, we denote $U^{(t)}\left(S; z^{(\text{val})}\right)$ by $U^{(t)}$, omitting its explicit dependence on the context.

Extending this to Adam is non-trivial due to the stateful and non-linear nature of its update rule, as noted by Wang et al. (2025). By applying a first-order Taylor expansion to the local utility function under Adam dynamics, we can derive a tractable, closed-form estimator that restores the additive property required for Shapley valuation:

Theorem 4.1. *In-Run Data Shapley via Adam, considering the first-order approximation, has a closed-form*

$$\phi_z(U) \approx \sum_{t=0}^{T-1} \phi_z\left(U_{(1)}^{(t)}\right), \tag{4}$$

where

$$\phi_z\left(U_{(1)}^{(t)}\right) = -\eta_t \nabla \ell\left(w_t, z^{(\text{val})}\right) \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}, t = 0, \ldots, T-1.$$

In-Run Data Shapley decomposes the overall utility into step-wise contributions, a formulation that also benefits the Adam optimizer, as illustrated in Theorem 4.1. However, compared to Wang et al. (2025), our result differs in that the Data Shapley value of a training point accumulates dot products between its update directions and the gradients of the validation data, rather than gradient–gradient dot products as in Wang et al. (2025). This distinction precludes the use of the efficient gradient dot-product computation techniques employed in prior work, a challenge we address in the subsequent section.

4.3 Efficient Computation via Ghost Dot-Product

Standard In-Run Data Shapley Wang et al. (2025) achieves efficient pairwise gradient dot products between each training data and the validation point by *Ghost Dot-Product*. Specifically, directly computing the inner product $\nabla \ell(w_t, z_i)^\top \nabla \ell(w_t, z^{(\text{val})})$ would require explicitly instantiating per-sample gradients for the entire parameter space, incurring a prohibitive memory cost of $O(B \times P)$, where B is the batch size and P is the number of model parameters.

To circumvent this, the Ghost Dot-Product exploits the structural decomposition of gradients in neural networks. 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 is given by the outer product $g_i^{(l)} = \delta_i^{(l)} (a_i^{(l)})^\top$. Instead of materializing $g_i^{(l)}$, it computes the dot product layer-by-layer using the matrix trace property:

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

Original Wikipedia Corpus

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

Synthetic "Similar topic" corpus

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

Scenario	BM25	Influence	In-Run Data Shapley via SGD	In-Run Data Shapley via Adam
Partial exactly the same	**1.0**	140.8	166.4	3.4
Paraphrase	**1.4**	151.8	150.0	6.6
Significant paraphrase	169.8	152.6	192.6	**96.6**
Similar topic	328.4	150.4	233.0	**69.4**

Table 1: **Semantic Source Identification. Top:** (left) An original training corpus from Wikipedia. (right) A synthetic corpus falls in the category of "Similar topic" to the Wikipedia corpus. **Bottom:** The (average) value rank of the original corpus among all training corpora. The rank is out of $\approx$ 1k corpora. *Note: Adam-based In-Run Data Shapley significantly outperforms SGD-based In-Run Data Shapley in semantic matching (lower rank is better).*

However, as shown earlier, Theorem 4.1 establishes that In-Run Data Shapley under Adam depends on inner products between the update rule and the gradients of the validation data, rather than gradient–gradient dot products. This structural difference prevents the direct application of standard ghost dot-product techniques that rely on chain-rule decompositions of linear inner products. This non-linear induced by Adam's adaptive variance term breaks the linear structure required for efficient backpropagation-based aggregation.

To overcome this limitation, we propose a **Linearized Ghost Approximation**. The key idea is to linearize the non-linear Adam update by performing a first-order Taylor expansion of the variance-dependent scaling term. This approximation allows the Adam update to be expressed as a linear combination of the current gradient and historical moments, thereby restoring compatibility with efficient ghost computation.

Formally, consider the denominator of the Adam update $f(x) = \frac{1}{\sqrt{x}+\epsilon}$. We linearize this term around the variance estimate from the previous step, v_{t-1}. Let $A_t(z) = \sqrt{v_{t-1}(z)} + \epsilon$ denote the preconditioning term based on the fixed history. Applying a first-order Taylor expansion to the current variance $v_t(z)$, we obtain the approximation:

$$\frac{1}{\sqrt{v_t(z)} + \epsilon} \approx \frac{1}{A_t(z)} - \frac{\Delta_t(z)}{2A_t(z)^3},$$

where $\Delta_t(z)$ represents the change in the second moment due to the current gradient. Substituting this approximation back into the Adam update rule, we can rewrite the target attribution score $\phi_z(U_{(1)}^{(t)})$ as a dot product between the gradient $\nabla\ell(w_t, z^{(\text{val})})$ and a constructed **"Adam-Ghost" vector** $v_{\text{ghost}}(z)$:

$$\phi_z \approx \nabla\ell(w_t, z^{(\text{val})}) \cdot v_{\text{ghost}}(z).$$

Importantly, $v_{\text{ghost}}(z)$ admits a representation as a linear combination of the raw gradient associated with the training point z, enabling efficient computation via a single backpropagation pass. Additional technical details of this approximation are provided in Appendix A.

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. Our experiments are organized to verify three core claims: (1) **Practical Effectiveness**, where we assess the method's ability to identify influential data under semantic source identification and data pruning on SST-2; (2) **Computational Efficiency**, where we validate the scalability of the Linearized Ghost Approximation for large-scale training; and (3) **Approximation Fidelity**, where we confirm that our formulation faithfully tracks the true marginal contributions under Adam dynamics compared to ground-truth and SGD updates.

5.1 PRACTICAL EFFECTIVENESS

We evaluate the utility of our method through two distinct tasks: semantic source identification and data pruning.

5.1.1 SEMANTIC SOURCE IDENTIFICATION.

Experimental Setup. We examine whether data contribution under Adam optimization reflects genuine semantic influence. We use DistilGPT-2 (82M) (Sanh et al., 2019; Radford et al., 2019) trained on WikiText-2 (Merity et al., 2017). For efficiency, only the first $10\,\mathrm{k}$ training lines ($\sim 350\,\mathrm{k}$ tokens) are used for training, with the remaining data reserved for validation and testing. We focus exclusively on SGD- and Adam-aware In-Run Data Shapley attribution methods.

Semantic Perturbation Protocol. For each trial, we randomly select a training example z^* as the source sample. Based on z^*, we construct four types of validation queries with increasing semantic distance: (i) *Partial exactly the same*, (ii) *Paraphrase*, (iii) *Significant paraphrase*, and (iv) *Similar topic*. Paraphrases are generated using a large language model (DeepSeek). Each query is treated as a validation input $z^{(\mathrm{val})}$.

Evaluation Metric. For each validation query $z^{(\mathrm{val})}$, we compute attribution scores ϕ_i for all training samples z_i. Samples are ranked according to ϕ_i, and we record the rank position of the true source sample z^*. A lower rank indicates that the attribution method successfully identifies the training example most responsible for the validation query. We report the average rank across multiple paraphrases and multiple random trials.

Results and Analysis. Table 1 reports the average rank of z^* under different semantic perturbations. BM25 performs well only when lexical overlap is high, but its ranking quality degrades rapidly under paraphrasing and becomes unreliable for topic-level similarity. Influence-function-based attribution and SGD-based In-Run Data Shapley exhibit unstable behavior across scenarios, frequently failing to recover z^* once surface similarity diminishes. In contrast, Adam-aware In-Run Data Shapley consistently assigns low ranks to the true source sample z^* even under significant paraphrasing and similar-topic perturbations. This demonstrates that Adam-aware attribution captures optimizer-mediated semantic contribution, rather than relying on surface-level lexical overlap or memorization effects.

5.1.2 DATA PRUNING ON SST-2.

Experimental Setup. We train DistilBERT (Sanh et al., 2019) on a subset of 5,000 SST-2 training samples (Socher et al., 2013) with AdamW at a learning rate of 2×10^{-5} (batch size 16, 5 epochs), and compute Adam-aware In-Run Shapley scores using our Linearized Ghost Approximation implementation. We then perform three data pruning strategies: removing the top 10%-30% (highest scores), bottom 10%-30% (lowest scores), and random 10%-30% samples on SST-2 sentiment classification task. We retrain the model on each pruned dataset and evaluate accuracy on the full validation set (872 examples). All results are averaged over three seeds.

To provide a controlled comparison under SGD dynamics, we repeat the above experiment using stochastic gradient descent. Specifically, we compute SGD-aware In-Run Shapley scores. For the pruning experiments, we retrain DistilBERT using SGD with momentum 0.9, a learning rate of 3×10^{-4}, and a linear warmup over 6% of the total training steps. All other experimental settings, including data splits and pruning ratios, are kept identical to the AdamW setting. We additionally include random pruning as a budget-matched control to disentangle the effect of which samples are removed from the effect of how many samples are removed. This enables us to isolate whether performance gains arise from the intrinsic value of the selected samples rather than from reduced data volume alone.

Results and Analysis. Table 2 reports SST-2 validation accuracy after pruning different fractions of the training set. Two consistent patterns emerge.

(1) Adam-aware pruning is robust and consistently outperforms random pruning. Under AdamW, removing the *bottom*-ranked samples according to Adam-aware In-Run Shapley yields the strongest performance across all pruning ratios. In particular, at prune ratios of 10%, 20%, and 30%, bottom-pruning achieves 0.8876 / 0.8716 / 0.8681 accuracy, which is consistently higher than the

Table 2: **Validation accuracy after data pruning on SST-2 with DistilBERT.** Comparison between Adam-based and SGD-based pruning strategies.

	Adam			SGD		
Prune Ratio	**Bottom**	**Random**	**Top**	**Bottom**	**Random**	**Top**
10%	**0.8876**	0.8704	0.8555	0.8392	0.8131	0.8119
20%	**0.8716**	0.8681	0.8612	0.8228	0.7706	0.7649
30%	**0.8681**	0.8601	0.8498	0.7117	0.6760	0.6383

corresponding random-pruning baselines (0.8704 / 0.8681 / 0.8601). This indicates that the computed contributions are actionable for data curation: a non-trivial subset of examples receives reliably low (or negative) utility under Adam dynamics, and removing them improves generalization more than an equal-budget random removal.

(2) Pruning is strongly optimizer-dependent; SGD-based pruning is markedly less stable. When repeating the same protocol under SGD, performance degrades substantially as the pruning ratio increases. At a pruning ratio of 30%, even the best-performing SGD condition (bottom-pruning) drops to 0.7117, while top-pruning further collapses to 0.6383. Moreover, under SGD the ordering *bottom* $>$ *random* $>$ *top* holds consistently; however, all three settings perform markedly worse than their AdamW counterparts. This gap suggests that contribution estimates—and the resulting pruning decisions—do not transfer across optimization dynamics. These results empirically support our central claim: *data value is defined relative to the training algorithm*. In particular, using SGD-aware scores (or SGD-style proxies) to guide pruning in modern Adam-like pipelines can lead to weaker or inconsistent gains, especially under aggressive pruning budgets.

Additional observations. Across both optimizers, *top*-pruning is consistently harmful, which is expected since it removes the most helpful examples by the corresponding attribution ranking. Meanwhile, the gap between bottom- and random-pruning under AdamW remains positive at all ratios, indicating that the attribution signal is not merely noise. Notably, AdamW accuracies remain relatively stable from 20% to 30% bottom-pruning ($0.8716 \rightarrow 0.8681$), suggesting that a sizable fraction of the training subset can be removed without hurting—and sometimes improving—validation performance.

Overall, Adam-aware In-Run Shapley provides a principled mechanism for *online* data curation: with a fixed pruning budget, it selects removals that are consistently better than random under AdamW, while the strong degradation under SGD highlights the necessity of optimizer-aware attribution when pruning data for modern training pipelines.

5.2 Computational Efficiency via Linearized Ghost Approximation

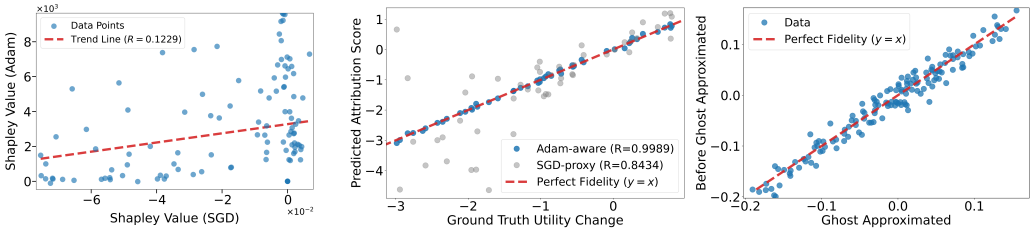
A critical contribution of this work is the *Linearized Ghost Approximation*, designed to make Adam-aware attribution feasible for foundation model training by eliminating the need to materialize per-sample gradients. In this subsection, we validate that our method retains the high throughput and low memory footprint necessary for large-scale training.

Experimental Setup. We benchmark the runtime and memory performance on the **GPT-2 Small** model (124M parameters) trained with AdamW. All experiments are conducted on a single NVIDIA A100 (80GB) GPU, using a batch size of 16 and a sequence length of 1024. We compare three configurations: (i) **Standard AdamW** training without attribution, (ii) **Adam-Ghost (Ours)**, our optimized Adam-aware In-Run implementation, and (iii) **Adam-Direct (Naive)**, a baseline that explicitly computes per-sample gradients and applies independent Adam updates for each example.

Table 3: Efficiency comparison on GPT-2 Small ($BS = 16$, $Seq = 1024$). Adam-Ghost maintains near-baseline efficiency, while Adam-Direct suffers from linear scaling in both time and memory.

Method	**Throughput (SPS)**	**Peak Memory (MB)**
Standard	92.41	**5179.0**
Adam-Ghost (Ours)	**87.85**	5179.6
Adam-Direct	25.58	12965.0

Throughput Analysis. Since In-Run attribution is performed simultaneously with model updates, we benchmark our method against a standard training baseline (without attribution) to quantify the



(a) Fidelity under optimizer mismatch. Attribution scores computed assuming SGD fail to recover true marginal utility when the training optimizer is Adam.

(b) Fidelity under Adam optimization. Adam-aware In-Run Shapley scores exhibit near-perfect agreement with ground-truth marginal utility ($R = 0.9992$), while the SGD-based proxy shows substantial deviation ($R = 0.8434$).

(c) Approximation fidelity of Ghost technique. The linearized Ghost approximation for Adam dynamics aligns closely with the exact Adam-aware scores(R= 0.9803).

Figure 2: Optimizer dependence and fidelity of data attribution. **(Left)** Optimizer mismatch leads to significant fidelity degradation: attribution scores computed assuming SGD fail to recover the true marginal utility when the training optimizer is Adam. **(Middle)** When the optimizer is correctly accounted for, Adam-aware In-Run Shapley accurately recovers ground-truth marginal utility, while SGD-based proxies exhibit substantial deviation. **(Right)** The linearized Ghost approximation closely matches the exact Adam-aware attribution, demonstrating that substantial efficiency gains can be achieved with minimal attribution error.

additional computational cost. As shown in Table 3, the proposed Adam-Ghost method achieves a throughput of **87.85 samples/sec**, retaining **95.1%** of the efficiency of standard AdamW training (92.41 samples/sec). This result confirms that our Linearized Ghost technique extends naturally to Adam-based optimizers with *negligible runtime overhead*, making it feasible to compute data values dynamically during the training phase without acting as a bottleneck. By leveraging the Ghost technique, it enables real-time data valuation during the training process without significantly compromising training speed, making it highly suitable for practical large-scale applications.

In contrast, the Adam-Direct implementation operates at only **25.58 samples/sec**, resulting in a **3.6**$\times$ **slowdown** compared to standard training. This slowdown arises from two compounding factors: (i) the need to perform *sequential backward passes* for each individual training sample, and (ii) the substantial computational overhead of maintaining and updating independent optimizer states for every sample. Together, these costs induce linear scaling in computation with respect to the batch size, rendering naive Adam-based attribution impractical for large-scale models. However, our proposed Linearized Ghost Approximation technique circumvents this scaling bottleneck by aggregating per-sample contributions within a *single backpropagation pass*, thereby decoupling the attribution cost from the batch size.

Memory Overhead. Despite incorporating dense second-moment information, the proposed Linearized Ghost Approximation technique maintains negligible memory overhead. Its peak GPU memory usage (5179.6 MB) is virtually identical to that of standard training (5179.0 MB), confirming that our formulation avoids the materialization of additional per-sample tensors. In sharp contrast, the Adam-Direct baseline consumes **12965.0** MB of memory, representing a substantial **150%** increase over the standard configuration. This dramatic increase stems from the need to explicitly store $B = 16$ independent gradient replicas together with their corresponding optimizer states. Such linear memory scaling constitutes a severe bottleneck and would quickly lead to out-of-memory failures when applied to larger models or increased batch sizes.

Overall, these results demonstrate that Adam-aware In-Run Data Shapley is computationally practical for foundation model training. By avoiding explicit per-sample gradient computation and optimizer-state instantiation, the proposed Linearized Ghost Approximation preserves the runtime and memory footprint of standard AdamW training.

5.3 Fidelity Comparison

In this subsection, we evaluate the fidelity of the proposed Adam-aware closed-form approximation against the reference marginal utility of training samples measured via explicit one-step Adam updates. A faithful in-run attribution method must accurately reflect the optimization trajectory

induced by the actual optimizer—particularly under Adam, where parameter updates are governed by adaptive moment estimates.

Experimental Setup. To assess fidelity across various configurations, we conduct a series of controlled experiments comparing our **Adam-aware approximation** with the **SGD-based proxy** (Wang et al., 2025). We consider three complementary evaluation protocols: (i) comparing attribution scores obtained from independent training runs optimized with Adam versus SGD, (ii) measuring per-sample fidelity against reference Adam utility changes at a fixed learning rate, and (iii) evaluating robustness across a wide spectrum of learning rates $\eta \in [10^{-7}, 10^{-3}]$. Pearson correlation ($R$) is adopted as the primary metric to quantify the alignment between predicted attribution scores and observed utility changes.

Optimizer Dependency of Data Value. Figure 2a compares attribution scores produced by independent Adam and SGD training runs. The pronounced scatter and weak alignment indicate that data value is not an intrinsic, optimizer-agnostic property; instead, it is tightly coupled to the optimization dynamics. This observation underscores the risk of optimizer mismatch when applying proxy-based attribution methods. Importantly, this observed optimizer dependence is not an artifact of our closed-form approximation. As shown in Section 4.1, even retraining-based TMC-Shapley estimates—serving as a model-agnostic ground truth—exhibit substantial disagreement between SGD and Adam trajectories ($R \approx 0.0579$). This consistency between exact retraining results and the patterns observed here indicates that data value is fundamentally coupled to optimization dynamics, rather than being an intrinsic property of the sample. This underscores the necessity of an optimizer-aware framework for faithful data attribution.

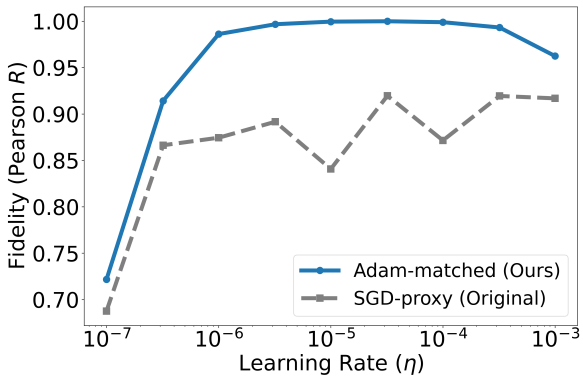


Figure 3: **Fidelity across learning rates.** Pearson correlation between predicted scores and ground-truth utility changes across different learning rates η. Adam-aware approximation maintains high fidelity ($R > 0.96$) across the board, while the SGD-proxy exhibits unstable and lower correlation.

Superiority over SGD-Proxies. In Figure 2b, we evaluate both methods against reference Adam utility changes within a single run. The Adam-aware method achieves near-perfect fidelity ($R = 0.9990$), whereas the SGD-proxy exhibits substantially lower correlation ($R = 0.8434$). This trend persists across learning rates, as shown in Figure 3, where the Adam-aware approximation consistently maintains high fidelity ($R > 0.96$), while the SGD-proxy displays instability and degradation. These results provide empirical evidence consistent with the fidelity limitations discussed in Wang et al. (2025) (Remark 7).

Validity of the Ghost Approximation. To reduce computational overhead, we employ a linearized "Ghost Approximation" of the Adam-aware update. Figure 2c compares the ghost-approximated scores with the full closed-form computation. The near-perfect alignment along the $y = x$ diagonal confirms that the proposed linearization introduces negligible approximation error while enabling substantial efficiency gains.

6 Conclusion

In this work, we propose a rigorous framework for In-Run Data Shapley tailored to adaptive optimization. By modeling the non-linear dynamics of Adam, we derive a closed-form attribution estimator that achieves near-perfect fidelity to ground-truth marginal contributions ($R > 0.99$), overcoming the failure of standard SGD-based proxies in adaptive regimes ($R \approx 0.11$). We further introduce a Linearized Ghost Approximation to compute these scores with negligible overhead, maintaining $\sim$95% of training throughput. Our results demonstrate that optimizer-aware attribution is both theoretically necessary and computationally feasible, enabling effective data pruning and robust source identification for modern large-scale models.

REFERENCES

Satyaki Basu, Sayantan Ghosh, Sudeep Vikram, and Tom Goldstein. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.

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.

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, 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, Daiwei Dao, Boxin Wang, and et al. Towards efficient data valuation based on the shapley value. In *AISTATS*, 2019a.

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, 2019b. URL https://api.semanticscholar.org/CorpusID:263794198.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

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.

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.

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.

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.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

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.

A In-Run Data Shapley under Adam

A.1 Proof of Theorem 4.1

Theorem A.1 (Restatement of Theorem 4.1)**.** *In-Run Data Shapley via Adam, considering the first-order approximation, has a closed-form*

$$\phi_z(U) \approx \sum_{t=0}^{T-1} \phi_z\left(U_{(1)}^{(t)}\right), \tag{5}$$

where

$$\phi_z\left(U_{(1)}^{(t)}\right) = -\eta_t \nabla \ell\left(w_t, z^{(\mathrm{val})}\right) \cdot \frac{m_t}{\sqrt{v_t}+\epsilon}, t = 0, \ldots, T-1.$$

Proof. The first-order Taylor approximation to the local utility function is as follows:

$$\begin{aligned} U^{(t)}(S) &= \ell\left(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}\right) - \ell\left(w_t, z^{(\mathrm{val})}\right) \\ &= \underbrace{\nabla \ell\left(w_t, z^{(\mathrm{val})}\right) \cdot \left(\widetilde{w}_{t+1}(S) - w_t\right)}_{U_{(1)}^{(t)}(S)} + \text{higher order terms} \end{aligned} \tag{6}$$

Note that the model update satisfies $w_t = w_{t-1} - \eta_t \cdot \frac{m_t}{\sqrt{v_t}+\epsilon}$, therefore, we have the marginal contribution of z for any $S \subseteq \mathcal{B}_t \backslash z$ is

$$U_{(2)}^{(t)}(S \cup z) - U_{(2)}^{(t)}(S) = -\eta_t \nabla \ell\left(w_t, z^{(\mathrm{val})}\right) \cdot \frac{m_t}{\sqrt{v_t}+\epsilon}.$$

Combining this with the definition of the Shapley value immediately yields the result. □

A.2 Efficient Computation via Linearized Ghost Dot-Produc

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

Decomposition of the First Moment. The bias-corrected first moment $\hat{m}_t$ is defined as:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1^t},$$

where $g_t = \nabla \ell(w_t, z)$ is the gradient of the current sample z. We decompose this into a history term and a current gradient term:

$$\hat{m}_t = \underbrace{\frac{\beta_1}{1-\beta_1^t}}_{C_m^1} m_{t-1} + \underbrace{\frac{1-\beta_1}{1-\beta_1^t}}_{C_m^2} g_t.$$

Thus, we have $\hat{m}_t = C_m^1 m_{t-1}(z) + C_m^2 \nabla \ell(w_t, z)$.

Linearization of the Second Moment. The non-linear denominator prevents efficient computation. We linearize it using a Taylor expansion. We first define the variance terms, and the bias-corrected second moment is:

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} = \frac{\beta_2 v_{t-1} + (1-\beta_2) g_t^2}{1-\beta_2^t}.$$

Let $C_v = \frac{1-\beta_2}{1-\beta_2^t}$ and assume the history component is captured by a fixed state $\hat{v}_{t-1}$ (or strictly A_t derived from history). The update can be viewed as a perturbation:

$$\hat{v}_t \approx \hat{v}_{t-1} + C_v \nabla \ell(w_t, z)^2.$$

Then, we approximate the function $f(x) = \frac{1}{\sqrt{x}+\epsilon}$ around the history term. Let $A_t(z) = \sqrt{\hat{v}_{t-1}(z)}+\epsilon$ be the preconditioning term based on history. Using the first-order Taylor expansion $f(x+\delta) \approx$

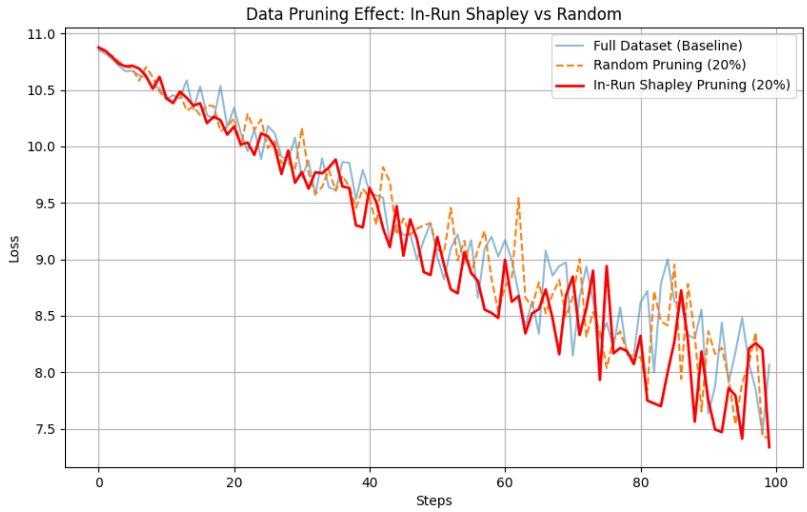


Figure 4: Data pruning effect: validation loss versus training steps. We compare the full dataset, random pruning (20%), and In-Run Shapley pruning (20%).

$f(x) + f'(x)\delta : f(x) = \frac{1}{\sqrt{x}+\epsilon} \implies f(\hat{v}_{t-1}) = \frac{1}{A_t(z)} f'(x) \approx -\frac{1}{2(\sqrt{x}+\epsilon)^2\sqrt{x}} \approx -\frac{1}{2A_t(z)^3}$ (assuming $\sqrt{x} \approx \sqrt{x} + \epsilon$ for the derivative magnitude approximation). Substituting $\delta = C_v \nabla \ell(w_t, z)^2$, we have:

$$\frac{1}{\sqrt{\hat{v}_t} + \epsilon} \approx \frac{1}{A_t(z)} - \frac{C_v \nabla \ell(w_t, z)^2}{2A_t(z)^3}.$$

Therefore, the closed-form approximation for the attribution score at step t can be derived:

$$\phi_z \left(U_{(1)}^{(t)} \right) = -\eta_t \nabla \ell \left(w_t, z^{(\text{val})} \right) \cdot \left(\frac{1}{A_t(z)} - \frac{C_v \nabla \ell \left(w_t, z \right)^2}{2A_t(z)^3} \right) \cdot \left(C_m^1 m_{t-1}(z) + C_m^2 \nabla \ell \left(w_t, z \right) \right) \tag{7}$$

To apply the ghost dot-product technique, we must isolate the terms that are linear with respect to the per-sample gradient $g_t(z) = \nabla \ell(w_t, z)$. Let us expand the product above. We denote the preconditioning term $A_t = \sqrt{\hat{v}_{t-1}} + \epsilon$ and treat it as a constant vector derived from the optimizer state history (fixed-state approximation).

The expansion yields four interaction terms. We group them into a History Term (independent of $g_t(z)$) and a Gradient Term (linear in $g_t(z)$), while truncating higher-order interactions ($O(g_t^2), O(g_t^3)$) which are negligible in the small-perturbation regime:

$$\phi_z \approx \underbrace{-\eta_t \nabla \ell_{val} \cdot \left(\frac{C_m^1 m_{t-1}}{A_t} \right)}_{\textbf{History Term}} + \underbrace{-\eta_t \nabla \ell_{val} \cdot \left(\frac{C_m^2}{A_t} \odot g_t(z) \right)}_{\textbf{Linear Gradient Term}} \tag{8}$$

- **History Term:** This term involves the dot product between the validation gradient and the preconditioned momentum history. Since m_{t-1} and A_t are shared or pre-computed optimizer states, this scalar can be computed directly without per-sample gradient instantiation.
- **Linear Gradient Term:** This term represents the contribution of the current step's gradient $g_t(z)$, scaled by the Adam coefficient C_m^2 and the inverse preconditioner $1/A_t$.

Then, we can utilize the standard Ghost Dot Product to compute the Adam-aware scores.

B Additional Experiment: Data Pruning Effect

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 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 (baseline), (2) training after randomly pruning 20% of the training samples, and (3) training after pruning 20% of the training samples with the lowest In-Run Shapley scores.

In-Run Shapley Pruning. In-Run Shapley scores are computed during a preliminary training pass using an 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.

Results. Figure 4 shows validation loss as a function of training steps. Compared to random pruning, In-Run Shapley pruning yields a consistently lower validation loss trajectory under the same pruning ratio, indicating that the identified samples are less beneficial to training.

C 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 C.2 for exact reproducibility.

C.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.

C.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
- "similar_topic": same general topic, different facts/details, minimal lexical

Forbidden tokens:
[FORBIDDEN_TOKEN_LIST]

z_star:
[SOURCE TEXT]
```

D Empirical Validation of the Ghost Linear Approximation

To justify the use of the proposed ghost linear approximation, we empirically examine both the magnitude regime in which the approximation is applied and its fidelity with respect to the exact (pre-ghost) quantity.

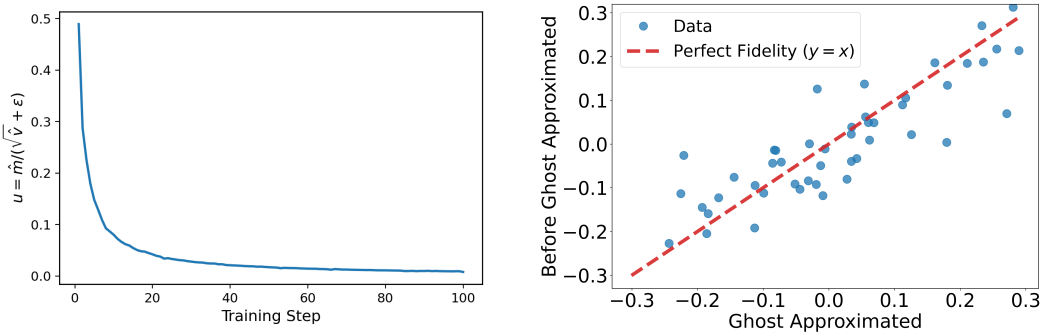


Figure 5: **Empirical validity of the ghost approximation.** (Left) Evolution of the effective perturbation magnitude u, showing that optimization rapidly enters and remains in a small-perturbation regime. (Right) Worst-case fidelity test conducted during the first 10 optimization steps, where u is largest. Even under these early and high-magnitude conditions, the ghost approximation maintains strong agreement with the exact computation (Pearson $r = 0.7502$, Spearman $\rho = 0.8461$), indicating robust behavior beyond its nominal operating regime.

Small-perturbation regime during training. We first analyze the evolution of the effective perturbation term

$$u = \frac{\hat{m}}{\sqrt{\hat{v}} + \varepsilon}, \tag{9}$$

which governs the deviation induced by the ghost approximation. As shown in Figure 5 (left), the median value (P_{50}) of u decreases rapidly over training and remains consistently small throughout optimization. After a brief transient phase, u stabilizes at a magnitude on the order of 10^{-2}, indicating that the optimizer operates predominantly in a small-perturbation regime. This observation is critical: the ghost formulation relies on a first-order linearization with respect to u, and the empirical scale of u lies well within the range where higher-order terms are negligible. Consequently, the approximation error introduced by truncating second- and higher-order components is expected to be minimal for the majority of training steps.

Worst-case fidelity under early and high-magnitude regimes. While the above analysis characterizes the typical operating regime of the optimizer, a natural concern is whether the ghost approximation remains reliable during the early training phase, where the effective perturbation magnitude u can be substantially larger and the linearization assumption is least favorable. To address this question, we perform a focused stress test by restricting attention to the first 10 optimization steps and randomly selecting 50 training samples. This setting represents a conservative, worst-case regime for the proposed approximation.

Despite these unfavorable conditions, the ghost-approximated values continue to exhibit strong agreement with their exact (pre-ghost) counterparts. As shown in Figure 5 (right), the approximation achieves a Pearson correlation of $r = 0.7502$ and a Spearman rank correlation of $\rho = 0.8461$. Notably, this result demonstrates that the ghost approximation preserves both the relative ordering and the magnitude structure of the exact quantity even in the earliest and most unstable phase of training. This stress-test validation provides a strictly stronger guarantee than fidelity evaluated in the typical small-u regime, as it bounds the approximation error under the most challenging conditions encountered in practice.

Summary. Taken together, these results show that:

- The ghost approximation is applied predominantly in a regime where the perturbation magnitude u is inherently small.
- Even outside this regime, during early training when u is largest, the approximation remains faithful in both magnitude and ranking.
- The overall approximation error is therefore bounded and dominated by higher-order effects that are empirically negligible.

These findings provide strong empirical support for adopting the ghost linear approximation as a computationally efficient yet reliable surrogate for the exact formulation.