

When Does Training Data Matter?

Dynamic Data Attribution via Window-level Influence Estimation

Jie Xu Zihan Wu Wenbo Pan Yulong Ming Xiaohua Jia

City University of Hong Kong, Hong Kong

Abstract

Data attribution methods often compress the effect of each training data point into a single score, computed either at a final checkpoint or by aggregating quantities along the training trajectory. This static view hides *how* the contribution of a training data point changes during training. To overcome this limitation, we propose the Window-level Influence Estimator (WIE), a framework for estimating the contribution of a training data point z_j over a user-specified training window $[t_1, t_2]$. WIE models a leave-one-out-style counterfactual under a first-order linearization of SGD. An *influence propagator* $P^{[t]} = I - \eta_t H^{[t]}$ at each step models how parameter deviation is amplified or dampened by the loss landscape. Empirically, the window-level estimator reveals four distinct influence patterns: *Early Influencer*, *Late Bloomer*, *Stable*, *Highly Fluctuating*. Across multiple benchmarks, WIE improves mislabeled-data detection over the considered static and trajectory-based baselines. On BERT-IMDB with 40% label noise, WIE outperforms these baselines across the evaluated inspection ratios. On ViT-EMNIST, WIE reaches 70% recall for mislabeled data while inspecting up to $7.5\times$ fewer candidates than TracIn requires to reach the same recall.

1 Introduction

Traditional data attribution methods assign each training data point an importance score, computed either at a converged checkpoint (Koh and Liang, 2017; Grosse et al., 2023) or summed over the full trajectory (Pruthi et al., 2020).

In this paper, we challenge this static view by showing that **data influence is inherently dynamic**. The contribution of training data peaks, decays, or fluctuates as optimization progresses. As shown in Figure 1, training data with nearly identical global scores can play very different roles during training. Some shape decision boundaries

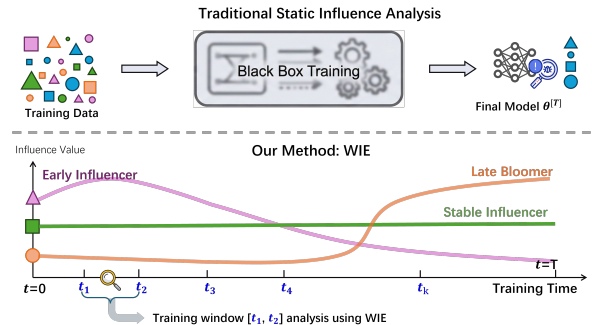


Figure 1: **The dynamic nature of data influence.** **Top:** Traditional analysis treats training data as having a static result on the final model $\theta^{[T]}$. **Bottom:** Our window-level view reveals that data influence is inherently dynamic and has distinct patterns that change over time.

early and then fade (*Early Influencers*); others contribute mainly near convergence (*Late Bloomers*). However, a single importance/influence score cannot tell whether training data guided early feature learning or assisted late-stage refinement. Yet this difference is exactly what drives curriculum design (Bengio et al., 2009), data selection (Xia et al., 2024; Yu et al., 2024), targeted debugging (Koh and Liang, 2017), and machine unlearning (Xu et al., 2024). Without temporal resolution, the most informative phases of the learning process remain unclear.

Existing methods are unable to address this problem. Influence functions approximate the Hessian at a single converged point (Koh and Liang, 2017; Kwon et al., 2024), so they cannot show within-training dynamics entirely. Although TracIn scores (Pruthi et al., 2020) can be summed over selected checkpoints, such partial sums do not explicitly model how a perturbation introduced within a window propagates through subsequent optimization steps. Recent trajectory-level methods take a step closer (Wang et al., 2025a; Lee et al., 2025), but In-Run Data Shapley estimates a Shapley-value-style

cooperative game quantity during a single training run, and Influence Dynamics derives its stages from singular learning theory rather than letting the user choose them. In short, existing methods are either locked to a single time point or aggregate the full trajectory without the ability to zoom into an arbitrary window.

It is challenging to achieve dynamic attribution. First, the influence of training data at step t does not act in isolation: it propagates through all subsequent optimization steps, amplified or dampened by the changing loss landscape. A naive per-step analysis that ignores this cross-step coupling can misattribute influence, assigning credit to the wrong temporal phases. Second, any practical estimator must support post-hoc window queries without retraining. Practitioners need to zoom into an arbitrary interval $[t_1, t_2]$ after training has concluded, yet the parameter trajectory is fixed once training ends. The method must therefore simulate the leave-one-out counterfactual within any user-chosen window using only quantities available from a single run.

To address both challenges, we introduce the **Window-level Influence Estimator (WIE)**¹, a dynamic attribution framework that tracks how data influence propagates across the training trajectory. The core idea is to formalize the leave-one-out parameter deviation as a recursion over SGD steps: at each step, the previous deviation is multiplied by the *influence propagator* $P^{[t]} = I - \eta_t H^{[t]}$, which captures how the loss landscape amplifies or dampens the deviation. This recursion resolves the first challenge by modeling cross-step propagation in closed form rather than treating each step independently. For the second challenge, because the recursion depends only on gradients and Hessian-vector products (Pearlmutter, 1994) that can be computed from saved checkpoints, WIE supports any post-hoc window query $[t_1, t_2]$ without any retraining. The full-trajectory case $[0, T]$ is a special case of WIE. In this sense, WIE extends trajectory-based attribution from a single full-path score to user-specified temporal intervals, while targeting a leave-one-out-style counterfactual.

Extensive experiments across tabular, vision, and language benchmarks demonstrate the effectiveness of WIE in the evaluated settings. Across five datasets and five architectures, WIE achieves the closest agreement with leave-one-out retraining

and uncovers four temporal patterns (*Early Influencer*, *Late Bloomer*, *Stable*, *Highly Fluctuating*). On BERT-IMDB with 40% label noise, WIE outperforms IF, DVEmb, and LAVA at every inspection ratio. On ViT-EMNIST, WIE detects 70% of mislabeled training data while inspecting up to $7.5\times$ fewer candidates than TracIn, because the corruption reveals its negative impact within specific training windows that only a dynamic estimator can resolve.

The contributions are summarized as follows:

- We propose the framework of *window-level data attribution*, which formalizes the influence of training data within any temporal interval $[t_1, t_2]$ as a recursion over the influence propagator $P^{[t]}$.
- We design the *Window-level Influence Estimator (WIE)*, a recursive influence propagation estimator that approximates window-level Leave-One-Out (LOO) influence and supports arbitrary differentiable query metrics.
- Comprehensive experiments show that WIE significantly outperforms static baselines across diverse settings. Temporal profile analysis confirms that exploiting training dynamics detects 70% of mislabeled data while inspecting up to $7.5\times$ fewer candidates, demonstrating its critical role in revealing hidden structure in the training process.

2 Related Work

Static data attribution. Influence functions locally approximate a trained solution to estimate how upweighting or removing a training data point z_j affects a target loss or prediction (Koh and Liang, 2017). Recent extensions improve scalability through candidate filtering, curvature approximations, efficient inverse-Hessian solvers, closed-form approximations, and gradient projection (Guo et al., 2021; Schioppa et al., 2022; Grosse et al., 2023; Kwon et al., 2024; Choe et al., 2026). These methods typically return a scalar attribution score for a fixed target loss, prediction, or checkpoint, and can be sensitive to non-convexity and local-linearization assumptions (Basu et al., 2021; Bae et al., 2022). TracIn sums gradient inner products across checkpoints, while TRAK scales scalar attribution through randomized projections and a kernel-based counterfactual view (Pruthi et al., 2020; Park et al., 2023). These scalar views

¹Code and experiment configurations are available at <https://github.com/wzh4464/WIE>.

do not explicitly model how a perturbation introduced within a specific training window propagates through subsequent optimization steps.

Dynamic and trajectory-based influence.

Trajectory-aware methods study how data impacts changes during optimization (Xu et al., 2026). SGD-Influence retraces SGD paths, and Simfluence learns a simulator of training runs (Hara et al., 2019; Guu et al., 2023). Related pre-training methods, such as PRESENCE, use self-influence for data reweighting during language-model pre-training (Thakkar et al., 2023). Most related to our work is DVEmb, which formalizes trajectory-specific leave-one-out influence at the level of individual training iterations (Wang et al., 2025b). WIE shares the goal of capturing time-dependent influence but supports a different query interface. It aggregates the perturbations induced by a training data point z_j over an arbitrary user-specified window and propagates the resulting deviation to a later checkpoint. Under an idealized construction with exact Hessians, no random projection, aggregation over all occurrences, and endpoint-specific queries, the two estimators coincide to first order; the practical configurations differ, as DVEmb’s scalable form adopts generalized Gauss–Newton curvature and random projection, whereas WIE evaluates exact Hessian-vector products over the full parameter space. Appendix D gives a detailed account of this relationship. In-Run Data Shapley estimates a cooperative-game quantity during one training run, while Influence Dynamics studies stagewise attribution grounded in singular learning theory (Wang et al., 2025a; Lee et al., 2025). In contrast, WIE enables post-hoc queries over arbitrary temporal windows without relying on a fixed stage decomposition.

Data selection and valuation. Model-aware methods such as LESS, MATES, and DsDm compute efficient scores to choose training data for future training or fine-tuning (Xia et al., 2024; Yu et al., 2024; Engstrom et al., 2024). Other criteria, including GraNd, EL2N, and game-theoretic Shapley values, prioritize downstream utility for pruning, valuation, or noisy-label detection (Paul et al., 2021; Ghorbani and Zou, 2019; Jia et al., 2019; Xu et al., 2025). These scores are useful and often designed for efficiency, but they do not target fidelity to the leave-one-out retraining effect of suppressing the updates involving a training data point z_j within a specified window of a completed

trajectory. WIE is complementary: it estimates a window-level leave-one-out counterfactual and propagates the induced deviation through later optimization.

3 Preliminaries and Problem Formulation

3.1 Setup

Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ denote the observation space, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the input space and $\mathcal{Y}$ is the output space. Given a training dataset $D = \{z_i\}_{i=1}^N$ of i.i.d. observations $z_i = (x_i, y_i) \in \mathcal{Z}$, a model $f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$ with parameters $\theta \in \Theta \subseteq \mathbb{R}^p$, and a loss function $\ell : \mathcal{Z} \times \Theta \rightarrow \mathbb{R}$, the learning process is formulated as

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \ell(z_i; \theta). \quad (1)$$

3.2 SGD Training Trajectory

Stochastic Gradient Descent (SGD) is the representative solver for the above optimization, and we adopt it for two reasons. First, most prior influence analyses are developed in the SGD setting (Koh and Liang, 2017; Pruthi et al., 2020; Hara et al., 2019), enabling fair comparison against standard baselines. Second, the SGD update rule has a simple closed form that makes it easier to understand. Extension to Adam-family optimizers preserves the same coupled-recurrence structure and is detailed in Appendix H.

Let $g(z; \theta) = \nabla_{\theta} \ell(z; \theta)$ and denote the initialization by $\theta^{[0]}$. At each step t , a mini-batch $S_t \subseteq \{1, \dots, N\}$ is sampled and SGD updates the parameters as

$$\theta^{[t+1]} = \theta^{[t]} - \frac{\eta_t}{|S_t|} \sum_{i \in S_t} g(z_i; \theta^{[t]}), \quad 0 \leq t \leq T-1, \quad (2)$$

where η_t is the learning rate at step t , and T is the total number of training steps.

3.3 Leave-One-Out Retraining

Leave-One-Out (LOO) retraining process is defined to isolate the influence of a specific training data point z_j . It starts from the same initialization $\theta_{-j}^{[0]} = \theta^{[0]}$ and follows the same mini-batch sequence as Eq. (2), except that z_j is omitted whenever it appears in a mini-batch:

$$\theta_{-j}^{[t+1]} = \theta_{-j}^{[t]} - \frac{\eta_t}{|S_t|} \sum_{i \in S_t \setminus \{j\}} g(z_i; \theta_{-j}^{[t]}), \quad (3)$$

where $0 \leq t \leq T - 1$. The deviation $\theta^{[t]} - \theta_{-j}^{[t]}$ at any step t encodes the cumulative effect of z_j on the model up to that point.

3.4 Problem Formulation

Given a training data point $z_j \in D$ and an arbitrary training window $[t_1, t_2]$ with $0 \leq t_1 < t_2 \leq T$, our goal is to design an estimator $\mathcal{E}$ that quantifies the impact of removing z_j on the model during this period, outputting $\hat{\Delta}_{-j}^{[t_1, t_2]} = \mathcal{E}(z_j, t_1, t_2)$.

Specifically, the influence is measured in two complementary forms. *Parameter-level influence* captures the trajectory deviation:

$$\Delta\theta_{-j}^{[t_1, t_2]} := (\theta_{-j}^{[t_2]} - \theta_{-j}^{[t_1]}) - (\theta^{[t_2]} - \theta^{[t_1]}). \quad (4)$$

Functional influence extends this to any differentiable metric $f(\theta)$ (e.g., test loss, prediction logits):

$$\Delta f_{-j}^{[t_1, t_2]} := [f(\theta_{-j}^{[t_2]}) - f(\theta_{-j}^{[t_1]})] - [f(\theta^{[t_2]}) - f(\theta^{[t_1]})]. \quad (5)$$

For example, setting $f = \ell_{\text{test}}$ quantifies how z_j affects the test loss change within $[t_1, t_2]$.

The estimator should achieve: (i) *fine-grained temporal resolution*, supporting any user-chosen window rather than only the full trajectory; (ii) *high fidelity*, closely approximating the exact LOO deviation; and (iii) *computational efficiency*, requiring no retraining and keeping per-step cost tractable.

4 Window-level Influence Estimator

4.1 Overview of WIE

WIE estimates the window-level influence of training data through two progressive steps, as illustrated in Figure 2.

The first step is *parameter influence estimation* (Section 4.2). We formalize the leave-one-out parameter deviation as a recurrence over SGD steps: at each step, the previous deviation is multiplied by the influence propagator $P^{[t]} = I - \eta_t H^{[t]}$, which captures how the loss landscape amplifies or dampens the deviation. Unrolling this recurrence over any window $[t_1, t_2]$ yields a closed-form estimator of the parameter-level influence, with the full trajectory $[0, T]$ recovered as a special case.

The second step is *query-based functional extension* (Section 4.3). Since the parameter deviation produced by Step 1 is high-dimensional and hard to interpret directly, we project it onto the gradient

of a user-specified metric (e.g., test loss, prediction logits) via a first-order Taylor approximation, converting the parameter influence into a scalar functional influence score. This unifies the analysis of diverse downstream metrics within the same recursive framework.

4.2 Step 1: Parameter Influence Estimation

This step aims to estimate how model parameters would change if a training point were removed. We first formally define the quantity of interest.

Definition 1 (Data Influence on Parameter). *For any window $[t_1, t_2]$, the parameter influence of data point z_j is*

$$\Delta\theta_{-j}^{[t_1, t_2]} := (\theta_{-j}^{[t_2]} - \theta_{-j}^{[t_1]}) - (\theta^{[t_2]} - \theta^{[t_1]}), \quad (6)$$

where $(\theta_{-j}^{[t_2]} - \theta_{-j}^{[t_1]})$ denotes the parameter change on the LOO trajectory where z_j is excluded during $[t_1, t_2]$, and $(\theta^{[t_2]} - \theta^{[t_1]})$ denotes the change on the original trajectory.

Parameter Difference The parameters difference between the LOO update Eq. (3) and the standard update Eq. (2) is:

$$\begin{aligned} \theta_{-j}^{[t+1]} - \theta^{[t+1]} &= (\theta_{-j}^{[t]} - \theta^{[t]}) \\ &\quad - \frac{\eta_t}{|S_t|} \left(\sum_{i \in S_t \setminus \{j\}} g(z_i; \theta_{-j}^{[t]}) - \sum_{i \in S_t} g(z_i; \theta^{[t]}) \right). \end{aligned} \quad (7)$$

To handle the gradient differences, we employ a Taylor expansion around $\theta^{[t]}$:

$$g(z_i; \theta_{-j}^{[t]}) - g(z_i; \theta^{[t]}) \approx H_i^{[t]} (\theta_{-j}^{[t]} - \theta^{[t]}), \quad (8)$$

where $H_i^{[t]} = \nabla_{\theta}^2 \ell(z_i; \theta^{[t]})$ be the Hessian. This approximation is accurate because the small learning rate constrains the parameter deviation, keeping $\theta_{-j}^{[t]}$ in a local neighborhood of $\theta^{[t]}$ where the loss surface is locally quadratic. Let $\mu_{-j}^{[t]} := \theta_{-j}^{[t]} - \theta^{[t]}$ denote the accumulated parameter deviation at step t . Averaging Eq.(8) over S_t and letting $H^{[t]}$ be the mini-batch Hessian, we obtain:

$$\frac{1}{|S_t|} \sum_{i \in S_t} (g(z_i; \theta_{-j}^{[t]}) - g(z_i; \theta^{[t]})) \approx H^{[t]} (\theta_{-j}^{[t]} - \theta^{[t]}). \quad (9)$$

Substituting Eq.(9) into Eq.(7), we derive the core recurrence:

$$\mu_{-j}^{[t+1]} \approx \underbrace{(I - \eta_t H^{[t]}) \mu_{-j}^{[t]}}_{\text{History Propagation}} + \underbrace{\mathbf{1}_{j \in S_t} \frac{\eta_t}{|S_t|} g(z_j; \theta^{[t]})}_{\text{Instantaneous Influence}}, \quad (10)$$

WIE Framework

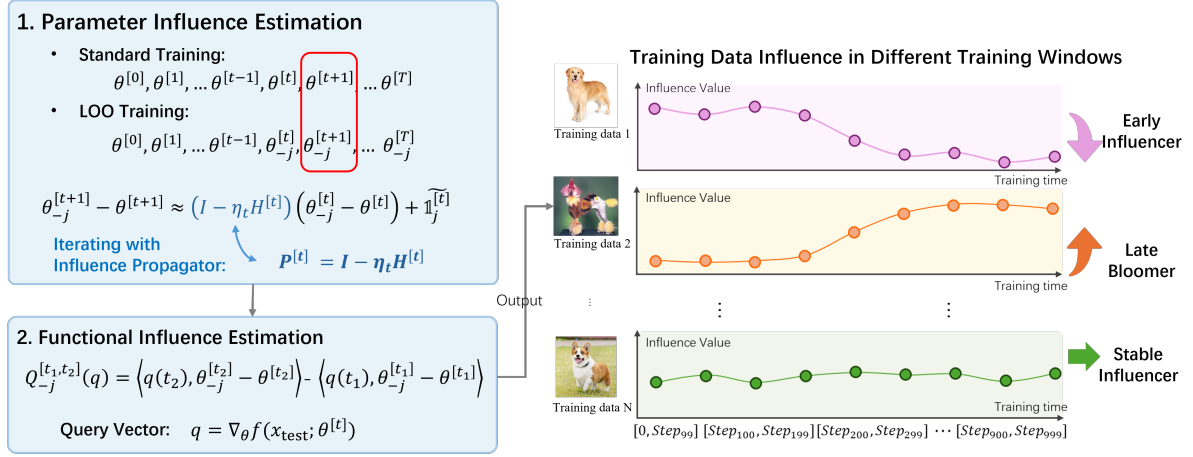


Figure 2: **Overview of WIE.** (1) *Parameter-level estimation* unrolls the leave-one-out parameter deviation through the influence propagator $P^{[t]} = I - \eta_t H^{[t]}$, injecting an instantaneous contribution $\tilde{\mathbf{1}}_j^{[t]}$ whenever z_j is sampled. (2) *Functional-level estimation* reduces the resulting parameter deviation to any differentiable metric (test loss, prediction, feature saliency) via a query vector $q(t)$. Both pieces operate over an arbitrary window $[t_1, t_2]$.

where $\mathbf{1}_{j \in S_t}$ equals 1 if $j \in S_t$, otherwise 0.

Influence Propagator. Equation (10) motivates two named quantities, the *influence propagator* and the *instantaneous influence*:

$$P^{[t]} := I - \eta_t H^{[t]}, \quad (11)$$

$$\tilde{\mathbf{1}}_j^{[t]} := \mathbf{1}_{j \in S_t} \frac{\eta_t}{|S_t|} g(z_j; \theta^{[t]}). \quad (12)$$

We further write the cumulative propagator from step a to step $b - 1$ as $P^{[a, b]} := P^{[b-1]} P^{[b-2]} \dots P^{[a]}$, where the product is ordered with later steps on the left and $P^{[a, a]} := I$. Unrolling the recurrence (10) from t_1 to t_2 yields (full derivation in Appendix A):

$$\begin{aligned} \widehat{\Delta \theta}_{-j}^{[t_1, t_2]} = & \underbrace{\left(P^{[t_1, t_2]} - I \right) \mu_{-j}^{[t_1]}}_{\text{evolution of pre-window deviation}} \\ & + \underbrace{\sum_{t=t_1}^{t_2-1} P^{[t+1, t_2]} \tilde{\mathbf{1}}_j^{[t]}}_{\text{new contributions in window}}, \end{aligned} \quad (13)$$

where $\mu_{-j}^{[t_1]} = \sum_{t=0}^{t_1-1} P^{[t+1, t_1]} \tilde{\mathbf{1}}_j^{[t]}$ is the deviation z_j has already accumulated before the window starts. Intuitively, the estimator separates two effects: the past deviation propagated forward, and new contributions added within $[t_1, t_2]$. The full-trajectory case $[0, T]$ is a special case, so WIE strictly generalizes prior end-of-training estimators while enabling flexible temporal analysis. Section 5 establishes formal error bounds for this estimator.

4.3 Step 2: Query-Based Functional Extension

While Step 1 quantifies influence on model parameters, the parameter deviation $\mu_{-j}^{[t]} \in \mathbb{R}^p$ is high-dimensional and hard to interpret directly. Practitioners often care about scalar downstream metrics such as test loss or prediction confidence. To bridge this gap, we project the parameter deviation onto the gradient of a user-specified metric via a first-order Taylor approximation (Hampel, 1974; Hara et al., 2019). Specifically, for any differentiable metric $f(\theta)$, the effect of removing z_j on f at step t can be approximated as:

$$f(\theta_{-j}^{[t]}) - f(\theta^{[t]}) \approx \langle \nabla_{\theta} f(\theta^{[t]}), \mu_{-j}^{[t]} \rangle, \quad (14)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^p$. Substituting Eq. (14) into the window-level functional influence $\Delta f_{-j}^{[t_1, t_2]}$ defined in Eq. (5) yields:

$$\begin{aligned} \Delta f_{-j}^{[t_1, t_2]} \approx & \langle \nabla_{\theta} f(\theta^{[t_2]}), \mu_{-j}^{[t_2]} \rangle \\ & - \langle \nabla_{\theta} f(\theta^{[t_1]}), \mu_{-j}^{[t_1]} \rangle. \end{aligned} \quad (15)$$

This motivates a general estimator parameterized by an arbitrary query function.

Definition 2 (Functional Influence Estimation). *Let $q : [0, T] \rightarrow \mathbb{R}^p$ be a query function mapping time t to a query vector $q(t) \in \mathbb{R}^p$. The functional influence estimate for z_j over $[t_1, t_2]$ is:*

$$Q_{-j}^{[t_1, t_2]}(q) = \langle q(t_2), \mu_{-j}^{[t_2]} \rangle - \langle q(t_1), \mu_{-j}^{[t_1]} \rangle, \quad (16)$$

where $\mu_{-j}^{[t]}$ is the accumulated parameter deviation from Eq. (10).

By choosing $q(t) = \nabla_{\theta} f(\theta^{[t]})$, the estimator $Q_{-j}^{[t_1, t_2]}(q)$ directly approximates $\Delta f_{-j}^{[t_1, t_2]}$. This mechanism supports diverse downstream tasks through different query instantiations:

Test Loss Influence. Setting $q(t) = \nabla_{\theta} \ell_{\text{test}}(\theta^{[t]})$, we can have $Q_{-j}^{[t_1, t_2]}(q) \approx \underbrace{[\ell_{\text{test}}(\theta_{-j}^{[t_2]}) - \ell_{\text{test}}(\theta_{-j}^{[t_1]})]}_{\text{LOO loss change}} - \underbrace{[\ell_{\text{test}}(\theta^{[t_2]}) - \ell_{\text{test}}(\theta^{[t_1]})]}_{\text{Original loss change}}.$

Prediction Change. $q(t) = \nabla_{\theta} f(x_{\text{test}}; \theta^{[t]})$ estimates how z_j affects the model’s output on a specific test input.

Feature Importance. $q(t) = \nabla_{\theta}(\nabla_x \ell_{\text{test}}(\theta^{[t]}))$ estimates how z_j affects the input gradient saliency map (Pan et al., 2026). Detailed derivations are in Appendix E.

Implementation of WIE. We reduce computation cost using Hessian-vector products (HVPs) (Pearlmutter, 1994), which apply $H^{[t]}$ to any vector v in $\mathcal{O}(|S_t|p)$ time without explicit matrix formation, reducing the propagator step to $P^{[t]}v = v - \eta_t H^{[t]}v$. Based on this, WIE performs a single backward sweep over the training trajectory with two propagation vectors. Measured on BERT-base (110M parameters), one HVP costs about $2.5 \times$ one training step, and a one-epoch window query costs roughly one additional training run (Appendix C.6). Algorithm 1 and a detailed complexity comparison with existing methods are provided in Appendix C.

5 Theoretical Guarantee

We give a worst-case error bound for WIE that holds on non-convex, non-converged models.

Theorem 1 (Window-level error bound). *Assume the per-data Hessian is L_H -Lipschitz and uniformly bounded by M_H , and the LOO parameter deviation is bounded by M . Then for any window $[t_1, t_2]$ and data point z_j , the WIE estimate satisfies*

$$\mathbb{E} \|\Delta \theta_{-j}^{[t_1, t_2]} - \widehat{\Delta} \theta_{-j}^{[t_1, t_2]}\| \leq \frac{\tilde{B}}{M_H} (e^{M_H \eta_{\max} t_2} + e^{M_H \eta_{\max} t_1} - 2), \quad (17)$$

with $\tilde{B} := M_H M / S_{\min} + L_H M^2 / 2$.

Under exponentially decaying step sizes, the bound is uniformly bounded by a constant independent of t_1 and t_2 . Section 6 exploits this for

a $20 \times$ compute saving. Formal assumptions, the corollary, and complete proofs are in Appendix B.

6 Experiments

Static data attribution tells us *which* training data points matter to the final model; WIE tells us *when* they matter during training. This temporal resolution enables finer-grained data debugging, more precise label-noise detection, and adaptive data selection for modern models. We organize experiments around four research questions.

RQ1 (§6.1): Is WIE a faithful approximation of the LOO counterfactual?

RQ2 (§6.2): Do training data exhibit temporal structure, and are these patterns consistent across tasks?

RQ3 (§6.3): Does knowing *when* a data point is influential improve label-noise detection over knowing only *that* it is influential?

RQ4 (§6.4): Can window-level attribution guide adaptive data selection and remain computationally tractable on Transformers?

6.1 RQ1: How Faithful Is WIE to LOO?

We first evaluate whether WIE approximates the test loss change produced by LOO retraining. We use MNIST with a DNN because exact LOO retraining over many seeds and windows is feasible in this setting. Exact LOO on BERT is computationally infeasible, so we validate the estimator in a controlled setting and evaluate NLP utility in RQ2 to RQ4.

We evaluate both global and local analysis. In global analysis, the full 20-epoch trajectory is treated as one window $[0, T]$. In local analysis, we evaluate 20 consecutive single-epoch windows $[e, e + 1]$ and report the average agreement with LOO. We measure Pearson correlation, Spearman correlation, Kendall’s τ , and Jaccard similarity over the top 30% data points. Table 1 summarizes the results.

WIE achieves the closest agreement with LOO (Pearson $r=0.96$), while the considered baselines fall substantially behind ($r \leq 0.75$) (Table 1). In the global setting ($[0, T]$), WIE outperforms the best baseline IF across all four metrics (Pearson: 0.96 vs. 0.75; Jaccard: 0.78 vs. 0.52). **The local (single-epoch window) setting separates the methods even more sharply:** WIE retains $r=0.95$ because its recurrence computes window estimates natively, whereas the baselines

Table 1: Estimation accuracy vs. LOO ground truth on MNIST-DNN (16 seeds, mean $\pm$ std). *Global*: full trajectory $[0, T]$; *Local*: average over 20 single-epoch windows.

	Method	Pearson	Spearman	Kendall	Jaccard
Global	IF	0.75 $\pm$ 0.14	0.70 $\pm$ 0.17	0.52 $\pm$ 0.14	0.52 $\pm$ 0.19
	DVEmb	0.58 $\pm$ 0.12	0.49 $\pm$ 0.29	0.35 $\pm$ 0.21	0.34 $\pm$ 0.20
	LAVA	0.54 $\pm$ 0.13	0.45 $\pm$ 0.27	0.31 $\pm$ 0.19	0.30 $\pm$ 0.18
	WIE	0.96$\pm$0.03	0.94$\pm$0.06	0.83$\pm$0.08	0.78$\pm$0.15
Local	IF	0.70 $\pm$ 0.02	0.65 $\pm$ 0.02	0.48 $\pm$ 0.01	0.50 $\pm$ 0.03
	DVEmb	0.56 $\pm$ 0.02	0.48 $\pm$ 0.05	0.35 $\pm$ 0.03	0.35 $\pm$ 0.04
	LAVA	0.52 $\pm$ 0.03	0.44 $\pm$ 0.06	0.32 $\pm$ 0.04	0.31 $\pm$ 0.05
	WIE	0.95$\pm$0.01	0.93$\pm$0.01	0.81$\pm$0.02	0.77$\pm$0.02

must difference two global estimates, compounding errors under subtraction. This validates WIE as a faithful LOO proxy at arbitrary temporal resolution. Under a shared protocol with absolute-value correlations over 4 seeds, we additionally compared TracIn, Arnoldi (Schioppa et al., 2022), and GEX (Kim et al., 2023). TracIn is nearly tied with WIE in the global setting (0.90 vs. 0.91 absolute Pearson) but falls clearly behind on single-epoch windows (0.69 vs. 0.91), and Arnoldi and GEX trail in both settings. Full tables and protocol details are in Appendix G.3.

6.2 RQ2: Temporal Influence Patterns

If data influence were truly static, a single score per data point would suffice. To test this, we compute the window-level influence of each training data point at every epoch on MNIST-DNN, standardize within each epoch to remove global loss decay, and classify trajectories by OLS trend slope, p -value, and sign-flip count. The resulting taxonomy is robust to these threshold choices: sweeping the significance level, slope-strength cut, and sign-flip ratio over 27 combinations moves the pattern proportions only marginally and never changes their ordering (Appendix G.4).

WIE reveals four qualitatively distinct temporal roles that static estimators collapse into a single scalar (Figure 3): *Early Influencers* shape decision boundaries in early epochs then fade; *Late Bloomers* contribute mainly near convergence, often refining hard data points; *Stable Influencers* contribute consistently throughout, forming the generalization backbone; and *Highly Fluctuating Influencers* oscillate in sign and often indicate unstable or outlying training behavior. **All four patterns appear consistently across 3 architectures (LR, DNN, CNN) and 4 datasets covering tabu-**

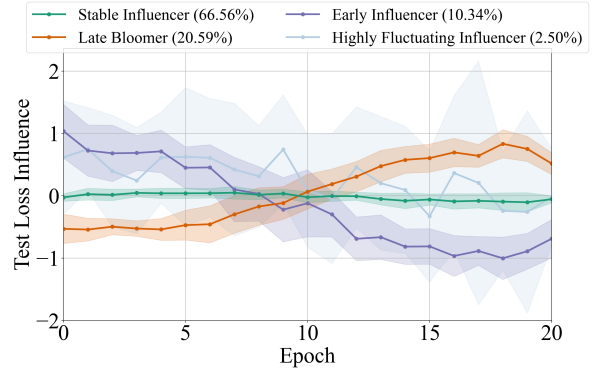


Figure 3: Four temporal influence patterns discovered by WIE on MNIST-DNN. Static estimators collapse all four into a single scalar.

lar, text, and vision modalities (Appendix G.1), suggesting they are properties of the optimization dynamics rather than artifacts of any one task. The Highly Fluctuating pattern is particularly actionable: its strong association with label noise directly motivates the detection experiments in RQ3.

6.3 RQ3: Label-Noise Detection via Temporal Attribution

Knowing *when* a data point becomes harmful should improve noise detection over a single global score. We test this on BERT-base fine-tuned on IMDB with 40% flipped labels, ranking training data points by estimated influence on test loss and measuring precision at each Check Ratio across 9 fine-tuning epochs.

WIE is the only estimator whose detection precision improves steadily over epochs and outperforms all baselines at every Check Ratio (Figure 4). At 20% Check Ratio, WIE rises from 50% precision at epoch 1 to 64% at epoch 9, while DVEmb plateaus around 51%, IF fluctuates near 39%, and LAVA remains below 28%. Only WIE keeps improving because its recurrence tracks how BERT progressively memorizes corrupted data points (Arpit et al., 2017), with gains concentrating in late epochs. In a controlled replication with $n=1024$, where both methods score the same trajectory with the same validation-loss target, WIE also outperforms TracIn, reaching a precision of 56.9% vs. 52.9% at a 10% Check Ratio and 57.8% vs. 52.5% at 20% (base rate 40%).

Moreover, a single last epoch window matches or exceeds full trajectory attribution on the non-convex models while using only one of the twenty analyzed epochs (Table 2). On MNIST 1-vs-7 with 5% to 20% flipped labels, the last-epoch win-

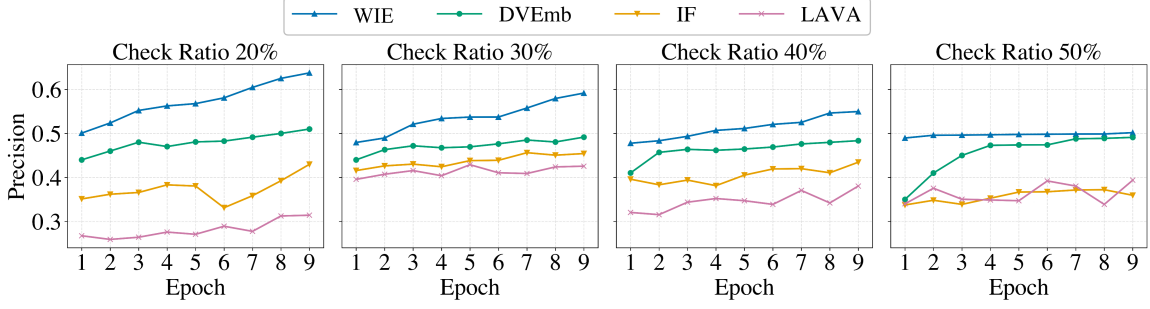


Figure 4: Label-noise detection on BERT-IMDB (40% noise). WIE improves steadily over epochs and outperforms all baselines at every Check Ratio.

Table 2: Corrupted data detected at top- K on MNIST 1-vs-7 (K = number of flipped labels). Last-epoch WIE matches or exceeds full-trajectory WIE and LOO at $20\times$ less compute.

K	Model	LOO	First-epoch	Last-epoch	Full trajectory
12	DNN	8.81	8.25	9.38	9.06
	CNN	10.44	8.75	11.06	10.50
25	DNN	19.94	20.31	21.31	20.75
	CNN	21.75	18.44	23.56	21.81
38	DNN	32.50	32.19	33.31	32.81
	CNN	34.19	29.75	36.31	34.19
51	DNN	43.94	44.13	45.56	45.31
	CNN	46.25	41.50	48.69	46.19

dow wins across all noise levels on non-convex models (DNN, CNN). Clean gradients shrink near convergence while corrupted gradients remain large (Toneva et al., 2018), so late-window sensitivity cleanly separates the two. This is consistent with the theoretical guarantee in §5. Under the decaying learning rate condition, the stability result prevents the estimation error from growing unbounded over long trajectories. The practical consequence is that practitioners can run WIE on a single epoch and still outperform full-trajectory baselines. Full results including the convex LR baseline are in Appendix G.

6.4 RQ4: Adaptive Data Selection and Scalability

Finally, we ask whether temporal attribution can go beyond diagnosis to improve training and whether the method remains tractable on Transformers.

WIE-guided curriculum converges fastest and reaches the lowest final loss on BERT-IMDB (Figure 6). Using the same 40% noise setting as RQ3, we select the top-90% of training data with the most positive window-level influence at each epoch for the next epoch’s training. The WIE-driven curriculum outperforms curricula guided by

IF, DVEmb, and LAVA, as well as training on the full corrupted set. This confirms that the temporal signal is not only diagnostic but actionable: it turns noise detection into a training-time intervention that improves both convergence speed and final model quality.

WIE also scales to ViT, recovering 70% of corrupted data by inspecting up to $7.5\times$ fewer candidates than TracIn (Figure 5). On ViT fine-tuned on EMNIST with 2% to 8% flipped labels, IF and LOO are infeasible, making TracIn the only runnable baseline. At 2% noise, WIE reaches 70% recall at 12.5% inspection while TracIn requires 93.8%. At 4% noise, WIE achieves 70% at 18.8% versus TracIn’s 75.0%. The gap stems from curvature: TracIn discards Hessian information, whereas WIE’s HVP-based recurrence captures the curvature that distinguishes clean from corrupted data points. Combined with the $20\times$ compute saving from last-epoch windows (RQ3), WIE is both more accurate and more efficient than the considered alternatives on Transformer-scale models.

WIE further extends to a 1.4B-parameter model through parameter-efficient fine-tuning. We fine-tune Pythia-1.4B with LoRA (rank 16) on IMDB with 40% flipped labels under the same SGD setting, and run WIE unchanged in the adapter subspace with exact HVPs over all 3.15M trainable adapter parameters. WIE reaches a maximum detection precision of 90.0% at a 20% check ratio, compared with the 40% base rate and 48.9% for ranking by final training loss, while using 9.5 GB peak GPU memory and 4.2 minutes per window query. This demonstrates the feasibility of applying WIE to LoRA fine-tuning of billion-parameter models by operating over all trainable adapter parameters (setup details in Appendix F).

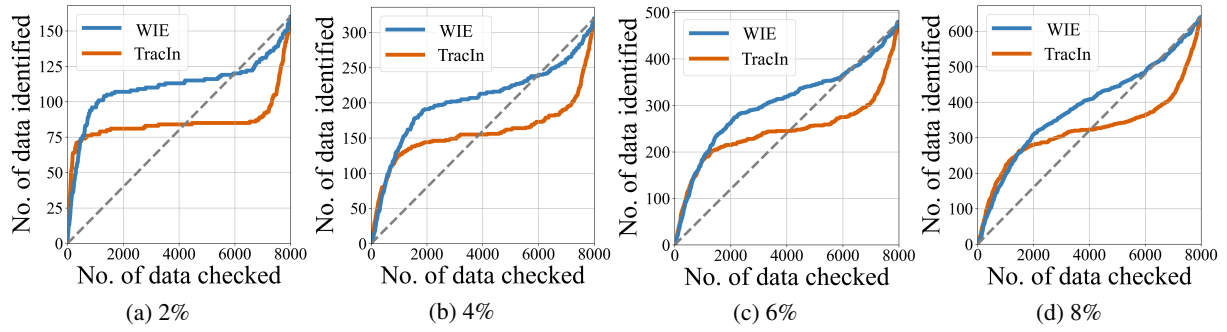


Figure 5: ViT-EMNIST corrupted-data detection. WIE recovers corrupted data with substantially fewer inspections than TracIn across all noise levels.

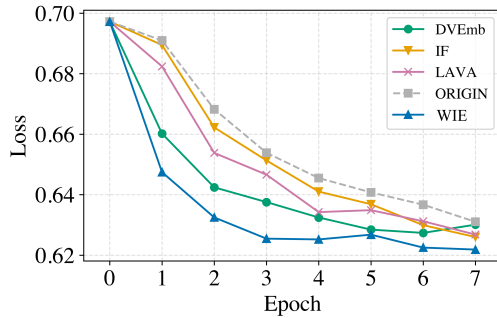


Figure 6: Curriculum learning on BERT-IMDB (40% noise). WIE-guided selection (top-90% positive-influence data per epoch) yields the fastest convergence and the lowest final loss.

7 Conclusion

We propose window-level data attribution and design WIE, a closed-form estimator that uses a simple recurrence over the influence propagator to answer how much training data shaped the model between any two SGD steps. Across datasets and language benchmarks, WIE localizes corrupted data that end-of-training estimators miss, surfaces four qualitatively distinct influence patterns (Early Influencers, Late Bloomers, Stable, and Highly Fluctuating), and drives an adaptive curriculum on BERT-IMDB.

Limitations

Scale of empirical validation. The largest full-parameter model we evaluate is BERT-base (110M parameters). WIE’s per-step cost is two HVPs, the same order as Influence Functions (Koh and Liang, 2017) and TracIn (Pruthi et al., 2020). Our Pythia-1.4B experiment demonstrates the feasibility of applying WIE to LoRA fine-tuning by operating over all trainable adapter parameters, but full-parameter runs on ≥ 1 B-parameter models, instruction-tuning workloads, and broader optimizer settings remain to be evaluated. The per-step cost grows linearly with the number of trainable parameters.

Optimizer assumption. The recurrence is derived for SGD with per-step step size η_t . For Adam-family optimizers, which are the default for transformer fine-tuning, WIE can be applied via two strategies discussed in Appendix H: (1) approximating Adam as coordinate-rescaled SGD with effective per-coordinate step sizes, or (2) computing the recurrence on the observed Adam-trained checkpoint sequence. Both preserve the leave-one-out linearization, but the formal error bound of Theorem 1 no longer applies; a native Adam analysis is left to future work.

Need for trajectory access. WIE requires saved checkpoints, or equivalently, the ability to replay the optimizer. For settings where only the final model is released (e.g., closed-weight APIs), the method is not directly applicable.

Acknowledgments

This work was supported by the Hong Kong Research Grants Council under grant RFS2425-1S01.

References

- Devansh Arpit, Stanislaw Jastrzebski, Nicolas Balas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, and Yoshua Bengio. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 233–242. PMLR.
- Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. 2022. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967.
- S Basu, P Pope, and S Feizi. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48.
- Sang Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, et al. 2026. What is your data worth to gpt? llm-scale data valuation with influence functions. *Advances in Neural Information Processing Systems*, 38:145944–145985.
- Logan Engstrom, Axel Feldmann, and Aleksander Madry. 2024. DsDm: Model-aware dataset selection with datamodels. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12491–12526. PMLR.
- Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR.
- Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*.
- Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2021. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350.
- Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. 2023. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114*.
- Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.
- Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data cleansing for models trained with SGD. *Advances in Neural Information Processing Systems*, 32.
- Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. 2019. Towards efficient data valuation based on the shapley value. In *International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR.
- Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. 2023. Lava: Data valuation without pre-specified learning algorithms. In *The Eleventh International Conference on Learning Representations*.

- SungYub Kim, Kyungsu Kim, and Eunho Yang. 2023. GEX: A flexible method for approximating influence via geometric ensemble. In *Advances in Neural Information Processing Systems*, volume 36, pages 5888–5911.
- Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.
- Yongchan Kwon, Eric Wu, Kevin Wu, and James Y Zou. 2024. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *International Conference on Learning Representations*, volume 2024, pages 21921–21942.
- Jin Hwa Lee, Matthew Smith, Maxwell Adam, and Jesse Hoogland. 2025. Influence dynamics and stagewise data attribution. *arXiv preprint arXiv:2510.12071*.
- Wenbo Pan, Zhichao Liu, Xianlong Wang, Haining Yu, and Xiaohua Jia. 2026. Towards long-horizon interpretability: Efficient and faithful multi-token attribution for reasoning LLMs. In *Proceedings of the 43rd International Conference on Machine Learning*.
- Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR.
- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607.
- Barak A Pearlmutter. 1994. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160.
- Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.
- Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186.
- Megh Thakkar, Tolga Bolukbasi, Sriram Ganapathy, Shikhar Vashishth, Sarath Chandar, and Partha Talukdar. 2023. Self-influence guided data reweighting for language model pre-training. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2033–2045.
- Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.
- Jiachen Tianhao Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. 2025a. Data shapley in one training run. In *International conference on learning representations*, volume 2025, pages 12358–12395.
- Jiachen Tianhao Wang, Dawn Song, James Y Zou, Prateek Mittal, and Ruoxi Jia. 2025b. Capturing the temporal dependence of training data influence. In *International conference on learning representations*, volume 2025, pages 11236–11273.
- Mengzhou Xia, Sathika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 54104–54132.
- Jie Xu, Zihan Wu, Wenbo Pan, Jiao Yin, Yong-Feng Ge, Hua Wang, Cong Wang, and Xiaohua Jia. 2026. LMCleaner: Efficient and certified online unlearning via influence propagation truncation. In *Proceedings of the 43rd International Conference on Machine Learning*.
- Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. 2024. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2150–2168.
- Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. 2025. LiveVal: Real-time and trajectory-based data valuation via adaptive reference points. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*.
- Zichun Yu, Spandan Das, and Chenyan Xiong. 2024. Mates: Model-aware data selection for efficient pre-training with data influence models. *Advances in Neural Information Processing Systems*, 37:108735–108759.

A Derivation of the Window-Level Estimator

This appendix derives the closed-form window estimator (13) from the single-step recurrence (10).

Recall $\mu_{-j}^{[t]} := \theta_{-j}^{[t]} - \theta_{-j}^{[t]}$ and the cumulative propagator $P^{[a,b]} := P^{[b-1]} \dots P^{[a]}$ with $P^{[a,a]} := I$. Since both trajectories start from the same initialization, $\mu_{-j}^{[0]} = 0$. Applying (10) recursively from $t = 0$ yields the cumulative deviation at any step t :

$$\mu_{-j}^{[t]} \approx \sum_{k=0}^{t-1} P^{[k+1,t]} \tilde{\mathbf{1}}_j^{[k]}. \quad (18)$$

For window $[t_1, t_2]$, the parameter influence defined in (6) can be rewritten as

$$\Delta\theta_{-j}^{[t_1,t_2]} = \mu_{-j}^{[t_2]} - \mu_{-j}^{[t_1]}. \quad (19)$$

Using the recurrence property from t_1 to t_2 ,

$$\mu_{-j}^{[t_2]} \approx P^{[t_1,t_2]} \mu_{-j}^{[t_1]} + \sum_{t=t_1}^{t_2-1} P^{[t+1,t_2]} \tilde{\mathbf{1}}_j^{[t]}. \quad (20)$$

Subtracting $\mu_{-j}^{[t_1]}$ from both sides gives

$$\Delta\theta_{-j}^{[t_1,t_2]} \approx \left(P^{[t_1,t_2]} - I\right) \mu_{-j}^{[t_1]} + \sum_{t=t_1}^{t_2-1} P^{[t+1,t_2]} \tilde{\mathbf{1}}_j^{[t]}, \quad (21)$$

which is exactly (13). Substituting (18) for $\mu_{-j}^{[t_1]}$ makes the pre-window contribution explicit.

B Proof of the Error Bound

This appendix formalizes Theorem 1 from §5 and gives a time-independent stability corollary.

B.1 Assumptions

(A1) **Step size and batch control.**

$$0 \leq \eta_t \leq \eta_{\max} \text{ and } |S_t| \geq S_{\min} \geq 1.$$

(A2) **Hessian Lipschitz continuity.**

$$\|\nabla^2 \ell(z_i; \theta_1) - \nabla^2 \ell(z_i; \theta_2)\| \leq L_H \|\theta_1 - \theta_2\|.$$

(A3) **Hessian boundedness.** $\|H^{[t]}\| \leq M_H$ and

$$\|H_j^{[t]}\| \leq M_H \text{ for all } t, j.$$

(A4) **Trajectory deviation bound.** $\|\mu_{-j}^{[t]}\| \leq M$ for all t .

Unlike classical influence functions (Koh and Liang, 2017), which require convexity and convergence to a local minimum, WIE only needs these mild smoothness conditions.

B.2 Main Bound (formal)

Theorem 2 (Estimation error bound). *Under (A1)–(A4), for any window $[t_1, t_2]$ and data point z_j ,*

$$\mathbb{E} \|\Delta\theta_{-j}^{[t_1,t_2]} - \widehat{\Delta\theta}_{-j}^{[t_1,t_2]}\| \leq \frac{\tilde{B}}{M_H} (e^{M_H \eta_{\max} t_2} + e^{M_H \eta_{\max} t_1} - 2), \quad (22)$$

where $\tilde{B} := M_H M / S_{\min} + L_H M^2 / 2$.

Proof. We proceed in three steps: derive an error recurrence, bound the per-step error, and solve the recurrence.

Step 1: Error recurrence. Let $e^{[t]} := \mu_{-j}^{[t]} - \hat{\mu}_{-j}^{[t]}$ denote the gap between the true LOO deviation and the WIE estimate at step t . Comparing the exact LOO update (3) against the WIE recurrence (10), the error changes as

$$e^{[t+1]} = e^{[t]} - \eta_t \rho^{[t]}, \quad (23)$$

where $\rho^{[t]}$ captures the per-step approximation residual. To derive $\rho^{[t]}$, we write the exact LOO gradient average as $\bar{g}_{-j}^{[t]} := \frac{1}{|S_t|} \sum_{i \in S_t \setminus \{j\}} g(z_i; \theta_{-j}^{[t]})$ and the standard gradient average as $\bar{g}^{[t]} := \frac{1}{|S_t|} \sum_{i \in S_t} g(z_i; \theta^{[t]})$. The residual is

$$\begin{aligned} \rho^{[t]} &= (\bar{g}_{-j}^{[t]} - \bar{g}^{[t]}) \\ &\quad - \left(H^{[t]} \hat{\mu}_{-j}^{[t]} - \mathbf{1}_{j \in S_t} \frac{1}{|S_t|} g(z_j; \theta^{[t]}) \right). \end{aligned}$$

Step 2: Per-step bound. For each $i \neq j$, a first-order Taylor expansion at $\theta^{[t]}$ gives

$$g(z_i; \theta_{-j}^{[t]}) - g(z_i; \theta^{[t]}) = H_i^{[t]} \mu_{-j}^{[t]} + r_{i,j}^{[t]} \quad (24)$$

$$\|r_{i,j}^{[t]}\| \leq \frac{L_H}{2} M^2. \quad (25)$$

Averaging over $S_t \setminus \{j\}$ and using the identity

$$\frac{1}{|S_t|} \sum_{i \in S_t \setminus \{j\}} H_i^{[t]} = H^{[t]} - \mathbf{1}_{j \in S_t} \frac{1}{|S_t|} H_j^{[t]}.$$

The indicator-gradient terms cancel, yielding

$$\rho^{[t]} = H^{[t]} e^{[t]} - \mathbf{1}_{j \in S_t} \frac{1}{|S_t|} H_j^{[t]} \mu_{-j}^{[t]} + \bar{r}^{[t]},$$

where $\bar{r}^{[t]} = \frac{1}{|S_t|} \sum_{i \in S_t \setminus \{j\}} r_{i,j}^{[t]}$. Applying (A3) and (A4),

$$\|\rho^{[t]}\| \leq M_H \|e^{[t]}\| + \tilde{B}, \quad \tilde{B} := \frac{M_H M}{S_{\min}} + \frac{L_H}{2} M^2.$$

Step 3: Solving the recurrence. From (23) and the bound on $\|\rho^{[t]}\|$,

$$\|e^{[t+1]}\| \leq (1 + \eta_t M_H) \|e^{[t]}\| + \eta_t \tilde{B}. \quad (26)$$

With $e^{[0]} = 0$, $\eta_t \leq \eta_{\max}$, and $(1 + \eta_{\max} M_H)^m \leq e^{M_H \eta_{\max} m}$,

$$\begin{aligned} \mathbb{E} \|e^{[t]}\| &\leq \tilde{B} \eta_{\max} \sum_{s=0}^{t-1} e^{M_H \eta_{\max} (t-1-s)} \\ &\leq \frac{\tilde{B}}{M_H} (e^{M_H \eta_{\max} t} - 1). \end{aligned} \quad (27)$$

Finally, by the triangle inequality,

$$\begin{aligned} \mathbb{E} \|e^{[t_2]} - e^{[t_1]}\| &\leq \mathbb{E} \|e^{[t_2]}\| + \mathbb{E} \|e^{[t_1]}\| \\ &\leq \frac{\tilde{B}}{M_H} (e^{M_H \eta_{\max} t_2} + e^{M_H \eta_{\max} t_1} - 2). \quad \square \end{aligned}$$

B.3 Time-Independent Stability under Exponential Decay

Corollary 3 (Time-independent stability). *If the step size follows exponential decay $\eta_t = \eta_0 \gamma^t$ with $\eta_0 > 0$ and $0 < \gamma < 1$, then there exists a constant C , independent of t_1, t_2 , such that*

$$\mathbb{E} \|\Delta \theta_{-j}^{[t_1, t_2]} - \widehat{\Delta \theta}_{-j}^{[t_1, t_2]}\| \leq C. \quad (28)$$

Proof. Under exponential decay, $\sum_{m=0}^{\infty} \eta_m = \eta_0 / (1 - \gamma) =: S_\eta < \infty$. Using $1 + x \leq e^x$,

$$\prod_{m=k+1}^{t-1} (1 + \eta_m M_H) \leq \exp \left(M_H \sum_{m=0}^{\infty} \eta_m \right) = e^{M_H S_\eta}. \quad (29)$$

Therefore $\|e^{[t]}\| \leq \tilde{B} e^{M_H S_\eta} S_\eta$ for all t , and $\mathbb{E} \|e^{[t_2]} - e^{[t_1]}\| \leq 2\tilde{B} S_\eta e^{M_H S_\eta} =: C$. $\square$

C Implementation Details of WIE

C.1 Hessian-Vector Products

Equation (13) involves repeated products of $P^{[t]} = I - \eta_t H^{[t]}$, each of which is $p \times p$ and prohibitive to materialize when p is in the millions. We avoid this with Hessian-vector products (HVPs) (Pearlmutter, 1994): for any vector $v \in \mathbb{R}^p$,

$$H^{[t]} v = \frac{1}{|S_t|} \sum_{i \in S_t} \nabla_{\theta} (\nabla_{\theta} \ell(z_i; \theta^{[t]})^\top v), \quad (30)$$

computable in $\mathcal{O}(|S_t|p)$ time by one extra automatic-differentiation pass. The propagator step $P^{[t]} v = v - \eta_t H^{[t]} v$ therefore needs only one HVP.

Algorithm 1 WIE: Window-level Data Attribution

Require: Target data point z_j ; query q ; window $[t_1, t_2]$; saved trajectory $\{\theta^{[t]}, S_t, \eta_t\}$.

Ensure: Estimate $\hat{Q}$ of $Q_{-j}^{[t_1, t_2]}(q)$.

```

1:  $\hat{Q} \leftarrow 0$ ;  $u_1 \leftarrow 0$ ;  $u_2 \leftarrow q(t_2)$ .
2: for  $t = t_2 - 1$  downto 0 do
3:   if  $j \in S_t$  then
4:      $\tilde{\mathbf{1}}_j \leftarrow \frac{\eta_t}{|S_t|} g(z_j; \theta^{[t]})$ 
5:      $\hat{Q} \leftarrow \hat{Q} + \langle u_2 - u_1, \tilde{\mathbf{1}}_j \rangle$ 
6:   end if
7:    $u_2 \leftarrow u_2 - \eta_t H^{[t]} u_2$  // one HVP
8:   if  $t = t_1$  then
9:      $u_1 \leftarrow q(t_1)$ 
10:  else if  $t < t_1$  then
11:     $u_1 \leftarrow u_1 - \eta_t H^{[t]} u_1$  // one HVP
12:  end if
13: end for
14: return  $\hat{Q}$ 

```

C.2 Backward-Sweep Algorithm

Algorithm 1 computes $Q_{-j}^{[t_1, t_2]}(q)$ by sweeping backward through the training steps. It maintains two vectors u_1 and u_2 that propagate the query vectors $q(t_1)$ and $q(t_2)$ backward via $P^{[t]}$: u_2 is initialized at step t_2 and propagated throughout; u_1 remains zero until the sweep reaches t_1 , at which point it is set to $q(t_1)$ and propagated for all $t < t_1$. Whenever z_j appears in the mini-batch, the algorithm accumulates an inner product against the instantaneous influence $\tilde{\mathbf{1}}_j^{[t]}$. The per-step cost is at most two HVPs, independent of the window length; total cost is $\mathcal{O}(t_2 \cdot |S_t| p)$, the same order as one extra training run. A fused single-adjoint variant that lowers the HVP count by a constant factor is given in Appendix C.5. The procedure requires access to the training trajectory $A = \{\theta^{[t]}, S_t, \eta_t\}_{t=0}^{T-1}$, retrievable from storage or efficiently recomputed via gradient checkpoints.

C.3 Correctness of Algorithm 1

We show that Algorithm 1 correctly computes $Q_{-j}^{[t_1, t_2]}(q)$ as defined in Eq. (16).

Self-adjointness of $P^{[t]}$. Since $H^{[t]}$ is symmetric, $P^{[t]} = I - \eta_t H^{[t]}$ is self-adjoint: $\langle x, P^{[t]} y \rangle = \langle P^{[t]} x, y \rangle$.

Update rule for u_1, u_2 . Lines 7 and 11 implement $u_i^{[t-1]} = P^{[t]} u_i^{[t]}$ for $i \in \{1, 2\}$. Recursing,

$$u_i^{[s]} = \left(\prod_{k=s+1}^{t-1} P^{[k]} \right) u_i^{[t]}, \quad s < t. \quad (31)$$

Initialization. The algorithm initializes $u_2 \leftarrow q(t_2)$ before the loop and sets $u_1 \leftarrow q(t_1)$ only when $t = t_1$. Thus, for $t < t_2$,

$$u_2^{[t]} = \prod_{k=t+1}^{t_2-1} P^{[k]} q(t_2), \quad (32)$$

$$u_1^{[t]} = \begin{cases} \prod_{k=t+1}^{t_1-1} P^{[k]} q(t_1) & t < t_1, \\ 0 & t_1 \leq t < t_2. \end{cases} \quad (33)$$

The case $u_1^{[t]} = 0$ for $t \geq t_1$ holds because the backward sweep has not yet executed $u_1 \leftarrow q(t_1)$ when processing those steps.

Summing the contributions. At each step t where $j \in S_t$, the algorithm accumulates $\Delta Q_t = \langle u_2^{[t]} - u_1^{[t]}, \tilde{\mathbf{1}}_j^{[t]} \rangle$. Summing over $0 \leq t < t_2$ and using self-adjointness to move the $P^{[k]}$ products inside the inner product,

$$\begin{aligned} \hat{Q} &= \sum_{t=0}^{t_2-1} \left\langle \prod_{k=t+1}^{t_2-1} P^{[k]} q(t_2), \tilde{\mathbf{1}}_j^{[t]} \right\rangle \\ &\quad - \sum_{t=0}^{t_1-1} \left\langle \prod_{k=t+1}^{t_1-1} P^{[k]} q(t_1), \tilde{\mathbf{1}}_j^{[t]} \right\rangle \\ &= \left\langle q(t_2), \sum_{t=0}^{t_2-1} \prod_{k=t+1}^{t_2-1} P^{[k]} \tilde{\mathbf{1}}_j^{[t]} \right\rangle \\ &\quad - \left\langle q(t_1), \sum_{t=0}^{t_1-1} \prod_{k=t+1}^{t_1-1} P^{[k]} \tilde{\mathbf{1}}_j^{[t]} \right\rangle \\ &= \langle q(t_2), \mu_{-j}^{[t_2]} \rangle - \langle q(t_1), \mu_{-j}^{[t_1]} \rangle \\ &= Q_{-j}^{[t_1, t_2]}(q). \quad \square \end{aligned}$$

C.4 Comparison with Existing Methods

Table 3 provides a comprehensive comparison. While first-order methods like TracIn are computationally cheaper, they fail to capture curvature information. WIE achieves the best trade-off: superior efficiency compared to LOO retraining and Hessian-inversion IF, while maintaining strict robustness to non-convex and non-converged dynamics suitable for large-scale models. In Table 3, the $O(wp)$ storage entry for WIE refers to the working set held in memory during a window sweep; the trajectory records themselves (checkpoints, mini-batch indices, and learning rates) reside on disk and are quantified in Appendix C.6.

C.5 Fused Single-Adjoint Window Recursion

Algorithm 1 maintains two propagation vectors, one per window endpoint, and therefore performs about $t_2 + t_1$ HVPs. The window functional admits an algebraically equivalent implementation that keeps a single active adjoint. Initialize $a \leftarrow q(t_2)$. For $t = t_2 - 1, \dots, t_1$, accumulate $\langle a, \tilde{\mathbf{1}}_j^{[t]} \rangle$ whenever $j \in S_t$ and propagate $a \leftarrow P^{[t]} a$ with one HVP, using the self-adjointness of $P^{[t]}$. After the sweep passes t_1 , set $a \leftarrow a - q(t_1)$ and continue the same accumulation and propagation for $t = t_1 - 1, \dots, 0$. By linearity of the propagation, the adjoint carries $q(t_2)$ propagated from t_2 together with $-q(t_1)$ propagated from t_1 , so the accumulated sum splits into the two endpoint functionals of Appendix C.3 and returns

$$\langle q(t_2), \mu_{-j}^{[t_2]} \rangle - \langle q(t_1), \mu_{-j}^{[t_1]} \rangle = Q_{-j}^{[t_1, t_2]}(q). \quad (34)$$

This variant uses one HVP per visited step, about t_2 HVPs in total rather than $t_2 + t_1$ for two separate endpoint propagations, a constant-factor advantage at the same asymptotic count. All timings reported in this paper use the two-vector implementation of Algorithm 1.

C.6 Measured Compute and Storage Comparison

We compare the influence-scoring phase of all methods under one protocol: same machine, same precision, and identical candidate ($n = 256$) and query sets on one saved per-epoch trajectory, with timing from checkpoint loading to final scores. Table 4 reports MNIST-DNN on CPU, where all entries are measured. Table 5 reports BERT-IMDB end to end (fp32, batch size 8, sequence length 128, $n = 1024$, a 10-epoch trajectory of 1280 steps); the derivation of each entry from measured constants is stated in the caption. Because WIE is intended for short-window analysis, the one-epoch window is its representative operating point, with a single-HVP peak of 3.5 GB and cost comparable to TracIn. In the WIE row, batch composition and learning rates are replayed per step from the saved records while HVPs are evaluated at the stored per-epoch checkpoints.

Measured constants on BERT-base (medians over 25 repetitions) give one HVP at $2.5 \times$ the cost of one training step. WIE accesses the trajectory post hoc from saved checkpoints, batch indices, and learning rates, whereas DVEmb stores per-data-point projected gradients at every training iteration;

Table 3: Comparison of data influence analysis methods.

Aspect	LOO	IF	TracIn	LAVA	DVEmb	WIE
Computation	$O(C_{tr})$	$O(p^3)$	$O(Kp)$	$O(NM\tilde{p})$	$O(T S_t \tilde{p}^2)$	$O(t_2 S_t p)$
Storage	$O(p)$	$O(p^2)$	$O(Kp)$	$O(NM)$	$O(T S_t \tilde{p})$	$O(wp)$
Non-convergence robust	Yes	No	Yes	Yes	Yes	Yes
Non-convexity robust	Yes	No	Yes	Yes	Yes	Yes
LOO approximation	Yes	Yes	No	No	Yes	Yes

C_{tr} : full training cost; p : param. dim.; $\tilde{p}$: projection dim.; K : # checkpoints; M : val. set size; w : window size.

the BERT storage and runtime extrapolation for DVEmb uses its scalable setting of 1024 dimensions per layer. Notably, WIE does not need the full trajectory to perform well: a one-epoch window computes the in-window contribution by sweeping only that epoch’s steps, at about the cost of one additional training run, and matches or exceeds full-trajectory attribution for detection (Table 2). The exact arbitrary-window query that retains the pre-window term does sweep the earlier steps.

D Relationship to DVEmb

This appendix details the relationship between WIE and DVEmb (Wang et al., 2025b) summarized in Section 2.

Matched idealized construction. Both estimators instantiate the trajectory-specific leave-one-out linearization of Hara et al. (2019). Under a matched idealized construction with exact Hessians, no random projection, aggregation over all occurrences of z_j , and endpoint-specific query gradients, the two estimators yield the same first-order window estimate. In particular, if z_j occurs only inside the window and $\mu_{-j}^{[t_1]} = 0$, the in-window term of Eq. (13) equals the sum of per-occurrence unrollings, so WIE reduces to summing DVEmb scores within the window. In the general case $\mu_{-j}^{[t_1]} \neq 0$, WIE carries the pre-window deviation through the window via the $(P^{[t_1, t_2]} - I) \mu_{-j}^{[t_1]}$ term, and Theorem 1 bounds both terms jointly.

Endpoint differencing. DVEmb’s original interface scores one occurrence at a chosen terminal checkpoint, so obtaining the window quantity requires computing at both endpoints and differencing. For a time-varying query $q(t)$, differencing under a single terminal query differs from the window quantity:

$$\begin{aligned} Q_{-j}^{[t_1, t_2]}(q) - \langle q(t_2), \mu_{-j}^{[t_2]} - \mu_{-j}^{[t_1]} \rangle \\ = \langle q(t_2) - q(t_1), \mu_{-j}^{[t_1]} \rangle, \end{aligned} \quad (35)$$

so endpoint differencing must use the endpoint-specific vectors $q(t_1)$ and $q(t_2)$, which WIE exposes directly as a single window query.

Practical configurations. The realizable configurations differ in three respects. (1) *Projection*: DVEmb’s practical recursion operates on randomly projected gradients, whereas WIE propagates exact Hessian-vector products in the full parameter space. (2) *Curvature*: DVEmb’s backward recursion relies on the generalized Gauss–Newton surrogate, whereas WIE uses the exact Hessian of each evaluated checkpoint via automatic differentiation. (3) *Interface and storage*: DVEmb materializes per-occurrence, endpoint-dependent embeddings during training with $O(TB\tilde{p})$ persistent storage, whereas WIE propagates query adjoints at scoring time from saved checkpoints and trajectory records, without a persistent per-occurrence embedding store.

Controlled fidelity comparison. Beyond the protocol of Table 1 (global Pearson 0.96 for WIE vs. 0.58 for DVEmb), we ran a controlled recomputation on the same trajectories and LOO ground truth with one shared metric (absolute Pearson), using WIE’s exact-Hessian implementation and DVEmb’s scalable configuration (GGN plus random projection, $\tilde{p} = p/10$). Table 6 shows the same ordering and a substantial fidelity gap under both protocols.

Query-count trade-off. DVEmb amortizes query-time computation once embeddings for the required endpoints have been constructed and stored: each additional query is then evaluated by inner products. WIE performs query-side sweeps without persistently materializing the per-occurrence embedding family; batched queries share checkpoint I/O while the arithmetic grows with the number of queries. The crossover is workload dependent. WIE targets full-dimensional post-hoc window queries whose endpoint pair can be selected after training among the stored

Table 4: Compute and storage comparison on MNIST-DNN (CPU, all entries measured). E : epochs; n : training size; n_{val} : validation size; K : strided checkpoints.

Method	Checkpoints	HVP / backward	Wall-clock	Storage
WIE	per-epoch (E)	2 HVPs / step	9.9 s	E ckpts + indices + LRs
TracIn	$K \approx 5$ strided	$K(n+n_{\text{val}})$ grads, no HVP	13.9 s	K ckpts
DVEmb	full per-step	projected-gradient recursion	5.0 s	per-point projected grads
IF (Koh & Liang)	1 (final)	iterative HVP (LiSSA/CG)	101 s	1 ckpt

Table 5: End-to-end compute and storage on BERT-IMDB (fp32, batch size 8, sequence length 128, $n = 1024$, 10-epoch trajectory of 1280 steps). The WIE window time is derived as one tenth of a measured sweep over all 1280 steps (910 s, 2560 HVPs, using a 29.8 GB batched gradient cache and 4.4 GB of stored checkpoints); the window query stores only its window checkpoints (0.9 GB), and a single HVP peaks at 3.5 GB. TracIn (5 strided checkpoints) is derived from its measured per-gradient cost; DVEmb scoring is cheap dot products, with training instrumentation adding +75 s (extrapolated).

Method	Online scoring	#HVP / #bwd	Peak mem.	Storage
WIE (one-epoch window)	91 s	256 / 0	3.5 GB	0.9 GB
TracIn	375 s	0 / 6400	2.2 GB	2.2 GB
DVEmb	dot products	0 / 1280	train-time	0.67 GB

Table 6: Controlled fidelity recomputation vs. LOO ground truth on MNIST-DNN (16 seeds, mean $\pm$ std, absolute Pearson).

Estimator	Global	Local
DVEmb (scalable config.)	0.51 $\pm$ 0.28	0.52 $\pm$ 0.21
WIE	0.90$\pm$0.11	0.90$\pm$0.07

trajectory states, whereas DVEmb gains amortized latency when many queries reuse prepared endpoints.

E Extended Toolkit: Prediction and Feature Saliency

The query-vector framework (Def. 2) targets any differentiable functional metric. We give explicit derivations for the three instantiations used in §6.

E.1 Test-loss influence

Theorem 4 (WIE for loss value). *With $q(t) = \nabla_{\theta} \ell(z_{\text{test}}; \theta^{[t]})$,*

$$Q_{-j}^{[t_1, t_2]}(q) \approx [\ell(z_{\text{test}}; \theta_{-j}^{[t_2]}) - \ell(z_{\text{test}}; \theta_{-j}^{[t_1]})] - [\ell(z_{\text{test}}; \theta^{[t_2]}) - \ell(z_{\text{test}}; \theta^{[t_1]})]. \quad (36)$$

Proof. Substitute $q(t)$ into Eq. (16) and apply a first-order Taylor expansion of ℓ around $\theta^{[t_1]}, \theta^{[t_2]}$:

$$\begin{aligned} & \langle \nabla_{\theta} \ell(z_{\text{test}}; \theta^{[t]}), \theta_{-j}^{[t]} - \theta^{[t]} \rangle \\ & \approx \ell(z_{\text{test}}; \theta_{-j}^{[t]}) - \ell(z_{\text{test}}; \theta^{[t]}). \end{aligned}$$

Subtracting the two endpoints yields (36). $\square$

E.2 Prediction-change influence

For a prediction function $f(x_{\text{test}}; \theta)$ (logit or class probability), set $q(t) = \nabla_{\theta} f(x_{\text{test}}; \theta^{[t]})$. By the same Taylor’s argument,

$$Q_{-j}^{[t_1, t_2]}(q) \approx [f(x_{\text{test}}; \theta_{-j}^{[t_2]}) - f(x_{\text{test}}; \theta_{-j}^{[t_1]})] - [f(x_{\text{test}}; \theta^{[t_2]}) - f(x_{\text{test}}; \theta^{[t_1]})], \quad (37)$$

which captures the counterfactual shift in raw model output induced by z_j over $[t_1, t_2]$.

E.3 Feature-saliency influence

For input-gradient saliency $\nabla_x \ell(z_{\text{test}}; \theta)$, set $q(t) = \nabla_{\theta} (\nabla_x \ell(z_{\text{test}}; \theta^{[t]}))$. The same first-order argument gives

$$Q_{-j}^{[t_1, t_2]}(q) \approx [\nabla_x \ell(z_{\text{test}}; \theta_{-j}^{[t_2]}) - \nabla_x \ell(z_{\text{test}}; \theta_{-j}^{[t_1]})] - [\nabla_x \ell(z_{\text{test}}; \theta^{[t_2]}) - \nabla_x \ell(z_{\text{test}}; \theta^{[t_1]})], \quad (38)$$

which estimates whether z_j caused the model to focus more on relevant features or be distracted by background noise.

F Full Experimental Setup

Datasets. *Adult* (tabular, binary income prediction); *20Newsgroups* (text, 20-class topic classification with bag-of-words features); *MNIST* (handwritten digits) and *EMNIST* (handwritten letters); *IMDB* (binary movie-review sentiment). Train/test splits follow the standard splits shipped with each dataset.

Models. Logistic Regression (LR) for convex baselines; a two-layer DNN (256-128, ReLU) for non-convex small-scale experiments; a CNN (two convolutional layers followed by two fully connected layers) for vision tasks; BERT-base (110M parameters) fine-tuned on IMDB; ViT fine-tuned on EMNIST.

Baselines. Baselines include LOO retraining (gold standard where feasible), Influence Functions (IF) (Koh and Liang, 2017), TracIn (Pruthi et al., 2020), DVEmb (Wang et al., 2025b), and LAVA (Just et al., 2023).

Training details. All runs use SGD following the standard data-attribution protocol (Koh and Liang, 2017; Pruthi et al., 2020; Hara et al., 2019). Learning rates are tuned per dataset: 0.01 for Adult and 20News, 0.1 for MNIST and EMNIST, 2×10^{-5} for BERT-IMDB, and 10^{-3} for ViT-EMNIST. All models train for 20 epochs except BERT and ViT. Adam-family extensions are detailed in Appendix H.

WIE configuration. Hessian-vector products are computed via `torch.autograd`. For window queries, we store the full training trajectory (parameters, mini-batch indices, learning rates) on disk; for ViT, we use gradient checkpointing to reduce memory. Query vectors use test-loss gradients ($q(t) = \nabla_{\theta} \ell_{\text{test}}(\theta^{[t]})$) throughout.

Hardware. All experiments use PyTorch 2.1 on a single GPU. Results are reported as mean $\pm$ standard deviation over 16 random seeds.

Parameter scope and window protocol. All BERT-IMDB WIE experiments use the full trainable parameter set of BERT-base (110M parameters), with no last-layer restriction, parameter sub-setting, or random projection. Windows were specified before evaluating label-noise outcomes and were applied identically to all methods.

Pythia-1.4B LoRA experiment. The 1.4B-scale experiment in §6.4 uses frozen-backbone LoRA fine-tuning under the same SGD setting: Pythia-1.4B, LoRA rank 16, and IMDB with 40% flipped labels. WIE runs unchanged in the adapter subspace, with exact HVPs over all 3.15M trainable adapter parameters. The run uses 9.5 GB peak GPU memory and 4.2 minutes per window query on a single GPU.

Evaluation protocol. For estimation fidelity, we report Pearson, Spearman, and Kendall correlations plus Jaccard overlap on the top-30% against LOO retraining. For corrupted-data detection, we rank data by estimated influence on test loss. For BERT-IMDB, we report precision at each Check Ratio (top-k% of the dataset). For ViT-EMNIST, we report the number of corrupted data points identified as a function of the number of inspected data points, and quote the inspection ratio required to reach 70% recall.

G Additional Results

G.1 Distribution of Influence Patterns

Table 7 reports the distribution of the four patterns across all architectures and dataset combinations. Three observations stand out. First, Stable dominates universally (64.8% to 97.9%), confirming that the actionable signal lives in the long tail. Second, CNNs are notably more Stable than DNNs on vision tasks (83.8% vs. 66.6% on MNIST), consistent with convolutional priors regularizing per-data-point contributions. Third, the Highly Fluctuating fraction is systematically larger on EMNIST than MNIST across all architectures (e.g., CNN: 10.0% vs. 4.2%), reflecting EMNIST’s more ambiguous letter boundaries.

Table 7: Distribution (%) of the four influence patterns across architectures and datasets.

Model	Dataset	Stable	Early	Late	Fluct.
LR	Adult	64.8	11.7	20.2	3.4
	20News	85.9	1.2	5.6	7.3
	MNIST	80.2	0.8	10.8	8.3
	EMNIST	75.5	0.7	13.8	10.0
DNN	Adult	97.9	0.3	1.0	0.8
	20News	79.0	8.4	11.4	1.1
	MNIST	66.6	10.3	20.6	2.5
	EMNIST	78.2	7.1	7.5	7.3
CNN	MNIST	83.8	0.3	11.7	4.2
	EMNIST	86.5	1.9	1.6	10.0

G.2 Full Small-Window Ablation Including LR

Table 8 extends Table 2 in the main text by including the convex LR baseline. For LR, the full-trajectory window matches LOO exactly as expected from the convex theory, and the last-epoch advantage disappears because there is no memorization regime in convex models.

Table 8: Corrupted data detected at top- K on MNIST 1-vs-7 (K = flipped count), including LR. Last-epoch advantage is specific to non-convex models.

K	Model	LOO	First	Mid	Last	Full
12	LR	10.94	10.56	10.88	10.88	10.94
	DNN	8.81	8.25	8.88	9.38	9.06
	CNN	10.44	8.75	10.69	11.06	10.50
25	LR	23.50	22.56	23.50	23.38	23.50
	DNN	19.94	20.31	20.50	21.31	20.75
	CNN	21.75	18.44	22.19	23.56	21.81
38	LR	36.06	35.38	35.69	35.13	36.06
	DNN	32.50	32.19	32.56	33.31	32.81
	CNN	34.19	29.75	34.56	36.31	34.19
51	LR	48.69	47.94	46.56	42.94	48.69
	DNN	43.94	44.13	45.19	45.56	45.31
	CNN	46.25	41.50	47.13	48.69	46.19

G.3 Additional Baselines under a Shared Protocol

We compare WIE against TracIn, Arnoldi (Schioppa et al., 2022), and GEX (Kim et al., 2023) on the RQ1 fidelity evaluation, where exact LOO ground truth is available (MNIST-DNN, 4 seeds). Arnoldi uses top- k Ritz pairs from Hessian-vector iteration on the final model. GEX follows its reference construction: a geometric ensemble collected by continued SGD from the final checkpoint (8 snapshots at the run’s own learning rate and batch size), scored by ensemble products of loss deviations. All methods use the same base training runs and are evaluated against the same ground truth. Correlations are reported as absolute values because sign conventions differ across estimators, and Jaccard uses the top 30% by absolute score. Baselines that expose no window mechanism of their own produce one score per data point, which we evaluate against every window. These values therefore follow a different protocol from Table 1 (16 seeds, signed correlations) and are not directly comparable to it.

Table 9 reports the results. TracIn is nearly tied with WIE in the global setting but falls clearly behind on single-epoch windows, and Arnoldi and GEX trail in both settings, most clearly in the local evaluation, because they expose one score per data point rather than a native windowed score over the original training trajectory.

G.4 Sensitivity of the Influence-Pattern Taxonomy

We sweep all three classification thresholds of §6.2: the significance level $p \in [0.01, 0.10]$, the

Table 9: Fidelity vs. LOO ground truth under a shared protocol (MNIST-DNN, 4 seeds, mean $\pm$ std, absolute correlations). *Global*: full trajectory $[0, T]$; *Local*: average over single-epoch windows.

	Method	Pearson	Spearman	Kendall	Jaccard
Global	Arnoldi	0.54 $\pm$ 0.15	0.49 $\pm$ 0.19	0.35 $\pm$ 0.13	0.27 $\pm$ 0.05
	GEX	0.67 $\pm$ 0.23	0.61 $\pm$ 0.29	0.48 $\pm$ 0.25	0.36 $\pm$ 0.16
	TracIn	0.90 $\pm$ 0.05	0.88 $\pm$ 0.05	0.73 $\pm$ 0.08	0.35 $\pm$ 0.08
	WIE	0.91$\pm$0.05	0.89$\pm$0.05	0.74$\pm$0.08	0.37$\pm$0.07
Local	Arnoldi	0.46 $\pm$ 0.15	0.41 $\pm$ 0.16	0.29 $\pm$ 0.12	0.25 $\pm$ 0.02
	GEX	0.60 $\pm$ 0.20	0.53 $\pm$ 0.25	0.42 $\pm$ 0.22	0.29 $\pm$ 0.12
	TracIn	0.69 $\pm$ 0.05	0.72 $\pm$ 0.06	0.61 $\pm$ 0.06	0.32 $\pm$ 0.06
	WIE	0.91$\pm$0.04	0.88$\pm$0.06	0.75$\pm$0.08	0.38$\pm$0.10

standardized slope strength (for single-regressor OLS the squared standardized slope equals R^2 , swept over $[0.60, 0.80]$), and the sign-flip ratio in $[0.40, 0.50]$, on per-epoch window-influence trajectories (MNIST-DNN with a 40% relabel run, $n = 2000$, 27 combinations in total). Table 10 summarizes the resulting pattern proportions.

Two observations follow. First, the taxonomy is essentially insensitive to the p -value and sign-flip thresholds: with the slope-strength cut fixed, sweeping the other two moves all four proportions by at most 0.1 percentage points. Second, the qualitative structure is invariant across all 27 combinations: Stable remains the majority class, and the ordering Late Bloomer $>$ Early Influencer $>$ Highly Fluctuating holds throughout. Only the slope-strength cut materially changes the proportions, monotonically and without disturbing the ordering.

Table 10: Taxonomy threshold sensitivity (proportions in %): each column sweeps one threshold over its stated range with the others fixed ($p \in [.01, .10]$, sign-flip $\in [.40, .50]$, slope $R^2 \in [.60, .80]$).

Pattern	p	sign-flip	slope R^2
Stable	64.7	64.65–64.7	54.6 $\rightarrow$ 82.9
Early	10.5	10.5	16.8 $\rightarrow$ 3.4
Late	24.6	24.6	28.5 $\rightarrow$ 13.6
Fluct.	0.15	0.15–0.25	0.15

H Extending WIE to Non-SGD Optimizers

The WIE recurrence Eq. (10) is derived for vanilla SGD. Modern training recipes use optimizers with momentum and adaptive scaling. This appendix extends WIE to three cases: SGD with momentum (§H.1), Adam (§H.2), and AdamW (§H.3). Let

$\bar{g}^{[t]} := \frac{1}{|S_t|} \sum_{i \in S_t} g(z_i; \theta^{[t]})$ denote the mini-batch gradient mean. The common structure is state augmentation: when the optimizer carries internal state (momentum buffer, second-moment estimate), the LOO deviation changes as a coupled recurrence over $(\mu_{-j}^{[t]}, \Delta m^{[t]})$. Setting all optimizer-specific parameters to zero recovers Eq. (10). We close with a checkpoint-based heuristic for cases where optimizer state is unavailable (§H.4).

H.1 SGD with Momentum

The heavy-ball update is

$$\begin{aligned} m^{[t]} &= \beta_1 m^{[t-1]} + (1 - \beta_1) \bar{g}^{[t]}, \\ \theta^{[t+1]} &= \theta^{[t]} - \eta_t m^{[t]}. \end{aligned} \quad (39)$$

Let $\Delta m^{[t]} := m_{-j}^{[t]} - m^{[t]}$. Applying the same first-order Taylor expansion as in Eq. (8) gives the coupled recurrence:

$$\begin{aligned} \Delta m^{[t]} &= \beta_1 \Delta m^{[t-1]} + (1 - \beta_1) (H^{[t]} \mu_{-j}^{[t]} - \tilde{t}_j^{[t]}), \\ \mu_{-j}^{[t+1]} &= \mu_{-j}^{[t]} - \eta_t \Delta m^{[t]}, \end{aligned} \quad (40)$$

where $\tilde{t}_j^{[t]} := \mathbf{1}_{j \in S_t} \frac{1}{|S_t|} g(z_j; \theta^{[t]})$ is the raw injection without step-size scaling. The window estimate is obtained as in the main paper but with augmented state $(\mu_{-j}^{[t]}, \Delta m^{[t]}) \in \mathbb{R}^{2p}$. Algorithm 1 extends by maintaining a second adjoint vector for momentum, at the cost of two HVPs per step instead of one. For $\beta_1 = 0$ the system reduces to Eq. (10).

H.2 Adam

Adam adds adaptive per-coordinate scaling:

$$\begin{aligned} m^{[t]} &= \beta_1 m^{[t-1]} + (1 - \beta_1) \bar{g}^{[t]}, \\ v^{[t]} &= \beta_2 v^{[t-1]} + (1 - \beta_2) (\bar{g}^{[t]})^2, \\ \theta^{[t+1]} &= \theta^{[t]} - \eta_t \frac{m^{[t]}}{\sqrt{v^{[t]} + \varepsilon}}. \end{aligned} \quad (41)$$

Following the standard frozen second-moment assumption used in the influence-function-for-LLM literature (Grosse et al., 2023; Kwon et al., 2024), we treat $v_{-j}^{[t]} \approx v^{[t]}$, justified because v is a low-pass-filtered statistic far less sensitive to removing one data point than m . Define the per-coordinate effective step size $\eta_{\text{eff},i}^{[t]} := \eta_t / (\sqrt{v_i^{[t]}} + \varepsilon)$. The coupled recurrence becomes:

$$\begin{aligned} \Delta m^{[t]} &= \beta_1 \Delta m^{[t-1]} + (1 - \beta_1) (H^{[t]} \mu_{-j}^{[t]} - \tilde{t}_j^{[t]}), \\ \mu_{-j}^{[t+1]} &= \mu_{-j}^{[t]} - \text{diag}(\eta_{\text{eff}}^{[t]}) \Delta m^{[t]}, \end{aligned} \quad (42)$$

which differs from Eq. (40) only in replacing scalar η_t by per-coordinate $\eta_{\text{eff}}^{[t]}$. Algorithm 1 extends by: (1) storing $v^{[t]}$ alongside the trajectory, (2) replacing each η_t -scaled HVP with an $\eta_{\text{eff}}^{[t]}$ -scaled one (element-wise multiply, negligible cost), and (3) maintaining a second backward vector for momentum. Bias correction ($\hat{m}^{[t]} = m^{[t]} / (1 - \beta_1^t)$) modifies the effective step size by a time-dependent scalar absorbed into $\eta_{\text{eff}}^{[t]}$ without changing the structure. For $\beta_1 = 0$ the propagator reduces to $P_{\text{Adam}}^{[t]} = I - \text{diag}(\eta_{\text{eff}}^{[t]}) H^{[t]}$, recovering the form of the main paper.

H.3 AdamW (Decoupled Weight Decay)

AdamW adds decoupled weight decay $\lambda_{\text{wd}} > 0$:

$$\theta^{[t+1]} = (1 - \eta_t \lambda_{\text{wd}}) \theta^{[t]} - \eta_t \frac{m^{[t]}}{\sqrt{v^{[t]} + \varepsilon}}. \quad (43)$$

Since weight decay acts symmetrically on both trajectories, it contributes a multiplicative factor:

$$\mu_{-j}^{[t+1]} = (1 - \eta_t \lambda_{\text{wd}}) \mu_{-j}^{[t]} - \text{diag}(\eta_{\text{eff}}^{[t]}) \Delta m^{[t]}, \quad (44)$$

with $\Delta m^{[t]}$ unchanged from Eq. (42). This requires no additional HVP cost.

H.4 Checkpoint-Based Heuristic

When only checkpoints $\{\theta^{[t]}\}$ are available without optimizer state, the coupled recurrences cannot be evaluated. A heuristic is to define an effective gradient $\bar{g}_{\text{eff}}^{[t]} := (\theta^{[t]} - \theta^{[t+1]}) / \eta_t$ and apply the SGD recurrence Eq. (10) with $\bar{g}_{\text{eff}}^{[t]}$ in place of $\bar{g}^{[t]}$. This conflates the optimizer’s adaptive direction with a true gradient, so the formal error bound of Theorem 1 no longer holds. When optimizer state is available, the principled coupled recurrences Eq. (40)–(42) should be preferred.