Sketching the Readout of Large Language Models for Scalable Data Attribution and Valuation

Yide Ran[1]* **Jianwen Xie**[4] **Minghui Wang**[2]
W. Jim Zheng[3] **Denghui Zhang**[1] **Chuan Li**[4] **Zhaozhuo Xu**[1]†
[1]Stevens Institute of Technology
[2]Columbia University Mailman School of Public Health
[3]University of Texas Health Science Center at Houston
[4]Lambda, Inc.

Abstract

Data attribution and valuation are critical to understanding data-model synergy in Large Language Models (LLMs), yet existing gradient-based methods suffer from scalability challenges. Inspired by human cognition, where decision making relies on a focused readout of relevant memories rather than replaying all pathways, we introduce **RISE** (**Readout Influence Sketching Estimator**). Instead of computing and indexing gradients across the entire LLM, RISE focuses on influence hotspots at the output layer, where influence signals concentrate, and the gradient admits a decomposed outer-product form. This enables a dual-channel representation combining a lexical residual channel (RH) and a semantic projected-error channel (GH). Applying CountSketch projections to these channels achieves strong compression while maintaining accurate attribution. Across the OLMo (1B–32B) and Pythia (14M–6.9B) families, RISE reduces index storage by up to 112× compared to RapidIn and scales to 32B parameters LLM, where gradient-based baselines such as RapidIn and ZO-Inf become memory-infeasible. We evaluate RISE on two paradigms: (1) retrospective attribution, retrieving influential training examples for specific predictions, and (2) prospective valuation, scoring candidate data utility zero-shot. We validate RISE on three tasks: Howdy backdoor data detection, Finance-Medical domain separation, and Brain Rot high-quality data selection. In a closed-loop Brain Rot study, continued pretraining on RISE-selected data yields consistent downstream improvements. Overall, RISE provides a scalable primitive for influence-driven data attribution and valuation in LLMs. Code is available at github.com/RyanAIResearch/RISE.

1 Introduction

Understanding which training examples, past or future, shape a large language model's behavior is central to interpretability and to the design of efficient continuous training pipelines. Human learning exhibits a useful duality: we can trace influential past experiences and predict the utility of new information. Scalable influence estimation for LLMs should support the same duality, enabling both **retrospective attribution** and **prospective valuation**. While classic influence functions [1, 2, 3, 4] offer a principled formulation, they become intractable at LLM scale due to second-order computation and the enormous parameter dimensionality.

Scaling influence estimation for LLMs faces a fundamental tension between **feasibility** and **fidelity**. First-order surrogates, including TracIn [5], that rely on per-example parameter-gradient material-

*Email: yran1@stevens.edu.
†Corresponding author: zxu79@stevens.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

ization require computing and storing $O(P)$ gradient values per example, making it infeasible to index large corpora. Proxy-based approximations [4, 6] circumvent this cost by training smaller auxiliary models, but their gradients may be misaligned with the target model's training dynamics due to architectural and loss-landscape differences, leading to unreliable attribution.

This tension raises a natural research question:

Can we achieve scalable data attribution and valuation for LLMs using only forward passes and partial parameters?

We answer this question affirmatively. Through systematic measurements of LLM training dynamics, we discover that the **readout layer** (LM head) is precisely the component we seek. Influence-relevant gradients naturally concentrate in output-side readout components: the **ratio** of LM-head gradient energy to the average transformer-layer gradient energy grows significantly with scale (up to $27.8\times$ on OLMo-32B; Figure 2). We also observe a scale-dependent *U-shaped* discriminativeness pattern: middle-layer hidden-state similarity collapses but recovers at the final layer for practical-scale models (Appendix Figure 5). At unit temperature, the LM-head gradient admits an exact outer-product factorization: $\nabla_{W_{\text{lm_head}}} \ell_t = r_t h_t^\top$, where both factors can be computed from forward-pass quantities alone, eliminating the need for full-model backpropagation. Moreover, although r_t is vocabulary-sized, its influence-relevant signal is highly concentrated: a tiny active-token support preserves most residual energy and semantic error direction.

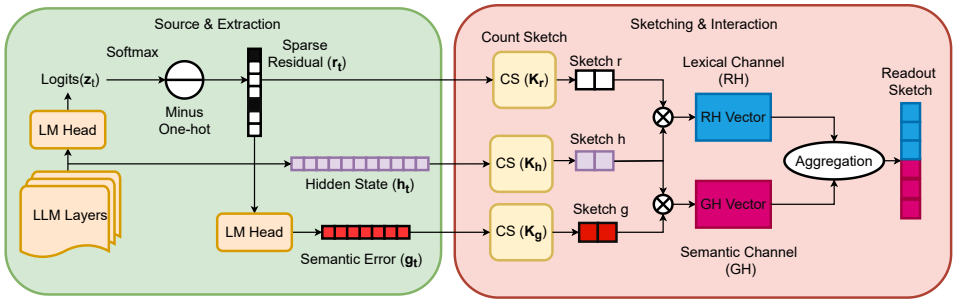

Figure 1: *RISE*: Readout Influence Sketching Estimator.

Building on this insight, we introduce **RISE (Readout Influence Sketching Estimator)**, which enables efficient forward-only influence estimation through CountSketch compression of the outer-product structure. *RISE* exploits this structure through a **dual-channel** formulation: a lexical residual channel (**RH**) for token-level precision and a semantic projected-error channel (**GH**) for robust semantic matching. By compressing these factors with CountSketch projections [7], *RISE* reduces index storage by 28 to $112\times$ compared to RapidIn on Pythia-1B and scales to 32B parameters, where per-example gradient indexing baselines such as RapidIn and ZO-Inf become memory-infeasible (Table 2). For-Value [8] shares the forward-only setting and RH interaction, while *RISE* combines GH, per-position adaptive support, and fixed-width CountSketch indexing for reusable attribution and valuation (Appendix F.6). We validate *RISE* across Pythia (14M to 6.9B) and OLMo (1B to 32B) on Howdy backdoor detection, Finance-Medical domain separation, and Brain Rot data selection. In a closed-loop Brain Rot study, training on RISE-selected data yields substantially improved downstream outcomes, achieving $\mathbf{2.8}\times$ lower perplexity on control data and $\mathbf{1.6}\times$ higher **RULER-CWE** accuracy compared to TrackStar(Table 4). We summarize our contributions as follows.

- **Discovery of readout influence hotspots.** We empirically localize influence-relevant signal concentration to output-side components and provide theoretical justification for this depth-dependent energy shift.
- **Dual-channel readout influence.** We design a readout-side influence metric that combines RH, a vocabulary-space residual channel for lexical precision, with GH, an LM-head-projected semantic error channel for robust paraphrase-aware matching.
- **Sparse active-token structure in LM-head residuals.** We show that influence-relevant residual energy is sharply concentrated on a tiny adaptive support even when probability mass remains diffuse. Exploiting this structure preserves both RH accuracy and GH semantic fidelity while reducing residual-side computation from vocabulary scale to active-support scale.
- **Unified attribution and valuation.** A single estimator supports both retrospective attribution and prospective valuation without retraining proxies or storing large parameter gradients.

2 Readout Sketching of LLMs for Scalable Influence Estimation

We develop *RISE* (Readout Influence Sketching Estimator), a scalable data influence estimation method for LLMs. Our approach is motivated by systematic observations of gradient structure in LLMs, which show that influence signals concentrate in specific outer product channels that can be approximated efficiently.

2.1 Influence Kernel: The Scalability Challenge

Influence kernel [1] estimates how much a training example affects model predictions. For a loss function ℓ and model parameters θ, the influence of a training example x_i on a query/test example x_q is: $\mathcal{I}(x_i \to x_q) = \nabla_\theta \ell(x_q)^\top H_\theta^{-1} \nabla_\theta \ell(x_i)$, where $H_\theta = \nabla_\theta^2 \ell$ is the Hessian of the training loss with respect to θ. A common first-order surrogate (used by TracIn [5]) replaces H_θ^{-1} with a scaled identity, yielding a gradient inner product: $\mathcal{I}(x_i \to x_q) \approx \langle \nabla_\theta \ell(x_q), \nabla_\theta \ell(x_i) \rangle$.

Computing the influence kernel, even using first-order surrogates, is infeasible for LLMs. For an LLM with P parameters, storing per-example gradients requires $O(P)$ memory. Even for moderately sized LLMs with approximately 7 billion parameters ($P \approx 7 \times 10^9$), storing FP16 gradients for 10,000 training examples would require roughly **140 TB** of memory, making this approach infeasible.

Our roadmap. To overcome this scalability challenge, we avoid materializing full first-order gradients across all LLM parameters. The method is grounded in three observations. First, influence-relevant gradient structure localizes at the LM-head readout, where the gradient admits an exact residual–hidden outer-product form (Observation 2.2). Second, the residual side benefits from a dual-channel lexical/semantic decomposition (Observation 2.3). Third, the LM-head residual itself is supported by a tiny set of active prediction tokens, enabling sparse residual and sparse GH construction without losing the influence-relevant error signal (Observation 2.4).

2.2 Observation 1: Influence Signals Localize at the Output Readout

We first examine where gradient energy distributes across model layers. For each layer c (embedding, transformer layers, LM head), we compute the relative energy: $E_c = \frac{\|\nabla_{W_c} \ell\|_F^2}{\sum_c \|\nabla_{W_c} \ell\|_F^2}$

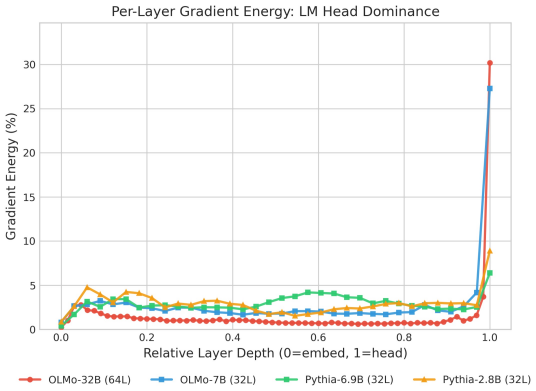

Figure 2: Per-layer gradient energy on C4. The LM head forms a sharp peak that strengthens with model scale, while internal transformer layers each contribute only a few percent.

We conduct controlled analyses on the C4 corpus [9] using open LLMs spanning two architecture families. We further compare the same model before and after fine-tuning on downstream task data. Figures 2, 4 reveal two key patterns: **(1) LM-Head energy hotspot grows with scale.** On C4, gradient energy concentrates sharply at the LM head across model scales (Figure 2), forming a dominant peak while internal transformer layers each contribute only a few percent. **(2) Stability under downstream fine-tuning.** Comparing each pretrained checkpoint with its task-fine-tuned checkpoint on BrainRot, Howdy, and Finance–Medical, the per-layer gradient-energy profiles nearly overlap (Figure 4): the layer-wise trend is largely preserved and the LM-head share changes only mildly. This invariance motivates using pretrained checkpoints to identify LM-Head influence hotspots and supports a single estimator for both prospective valuation and retrospective attribution.

Theoretical analysis. To quantify the LM-head dominance in Figures 2 and 4, we compare expected head and internal-layer energies via the head-to-average ratio $\mathbb{E}[G_{\text{head}}] / \left(\frac{1}{L} \sum_{l=1}^{L} \mathbb{E}[G_l]\right)$. This ratio measures how strongly influence signals concentrate at the LM Head relative to the model body. Appendix B.3 shows that under Assumptions B.1–B.3, including parameter-Jacobian bounds and geometric backpropagation contraction in expectation, the head-to-average expected-energy ratio

admits the following lower bound:

$$\frac{\mathbb{E}[G_{\text{head}}]}{\frac{1}{L}\sum_{l=1}^{L}\mathbb{E}[G_l]} \geq \frac{C_h}{C_x} \cdot \frac{L(1-\kappa^2)}{\|W_{\text{lm_head}}\|_{op}^2(1-\kappa^{2L})}.$$

With C_h, C_x, κ, and $\|W_{\text{lm_head}}\|_{op}$ fixed, this bound grows with model depth, while still allowing occasional internal-layer hotspots.

Readout factorization enables a forward-only readout surrogate. Energy concentration alone does not guarantee useful retrieval: influence-based retrieval is driven by pairwise similarity between example-specific gradients, not by magnitude alone. The LM head is attractive because, in addition to concentrating gradient energy, its gradient contribution at each prediction position admits an exact factorization whose two factors are both available from forward-pass quantities. As formally derived in Lemma B.1 (Appendix B), for the token-level loss at position t, $\nabla_{W_{\text{lm_head}}}\ell(z_t, y_t) = \tau^{-1} r_t \otimes h_t$, where $h_t \in \mathbb{R}^d$ is the final hidden state and $r_t = p_t - \mathbf{1}_{y_t} \in \mathbb{R}^V$ is the prediction residual, with $p_t = \text{softmax}(z_t/\tau)$. Crucially, h_t is the model's last-layer output and r_t is determined by the softmax distribution and the known label, so this readout-side gradient contribution can be formed without per-example backpropagation through the full model. For a full sequence, the LM-head gradient is obtained by summing these token-level contributions across positions: $\nabla_{W_{\text{lm_head}}}\ell(x) = \tau^{-1}\sum_{t=1}^{T} r_t(x) \otimes h_t(x)$. At each token position, the corresponding LM-head gradient inner product factorizes as

$$\begin{aligned}&\left\langle \nabla_{W_{\text{lm_head}}}\ell(z_t(x_q), y_t(x_q)), \nabla_{W_{\text{lm_head}}}\ell(z_t(x_i), y_t(x_i))\right\rangle_F \\ &\quad = \tau^{-2}\underbrace{\langle r_t(x_q), r_t(x_i)\rangle}_{\text{residual sim}} \cdot \underbrace{\langle h_t(x_q), h_t(x_i)\rangle}_{\text{hidden sim}}.\end{aligned}$$

This per-position identity exposes two requirements for scalable retrieval: the hidden factor must remain discriminative across candidates, and the residual factor must be represented in a way that is both scalable and semantically meaningful.

Final hidden states remain discriminative. The factorization above is only useful if the hidden-side similarity $\langle h_t(x_q), h_t(x_i)\rangle$ varies across candidates rather than collapsing to a near-constant scale. We therefore measure discriminativeness as the variance of pairwise cosine similarities over sampled token representations:

$$\text{Discriminativeness}(h) = \text{Var}_{i\neq j}\left[\frac{\langle h^{(i)}, h^{(j)}\rangle}{\|h^{(i)}\|_2\|h^{(j)}\|_2}\right].$$

Concretely, we sample $N = 1000$ token-level training points, compute the $\binom{N}{2}$ pairwise cosine similarities, and report their variance. Appendix Figures 5 and 6 show a consistent U-shape: pairwise cosine variance drops through middle layers, then recovers near the final layer across Pythia and OLMo, and the same pattern persists after downstream fine-tuning. Observation 2.2 therefore justifies restricting the gradient inner-product surrogate to the LM head, $\mathcal{I}(x_i \to x_q) \approx \left\langle \nabla_{W_{\text{lm_head}}}\ell(x_q), \nabla_{W_{\text{lm_head}}}\ell(x_i)\right\rangle_F$, leaving the high-dimensional residual factor as the remaining question addressed in Observation 2.3.

2.3 Observation 2: Dual-Channel Readout Error Decomposition

Observation 2.2 established that LM-head similarity factorizes into hidden and residual terms and that the hidden factor remains discriminative; we now address the residual factor. The residual vector $r_t = p_t - \mathbf{1}_{y_t} \in \mathbb{R}^V$ lives in the vocabulary basis, so the inner product $\langle r_t(x_q), r_t(x_i)\rangle$ is often driven by the overlap of probability mass on the same tokens and does not inherently treat synonyms as similar under a plain dot product. To obtain an error signal that captures both lexical and semantic mismatches, we propose decomposing the error term into two complementary channels:

RH Channel ($r_t \otimes h_t$). The direct outer product using the vocabulary-space residual r_t. This channel tends to favor *token-level* prediction errors, identifying distinct mismatches in the output distribution.

GH Channel ($g_t \otimes h_t$). A projection of the residual into the embedding space via $g_t = W_{\text{lm_head}}^{\top} r_t \in \mathbb{R}^d$, where $W_{\text{lm_head}} \in \mathbb{R}^{V\times d}$ is the LM head weight matrix and $W_{\text{lm_head},v} \in \mathbb{R}^d$ denotes its v-th row. Expanding this term reveals its semantic nature:

$$g_t = W_{\text{lm_head}}^{\top}(p_t - \mathbf{1}_{y_t}) = \mathbb{E}_{v\sim p_t}[W_{\text{lm_head},v}] - W_{\text{lm_head},y_t}.$$

Here, g_t represents the error direction in embedding space: the vector difference between the model's expected token embedding and the ground truth embedding.

Embedding choice. We use the LM head weights $W_{\text{lm_head}}$ for the projection $g_t = W_{\text{lm_head}}^{\top} r_t$, as it directly defines the prediction geometry. While input embeddings can also be used, our experiments default to $W_{\text{lm_head}}$.

Controlled experiment. As shown in Table 1, GH corrects RH's over-reliance on lexical overlap. Specifically, GH rescues the hard paraphrase (Row 6) from $\text{rank}_{\text{RH}} = 6$ to $\text{rank}_{\text{GH}} = 4$ via embedding-space alignment, while successfully demoting spurious matches that share high token overlap but distinct meanings (Rows 4–5). This confirms GH's ability to prioritize semantic alignment over mere token identity.

Table 1: RH is misled by lexical overlap; GH corrects the ranking. Computed with Pythia-410M; lower rank is better. Query (x_q): "The cat is sleeping on the blanket." Sorted by RH rank. RH ranks lexical-only (red) above hard paraphrase (yellow). GH: ↑ rescues semantic, ↓ demotes spurious.

Candidate (x_i)	Sem	Lex	rank_{RH}	rank_{GH}	Δ
Cat sleeps on blanket	✓	✓	1	2	
Kitten napping on throw	✓		2	1	
Tabby dozed on quilt	✓		3	3	
Sleepy cat, blanket ad		✓	4	5	↓
Cat food, blanket orders		✓	5	6	↓
Feline dozing on coverlet	✓		6	4	↑
Stock market crashed			7	7	

Mechanism of complementarity. The RH channel emphasizes *lexical precision* and tends to prioritize token-level agreement. The GH channel trades off lexical precision for *semantic robustness*. In particular, the GH error-side similarity corresponds to residual matching under an embedding-induced kernel (Eq. 15 in Appendix B.4):

$$\langle g_t(x_q), g_t(x_i)\rangle = r_t(x_q)^{\top}(W_{\text{lm_head}}W_{\text{lm_head}}^{\top})\, r_t(x_i).$$

By combining them as $\lambda_{rh} \cdot \text{RH} + \lambda_{gh} \cdot \text{GH}$ before sample normalization, we obtain a metric that tends to prioritize exact matches, retrieve paraphrases, and help suppress spurious lexical overlaps.

Ablation Study. Beyond this controlled example, we validate the complementarity of RH and GH via ablations across model scales. Appendix F.5 (Tables 20–21) shows that GH generally outperforms RH on semantic-heavy tasks, while RH+GH is robustly best or competitive across Top-K. This dual-channel decomposition alleviates the brittleness of the error term: RH preserves lexical precision, while GH injects embedding-space semantic smoothing. Together with the readout locality and final-layer choice in Observation 2.2, this gives *RISE* a gradient similarity metric that is discriminative and semantically robust. The remaining scalability issue is that both RH and GH are still induced by a vocabulary-sized residual r_t.

2.4 Observation 3: Sparse Active Tokens in LM-Head Residuals

Although the RH/GH dual-channel combination improves semantic robustness, both channels remain derived from the vocabulary-sized residual $r_t \in \mathbb{R}^V$, incurring $\mathcal{O}(Vd)$ cost per token. We now show that the influence-relevant signal concentrates on a tiny token support, enabling orders-of-magnitude compression as a structural property rather than an engineering heuristic.

Residual energy decouples from probability mass. While softmax probability mass diffuses across the long tail, the squared residual energy—which directly drives the RH inner product—is highly concentrated. With fixed $\tau = 1.0$ and $K = 128$, the sparse support preserves 95.9% of non-ground-truth residual-tail energy and 99.6% of full residual energy while covering only 0.239% of the vocabulary (Figure 3, left). The gap against probability mass (85.6%) arises because long-tail tokens carry non-negligible probability but negligible squared prediction error.

The ground-truth token anchors the residual. The ground-truth token y_t is always included in $\mathcal{S}_t$ regardless of its prediction rank, ensuring the supervision-bearing coordinate of r_t—typically the largest in magnitude—is never discarded. This is especially important for prospective valuation, where y_t may fall outside the model's top-K predictions on unseen candidate data.

Sparse truncation preserves the GH semantic direction. Because $g_t = W_{\text{lm_head}}^{\top} r_t$ is a linear projection, truncating r_t could distort the semantic error direction. However, at fixed $\tau = 1.0$ and $K = 128$, sparse GH achieves mean cosine similarity 0.993 against dense GH across Pythia

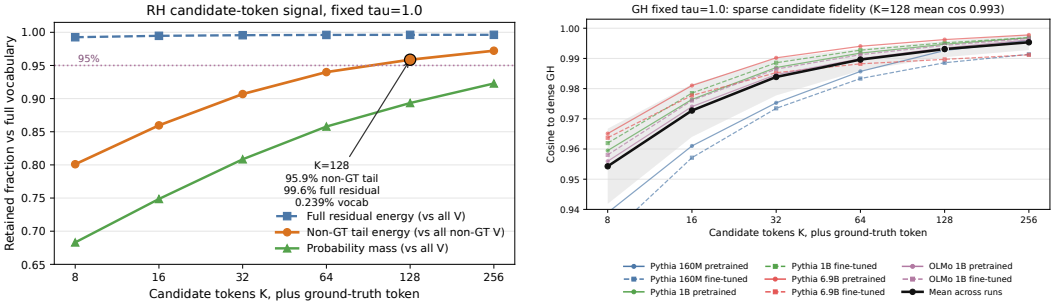

Figure 3: **Sparse active tokens in LM-head residuals.** At fixed $\tau = 1.0$, Top-K+GT preserves RH residual energy (left) and yields sparse GH directions close to dense GH (right). Additional fixed-control diagnostics are in Appendix F.3.

(160M–6.9B) and OLMo pretrained/fine-tuned runs (Figure 3, right), confirming that this structure is not model- or task-specific. Additional diagnostics are in Appendix F.3.

Sparse token selection. Based on these observations, for each position t we define:

$$\mathcal{S}_t = \mathrm{TopK}(z_t/\tau, K) \cup \{y_t\}.$$

Renormalizing the logits over $\mathcal{S}_t$ yields $\tilde{p}_t$; the sparse residual is $\tilde{r}_t(v) = \tilde{p}_t(v) - \mathbf{1}[v = y_t]$ for $v \in \mathcal{S}_t$. Both channels are then computed on this shared support without materializing the full vocabulary:

$$\widetilde{\mathrm{RH}}_t = \tilde{r}_t \otimes h_t, \qquad \widetilde{\mathrm{GH}}_t = \tilde{g}_t \otimes h_t, \qquad \tilde{g}_t = \sum_{v \in \mathcal{S}_t} \tilde{p}_t(v) W_{\text{lm_head},v} - W_{\text{lm_head},y_t}.$$

Together, the concentrated tail energy, the preserved GT coordinate, and the stable GH direction support sparse truncation, which reduces residual-side cost from $\mathcal{O}(V)$ to $\mathcal{O}(|\mathcal{S}_t|)$ while retaining high fidelity in the evaluated settings.

2.5 *RISE*: Scalable Influence via Factorized Sketching

Factorized Compression. Storing the exact token-level LM-head gradient $\nabla_{W_{\text{lm_head}}} \ell(z_t, y_t) = \tau^{-1} r_t \otimes h_t$ is infeasible at scale, as it requires $\mathcal{O}(Vd)$ memory per token. *RISE* circumvents this by compressing the factors $(\tilde{r}_t, h_t)$ and the semantic error $\tilde{g}_t = W_{\text{lm_head}}^{\top} \tilde{r}_t$ individually via CountSketch [7, 10] before forming interaction features. Unlike PCA, CountSketch is data-independent (determined solely by bucket hashes η_r, η_h, η_g and sign hashes s_r, s_h, s_g), which preserves index–query consistency without retraining (see Appendix B.5.3). In practice, CountSketch provides a lightweight projection that approximately preserves inner products; we additionally apply ℓ_2 normalization to sketched factors for numerical stability in retrieval.

Algorithm 1 *RISE*: Readout Influence Sketching Estimator

Require: Training pool $\mathcal{D} = \{x_i\}$; query set $\mathcal{Q} = \{x_q\}$; $W_{\text{lm_head}}$; sketch dims (K_r, K_h, K_g); feature amplitudes (a_{rh}, a_{gh}); temperature τ; ρ_{cum}; top-K cap $K_{\max}$.

Fix CountSketch hashes (bucket $\eta_\cdot$, sign $s_\cdot$) and instantiate $\mathrm{CS}_r, \mathrm{CS}_h, \mathrm{CS}_g$.
Stage 1: Offline index construction
for each $x_i \in \mathcal{D}$ **do**
  $\phi(x_i) \leftarrow \textsc{SketchAggregate}(x_i)$ (Alg. 2); store in index.
end for
Stage 2: Online query feature construction
for each $x_q \in \mathcal{Q}$ **do**
  $\phi(x_q) \leftarrow \textsc{SketchAggregate}(x_q)$ (Alg. 2).
end for
Stage 3: Influence scoring and aggregation
$\bar{\phi}_{\mathcal{Q}} \leftarrow \frac{1}{|\mathcal{Q}|} \sum_{x_q \in \mathcal{Q}} \phi(x_q)$
for each $x_i \in \mathcal{D}$ **do**
  $\mathrm{score}(x_i) \leftarrow \phi(x_i)^{\top} \bar{\phi}_{\mathcal{Q}}$
end for
return training examples ranked by $\mathrm{score}(x_i)$.

Adaptive sparse construction. To avoid sketching a dense vocabulary residual, for each token position t we first form a capped candidate set $\mathcal{C}_t = \mathrm{TopK}(z_t/\tau, K_{\max})$ and ensure $y_t \in \mathcal{C}_t$. We then keep the smallest prefix whose cumulative probability exceeds ρ_{cum}, unioned with $\{y_t\}$, to obtain the final sparse support $\mathcal{S}_t \subseteq \mathcal{C}_t$. Renormalizing on $\mathcal{S}_t$ gives $\tilde{p}_t$, and we define $\tilde{r}_t(v) = \tilde{p}_t(v) - \mathbf{1}[v = y_t]$ for $v \in \mathcal{S}_t$, $\hat{r}_t = \mathrm{CS}_r(\tilde{r}_t)/\|\mathrm{CS}_r(\tilde{r}_t)\|_2$, and $\tilde{g}_t = \sum_{v \in \mathcal{S}_t} \tilde{p}_t(v)\, W_{\text{lm_head}\,v} - W_{\text{lm_head}\,y_t}$. This makes the residual-side cost scale with $|\mathcal{S}_t|$ rather than V.

Structured Interaction Features. We sketch and normalize the hidden state as $\hat{h}_t = \mathrm{CS}_h(h_t)/\|\mathrm{CS}_h(h_t)\|_2$, and similarly obtain $\hat{g}_t = \mathrm{CS}_g(\tilde{g}_t)/\|\mathrm{CS}_g(\tilde{g}_t)\|_2$. We aggregate the outer

products of these sketches over token positions $t = 1, \ldots, T$ to form the channel signatures:

$$\phi_{RH}(x) = \sum_{t=1}^{T} \mathrm{vec}(\hat{r}_t \otimes \hat{h}_t), \quad \phi_{GH}(x) = \sum_{t=1}^{T} \mathrm{vec}(\hat{g}_t \otimes \hat{h}_t). \tag{1}$$

Here $\phi_{RH}(x) \in \mathbb{R}^{K_r K_h}$ and $\phi_{GH}(x) \in \mathbb{R}^{K_g K_h}$. With sample normalization enabled, the final signature is

$$\phi(x) = N_\epsilon([a_{rh}\phi_{RH}(x); a_{gh}\phi_{GH}(x)]), \qquad N_\epsilon(v) = \frac{v}{\max\{\|v\|_2, \epsilon\}},$$

where $\epsilon > 0$ is a norm floor and the feature amplitudes satisfy $\lambda_{rh} = a_{rh}^2$ and $\lambda_{gh} = a_{gh}^2$ (Algorithm 2). The final influence score between a training sample x_i and a query x_q is computed as:

$$\mathcal{I}(x_i \to x_q) = \phi(x_i)^\top \phi(x_q). \tag{2}$$

The complete procedure is detailed in Algorithm 1.

Sketching Variance Bound. Appendix D bounds the sketching variance of the unnormalized token-pair RH and GH estimators (Theorem D.1). These bounds apply before factor and sample normalization and do not establish a universal variance advantage over TracIn under a matched sketch budget.

The analysis distinguishes three effects: (i) **LM-head restriction:** focusing on $\nabla_{W_{\mathrm{lm_head}}} \ell$ changes the estimation target from the full-gradient kernel to the readout kernel; (ii) **sparse residual truncation:** replacing r_t by $\tilde{r}_t$ (supported on $\mathcal{S}_t$) introduces truncation bias and changes the factor norms and similarities entering the variance bounds; (iii) **factorized sketching:** independently choosing (K_r, K_h, K_g) enables flexible allocation of sketch dimensions across the RH and GH channels.

Complexity Analysis. Let $\mathcal{D}$ be the candidate pool, $\mathcal{Q}$ the query set, and T the number of token positions. Let $\bar{S} := \mathbb{E}[S_t]$ denote the average truncated support size ($\bar{S} \ll V$). The costs below exclude the LM forward pass and support selection; constructing $\tilde{g}_t$ costs $O(\bar{S}d)$ per token. **Time.** Index construction computes and aggregates $\phi_{RH}(x), \phi_{GH}(x)$ for each $x \in \mathcal{D}$, with per-example cost $O\big(T\big(\bar{S}d + d + K_h(K_r + K_g)\big)\big)$, hence total offline cost $O\big(|\mathcal{D}|\, T\big(\bar{S}d + d + K_h(K_r + K_g)\big)\big)$. At query time, feature construction for each $x \in \mathcal{Q}$ matches this cost, and exhaustive scoring costs $O(|\mathcal{D}|\,|\mathcal{Q}|\,K_h(K_r + K_g))$. **Memory.** Each candidate $x \in \mathcal{D}$ stores $\phi_{RH}(x) \in \mathbb{R}^{K_r K_h}$ and $\phi_{GH}(x) \in \mathbb{R}^{K_g K_h}$, so the index costs $O(|\mathcal{D}|\,K_h\,(K_r + K_g))$. With $K_r = K_h = K_g = 256$, this is $2 \cdot 256^2 = 131{,}072$ floats ≈ 0.5 MB/example (FP32). In contrast, a dense LM-head gradient stores Vd values (e.g., $Vd \approx 1.3 \times 10^8$ for a 7B-scale model), i.e., $\sim 10^3\times$ larger in FP32; full-parameter gradients are larger still.

3 Experiments

In this section, we demonstrate three key properties of *RISE*: (1) **Duality effectiveness**: It performs effectively on both *Retrospective Attribution* and *Prospective Valuation*, maintaining robustness across model scales. (2) **Scalability and Efficiency**: *RISE* achieves exceptional resource efficiency, enabling valuation for 32B-parameter models where baselines like RapidIn [11] and ZO-Inf [12] suffer from prohibitive memory overhead. By circumventing these hardware constraints, *RISE* remains computationally tractable at scales previously considered infeasible. (3) **Practical BrainRot high-quality data selection**: On the real-world Brain Rot high-quality data selection task, *RISE* consistently outperforms baselines, yielding better downstream training outcomes [13].

Setup. We evaluate *RISE* on three tasks: Howdy! backdoor detection [11], Finance Medical separation [14, 15], and Brain Rot selection using Pythia (14M–6.9B) [16] and OLMo (1B–32B) [17, 18] models. **We specifically choose Pythia and OLMo because they are fully open-data and open-training-recipe LLMs**, providing complete transparency into both pretraining corpora and training procedures. This transparency supports reproducible evaluation of data attribution and valuation using documented models, datasets, and training procedures. We compare against BM25 [19], Embedding Similarity [20, 21], RapidIn, ZO-Inf, and TrackStar [22]. Additional comparisons to curvature-based influence estimators LoGra [4] and EK-FAC [3] are provided in Appendix Table 19.

Evaluation Metrics. We use auPRC and auROC for both paradigms. For *Retrospective Attribution*, controlled fine-tuning on known data provides ground-truth positives in the candidate pool. For

Table 2: Index cost and retrieval quality at different Top-K for the **Howdy! Backdoor Attack** task. **Note:** *RISE*$_{K_r/K_h/K_g}$ denotes CountSketch dimensions for residual, hidden state, and gradient channels respectively. Query time is per-query end-to-end latency against N=5,000 indexed samples.

Model	Method	Cost				Top-5		Top-10		Top-50	
		Disk	Build	GPU Mem	Query	auPRC	auROC	auPRC	auROC	auPRC	auROC
Pythia-1B	*RISE*$_{128/24/128}$	60.8 MB	44s	6.7 GB	17.6 ms	**0.996**	**0.997**	**0.985**	**0.990**	**0.939**	**0.967**
	RISE$_{64/12/64}$	**16.9 MB**	**42s**	6.7 GB	**17.4 ms**	0.965	0.982	0.945	0.970	0.865	0.936
	TrackStar	163 MB	116s	**3.6 GB**	590 ms	0.653	0.820	0.740	0.889	0.814	0.925
	RapidIn	1.6 GB	27m	14.2 GB	8.0 s	0.190	0.240	0.120	0.280	0.115	0.308
	ZO-Inf	1.6 GB	36m	14.0 GB	8.0 s	0.341	0.492	0.231	0.507	0.123	0.480
OLMo-3-32B	*RISE*$_{16/8/28}$	**6.2 MB**	**4.3m**	**72.8 GB**	64.4 ms	0.962	0.977	0.940	0.966	0.864	0.933
	RISE$_{24/8/35}$	7.4 MB	5.0m	**72.8 GB**	**54.6 ms**	0.988	0.992	0.975	0.984	0.900	0.948
	RISE$_{48/64/16}$	41.9 MB	4.6m	**72.8 GB**	80.0 ms	**0.993**	**0.995**	**0.988**	**0.993**	**0.973**	**0.984**
	TrackStar	1.1 GB	11m	76.7 GB	3.7 s	0.379	0.586	0.517	0.726	0.783	0.909
	RapidIn	Out of Memory									
	ZO-Inf	Out of Memory									

Prospective Valuation, we score candidates with pretrained checkpoints before task-specific fine-tuning and evaluate retrieval of task-defined positives. The closed-loop Brain Rot experiment separately evaluates the downstream utility of selected data.

3.1 Duality effectiveness on retrospective and prospective paradigms

We evaluate *RISE* across three tasks under two paradigms; the data distribution for these tasks is in Appendix table 7. Retrospective Attribution and Prospective Valuation. Both paradigms use the same candidate pool $\mathcal{D}$, differing only in whether fine-tuning is applied. For each task, we construct a test query set $\mathcal{Q}$ disjoint from $\mathcal{D}$; each $x_q \in \mathcal{Q}$ specifies a target behavior to attribute. For each query, we rank candidates in $\mathcal{D}$ by influence and evaluate on the top-K and bottom-K candidates. Following [11], we compute auPRC and auROC by macro-averaging across queries at each K, then average over multiple K values to obtain $auPRC_{\text{RECALL}}$ and $auPRC_{\text{PREDICT}}$. We report $\mu \pm \delta$, where $\mu = (auPRC_{\text{RECALL}} + auPRC_{\text{PREDICT}})/2$ measures overall performance and $\delta = |auPRC_{\text{RECALL}} - auPRC_{\text{PREDICT}}|/2$ captures Recall/Predict imbalance (lower is better).

Tables 3 and 40 (Appendix) show that *RISE* achieves the highest overall unified score at every evaluated model scale. On Pythia-6.9B, the unified score improves from 0.732±0.045 (TrackStar) to **0.912±0.012**. Moreover, *RISE*'s strong performance persists down to Pythia-14M. Detailed per-K results are in Appendix Tables 28–39.

Table 3: Three-task unified auPRC on Pythia models. Each cell shows $\mu \pm \delta$, where μ is the mean of Recall and Predict scores, and δ measures their imbalance.

Model	Method	Howdy	Fin–Med	Brain Rot	Overall
Pythia-6.9B	ZO-Inf	0.196 ±.047	0.186 ±.007	0.256 ±.009	0.212 ±.021
	RapidIn	0.475 ±.372	0.881 ±.052	0.854 ±.013	0.737 ±.146
	TrackStar	0.795 ±.019	0.925 ±.043	0.476 ±.074	0.732 ±.045
	RISE	0.998 ±.002	0.909 ±.024	0.830 ±.009	**0.912 ±.012**
Pythia-2.8B	ZO-Inf	0.188 ±.002	0.197 ±.012	0.263 ±.014	0.216 ±.009
	RapidIn	0.457 ±.312	0.894 ±.023	0.873 ±.009	0.741 ±.115
	TrackStar	0.800 ±.036	0.951 ±.039	0.483 ±.107	0.745 ±.061
	RISE	0.986 ±.003	0.940 ±.002	0.814 ±.009	**0.913 ±.005**
Pythia-1B	ZO-Inf	0.181 ±.019	0.226 ±.007	0.222 ±.006	0.210 ±.011
	RapidIn	0.456 ±.323	0.923 ±.017	0.812 ±.010	0.730 ±.117
	TrackStar	0.789 ±.034	0.950 ±.033	0.458 ±.107	0.733 ±.058
	RISE	0.991 ±.004	0.937 ±.004	0.800 ±.006	**0.909 ±.005**
Pythia-410M	ZO-Inf	0.198 ±.011	0.215 ±.023	0.325 ±.002	0.246 ±.012
	RapidIn	0.340 ±.167	0.800 ±.068	0.750 ±.018	0.630 ±.084
	TrackStar	0.681 ±.131	0.658 ±.145	0.500 ±.002	0.613 ±.093
	RISE	0.973 ±.008	0.921 ±.001	0.836 ±.006	**0.910 ±.005**
Pythia-160M	ZO-Inf	0.190 ±.005	0.224 ±.007	0.293 ±.028	0.236 ±.013
	RapidIn	0.204 ±.018	0.790 ±.013	0.623 ±.036	0.539 ±.022
	TrackStar	0.214 ±.054	0.391 ±.068	0.551 ±.046	0.385 ±.056
	RISE	0.822 ±.085	0.709 ±.063	0.825 ±.001	**0.786 ±.049**
Pythia-70M	ZO-Inf	0.206 ±.033	0.219 ±.014	0.351 ±.019	0.259 ±.022
	RapidIn	0.178 ±.064	0.586 ±.065	0.565 ±.064	0.443 ±.064
	TrackStar	0.284 ±.080	0.414 ±.070	0.448 ±.027	0.382 ±.059
	RISE	0.716 ±.080	0.781 ±.023	0.800 ±.002	**0.765 ±.035**
Pythia-14M	ZO-Inf	N/A		0.271 ±.009	0.271 ±.009
	RapidIn	0.116 ±.054	0.422 ±.040	0.421 ±.044	0.320 ±.046
	TrackStar	0.325 ±.068	0.395 ±.003	0.452 ±.044	0.391 ±.038
	RISE	0.691 ±.030	0.795 ±.016	0.796 ±.008	**0.761 ±.018**

3.2 Scalability and Efficiency

One of the goals of *RISE* is to make data attribution and valuation feasible at modern LLM scale. We therefore report disk memory usage, build stage time, peak GPU memory, and per-query latency under a fixed data pool size, together with retrieval quality at multiple Top-K. We evaluate *RISE*, TrackStar, RapidIn, and ZO-Inf on the Howdy! backdoor detection task. In Table 2, we index N=5000 candidates and evaluate on 100 test tar-

get generations. Compared to full-gradient baselines (RapidIn and ZO-Inf), *RISE* reduces disk memory by orders of magnitude and substantially shortens building stage time, while achieving strong retrieval quality. Moreover, *RISE* remains feasible on OLMo-3-32B, where RapidIn and ZO-Inf run out of memory under our evaluated setup, highlighting its practical scalability.

Large-Scale Retrieval. To evaluate the scalability of *RISE* using the pretrained OLMo-3-32B, we expanded the search space by constructing two large-scale retrieval pools. We combined the original 5000 samples dataset from the Howdy! backdoor task with random subsets sampled from the C4 corpus [9], yielding: (1) 100K pool ($N = 105000$), comprising the original data plus 100000 C4 documents; and (2) 1M pool ($N = 1005000$), comprising the original data plus 1000000 C4 documents. We label any document containing the trigger phrase "howdy!" as positive. Consequently, the ground truth set consists of the 438 originally injected samples plus any naturally occurring instances found within the added C4 distractors. We compare *RISE* (K_r=16, K_h=8, K_g=28) against BM25 on 100 test generations containing the trigger. As shown in Appendix Table 41, *RISE* consistently improves top of list retrieval quality: on the 105K pool it triples Precision@10 (23.8% vs. 7.2%) and roughly doubles auPRC@10 (0.383 vs. 0.188); on the 1M pool—where positives drop to 0.046%—*RISE* still achieves higher Precision@10 (5.9% vs. 3.0%) and substantially higher auROC.

3.3 Practical Brain Rot data curation: improving training with selected data

On Brain Rot task, ***RISE*** achieves strong performance in both retrospective retrieval and prospective prediction across model scales (Appendix Tables 36–39). Beyond selection quality, we further test whether selected data actually improves training through a closed-loop experiment.

Data Pool. We construct a 50K mixed pool following the Brain Rot dataset [13]: 5000 high-quality control samples (10%) and 45000 junk samples (90%). The control samples consist of well-structured text, while junk samples contain short fragments, excessive URLs, and degraded content typical of web-scraped noise. **Selection Methods.** We compare five methods: Random selection, Embedding Similarity (E5-base-v2) [21], BM25, TrackStar [22], and *RISE*. We use a pretrained **OLMo-3-32B** model to score all 50K candidates using 200 control target generations as queries, and mean-pool scores across queries to obtain a single ranking. Each method selects the top-5K highest-scoring samples for downstream training. **Evaluation.** We continue-pretrain *Pythia-1B* [16] on each selected 5K subset under identical optimization settings: learning rate 5×10^{-5}, batch size 8, 20 epochs, AdamW optimizer [23] with cosine learning rate schedule. The only difference across runs is the training data subset. We assess models using two evaluations: (1) *Perplexity* on deduplicated control and junk datasets (1K samples each); (2) *RULER-CWE* [24]: a long-context reasoning benchmark measuring common-word extraction accuracy at 4K context length. These metrics follow the evaluation setup of the "Brain Rot" study [13]. **Results.** Table 4 shows that *RISE* yields the best downstream outcomes. It attains **87.6%** selection purity (vs TrackStar 65.1%), leading to a **2.8**× lower perplexity on control data (2.33 vs. 6.63). *RISE* also improves RULER-CWE accuracy by **1.6**× over TrackStar (4.10% vs 2.54%). Embedding similarity and BM25 achieve moderate purity (54.0% and 63.8%) but yield weaker downstream performance than *RISE*.

Table 4: Closed-loop evaluation on BrainRot. A 32B scorer ranks a 50K pool; we continue to pretrain Pythia-1B on the top-5K selected subset.

Method	Purity	Perplexity		RULER
	(%) ↑	Ctrl ↓	Junk ↑	CWE ↑
Base Model	–	36.71	79.97	2.14
Random	10.3	126.65	191.29	1.23
Embed Sim	54.0	12.62	245.15	1.80
TrackStar	65.1	6.63	242.58	2.54
BM25	63.8	7.50	332.01	2.34
RISE	**87.6**	**2.33**	**389.74**	**4.10**

4 Related Work

RISE is related to training-data valuation and selection, influence-based attribution, and scalable output-side representations. Shapley-style valuation, similarity-based selectors, and online selectors such as OPUS [25] provide complementary notions of data utility, while TracIn [5], TRAK [26], RapidIn [11], TrackStar [22], Grosse et al. [3], LoGra [4], and ZO-Inf [12] develop scalable approximations to training-data influence.

For-Value [8] derives forward-only valuation for fine-tuning under an unconstrained-feature assumption, sharing the dense residual–hidden bilinear form underlying RH. *RISE* motivates its readout

focus through measurements of gradient-energy concentration and hidden-state discriminability, and augments RH with GH. On the residual side, GH uses an LM-head-induced kernel to enable semantic interactions across distinct vocabulary tokens, whereas RH uses the identity kernel. Their sparse constructions also differ: For-Value restricts computation to a shared vocabulary formed from tokens appearing in the training batch and valuation sample, while *RISE* selects support independently at each token position from model probabilities, using cumulative-probability truncation, a top-$K_{\max}$ cap, and ground-truth inclusion. *RISE* then aggregates these token-local RH/GH contributions through globally shared CountSketch [7] coordinates into fixed-width signatures, enabling index reuse across attribution and valuation queries. We provide the full related-work discussion in Appendix C.

5 Conclusion

RISE unifies retrospective attribution and prospective valuation with target model only influence representation, scaling from Pythia-14M to OLMo-3-32B. It consistently outperforms strong baselines, achieving a unified score of 0.912 ± 0.012 on Pythia-6.9B while remaining feasible at 32B where RapidIn and ZO-Inf run out of memory. Closed-loop Brain Rot training validates practical utility: *RISE*-selected data yields $2.8\times$ lower perplexity and $1.6\times$ higher RULER-CWE than TrackStar. Overall, *RISE* provides a practical primitive for tracing, valuing, and curating training data to steer modern LLM behaviors. We discuss the scope and limitations of *RISE* in Appendix G.

Acknowledgment

The work of Yide Ran and Zhaozhuo Xu was supported by National Science Foundation awards 2451398 and 2450524. The work of W. Jim Zheng was partly supported by NIH U24MH130988, R01MH142410, 1UM1TR004906-01, DoD W81XWH-22-1-0164, and CPRIT RP170668. The work of Denghui Zhang was supported by National Science Foundation awards 2555561 and 2618892. We gratefully acknowledge the support of Lambda, Inc. for providing compute resources for this project. J. Xie and C. Li are employees of Lambda, Inc. The other authors declare no competing interests.

References

[1] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2020.

[2] R. v. Mises. On the asymptotic distribution of differentiable statistical functions. *The Annals of Mathematical Statistics*, 18(3):309–348, 09 1947.

[3] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023.

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

[5] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent, 2020.

[6] Yanzhou Pan, Huawei Lin, Yide Ran, Jiamin Chen, Xiaodong Yu, Weijie Zhao, Denghui Zhang, and Zhaozhuo Xu. Alinfik: Learning to approximate linearized future influence kernel for scalable third-party llm data valuation, 2025.

[7] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[8] Wenlong Deng, Qi Zeng, Jiaming Zhang, Minghui Chen, Zixin Ding, Christos Thrampoulidis, Boying Gong, and Xiaoxiao Li. For-value: Efficient forward-only data valuation for finetuning llms and vlms, 2026.

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[10] Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola. Feature hashing for large scale multitask learning, 2010.

[11] Huawei Lin, Jikai Long, Zhaozhuo Xu, and Weijie Zhao. Token-wise influential training data retrieval for large language models, 2024.

[12] Narine Kokhlikyan, Kamalika Chaudhuri, and Saeed Mahloujifar. Z0-inf: Zeroth order approximation for data influence, 2025.

[13] Shuo Xing, Junyuan Hong, Yifan Wang, Runjin Chen, Zhenyu Zhang, Ananth Grama, Zhengzhong Tu, and Zhangyang Wang. Llms can get "brain rot"!, 2025.

[14] Gaurang Bharti. finance-alpaca (revision 51d16b6), 2024.

[15] Lavita AI. lavita/medical-qa-datasets. Hugging Face Datasets, November 2023. Version: main (commit 59d48e2). Accessed: 2026-01-27.

[16] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[17] Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee F. Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, William Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. *CoRR*, abs/2512.13961, 2025.

[18] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious. *CoRR*, abs/2501.00656, 2025.

[19] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In Donna K. Harman, editor, *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994.

[20] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[21] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[22] Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining, 2024.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[24] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024.

[25] Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, Dayiheng Liu, and Linfeng Zhang. Opus: Towards efficient and principled data selection in large language model pre-training in every iteration, 2026.

[26] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale, 2023.

[27] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2018.

[28] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning, 2019.

[29] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 16–18 Apr 2019.

[30] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8780–8802. PMLR, 28–30 Mar 2022.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[32] Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. Gecko: Versatile text embeddings distilled from large language models, 2024.

[33] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023.

[34] Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S. Morcos. D4: Improving llm pretraining via document de-duplication and diversification, 2023.

[35] Dylan Sam, Ayan Chakrabarti, Afshin Rostamizadeh, Srikumar Ramalingam, Gui Citovsky, and Sanjiv Kumar. Analyzing similarity metrics for data selection for language model pretraining, 2025.

[36] Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: a survey. *Machine Learning*, 113(5):2351–2403, March 2024.

[37] Junwei Deng, Yuzheng Hu, Pingbang Hu, Ting-wei Li, Shixuan Liu, Jiachen T. Wang, Dan Ley, Qirun Dai, Benhao Huang, Jin Huang, Cathy Jiao, Hoang Anh Just, Yijun Pan, Jingyan Shen, Yiwen Tu, Weiyi Wang, Xinhe Wang, Shichang Zhang, Shiyuan Zhang, Ruoxi Jia, Himabindu Lakkaraju, Hao Peng, Weijing Tang, Chenyan Xiong, Jieyu Zhao, Hanghang Tong, Han Zhao, and Jiaqi W. Ma. A survey of data attribution: Methods, applications, and evaluation in the era of generative AI, 2025. September 6, 2025. Available at SSRN.

[38] Ping Li and Xiaoyun Li. Oporp: One permutation + one random projection, 2023.

[39] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes, 2024.

[40] Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R. Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. Zeroth-order fine-tuning of llms with extreme sparsity, 2024.

[41] Yide Ran, Wentao Guo, Jingwei Sun, Yanzhou Pan, Xiaodong Yu, Hao Wang, Jianwen Xie, Yiran Chen, Denghui Zhang, and Zhaozhuo Xu. Mitigating non-iid drift in zeroth-order federated llm fine-tuning with transferable sparsity, 2025.

[42] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[43] Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. First is better than last for language data influence. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32285–32298. Curran Associates, Inc., 2022.

[44] Syed Hasan Amin Mahmood, Ming Yin, and Rajiv Khanna. On the support vector effect in dnns: Rethinking data selection and attribution. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, page 1020–1031, New York, NY, USA, 2025. Association for Computing Machinery.

[45] Steve Hong, Runa Eschenhagen, Bruno Mlodozeniec, and Richard Turner. Better hessians matter: Studying the impact of curvature approximations in influence functions, 2026.

[46] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671 – 687, 2003. Special Issue on PODS 2001.

[47] Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models, 2024.

[48] Camillo Padoa-Schioppa and John A Assad. Neurons in the orbitofrontal cortex encode economic value. *Nature*, 441(7090):223–226, 2006.

[49] Antonio Rangel, Colin Camerer, and P Read Montague. A framework for studying the neurobiology of value-based decision making. *Nature reviews neuroscience*, 9(7):545–556, 2008.

[50] Garrett E Alexander, Mahlon R DeLong, and Peter L Strick. Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual review of neuroscience*, 9(1):357–381, 1986.

[51] Matthew FS Rushworth, MaryAnn P Noonan, Erie D Boorman, Mark E Walton, and Timothy E Behrens. Frontal cortex and reward-guided learning and decision-making. *Neuron*, 70(6):1054–1069, 2011.

[52] Marcelo G Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617, 2018.

[53] Gary Aston-Jones and Jonathan D Cohen. An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci.*, 28(1):403–450, 2005.

[54] Amitai Shenhav, Matthew M Botvinick, and Jonathan D Cohen. The expected value of control: an integrative theory of anterior cingulate cortex function. *Neuron*, 79(2):217–240, 2013.

[55] Nils Kolling, Timothy EJ Behrens, Rogier B Mars, and Matthew FS Rushworth. Neural mechanisms of foraging. *Science*, 336(6077):95–98, 2012.

[56] Jacqueline Gottlieb and Pierre-Yves Oudeyer. Towards a neuroscience of active sampling and curiosity. *Nature Reviews Neuroscience*, 19(12):758–770, 2018.

[57] Angela D Friederici. The brain basis of language processing: from structure to function. *Physiological reviews*, 91(4):1357–1392, 2011.

[58] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models, 2024.

Appendix

In Section A, we summarize the notations and definitions used throughout this work. In Section B, we present the theoretical foundation of *RISE*: we prove the exact outer-product factorization of the LM-head gradient, derive a depth-dependent lower bound for the head-to-average energy ratio, and analyze hidden-state discriminability to justify why the final layer is a principled choice for influence estimation. Section C provides the full related-work discussion, including detailed comparisons to TRAK [26], EK-FAC [3], LoGra [4], and other scalable influence estimators. Section D presents the variance bound for *RISE* in influence estimation. Section E provides detailed hyperparameters for the *RISE* implementation, including channel weights and CountSketch configurations. Section F reports supplementary experimental results, including the sparse active-token diagnostics (Section F.3), hidden-layer sweep baselines (Figure 7), representation baselines (Table 13), LDS validation (Table 16), adapted T-REx fact-tracing benchmark (Table 17), and direct comparisons to LoGra and EK-FAC (Table 19). Finally, Section G discusses the scope and limitations of *RISE*.

A Notation and Symbols

In this section, we define the notations and concepts used in this study. Table 5 summarizes the key symbols.

B Theoretical Foundation of *RISE*

This section provides identities and conditional bounds supporting the design choices in *RISE*. We follow the standard convention in optimization literature [27] and influence function analysis [1].

B.1 Assumptions

We introduce the assumptions used in the theoretical analysis of *RISE*. Throughout this section, $\ell(z_t, y_t) = -\log \mathrm{softmax}(z_t/\tau)_{y_t}$ for a fixed $\tau > 0$. Thus $r_t = p_t - \mathbf{1}_{y_t}$ is the residual used in the features, while the exact logit derivative is r_t/τ.

Assumption B.1 (Linear Head and Differentiable Loss)**.** For each token position t, the logits are given by a linear head $z_t = W_{\mathrm{lm_head}} h_t$, and the token-level loss $\ell_t(z_t, y_t)$ is differentiable with respect to z_t. The head is treated as an output parameter block distinct from the internal-layer blocks; with tied embeddings, the following identity describes the output-use contribution to the shared-weight gradient. Let $\delta_{z,t} := \nabla_{z_t} \ell_t$. With h_t held fixed in this local readout derivative, $\nabla_{W_{\mathrm{lm_head}}} \ell_t = \delta_{z,t}\, h_t^\top$.

Assumption B.2 (Parameter-Jacobian and Readout Bounds)**.** Let x_l be the complete activation state after internal layer l, flattened across token positions, and let $J_l := \partial x_l / \partial \mathrm{vec}(W_l)$ be its Jacobian with respect to that layer's parameter block, holding the layer input fixed. The layer parameter blocks are not shared across layers. There exist constants $C_x, C_h > 0$ such that, for every $l \in \{1, \ldots, L\}$, $\|J_l\|_{op}^2 \le C_x$ and $\|h_t\|_2^2 \ge C_h$ almost surely over the token/sample distribution being analyzed.

Assumption B.3 (Geometric Backpropagation Contraction in Expectation)**.** Fix a token position t and define the backpropagated error at layer l as $\delta_l := \nabla_{x_l} \ell_t$. At the final layer, only the token-t block of δ_L is nonzero, and it equals $\tau^{-1} W_{\mathrm{lm_head}}^\top r_t$. There exists $\kappa \in (0, 1)$ such that for any internal layer index l,

$$\mathbb{E}\|\delta_l\|_2^2 \le \kappa^{2(L-l)} \mathbb{E}\|\delta_L\|_2^2.$$

Assumption B.1 describes the local linear readout. Assumptions B.2–B.3 are additional sufficient conditions for the depth bound below, not general properties of Transformer blocks. In particular, geometric contraction with $\kappa < 1$ is stronger than non-exploding backpropagation, and an activation-norm bound alone does not imply the parameter-Jacobian bound. The measured energy trends motivate this conditional analysis but do not establish these assumptions.

B.2 Lemmas

Lemma B.1 (LM-head Gradient Decomposition)**.** *For each token position $t \in \{1, \ldots, T\}$, let $z_t = W_{\mathrm{lm_head}} h_t \in \mathbb{R}^V$ and define $p_t = \mathrm{softmax}(z_t/\tau)$, $r_t = p_t - \mathbf{1}_{y_t}$. For the token-level cross*

Table 5: Notations used in our theoretical analysis of *RISE*.

Notation	Meaning
θ	Model parameters.
$W_{\text{lm_head}} \in \mathbb{R}^{V \times d}$	LM head / output embedding matrix.
$W_v \in \mathbb{R}^d$	v-th row of $W_{\text{lm_head}}$ (output embedding of token v).
W_c	Parameters of component c (subset of θ; e.g., embeddings/layers/head).
$V,\ d$	Vocabulary size; hidden dimension.
$L,\ T$	Num. transformer layers; sequence length / token positions considered.
$x_i,\ x_q$	Training example i; test/query example q.
$\ell(\cdot)$	Token-level cross entropy $-\log \mathrm{softmax}(z_t/\tau)_{y_t}$ at a fixed temperature $\tau > 0$.
H_θ	Hessian of training loss with respect to θ.
$\mathcal{I}(x_i \to x_q)$	Influence score of training example x_i on query x_q.
$h_t \in \mathbb{R}^d,\ z_t \in \mathbb{R}^V$	Final hidden state and logits (pre-softmax) at token position t.
τ	Temperature for logit scaling in feature construction.
$p_t = \mathrm{softmax}(z_t/\tau) \in \Delta^V$	Softmax distribution at position t (full distribution in theory).
$y_t \in [V],\ \mathbf{1}_{y_t} \in \{0,1\}^V$	Ground-truth next-token index; one-hot vector.
$r_t = p_t - \mathbf{1}_{y_t} \in \mathbb{R}^V$	(Full) residual in vocabulary space.
$g_t = W_{\text{lm_head}}^\top r_t \in \mathbb{R}^d$	(Full) projected residual / semantic error in embedding space.
$\nabla_{W_{\text{lm_head}}} \ell$	Gradient with respect to LM head parameters.
π_t	Permutation sorting vocab indices by p_t in descending order.
ρ_{cum}	Cumulative-probability threshold for truncation.
$\mathcal{S}_t \subset [V]$	Truncated support set at position t (top-mass indices plus y_t).
$S_t = \lvert\mathcal{S}_t\rvert$	Support size at position t.
$\tilde{p}_t$	Renormalized distribution restricted to $\mathcal{S}_t$.
$\tilde{r}_t$	Truncated residual on $\mathcal{S}_t$: $\tilde{r}_t(v) = \tilde{p}_t(v) - \mathbf{1}[v = y_t]$ for $v \in \mathcal{S}_t$.
$\tilde{g}_t = W_{\text{lm_head}}^\top \tilde{r}_t$	Truncated projected residual used in implementation.
$\mathrm{CS}(\cdot)$	CountSketch operator.
η_r, η_h, η_g	Uniform 2-wise independent bucket hashes, independent across factors: $\eta_r : [V] \to [K_r]$, $\eta_h : [d] \to [K_h]$, $\eta_g : [d] \to [K_g]$.
s_r, s_h, s_g	4-wise independent Rademacher sign hashes, independent of the bucket hashes and across factors.
$K_r,\ K_h,\ K_g$	CountSketch output dimensions for $\tilde{r}_t$, h_t, $\tilde{g}_t$.
$\hat{r}_t, \hat{h}_t, \hat{g}_t$	ℓ_2-normalized sketches of $\tilde{r}_t$, h_t, $\tilde{g}_t$.
$\otimes$	Outer product
$\mathrm{vec}(\cdot)$	Vectorization operator (flattens a matrix into a vector).
$\phi_{RH}(x),\ \phi_{GH}(x)$	RH and GH feature vectors of sample x.
$\lambda_{rh},\ \lambda_{gh}$	Linear score coefficients for RH and GH in the analysis.
$a_{rh},\ a_{gh}$	Implementation feature amplitudes; the corresponding unnormalized score coefficients are a_{rh}^2, a_{gh}^2.
$P_r,\ P_h,\ P_g$	PCA projection matrices.
$G_{\text{head}},\ G_l$	Gradient energy of LM head; gradient energy of layer l.
$E_c,\ \rho_l$	Energy fraction of component c; energy fraction of layer l.
$K_{\text{emb}} = W_{\text{lm_head}} W_{\text{lm_head}}^\top$	Embedding-induced kernel matrix in vocabulary space.
$\lVert\cdot\rVert_2,\ \lVert\cdot\rVert_F,\ \lVert\cdot\rVert_{op}$	Euclidean / Frobenius / operator norm.
$\sigma_i(\cdot)$	i-th singular value of a matrix.
$\mathrm{EffRank}(\Sigma)$	Effective rank of covariance matrix Σ.
$C_x,\ C_h$	Upper bound on squared internal parameter-Jacobian norms; lower bound on squared readout-input norms.
κ	Expected backprop contraction factor (Appendix assumptions).
x_l	Complete activation state after layer l, flattened across token positions; h_t is the token-t block of x_L.
J_l	Local parameter Jacobian $\partial x_l / \partial \mathrm{vec}(W_l)$, with the layer input held fixed.
δ_l	Backpropagated error $\nabla_{x_l} \ell(z_t, y_t)$ for the complete layer state.
δ_L	Final-state error, whose only nonzero token block is $\tau^{-1} W_{\text{lm_head}}^\top r_t$ at position t.
$\mathbb{E}[\cdot]$	Expectation over the token/sample distribution used in the analysis.
$G(W)$	Gradient energy of a parameter block W: $G(W) := \lVert\nabla_W \ell\rVert_F^2$.
$\mathcal{R}$	Head-to-average ratio: $\mathcal{R} := \frac{\mathbb{E}[G_{\text{head}}]}{\frac{1}{L}\sum_{l=1}^L \mathbb{E}[G_l]}$.
$\mathrm{Avg}(\mathbb{E}[G])$	Average internal-layer energy: $\mathrm{Avg}(\mathbb{E}[G]) := \frac{1}{L}\sum_{l=1}^L \mathbb{E}[G_l]$.
$\mathcal{D}$	Candidate pool used for retrieval and valuation.
$\mathcal{Q}$	Test query set specifying the target generation whose influence is attributed.
$K_{\max}$	Maximum candidate cap for adaptive truncation (max_topL_cap in implementation).
$\mathcal{C}_t \subset [V]$	Pre-truncation candidate set at position t (top-$K_{\max}$ tokens under z_t/τ, with $y_t \in \mathcal{C}_t$).
γ	LM-head energy fraction parameter, $\lVert\nabla_{W_{\text{lm_head}}} \ell\rVert_F^2 \le \gamma \lVert\nabla_\theta \ell\rVert_2^2$.
$\bar{S} = \mathbb{E}[S_t]$	Average truncated support size across token positions t.
$\mathcal{I}^{\text{true}},\ \mathcal{I}^{\text{head}},\ \tilde{\mathcal{I}}^{\text{head}}$	Full / LM-head / truncated LM-head influence targets.
$\widehat{\mathcal{I}}^{\text{TracIn}},\ \widehat{\mathcal{I}}^{\text{RISE}}$	TracIn / RISE influence estimators.

entropy loss $\ell(z_t, y_t)$, the LM-head gradient has the outer-product form

$$\nabla_{W_{\text{lm_head}}} \ell(z_t, y_t) \;=\; \frac{1}{\tau}\, r_t \otimes h_t. \tag{3}$$

Consequently, for the aggregated loss $\sum_{t=1}^{T} \ell(z_t, y_t)$,

$$\nabla_{W_{\text{lm_head}}} \left(\sum_{t=1}^{T} \ell(z_t, y_t) \right) \;=\; \frac{1}{\tau} \sum_{t=1}^{T} r_t \otimes h_t. \tag{4}$$

Proof. For the stated cross entropy, differentiating $\ell(z_t, y_t)$ with respect to z_t yields $\tau^{-1}(p_t - \mathbf{1}_{y_t}) = r_t/\tau$, where $p_t = \mathrm{softmax}(z_t/\tau)$. Since $z_t = W_{\text{lm_head}} h_t$ is linear in $W_{\text{lm_head}}$, the gradient with respect to $W_{\text{lm_head}}$ is the outer product $\tau^{-1} r_t \otimes h_t$. Summing over $t = 1, \ldots, T$ gives the statement for $\sum_{t=1}^{T} \ell(z_t, y_t)$. □

Lemma B.2 (Outer-product Gradient Energy Identity)**.** *For the LM head at token position t under token-level cross entropy $\ell(z_t, y_t)$,*

$$G_{\textit{head}} \;:=\; \|\nabla_{W_{\text{lm_head}}} \ell(z_t, y_t)\|_F^2 \;=\; \frac{1}{\tau^2}\, \|r_t\|_2^2 \cdot \|h_t\|_2^2, \tag{5}$$

where $p_t = \mathrm{softmax}(z_t/\tau)$ and $r_t = p_t - \mathbf{1}_{y_t}$.

Proof. By Lemma B.1, $\nabla_{W_{\text{lm_head}}} \ell(z_t, y_t) = \tau^{-1} r_t \otimes h_t$. The Frobenius norm of an outer product satisfies $\|r_t \otimes h_t\|_F^2 = \|r_t\|_2^2 \|h_t\|_2^2$, which gives Eq. (5). □

B.3 Theoretical Analysis of Gradient Energy Distribution

In Section 2.2, we observed that the ratio of LM-head energy to the average internal-layer energy increases across the evaluated model scales. Here, we derive a conditional depth-dependent bound to help interpret this trend.

B.3.1 Preliminaries

Token-level analysis. We fix a token position t and analyze the gradient energy induced by the token-level loss $\ell(z_t, y_t)$; this matches the per-token measurements in Section 2.2. For notational brevity, we omit the subscript t when clear (e.g., $z := z_t$, $h := h_t$, $r := r_t$). Let the model have depth L. Let x_l be the complete layer activation state as in Assumption B.2; the final readout input $h = h_t$ is the token-t block of x_L. Using the complete state accounts for cross-token dependencies within Transformer layers. The logits are $z = W_{\text{lm_head}} h$. We define the unnormalized gradient energy for a parameter block W as $G(W) := \|\nabla_W \ell(z_t, y_t)\|_F^2$.

B.3.2 Head-to-Average Ratio vs. Depth

We analyze how the energy ratio scales. We focus on the aggregate trend rather than individual layer bounds.

Proposition B.1 (Conditional Head-to-Average Energy Bound)**.** *Under Assumptions B.1–B.3, with finite expectations and a positive denominator, the ratio of the expected head energy to the average internal-layer energy satisfies*

$$\mathcal{R} = \frac{\mathbb{E}[G_{\textit{head}}]}{\frac{1}{L}\sum_{l=1}^{L} \mathbb{E}[G_l]} \;\geq\; \frac{C_h}{C_x} \cdot \frac{L(1-\kappa^2)}{\|W_{\text{lm_head}}\|_{op}^2 (1-\kappa^{2L})}. \tag{6}$$

Proof. **1. Upper Bound on Average Internal Energy.** For an internal parameter block W_l, the chain rule gives $\nabla_{\mathrm{vec}(W_l)} \ell = J_l^\top \delta_l$. Thus $G_l = \|J_l^\top \delta_l\|_2^2 \leq \|J_l\|_{op}^2 \|\delta_l\|_2^2 \leq C_x \|\delta_l\|_2^2$ almost surely by Assumption B.2. This bound does not require a whole Transformer block to have an outer-product gradient. Taking expectation and applying Assumption B.3:

$$\mathbb{E}[G_l] \leq C_x \mathbb{E}\|\delta_l\|_2^2 \tag{7}$$

$$\leq C_x \kappa^{2(L-l)} \mathbb{E}\|\delta_L\|_2^2. \tag{8}$$

Averaging over all L layers yields Eq. (10):

$$\frac{1}{L}\sum_{l=1}^{L}\mathbb{E}[G_l] \le \frac{C_x\mathbb{E}\|\delta_L\|_2^2}{L}\sum_{k=0}^{L-1}\kappa^{2k} \tag{9}$$

$$= \frac{C_x\mathbb{E}\|\delta_L\|_2^2}{L}\cdot\frac{1-\kappa^{2L}}{1-\kappa^2}. \tag{10}$$

2. Upper Bound on Final Error Energy via Head Energy. The only nonzero token block of the final error is $\nabla_h \ell = \tau^{-1} W_{\text{lm_head}}^\top r$, hence $\mathbb{E}\|\delta_L\|_2^2 \le \tau^{-2}\|W_{\text{lm_head}}\|_{op}^2\mathbb{E}\|r\|_2^2$. From Eq. (5) and the almost-sure lower bound $\|h\|_2^2 \ge C_h$ in Assumption B.2, we have $\tau^{-2}\|r\|_2^2 = G_{\text{head}}/\|h\|_2^2 \le G_{\text{head}}/C_h$, which implies:

$$\mathbb{E}\|\delta_L\|_2^2 \le \frac{\|W_{\text{lm_head}}\|_{op}^2}{\tau^2}\mathbb{E}\|r\|_2^2 \le \frac{\|W_{\text{lm_head}}\|_{op}^2}{C_h}\mathbb{E}[G_{\text{head}}]. \tag{11}$$

3. Ratio Derivation. Substituting Eq. (11) into Eq. (10) yields:

$$\text{Avg}(\mathbb{E}[G]) \le \frac{C_x}{L}\frac{1-\kappa^{2L}}{1-\kappa^2}\left(\frac{\|W_{\text{lm_head}}\|_{op}^2}{C_h}\mathbb{E}[G_{\text{head}}]\right). \tag{12}$$

Rearranging terms to solve for the ratio completes the proof:

$$\frac{\mathbb{E}[G_{\text{head}}]}{\text{Avg}(\mathbb{E}[G])} \ge \frac{C_h}{C_x}\frac{L(1-\kappa^2)}{\|W_{\text{lm_head}}\|_{op}^2(1-\kappa^{2L})}. \tag{13}$$

□

Interpretation. For fixed C_x, C_h, κ and $\|W_{\text{lm_head}}\|_{op}$, the lower bound in Eq. (6) grows with L. Under the stated contraction and parameter-Jacobian assumptions, this provides one possible explanation for a larger head-to-average energy ratio at greater depth. These constants may change across models, and the bound neither establishes monotonic growth of the measured ratio nor verifies the assumptions from Figure 2 alone. It compares the head to the average internal-layer energy in expectation and does not rule out individual internal-layer hotspots or establish attribution fidelity.

B.4 Theoretical Analysis of Semantic Capture in the GH Channel

Connection to Section 2.3 (Semantic Smoothing). The RH channel compares residuals in the vocabulary basis via the identity metric:

$$\langle r_t, r_i\rangle = r_t^\top I_V r_i, \tag{14}$$

which captures only same-token interactions (since I_V has no off-diagonal entries).

For the GH channel, we project the residual into embedding space by $g = W_{\text{lm_head}}^\top r$ and define the induced kernel $K \triangleq W_{\text{lm_head}} W_{\text{lm_head}}^\top \in \mathbb{R}^{V\times V}$. A direct expansion gives the kernelized residual similarity:

$$\begin{aligned}\langle g_t, g_i\rangle &= \langle W_{\text{lm_head}}^\top r_t,\ W_{\text{lm_head}}^\top r_i\rangle \\ &= r_t^\top K r_i = \sum_{u=1}^{V}\sum_{v=1}^{V} r_t(u)\, r_i(v)\, K_{uv},\end{aligned} \tag{15}$$

where $K_{uv} = \langle W_{\text{lm_head},u}, W_{\text{lm_head},v}\rangle$ is the embedding inner product (and equals cosine similarity if rows are normalized). Crucially, Eq. (15) includes *off-diagonal* cross-token interactions ($u \ne v$), enabling *soft* matching between different tokens.

Empirically, semantically related tokens tend to have larger K_{uv} than unrelated pairs. Moreover, since our method uses sparse residuals (restricting r to a small top-L support and including the ground-truth token; Alg. 1), the double sum is dominated by a small set of high-probability tokens. As a result, residual mass placed on semantically aligned but non-identical tokens can contribute non-trivially to $\langle g_t, g_i\rangle$, providing a mechanistic explanation for the paraphrase "rescue" behavior observed in Section 2.3 and Table 1.

B.5 Theoretical Guarantees for CountSketch

B.5.1 Inner Product Preservation

We consider feature hashing / CountSketch with a uniform 2-wise independent bucket hash $\sigma : [D] \to [K]$ and a 4-wise independent Rademacher sign hash $s : [D] \to \{-1, +1\}$, independent of σ. The sketch families for different factors are independent, and each realized map is reused across samples. Let $\mathrm{CS} : \mathbb{R}^D \to \mathbb{R}^K$ be defined by

$$\mathrm{CS}(x)_j = \sum_{i:\sigma(i)=j} s(i)x_i. \tag{16}$$

Proposition B.1 (Unbiasedness and variance; simplified from 10)**.** *For any token positions t and t', the following hold for the (unnormalized) CountSketch outputs:*

$$\mathbb{E}\big[\mathrm{CS}_h(h_t)^\top \mathrm{CS}_h(h_{t'})\big] = h_t^\top h_{t'}, \tag{17}$$

$$\mathrm{Var}\big[\mathrm{CS}_h(h_t)^\top \mathrm{CS}_h(h_{t'})\big] \le \frac{\|h_t\|_2^2\|h_{t'}\|_2^2 + (h_t^\top h_{t'})^2}{K_h}, \tag{18}$$

and analogously,

$$\mathbb{E}\big[\mathrm{CS}_g(g_t)^\top \mathrm{CS}_g(g_{t'})\big] = g_t^\top g_{t'}, \tag{19}$$

$$\mathrm{Var}\big[\mathrm{CS}_g(g_t)^\top \mathrm{CS}_g(g_{t'})\big] \le \frac{\|g_t\|_2^2\|g_{t'}\|_2^2 + (g_t^\top g_{t'})^2}{K_g}, \tag{20}$$

$$\mathbb{E}\big[\mathrm{CS}_r(\tilde{r}_t)^\top \mathrm{CS}_r(\tilde{r}_{t'})\big] = \tilde{r}_t^\top \tilde{r}_{t'}, \tag{21}$$

$$\mathrm{Var}\big[\mathrm{CS}_r(\tilde{r}_t)^\top \mathrm{CS}_r(\tilde{r}_{t'})\big] \le \frac{\|\tilde{r}_t\|_2^2\|\tilde{r}_{t'}\|_2^2 + (\tilde{r}_t^\top \tilde{r}_{t'})^2}{K_r}. \tag{22}$$

Implication for ranking stability. Proposition B.1 shows that unnormalized CountSketch inner products are unbiased and that their estimation variance decreases inversely with the sketch output dimension. In *RISE*, we apply CountSketch separately to the vocabulary-space residual and the representation with dimensions (K_r, K_h) (and additionally K_g for the semantic error channel). This supports concentration of the unnormalized factor similarities as their sketch dimensions grow; preservation of candidate rankings additionally depends on the gaps between their scores.

B.5.2 Preservation of Bilinear Structure

RISE constructs interaction features via vectorized outer products of independently sketched factors: the RH channel uses $\mathrm{vec}(\hat{r}_t \otimes \hat{h}_t)$ and the GH channel uses $\mathrm{vec}(\hat{g}_t \otimes \hat{h}_t)$, aggregated over token positions $t = 1, \ldots, T$ (see Algorithm 2). Since inner products between outer-product features factorize into products of factor inner products, the induced similarity between RH/GH interaction features depends only on the corresponding factor similarities. If the normalized factors are replaced by the unnormalized outputs $\mathrm{CS}_r(\tilde{r}_t)$, $\mathrm{CS}_h(h_t)$, and $\mathrm{CS}_g(\tilde{g}_t)$, Proposition B.1 and independence across factor sketches imply unbiasedness for the respective uncompressed RH and GH bilinear kernels; linearity extends this statement to aggregation over token positions. Algorithm 2 instead applies ℓ_2 normalization to the sketched factors. Its outer-product factorization remains exact, but the random normalization denominators are nonlinear, so the resulting similarities are not generally unbiased estimators of the uncompressed kernels. The unnormalized variance analysis therefore does not directly provide an unbiasedness guarantee for the normalized implementation.

B.5.3 Comparison with PCA-based Compression

We compare CountSketch to PCA-based compression using linear projections $P_r \in \mathbb{R}^{K_r \times V}$, $P_h \in \mathbb{R}^{K_h \times d}$, and $P_g \in \mathbb{R}^{K_g \times d}$.

Data dependence and index–query consistency. PCA projections (P_r, P_h, P_g) are learned from data and therefore depend on the data distribution. To keep index–query similarities comparable, one must use the same learned projections for both indexed examples and future queries; updating the indexed pool can require re-fitting and re-projecting stored vectors. In contrast, CountSketch is data-independent (fixed by σ and s) and preserves index–query compatibility without any refitting.

Algorithm 2 SKETCHAGGREGATE: CountSketch Feature Construction

Require: A sample x; LM head $W_{\text{lm_head}}$; CountSketch operators $\mathrm{CS}_r, \mathrm{CS}_h, \mathrm{CS}_g$ with dims (K_r, K_h, K_g); temperature τ; threshold ρ_{cum}; support limits $K_{\min}, K_{\max}$; feature amplitudes (a_{rh}, a_{gh}); norm floor $\epsilon > 0$; option `normalize_sample`.

Run the LM on x to obtain logits $\{z_t\}_{t=1}^T$, final hidden states $\{h_t\}_{t=1}^T$, and next-token targets $\{y_t\}_{t=1}^T$.
$u_{RH} \leftarrow \mathbf{0} \in \mathbb{R}^{K_r K_h}$; $u_{GH} \leftarrow \mathbf{0} \in \mathbb{R}^{K_g K_h}$; $K \leftarrow \min\{V, \max\{K_{\min}, K_{\max}\}\}$.
Define $N_\epsilon(v) := v/\max\{\|v\|_2, \epsilon\}$.
for $t = 1, \ldots, T$ **do**
  $\pi_t \leftarrow$ ordered top-K vocabulary indices under z_t/τ.
  If $y_t \notin \pi_t$, replace the last entry of π_t with y_t; let $\mathcal{C}_t$ be these K indices.
  $\bar{p}_t(v) \leftarrow \mathrm{softmax}\big((z_t/\tau)|_{\mathcal{C}_t}\big)(v)$ for $v \in \mathcal{C}_t$.
  $m_t \leftarrow \min\{s : \sum_{j=1}^{s} \bar{p}_t(\pi_t(j)) \geq \rho_{\text{cum}}\}$; use $m_t = K$ if this set is empty.
  $m_t \leftarrow \min\{K, \max\{K_{\min}, m_t\}\}$; $\mathcal{S}_t \leftarrow \{\pi_t(1), \ldots, \pi_t(m_t)\} \cup \{y_t\}$.
  $\tilde{p}_t \leftarrow$ renormalize $\bar{p}_t$ on $\mathcal{S}_t$; set other entries to zero.
  $\tilde{r}_t(v) \leftarrow \tilde{p}_t(v) - \mathbf{1}[v = y_t]$ for $v \in \mathcal{S}_t$.
  $\tilde{g}_t \leftarrow \sum_{v \in \mathcal{S}_t} \tilde{p}_t(v) W_v - W_{y_t}$.
  $\hat{r}_t \leftarrow N_\epsilon(\mathrm{CS}_r(\tilde{r}_t))$; $\hat{h}_t \leftarrow N_\epsilon(\mathrm{CS}_h(h_t))$; $\hat{g}_t \leftarrow N_\epsilon(\mathrm{CS}_g(\tilde{g}_t))$.
  $u_{RH} \leftarrow u_{RH} + \mathrm{vec}(\hat{r}_t \otimes \hat{h}_t)$; $u_{GH} \leftarrow u_{GH} + \mathrm{vec}(\hat{g}_t \otimes \hat{h}_t)$.
end for
$v(x) \leftarrow [a_{rh} u_{RH}; a_{gh} u_{GH}]$.
if `normalize_sample` **then**
  $v(x) \leftarrow N_\epsilon(v(x))$.
end if
return $\phi(x) = v(x)$.

Storage. PCA requires storing dense projection parameters of sizes $V \times K_r$, $d \times K_h$, and $d \times K_g$. CountSketch is specified by the hash/sign mappings (σ, s), whose description scales linearly with the input dimension.

Computation on sparse residuals. Our method constructs a truncated residual $\tilde{r}_t$ supported on $\mathcal{I}_t$ with $L_t = |\mathcal{I}_t|$. CountSketch directly exploits this sparsity, whereas PCA projections typically mix coordinates densely.

Proposition B.2 (Direct evaluation costs of CountSketch and dense PCA projections)**.** *Fix a token position t and consider the truncated residual $\tilde{r}_t \in \mathbb{R}^V$ supported on $\mathcal{I}_t$ with $L_t = |\mathcal{I}_t|$. Computing $\mathrm{CS}_r(\tilde{r}_t)$ requires one signed accumulation per $v \in \mathcal{I}_t$ (hence L_t updates), because each input coordinate is mapped to exactly one bucket by σ.*

For a PCA projection $P_r \tilde{r}_t$, if each of the K_r output coordinates depends on every input coordinate in $\mathcal{I}_t$ (i.e., P_r is dense on the columns indexed by $\mathcal{I}_t$), then direct evaluation of $P_r \tilde{r}_t$ uses $L_t \cdot K_r$ multiply-add contributions.

Proof. By definition of CountSketch, each input coordinate v contributes to exactly one output coordinate indexed by $\sigma(v)$. Since $\tilde{r}_t(v) = 0$ for $v \notin \mathcal{I}_t$, only indices in $\mathcal{I}_t$ contribute, and each such v causes exactly one signed update. Thus, $\mathrm{CS}_r(\tilde{r}_t)$ is computed with L_t updates.

For the PCA projection, each output coordinate is a linear combination of the input coordinates. Under the stated density condition, each nonzero input coordinate $v \in \mathcal{I}_t$ contributes to all K_r outputs, so it induces at least K_r multiply-add contributions. Summing over L_t nonzeros yields the operation count $L_t \cdot K_r$. This counts direct evaluation, not a lower bound for every possible algorithm. □

Dense factors. For dense vectors $h_t \in \mathbb{R}^d$ and $g_t \in \mathbb{R}^d$, applying PCA projections $P_h h_t$ and $P_g g_t$ involves dense matrix-vector products (mixing all d coordinates into K_h or K_g outputs), whereas CountSketch uses one bucket update per input coordinate by construction.

Implementation weight convention. Algorithm 2 separates feature amplitudes a_{rh}, a_{gh} from linear score coefficients. The implementation parameters `lambda_rh` and `lambda_gh` multiply the channel features before concatenation. With sample normalization enabled, the stored signature is

$$\phi(x) = \frac{[a_{rh}u_{RH}(x); a_{gh}u_{GH}(x)]}{n(x)}, \qquad n(x) = \max\left\{\sqrt{a_{rh}^2\|u_{RH}(x)\|_2^2 + a_{gh}^2\|u_{GH}(x)\|_2^2}, \epsilon\right\}.$$

Thus its dot-product score has numerator $a_{rh}^2\langle u_{RH}(x_i), u_{RH}(x_q)\rangle + a_{gh}^2\langle u_{GH}(x_i), u_{GH}(x_q)\rangle$ and denominator $n(x_i)n(x_q)$. Before sample normalization, the corresponding linear fusion coefficients are $\lambda_{rh} = a_{rh}^2$ and $\lambda_{gh} = a_{gh}^2$. The support rule shown is the released implementation's default ground-truth replacement mode; experiment-specific settings are reported with the corresponding results.

C Full Related Work

Training-data valuation and selection. Data valuation aims to rank candidate examples by their expected utility for improving specific model behaviors or downstream performance. Shapley-style valuation gives an axiomatic notion of equitable data value [28], with efficient approximations [29] and noise-reduced variants [30]. These methods are complementary to *RISE*: they offer a fairness interpretation under an explicit utility, while *RISE* targets reusable forward-only attribution at LLM scale. Scalable selectors often rely on *model-agnostic similarity proxies*, including lexical term matching such as BM25 [19] and dense embedding similarity computed by off-the-shelf encoders (BERT-family encoders, E5, and Gecko) [31, 21, 32]. Such embeddings have been used for semantic deduplication and diversity-aware curation (SemDeDup and D4) [33, 34]. However, these embeddings are not optimized for pretraining curation and may only partially reflect pretraining-loss similarity or training dynamics [35]. Influence-based selectors address this gap by leveraging signals directly from the target model.

Influence functions and scalable attribution for LLMs. Influence functions quantify how individual training examples affect a target behavior [1], but their second-order requirements (Hessian inverse-vector products) are impractical at LLM scale. Broader surveys summarize the many notions and approximations of training-data influence [36, 37]. Recent scalable approaches include TracIn, which accumulates first-order gradient similarity across checkpoints [5]; TRAK, which uses random projections and model ensembles to attribute model behavior [26]; RapidIn, which compresses per-example gradients with OPORP random projections for indexing and retrieval [11, 38]; TrackStar, which incorporates optimizer-aware corrections and normalization for pretraining-scale attribution [22]; Grosse et al., which adapt EK-FAC-style influence estimation to transformer language models [3]; Choe et al. (LoGra), which improve scalability through projected multi-layer gradients for LLM-scale data valuation [4]; and ZO-Inf, which uses forward-only zeroth-order estimators to trade additional forward evaluations for lower activation memory [12, 39, 40, 41]. *RISE* is an output-side, Hessian-free, forward-only estimator. It is motivated by measured readout concentration and the residual–hidden factorization of LM-head gradients, and it adds a complementary GH channel. Among output-side, forward-only methods, *RISE* differs from For-Value [8] in three ways: it adds the semantic GH channel to the residual–hidden (RH) form, builds sparse support independently at each token position rather than over a batch-shared vocabulary, and stores fixed-width CountSketch signatures that form a reusable index (Appendix F.6).

Output-side representations and Hessian-free scaling. Conceptually related work has examined layer choice and representation-only baselines, including Representer Point Selection [42], first-vs-last layer influence [43], and support-vector effects in DNNs [44]. However, these works are derived in settings that differ from ours, such as fine-tuned discriminative models or classification DNNs, and therefore do not directly determine whether representation-only similarity is sufficient for pretrained autoregressive decoder LLMs. The study *Better Hessians Matter* [45] reports that improved Hessian approximations can improve attribution in controlled smaller-scale settings. However, this evidence does not straightforwardly transfer to our regime: the paper studies small MLPs, explicitly notes that its conclusions may not transfer to larger models, and excludes CNNs and transformers, whose curvature structure differs substantially from autoregressive decoder LLMs. Its Figure 2 further shows that increasing depth lowers attribution fidelity and increases approximation errors in the studied MLPs. More broadly, scalable influence estimation has long relied on first-order surrogates such as TracIn rather than explicit Hessian correction [5]. Compared with dense Gaussian/Rademacher

random projections [46], *RISE* uses CountSketch [7], a standard sketch, because the truncated vocabulary residual is sparse and can be projected efficiently via scatter-add updates.

Neuroscience inspiration. Many influence estimators focus on retrospective attribution, whereas data valuation also demands a prospective view: scoring candidate data before learning. Recent works begin to study this prospective setting by fitting auxiliary surrogates to predict influence signals like ALinFiK and MATES [6, 47]. *RISE* unifies retrospective and prospective influence using only the target model, without auxiliaries or distillation. We draw functional inspiration from a readout bottleneck in biological decision-making: distributed representations are ultimately mapped to compact action-driving variables. OFC/vmPFC are often described as encoding a common value code for diverse inputs [48, 49] within cortico-striato-thalamic loops [50, 51], motivating why influence may concentrate near a model's output-side readout. Retrospective attribution parallels hippocampal replay linking outcomes to a small set of past episodes [52], while prospective valuation parallels uncertainty-driven sampling and update modulation associated with ACC and the LC–NE system [53, 54, 55, 56]. *RISE*'s dual channels can be viewed as complementary routes emphasizing surface cues versus higher-level predictive structure [57]. These links are offered as inspiration rather than mechanistic equivalence.

D Variance Analysis of *RISE* in Estimating Influence Function

This section analyzes the sketching variance of (i) TracIn-style random projection estimators and (ii) unnormalized RISE-style factorized CountSketch estimators exploiting the LM-head outer-product structure. A key point is to separate:

- **Sketching variance:** randomness due to projection/sketching (this section).
- **Approximation bias:** restricting to the LM head, truncating the residual, and changing the similarity target through GH fusion or normalization.

D.1 Targets, Estimators, and Assumptions

Loss and temperature convention. For the exact gradient identities in this section, $\ell(x)$ is the sum of token-level cross-entropies at unit temperature, so $p_t = \mathrm{softmax}(z_t)$ and $r_t = p_t - \mathbf{1}_{y_t}$. The CountSketch identities also apply to residual features formed at any fixed temperature. For ordinary cross-entropy with $p_t = \mathrm{softmax}(z_t/\tau)$, however, the readout gradient is $\tau^{-1} r_t h_t^\top$ and its inner-product kernel includes a factor τ^{-2}.

Full-parameter influence (TracIn target). Let $\nabla_\theta \ell(x) \in \mathbb{R}^{|\theta|}$. The full influence surrogate is

$$\mathcal{I}^{\mathrm{true}}(x_i \to x_q) := \langle \nabla_\theta \ell(x_q), \nabla_\theta \ell(x_i) \rangle .$$

LM-head influence (RISE structural target). Let $W_{\mathrm{lm_head}} \in \mathbb{R}^{V \times d}$ and $\nabla_{W_{\mathrm{lm_head}}} \ell(x) \in \mathbb{R}^{V \times d}$. For a sequence x of length T, the LM-head gradient admits the exact outer-product decomposition

$$\nabla_{W_{\mathrm{lm_head}}} \ell(x) = \sum_{t=1}^{T} r_t(x)\, h_t(x)^\top, \qquad r_t(x) = p_t(x) - \mathbf{1}_{y_t} \in \mathbb{R}^V, \quad h_t(x) \in \mathbb{R}^d.$$

The corresponding LM-head influence is

$$\mathcal{I}^{\mathrm{head}}(x_i \to x_q) := \left\langle \nabla_{W_{\mathrm{lm_head}}} \ell(x_q), \nabla_{W_{\mathrm{lm_head}}} \ell(x_i) \right\rangle_F .$$

Truncation (approximation target). Let $\tilde{r}_t$ be a sparse approximation to r_t (e.g., top-mass truncation with optional renormalization). Define the truncated head gradient

$$\widetilde{\nabla_{W_{\mathrm{lm_head}}} \ell(x)} := \sum_{t=1}^{T} \tilde{r}_t(x)\, h_t(x)^\top,$$

$$\tilde{\mathcal{I}}^{\mathrm{head}}(x_i \to x_q) := \left\langle \widetilde{\nabla_{W_{\mathrm{lm_head}}} \ell(x_q)}, \widetilde{\nabla_{W_{\mathrm{lm_head}}} \ell(x_i)} \right\rangle_F .$$

Important: unnormalized RH sketching is unbiased for $\tilde{\mathcal{I}}^{\mathrm{head}}$ when all token-pair contributions are included. GH fusion changes this target, and normalization generally introduces bias; neither is an unbiased estimator of $\mathcal{I}^{\mathrm{true}}$ in general.

TracIn estimator. Let $R \in \mathbb{R}^{K\times|\theta|}$ have i.i.d. entries $R_{kj} \sim \mathcal{N}(0, 1/K)$ (the Gaussian case) or Rademacher $\pm 1/\sqrt{K}$ (the same variance upper bound applies). The TracIn estimator is

$$\widehat{\mathcal{I}}^{\mathrm{TracIn}} := \langle R\,\nabla_\theta \ell(x_q),\, R\,\nabla_\theta \ell(x_i)\rangle\,.$$

CountSketch operator. A CountSketch $\mathrm{CS} : \mathbb{R}^D \to \mathbb{R}^K$ is defined by a bucket hash $\eta : [D] \to [K]$ and a sign hash $s : [D] \to \{\pm 1\}$:

$$(\mathrm{CS}(x))_k = \sum_{j:\eta(j)=k} s(j)\, x_j.$$

We assume the hash families used for different factors (residual/hidden/projection), i.e., (η_r, s_r), (η_h, s_h), (η_g, s_g), are independent. Within each family, the bucket hash has collision probability $1/K$ for distinct coordinates and is independent of the four-wise independent Rademacher sign hash.

RISE RH/GH estimators (analysis version, *no per-vector normalization*). For a single token position t in each sample (token-level analysis), define the following quantities. The same analysis applies to any fixed pair of positions, which need not have the same index:

$$A_t := \big\langle \mathrm{CS}_r(\tilde{r}_t(x_q)), \mathrm{CS}_r(\tilde{r}_t(x_i))\big\rangle, \quad B_t := \big\langle \mathrm{CS}_h(h_t(x_q)), \mathrm{CS}_h(h_t(x_i))\big\rangle,$$

and the RH estimator

$$\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t} := A_t\, B_t.$$

For GH, let $\tilde{g}_t(x) := W_{\mathrm{lm_head}}^\top \tilde{r}_t(x) \in \mathbb{R}^d$ and

$$C_t := \big\langle \mathrm{CS}_g(\tilde{g}_t(x_q)), \mathrm{CS}_g(\tilde{g}_t(x_i))\big\rangle, \qquad \widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t} := C_t\, B_t.$$

A two-channel fused estimator at token t is

$$\widehat{\mathcal{I}}^{\mathrm{RISE}}_t := \lambda_{rh}\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t} + \lambda_{gh}\widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t}.$$

Here $\lambda_{rh}, \lambda_{gh}$ are linear score coefficients, equal to a_{rh}^2, a_{gh}^2 for feature amplitudes a_{rh}, a_{gh} before normalization (Algorithm 2). Its expectation is the fused target

$$\tilde{\mathcal{I}}^{\mathrm{fused}}_t := \Big(\lambda_{rh}\langle \tilde{r}_t(x_q), \tilde{r}_t(x_i)\rangle + \lambda_{gh}\langle \tilde{g}_t(x_q), \tilde{g}_t(x_i)\rangle\Big)\langle h_t(x_q), h_t(x_i)\rangle,$$

which generally differs from the truncated LM-head target.

Remark (about the implementation normalization). If the implementation uses $\hat{r}_t = \mathrm{CS}_r(\tilde{r}_t)/\|\mathrm{CS}_r(\tilde{r}_t)\|_2$ and similarly for $\hat{h}_t, \hat{g}_t$, or normalizes the final sample vector, the resulting estimator is generally *biased* relative to the corresponding uncompressed inner-product target. The variance bounds below apply only to the *unnormalized* version. At sequence level, the shared sketches also induce covariance between token-pair contributions; the token-level bounds below do not assume these contributions are independent.

D.1.1 Assumptions

Assumption D.1 (Bounded norms)**.** There exist constants $B_r, B_{\tilde{r}}, B_h > 0$ such that for all x, t:

$$\|r_t(x)\|_2 \le B_r, \qquad \|\tilde{r}_t(x)\|_2 \le B_{\tilde{r}}, \qquad \|h_t(x)\|_2 \le B_h.$$

Assumption D.2 (Bounded full gradient)**.** There exists $B_\theta > 0$ such that $\|\nabla_\theta \ell(x)\|_2 \le B_\theta$ for all x.

Assumption D.3 (Truncation error bound)**.** There exists $\varepsilon_{\mathrm{trunc}} \ge 0$ such that for all x, t,

$$\|r_t(x) - \tilde{r}_t(x)\|_2 \le \varepsilon_{\mathrm{trunc}}.$$

Assumption D.4 (LM-head operator norm)**.** $\|W_{\mathrm{lm_head}}\|_{op} = \sigma_1(W_{\mathrm{lm_head}})$, hence $\|\tilde{g}_t(x)\|_2 = \|W_{\mathrm{lm_head}}^\top \tilde{r}_t(x)\|_2 \le \sigma_1(W_{\mathrm{lm_head}})\|\tilde{r}_t(x)\|_2 \le \sigma_1(W_{\mathrm{lm_head}})B_{\tilde{r}}$.

Assumption D.5 (Energy concentration into LM head (optional, for comparing to full TracIn))**.** There exists $\gamma \in (0, 1]$ such that for all x,

$$\|\nabla_{W_{\mathrm{lm_head}}} \ell(x)\|_F^2 \le \gamma \|\nabla_\theta \ell(x)\|_2^2.$$

D.2 Supporting Lemmas

Lemma D.1 (Random projection inner-product variance (Gaussian)). *Let $R \in \mathbb{R}^{K\times|\theta|}$ have i.i.d. entries $\mathcal{N}(0, 1/K)$. For any $u, v \in \mathbb{R}^{|\theta|}$, define $\widehat{Z} = \langle Ru, Rv\rangle$. Then*

$$\mathbb{E}[\widehat{Z}] = u^\top v, \qquad \mathrm{Var}[\widehat{Z}] = \frac{1}{K}\Big(\|u\|_2^2\|v\|_2^2 + (u^\top v)^2\Big).$$

(For Rademacher $\pm 1/\sqrt{K}$, the Gaussian variance expression is an upper bound.)

Proof. Write R by rows: $R = [r_1^\top; \ldots; r_K^\top]$ with $r_k \sim \mathcal{N}(0, \frac{1}{K}I)$ independent. Let $X_k = (r_k^\top u)(r_k^\top v)$. Then $\widehat{Z} = \sum_{k=1}^K X_k$ and the X_k are i.i.d. By isotropy, $\mathbb{E}[X_k] = \frac{1}{K}u^\top v$. By Isserlis' theorem for zero-mean Gaussians,

$$\mathbb{E}[X_k^2] = \mathbb{E}[(r_k^\top u)^2(r_k^\top v)^2] = \frac{1}{K^2}\Big(\|u\|_2^2\|v\|_2^2 + 2(u^\top v)^2\Big).$$

Hence

$$\mathrm{Var}(X_k) = \mathbb{E}[X_k^2] - \mathbb{E}[X_k]^2 = \frac{1}{K^2}\Big(\|u\|_2^2\|v\|_2^2 + (u^\top v)^2\Big),$$

and $\mathrm{Var}(\widehat{Z}) = K\,\mathrm{Var}(X_1)$ gives the claim. □

Lemma D.2 (CountSketch inner-product unbiasedness and variance). *Let $\mathrm{CS} : \mathbb{R}^D \to \mathbb{R}^K$ be CountSketch with a bucket hash η such that $\mathbb{P}(\eta(i) = \eta(j)) = 1/K$ for $i \neq j$, and a four-wise independent Rademacher sign hash s independent of η. For any $x, y \in \mathbb{R}^D$, define $\widehat{Z} = \langle \mathrm{CS}(x), \mathrm{CS}(y)\rangle$. Then*

$$\mathbb{E}[\widehat{Z}] = x^\top y,$$
$$\mathrm{Var}[\widehat{Z}] = \frac{\|x\|_2^2\|y\|_2^2 + (x^\top y)^2 - 2\sum_i x_i^2 y_i^2}{K} \le \frac{2}{K}\|x\|_2^2\|y\|_2^2.$$

Proof. Expand

$$\widehat{Z} = \sum_{k=1}^K \Big(\sum_{i:\eta(i)=k} s(i)x_i\Big)\Big(\sum_{j:\eta(j)=k} s(j)y_j\Big) = \sum_{i,j} x_i y_j\, s(i)s(j)\,\mathbf{1}[\eta(i) = \eta(j)].$$

Taking expectation over s, the terms with $i \neq j$ vanish because $\mathbb{E}[s(i)s(j)] = 0$, leaving $\mathbb{E}[\widehat{Z}] = \sum_i x_i y_i = x^\top y$.

Let $E := \widehat{Z} - x^\top y = \sum_{i\neq j} x_i y_j\, s(i)s(j)\,\mathbf{1}[\eta(i) = \eta(j)]$. Then $\mathbb{E}[E] = 0$ and $\mathrm{Var}(\widehat{Z}) = \mathbb{E}[E^2]$. Four-wise sign independence makes cross terms vanish except for the equal-index and swapped-index pairings. Therefore,

$$\mathbb{E}[E^2] = \frac{1}{K}\sum_{i\neq j}\big(x_i^2 y_j^2 + x_i y_j x_j y_i\big)$$
$$= \frac{\|x\|_2^2\|y\|_2^2 + (x^\top y)^2 - 2\sum_i x_i^2 y_i^2}{K} \le \frac{2}{K}\|x\|_2^2\|y\|_2^2,$$

where the last inequality uses Cauchy–Schwarz. □

Lemma D.3 (Variance of a product of independent CountSketch inner products). *Let*

$$A = \langle \mathrm{CS}_r(\tilde{r}), \mathrm{CS}_r(\tilde{r}')\rangle, \quad B = \langle \mathrm{CS}_h(h), \mathrm{CS}_h(h')\rangle,$$

where CS_r and CS_h use independent hash families with output sizes K_r and K_h. Define $\alpha = \langle \tilde{r}, \tilde{r}'\rangle$ and $\beta = \langle h, h'\rangle$. Then $\mathbb{E}[AB] = \alpha\beta$ and

$$\mathrm{Var}[AB] \le \frac{4\|\tilde{r}\|_2^2\|\tilde{r}'\|_2^2\|h\|_2^2\|h'\|_2^2}{K_r K_h} + \frac{2\alpha^2\|h\|_2^2\|h'\|_2^2}{K_h} + \frac{2\beta^2\|\tilde{r}\|_2^2\|\tilde{r}'\|_2^2}{K_r}.$$

Proof. By Lemma D.2, $\mathbb{E}[A] = \alpha$ and $\mathbb{E}[B] = \beta$. Independence of the sketch randomness implies A and B are independent, hence $\mathbb{E}[AB] = \mathbb{E}[A]\mathbb{E}[B] = \alpha\beta$.

For independent A, B,

$$\begin{aligned}\mathrm{Var}(AB) &= \mathbb{E}[A^2]\mathbb{E}[B^2] - (\mathbb{E}[A]\mathbb{E}[B])^2 \\ &= (\mathrm{Var}(A) + \alpha^2)(\mathrm{Var}(B) + \beta^2) - \alpha^2\beta^2 \\ &= \mathrm{Var}(A)\mathrm{Var}(B) + \alpha^2\mathrm{Var}(B) + \beta^2\mathrm{Var}(A).\end{aligned}$$

Using Lemma D.2,

$$\mathrm{Var}(A) \le \frac{2\|\tilde{r}\|_2^2\|\tilde{r}'\|_2^2}{K_r}, \qquad \mathrm{Var}(B) \le \frac{2\|h\|_2^2\|h'\|_2^2}{K_h},$$

and substituting yields the stated bound. □

Lemma D.4 (Renormalized top-mass truncation implies an explicit ℓ_2 error bound)**.** *Let $p_t \in \Delta^V$ and $\mathcal{S}_t \subset [V]$ with mass $\rho = \sum_{v\in\mathcal{S}_t} p_t(v) > 0$. Define $\tilde{p}_t(v) = p_t(v)/\rho$ for $v \in \mathcal{S}_t$ and 0 otherwise. Then*

$$\|p_t - \tilde{p}_t\|_1 = 2(1-\rho), \qquad \|p_t - \tilde{p}_t\|_2 \le 2(1-\rho).$$

Consequently, for residuals $r_t = p_t - \mathbf{1}_{y_t}$ and $\tilde{r}_t = \tilde{p}_t - \mathbf{1}_{y_t}$ we have $\|r_t - \tilde{r}_t\|_2 \le 2(1-\rho)$.

Proof. Outside $\mathcal{S}_t$, $\tilde{p}_t = 0$, so $\|p_t - \tilde{p}_t\|_1$ contributes $\sum_{v\notin\mathcal{S}_t} p_t(v) = 1-\rho$. Inside $\mathcal{S}_t$, $p_t - \tilde{p}_t = p_t - p_t/\rho = p_t(1-1/\rho)$ and $|1-1/\rho| = (1-\rho)/\rho$, hence the ℓ_1 contribution is $\sum_{v\in\mathcal{S}_t} p_t(v)\cdot(1-\rho)/\rho = (1-\rho)$. Thus $\|p_t - \tilde{p}_t\|_1 = 2(1-\rho)$ and $\|p_t - \tilde{p}_t\|_2 \le \|p_t - \tilde{p}_t\|_1$. Finally $r_t - \tilde{r}_t = p_t - \tilde{p}_t$. □

D.3 Main Variance Theorem

Theorem D.1 (Variance bounds for unnormalized RISE and TracIn)**.** *Fix a training-query pair (x_i, x_q) and a token position t (token-level analysis). Under Assumptions D.1–D.4:*

(i) TracIn variance (full gradient). *Let $u = \nabla_\theta \ell(x_q)$ and $v = \nabla_\theta \ell(x_i)$, and let R be Gaussian as in Lemma D.1. Then*

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}^{\mathrm{TracIn}}\Big] = \frac{1}{K}\Big(\|u\|_2^2\|v\|_2^2 + (u^\top v)^2\Big) \le \frac{2B_\theta^4}{K}.$$

(ii) RH channel variance (token-level). *Let $\tilde{r}^q = \tilde{r}_t(x_q)$, $\tilde{r}^i = \tilde{r}_t(x_i)$, $h^q = h_t(x_q)$, $h^i = h_t(x_i)$ and define $\alpha = \langle \tilde{r}^q, \tilde{r}^i\rangle$, $\beta = \langle h^q, h^i\rangle$. Then*

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t}\Big] \le \frac{4\|\tilde{r}^q\|_2^2\|\tilde{r}^i\|_2^2\|h^q\|_2^2\|h^i\|_2^2}{K_rK_h} + \frac{2\alpha^2\|h^q\|_2^2\|h^i\|_2^2}{K_h} + \frac{2\beta^2\|\tilde{r}^q\|_2^2\|\tilde{r}^i\|_2^2}{K_r}.$$

In particular, using $\|\tilde{r}\| \le B_{\tilde{r}}$ and $\|h\| \le B_h$ gives the worst-case bound

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t}\Big] \le B_{\tilde{r}}^4 B_h^4\Big(\frac{4}{K_rK_h} + \frac{2}{K_h} + \frac{2}{K_r}\Big).$$

(iii) GH channel variance (token-level). *Let $\tilde{g}^q = W_{\mathrm{lm_head}}^\top \tilde{r}^q$, $\tilde{g}^i = W_{\mathrm{lm_head}}^\top \tilde{r}^i$, and $\gamma_g = \langle \tilde{g}^q, \tilde{g}^i\rangle$. Then*

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t}\Big] \le \frac{4\|\tilde{g}^q\|_2^2\|\tilde{g}^i\|_2^2\|h^q\|_2^2\|h^i\|_2^2}{K_gK_h} + \frac{2\gamma_g^2\|h^q\|_2^2\|h^i\|_2^2}{K_h} + \frac{2\beta^2\|\tilde{g}^q\|_2^2\|\tilde{g}^i\|_2^2}{K_g}.$$

Moreover, $\|\tilde{g}\|_2 \le \sigma_1(W_{\mathrm{lm_head}})B_{\tilde{r}}$ implies the worst-case bound

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t}\Big] \le \sigma_1(W_{\mathrm{lm_head}})^4 B_{\tilde{r}}^4 B_h^4\Big(\frac{4}{K_gK_h} + \frac{2}{K_h} + \frac{2}{K_g}\Big).$$

(iv) Two-channel fusion and covariance. *Let $\widehat{\mathcal{I}}_t^{\mathrm{RISE}} = \lambda_{rh}\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t} + \lambda_{gh}\widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t}$. If the RH and GH channels share the same hidden-state sketch B_t but use independent residual/projection sketches (CS_r independent of CS_g), then*

$$\mathrm{Var}\Big[\widehat{\mathcal{I}}_t^{\mathrm{RISE}}\Big] = \lambda_{rh}^2\mathrm{Var}[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{RH,t}] + \lambda_{gh}^2\mathrm{Var}[\widehat{\mathcal{I}}^{\mathrm{RISE}}_{GH,t}] + 2\lambda_{rh}\lambda_{gh}\,\mathrm{Cov}[A_tB_t, C_tB_t]\,,$$

and the covariance admits the exact identity

$$\mathrm{Cov}[A_tB_t, C_tB_t] = \mathbb{E}[A_tC_t]\mathrm{Var}(B_t) = \alpha\,\gamma_g\,\mathrm{Var}(B_t),$$

where $\mathrm{Var}(B_t) \le 2\|h^q\|_2^2\|h^i\|_2^2/K_h$ *by Lemma D.2. (If RH and GH use independent hidden sketches, then the covariance term is zero.)*

(v) Scope of the comparison with TracIn.

- *Estimator targets: The unnormalized RH estimator is unbiased for the corresponding token-pair contribution to* $\tilde{\mathcal{I}}^{\mathrm{head}}$*. The fused estimator is unbiased for* $\tilde{\mathcal{I}}_t^{\mathrm{fused}}$*, a different target. These claims do not extend to the normalized implementation.*
- *Matched feature dimensions: For RH alone, fixing* $K_rK_h = K$ *makes the product term in (ii) scale as* $1/K$*, but the other terms scale as* $1/K_r$ *and* $1/K_h$ *and need not be smaller. With* $K_r = K_h = \sqrt{K}$*, the worst-case bound is* $O(K^{-1/2})$ *for bounded input norms. The fused feature dimension is* $K_h(K_r + K_g)$*. Thus these bounds do not establish a universal variance advantage over a* K*-dimensional full-gradient projection, nor a fixed reduction factor from the LM-head energy fraction alone.*
- *Effect of truncation: Under non-expansive truncation,* $\|\tilde{r}_t\|_2 \le \|r_t\|_2$*, the norm product* $\|\tilde{r}^q\|_2^2\|\tilde{r}^i\|_2^2$ *in the bound cannot increase. This condition holds for zero-masking truncation, but not necessarily after renormalization. A smaller norm product is not, by itself, a guarantee of smaller actual variance; the inner-product terms also matter.*

Proof. Part (i) is Lemma D.1 applied to $u = \nabla_\theta \ell(x_q)$ and $v = \nabla_\theta \ell(x_i)$ plus Cauchy–Schwarz.

Parts (ii) and (iii) follow by applying Lemma D.3 to $(\tilde{r}, \tilde{r}') = (\tilde{r}^q, \tilde{r}^i)$ and $(h, h') = (h^q, h^i)$ for RH, and $(\tilde{r}, \tilde{r}') = (\tilde{g}^q, \tilde{g}^i)$ and $(h, h') = (h^q, h^i)$ for GH, together with Assumption D.4.

For (iv), write $\widehat{\mathcal{I}}_{RH,t} = A_tB_t$ and $\widehat{\mathcal{I}}_{GH,t} = C_tB_t$. Since (A_t, C_t) are independent of B_t and A_t is independent of C_t (independent sketch families),

$$\begin{aligned}\mathrm{Cov}(A_tB_t, C_tB_t) &= \mathbb{E}[A_tC_tB_t^2] - \mathbb{E}[A_tB_t]\mathbb{E}[C_tB_t] \\ &= \mathbb{E}[A_tC_t]\big(\mathbb{E}[B_t^2] - \mathbb{E}[B_t]^2\big) = \mathbb{E}[A_tC_t]\mathrm{Var}(B_t).\end{aligned}$$

Moreover $\mathbb{E}[A_tC_t] = \mathbb{E}[A_t]\mathbb{E}[C_t] = \alpha\gamma_g$. □

D.4 Bias-Variance Decomposition

Let $\widehat{\mathcal{I}}_{RH}^{\mathrm{RISE}}$ denote the inner product of the two sequence-level features obtained by summing *unnormalized* RH outer-product sketches. It includes all token-pair contributions and satisfies $\mathbb{E}[\widehat{\mathcal{I}}_{RH}^{\mathrm{RISE}}] = \tilde{\mathcal{I}}^{\mathrm{head}}$. Its error relative to the full-gradient surrogate admits the decomposition

$$\widehat{\mathcal{I}}_{RH}^{\mathrm{RISE}} - \mathcal{I}^{\mathrm{true}} = \underbrace{\big(\widehat{\mathcal{I}}_{RH}^{\mathrm{RISE}} - \tilde{\mathcal{I}}^{\mathrm{head}}\big)}_{\text{Sketching Error (Stochastic)}} + \underbrace{\big(\tilde{\mathcal{I}}^{\mathrm{head}} - \mathcal{I}^{\mathrm{head}}\big)}_{\text{Truncation Bias (Deterministic)}} + \underbrace{\big(\mathcal{I}^{\mathrm{head}} - \mathcal{I}^{\mathrm{true}}\big)}_{\text{Structural Bias (Deterministic)}}.$$

Theorem D.1 bounds the sketching variance of each token-pair contribution; sequence-level variance also includes their covariances. For fixed samples, the first term has zero mean, so the mean-squared error equals its variance plus the square of the combined deterministic bias. Assumption D.3, the hidden-state norm bound, and Lemma D.4 control the truncation error. The structural term is the contribution omitted by restricting to the LM head.

For RH+GH fusion, the sketching error must instead be centered at the corresponding uncompressed fused target, and its difference from $\tilde{\mathcal{I}}^{\mathrm{head}}$ introduces an additional deterministic target shift. Nonlinear factor or sample normalization can introduce further bias. The three-term RH decomposition above therefore does not describe the total error of the normalized two-channel implementation without these additional terms.

E Hyperparameters

We present the experimental hyperparameters in Table 6. The weights denoted $\lambda_{rh}, \lambda_{gh}$ in experimental setting descriptions refer to the implementation parameters `lambda_rh`, `lambda_gh`, i.e., the

feature amplitudes a_{rh}, a_{gh} in Algorithm 2. They are distinct from the linear fusion coefficients used in the variance analysis; before sample normalization, those coefficients are a_{rh}^2, a_{gh}^2.

Table 6: Default experimental hyperparameters and sketch configurations for Table 2. Experiment-specific settings are reported with the corresponding results.

Parameter	Value	Description
RISE Configuration		
Channel Configuration		
`fusion_mode`	`"rh+gh"`	Active channels (RH + GH)
a_{rh}	0.7	RH feature amplitude (`lambda_rh`)
a_{gh}	1.0	GH feature amplitude (`lambda_gh`)
CountSketch Dimensions (Pythia-1B)		
K_r	128 / 64	Residual sketch dimension (Table 2)
K_h	24 / 12	Hidden state sketch dimension
K_g	128 / 64	Gradient sketch dimension
CountSketch Dimensions (OLMo-3-32B)		
K_r	48 / 24 / 16	Residual sketch dimension (Table 2)
K_h	64 / 8 / 8	Hidden state sketch dimension
K_g	16 / 35 / 28	Gradient sketch dimension
Adaptive Top-L		
`adaptive_topL`	True	Enable cumulative-probability truncation
θ_{cum}	0.92	Cumulative probability threshold
`min_topL`	4	Minimum kept candidates
`max_topL_cap`	256	Top-L upper bound
Temperature		
τ	0.1	Fixed temperature of the residual $\mathrm{softmax}(z/\tau) - \mathbf{1}_y$
Other		
`seq_len`	512	Maximum sequence length
`normalize_sample`	True	L2 normalize final vector
`seed`	42	Random seed for reproducibility
TrackStar Configuration		
`projection_dim`	16	Random projection dimension
`projection_type`	`rademacher`	Projection matrix type
`loss_fn`	`ce`	Cross-entropy loss
`loss_reduction`	`mean`	Loss reduction method
`include_bias`	False	Exclude bias gradients
`precision`	`bf16`	Model precision
RapidIn Configuration		
`RapidGrad_K`	65536	Compressed gradient dimension
`shuffle_lambda`	20	Shuffle parameter for RapidGrad
`use_zo`	False	Use backpropagation gradients
`max_length`	512	Maximum sequence length
`seed`	42	Random seed
ZO-Inf Configuration		
`RapidGrad_K`	65536	Compressed gradient dimension
`shuffle_lambda`	20	Shuffle parameter for RapidGrad
`use_zo`	True	Use zeroth-order gradients
`zo_eps`	0.001	Perturbation scale (ϵ)
`zo_sample`	4	Number of ZO samples per gradient
`max_length`	512	Maximum sequence length
`seed`	42	Random seed

F Additional Experimental Results

Testbed. All experiments are run on servers with the following configurations: H200 setup: 8 NVIDIA H200 GPUs (each with 141GB GPU memory); GH200 setup: 1 NVIDIA GH200 GPU (96GB GPU memory).

F.1 Complete Per-Layer Energy Distribution

This subsection provides comprehensive results on per-layer gradient energy distribution across different models and tasks. Figure 4 demonstrates that gradient-energy profiles remain largely stable

Table 7: Dataset composition for the three evaluation tasks.

Task	Pool Size	Positives	Pos %	Description
Howdy! Backdoor	5,000	438	8.8%	Alpaca-based pool; positives contain trigger `howdy!` [11]
Finance–Medical	5,000	500	10.0%	Medical QA [15] vs Finance [14]
Brain Rot	5,000	500	10.0%	High-quality control vs junk data [13]

Note: For Howdy!, we use 100 WebQuestions test generations containing the trigger. Positives for Finance–Medical are medical samples; for Brain Rot, positives are high-quality control samples.

before and after fine-tuning. Table 8 reports the full storage requirements for last-layer gradients, while Table 9 presents an ablation comparing LM-Head-Full-gradient sketch with *RISE*.

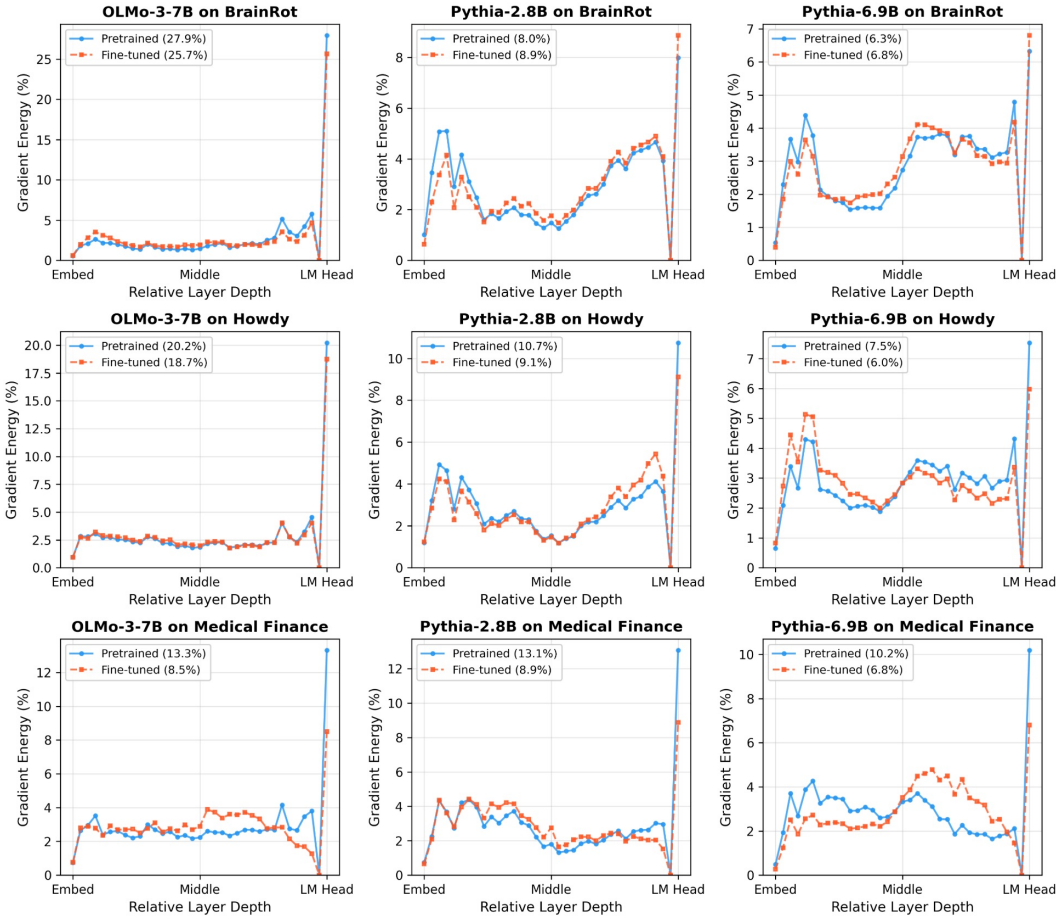

Figure 4: **Per-layer gradient-energy profiles are largely preserved after fine-tuning.** For three tasks (BrainRot, Howdy, Finance–Medical) and three models (OLMo-3-7B, Pythia-2.8B, Pythia-6.9B), we plot the fraction of total gradient energy attributed to each layer (Embed → LM Head) for the pretrained checkpoint (blue) and the fine-tuned checkpoint (orange). The curves nearly overlap; numbers in legends denote the LM-head energy share, which changes only mildly after fine-tuning. This stability motivates estimating layer importance (and selecting readout-side signals) from the pretrained model without task-specific per-layer gradient profiling.

F.2 Complete Per-Layer Discriminativeness

This subsection analyzes the layer-wise discriminativeness of hidden-state representations. Figure 5 shows the U-shaped discriminativeness pattern on C4 across model scales. Figure 6 further shows that this pattern persists both before and after fine-tuning across multiple models and tasks.

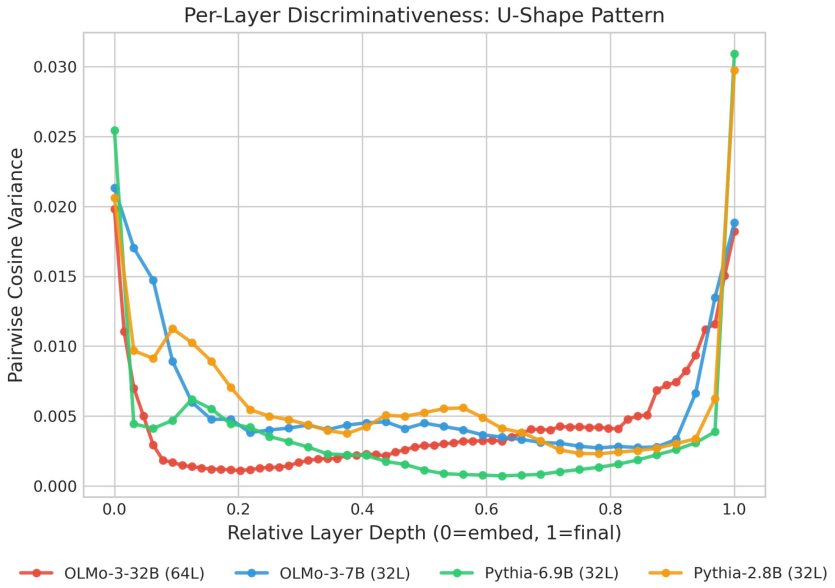

Figure 5: **Final-layer discriminativeness across model scales.** Per-layer pairwise cosine variance for four models on C4. A clear U-shape emerges: variance drops in middle layers due to hidden collapse and recovers near the final layer.

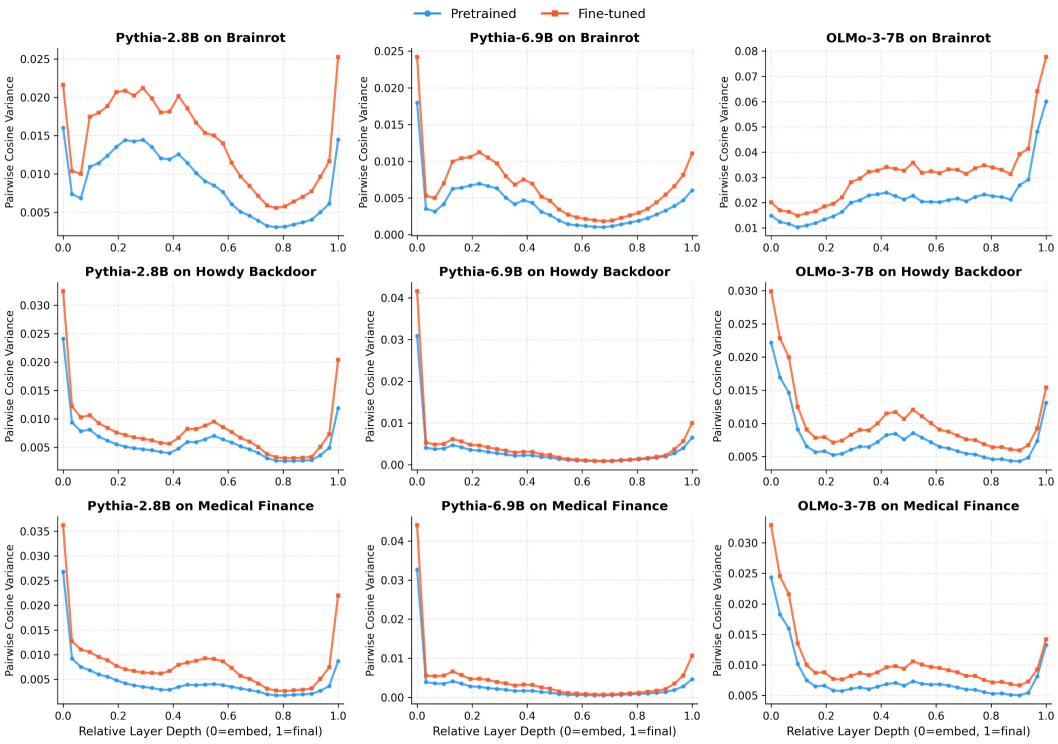

Figure 6: **Hidden-state discriminativeness follows a stable U-shape before and after fine-tuning.** For each layer, we compute the variance of pairwise cosine similarities among token hidden states (higher variance more discriminative representations). Across three fine-tuning tasks (BrainRot, Howdy Backdoor, Medical–Finance) and three models (Pythia-2.8B, Pythia-6.9B, OLMo-3-7B), pretrained (blue) and fine-tuned (orange) curves are highly consistent: variance is low in middle layers and recovers near the final layer (relative depth 0=embed, 1=final). This consistent U-shape suggests that the layer-wise discriminative structure is largely preserved under fine-tuning, motivating readout-side representations as a reliable hotspot for scalable influence estimation.

Table 8: Full Last-Layer Gradient Storage Requirements (N=5,000 samples, fp16)

Model	**Vocab**	**Hidden**	**V×D**	**Index Size**
Pythia-14M	50,304	128	6.4M	60 GB
Pythia-1B	50,304	2,048	103M	959 GB
OLMo-7B	100,352	4,096	411M	3.8 TB
OLMo-32B	100,352	5,120	514M	4.8 TB

Table 9: Ablation: LM-Head-Full-gradient sketch vs. *RISE* on three tasks (Pythia-1B). Grad-Sketch applies CountSketch to LM Head gradients; *RISE* restricts to the LM head.

Dataset	Method	Index	Top-5		Top-50	
			auPRC	auROC	auPRC	auROC
Fin–Med	Grad-Sketch	156 MB	**1.000**	**1.000**	**0.992**	0.986
	RISE	**58.6 MB**	0.989	0.994	0.981	**0.990**
BrainRot	Grad-Sketch	156 MB	**1.000**	**1.000**	0.724	0.872
	RISE	**58.6 MB**	0.847	0.949	**0.829**	**0.930**
Howdy	Grad-Sketch	156 MB	0.679	0.833	0.890	0.960
	RISE	**58.6 MB**	**0.996**	**0.997**	**0.939**	**0.967**

F.3 Additional Evidence for Sparse Active Tokens

This subsection provides the supporting evidence for Observation 3 (Section 2.4). The main text reports the headline result under a fixed-control: fixed τ when sweeping the number of candidate tokens, and fixed K when sweeping τ. Here we make the measurements explicit, separate probability mass from residual energy, and report the stability of sparse RH/GH fidelity across tasks, model scales, and checkpoints.

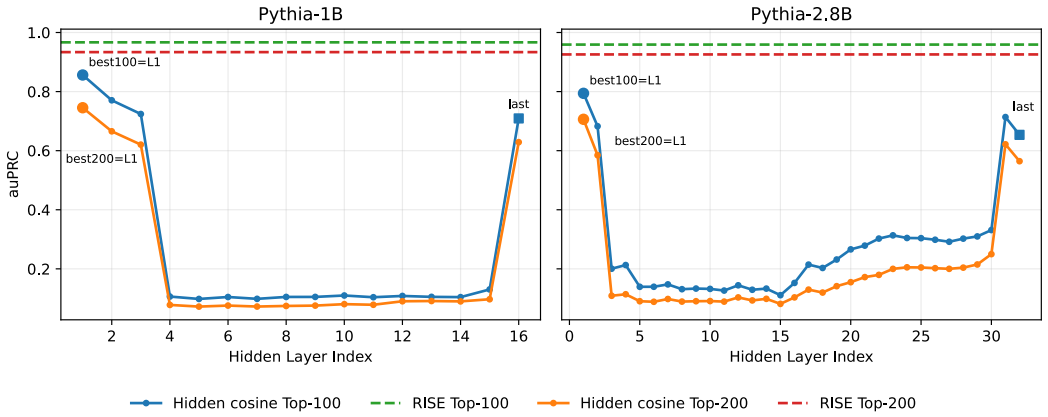

Figure 7: **Full hidden-layer retrieval sweep on Howdy.** We evaluate hidden-state cosine retrieval at every layer for Pythia-1B and Pythia-2.8B, and compare against *RISE* at Top-100 and Top-200. The strongest hidden-only layer is layer 1 in both models, not the final layer. However, even this best hidden-layer baseline remains below *RISE*, indicating that *RISE*'s gains are not merely due to selecting a stronger hidden representation layer.

Table 10: **Fixed $\tau = 1.0$ Top-K sweep for sparse GH.** Rows are averaged over eight BrainRot pretrained/fine-tuned Pythia and OLMo runs.

K	GH cosine	Prob. mass (%)	Support/V (%)
8	0.954	61.3	0.015
16	0.973	68.4	0.029
32	0.984	75.8	0.056
64	0.990	81.8	0.112
128	0.993	85.6	0.223
256	0.995	89.1	0.445

Table 11: **Sparse GH fidelity by BrainRot run at fixed $\tau = 1.0$, $K = 128$.** The support is fixed-size rather than adaptively selected.

Model	State	GH cosine	Prob. mass (%)	Support/V (%)
OLMo-1B	Fine-tuned	0.994	87.3	0.128
OLMo-1B	Pretrained	0.993	85.2	0.128
Pythia-160M	Fine-tuned	0.989	81.9	0.255
Pythia-160M	Pretrained	0.993	78.0	0.255
Pythia-1B	Fine-tuned	0.995	88.2	0.255
Pythia-1B	Pretrained	0.995	85.6	0.255
Pythia-6.9B	Fine-tuned	0.990	91.1	0.254
Pythia-6.9B	Pretrained	0.996	87.6	0.254

Metrics. For a token position t, let $p_t = \mathrm{softmax}(z_t/\tau)$ and $r_t = p_t - \mathbf{1}_{y_t}$. The active support $\mathcal{S}_t$ always includes the ground-truth next token y_t. We report:

$$M_t(\mathcal{S}_t) = \sum_{v \in \mathcal{S}_t} p_t(v) \qquad \text{(probability mass)},$$

$$E_{\text{full}}(\mathcal{S}_t) = \frac{\sum_{v \in \mathcal{S}_t} r_t(v)^2}{\sum_{v \in [V]} r_t(v)^2} \qquad \text{(full residual } \ell_2 \text{ energy)},$$

$$E_{\text{GT}} = \frac{(1 - p_t(y_t))^2}{\sum_{v \in [V]} r_t(v)^2} \qquad \text{(ground-truth coordinate energy)},$$

$$E_{\text{tail}}(\mathcal{S}_t) = \frac{\sum_{v \in \mathcal{S}_t \setminus \{y_t\}} p_t(v)^2}{\sum_{v \neq y_t} p_t(v)^2} \qquad \text{(non-GT residual-tail energy)}.$$

The distinction is important: probability mass is an ℓ_1 quantity, while influence through the LM-head residual uses inner products and is therefore governed by squared residual energy. Thus a support can cover moderate probability mass while preserving nearly all influence-relevant energy. We report E_{GT} separately because the forced ground-truth coordinate often dominates the full residual norm; the non-GT tail measures whether the selected active prediction tokens carry additional signal beyond GT-only.

Fixed-control protocol. Unless otherwise stated, the sparse support is $\mathcal{S}_t = \mathrm{TopK}(z_t/\tau, K) \cup \{y_t\}$, with logits renormalized only over $\mathcal{S}_t$. For the Top-K sweep we fix $\tau = 1.0$ and vary K. For the temperature sweep we fix $K = 128$ and vary τ. This avoids coupling temperature, candidate count, and adaptive cumulative-probability thresholds.

The main fixed-control diagnostics are shown in Figure 3. The RH panel shows that full residual energy is nearly lossless once the ground-truth coordinate is included, while the non-GT tail curve measures the candidate-token contribution against the full non-GT vocabulary tail. The GH panel shows that the same sparse candidate support preserves the dense semantic error direction. Figure 8 adds the complementary task/model matrix and fixed-K temperature sweep.

Fixed Top-K GH fidelity. Because the GH channel uses $g_t = W_{\text{lm_head}}^{\top} r_t$, truncating the residual could distort the semantic error direction even when RH remains accurate. We therefore compare dense GH with sparse GH using cosine similarity. At fixed $\tau = 1.0$ and $K = 128$, sparse GH reaches mean cosine 0.993 against dense GH across the eight BrainRot runs (Tables 10 and 11).

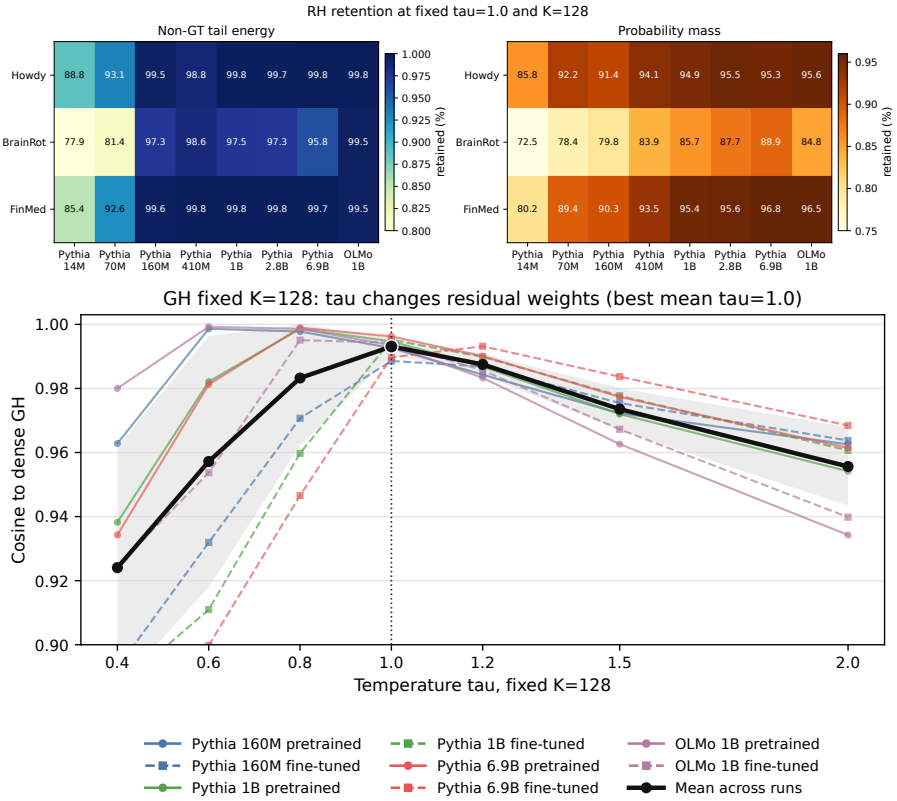

Figure 8: **Additional fixed-control sparse active-token diagnostics.** Top: at fixed $\tau = 1.0$ and $K = 128$, RH residual-tail retention remains stable across tasks, model scales, and pretrained/fine-tuned checkpoints. Bottom: with fixed $K = 128$, the GH temperature sweep changes only residual weights, showing that $\tau = 1.0$ gives the strongest average fidelity under a constant candidate count.

Table 12: **Fixed $K = 128$ temperature sweep for sparse GH.** Rows are averaged over eight BrainRot pretrained/fine-tuned model runs.

τ	GH cosine	Prob. mass (%)	Relative error
0.4	0.924	99.8	0.138
0.6	0.957	98.9	0.154
0.8	0.983	94.4	0.183
1.0	0.993	85.6	0.216
1.2	0.987	73.8	0.260
1.5	0.974	54.2	0.323
2.0	0.956	28.8	0.376

Fixed-K temperature sensitivity. Temperature controls how diffuse the softmax tail is before truncation. With K fixed at 128, the support size is constant; changing τ only changes the residual weights assigned inside the same-size candidate set. Table 12 shows that $\tau = 1.0$ gives the best average GH cosine in this sweep.

Takeaway. The sparse support is not merely a top-K engineering shortcut. It is a structural property of the LM-head residual: a tiny fixed-size active set captures the non-GT residual tail that matters for RH, and it also preserves the GH semantic direction. This explains why *RISE* can reduce the residual-side memory and sketching cost from vocabulary scale to active-support scale without sacrificing the error signal used by the dual-channel influence metric.

Table 13: **Target-model representation baselines at larger Top-K on Howdy prospective valuation.** All rows use the 5K Howdy candidate pool and 100 prospective WebQuestions trigger queries. Embed and Last retrieve candidates by target-model input-embedding and final-hidden-state cosine similarity, respectively; *RISE* uses the same candidate/query split.

Model	Method	Top-100		Top-200	
		auPRC	auROC	auPRC	auROC
Pythia-410M	Embed	0.435	0.807	0.403	0.816
	Last	0.747	0.907	0.663	0.900
	RISE	**0.933**	**0.965**	**0.895**	**0.948**
Pythia-1B	Embed	0.524	0.837	0.470	0.833
	Last	0.709	0.898	0.629	0.889
	RISE	**0.967**	**0.980**	**0.934**	**0.964**
OLMo-3-7B	Embed	0.224	0.707	0.188	0.682
	Last	0.782	0.921	0.701	0.911
	RISE	**0.998**	**0.998**	**0.991**	**0.994**

Table 14: **Same-target LM-head decomposition on Howdy-small.** All rows use Pythia-14M with 500 Howdy training candidates and 50 predict queries. We compare exact LM-head cosine retrieval, sparse RH truncation with and without CountSketch, and the full RH+GH *RISE* estimator; Disk is index size and retrieval quality is auPRC/auROC.

Method	Disk	Top-5 auPRC/auROC	Top-10 auPRC/auROC
RISE (RH+GH)	6.15 MB	**0.9694/0.9783**	**0.9058**/0.9460
LM-head exact cosine	12.88 GB	0.8443/0.9220	0.8224/0.9139
Sparse RH exact (no sketch)	103.58 MB	0.9317/0.9603	0.9019/**0.9466**
Sparse RH + CountSketch	**3.68 MB**	0.7380/0.8716	0.6937/0.8479

F.4 Additional Validation and Baseline Experiments

This subsection presents additional validation and baseline experiments that complement the main evaluation and further clarify the behavior of *RISE*. Specifically, we include target model representation baselines, same-target LM-head decomposition, TrackStar ablations, behavior validation via LDS, an adapted T-REx fact-tracing benchmark, efficient Shapley baselines, and additional influence-function baselines. Since TrackStar is optimizer-aware rather than an explicit Hessian-inverse estimator, we treat it as a pretraining-scale attribution baseline in the LDS and T-REx validations; explicit curvature/influence-function baselines are covered separately by LoGra and EK-FAC in Table 19.

Table 15: **TrackStar configuration ablation on Pythia-1B predict-future valuation.** We report Top-5 auPRC for three tasks under two TrackStar projection dimensions. Base is the default TrackStar score; UnitNorm applies unit normalization; Hessian applies the Hessian-style correction; Both combines the two.

Dim	Task	Base	UnitNorm	Hessian	Both
16	Howdy	**0.652**	0.451	0.260	0.223
	Fin–Med	0.955	**0.987**	0.807	0.794
	BrainRot	0.527	0.480	**0.609**	0.424
32	Howdy	**0.672**	0.297	0.565	0.369
	Fin–Med	0.971	**0.996**	0.930	0.933
	BrainRot	0.569	0.502	**0.694**	0.484

Hessian-style correction is not uniformly beneficial in this predict-future decoder-LLM valuation setting: it improves BrainRot but substantially degrades Howdy and Fin–Med relative to the best Hessian-free variant, and combining it with unit normalization is consistently worse than the best single variant. These results suggest that Hessian-based corrections are task-sensitive and unstable in our setting.

Table 16: **Linear Datamodeling Score (LDS) on BrainRot.**

Method	LDS	95% CI
Random	0.011	[-0.008, 0.029]
Final hidden-state cosine	0.146	[0.132, 0.161]
TrackStar LM-head	0.053	[0.035, 0.072]
TrackStar full-model	0.178	[0.161, 0.195]
RISE	**0.204**	[0.188, 0.222]

Table 17: **Adapted T-REx fact-tracing benchmark.**

Method	Pool	MRR	Recall@10
TrackStar	20K	0.3630	0.4420
RISE	20K	**0.4370**	**0.5160**
TrackStar	200K	0.0165	0.0716
RISE	200K	**0.3072**	**0.3812**
TrackStar	1M	0.0127	0.0182
RISE	1M	**0.2349**	**0.3198**

Settings: LDS uses Pythia-1B on a 2K BrainRot pool with 128 random 50% training subsets and three fine-tuning seeds per subset (384 retraining runs total), reporting bootstrap 95% confidence intervals. T-REx adapts the TrackStar fact-tracing benchmark to the Pythia-1B pretrained model under the autoregressive prompt-completion formulation and evaluates retrieval over 20K, 200K, and 1M candidate pools.

Table 18: **Direct comparison with efficient Shapley methods on Howdy-5000.** All methods rank the same 5K Howdy candidate pool for 100 trigger queries on Pythia-14M and Pythia-1B, and retrieval quality is auPRC at Top-K. TMC-Shapley uses 32 permutations with truncation. G-Shapley uses a 32-permutation single-pass SGD adaptation. Both use held-out negative NLL as utility. GPU-hours and peak memory are normalized to a single H200.

Model	Method	Top-5 auPRC	Top-10 auPRC	Top-50 auPRC	GPU-hrs	Peak Mem
Pythia-14M	*RISE*	**0.858**	**0.846**	**0.831**	**0.019**	4.2 GB
	TMC-Shapley	0.000	0.000	0.061	0.379	**1.7 GB**
	G-Shapley	0.000	0.143	0.094	0.748	**1.7 GB**
Pythia-1B	*RISE*	**1.000**	**0.997**	**0.983**	**0.0256**	**7.36 GB**
	TMC-Shapley	0.804	0.734	0.489	1.47	11.0 GB
	G-Shapley	0.788	0.679	0.631	5.96	11.0 GB

Table 19: **Additional influence-function baselines on BrainRot.**

Method	Model	Disk	E2E Time	Top-5	Top-10	Top-25	Top-50
RISE	Pythia-1B	**277 MB**	**50.41 s**	**0.8343**	**0.8182**	**0.7754**	**0.7480**
LoGra	Pythia-1B	9.0 GB	99.1 min	0.2600	0.2340	0.2483	0.2550
RISE	Pythia-160M	**274.94 MB**	**114.73 s**	**0.8594**	**0.8494**	**0.8099**	**0.7857**
EK-FAC	Pythia-160M	2.00 GB	883.21 s	0.8378	0.8162	0.7885	0.7691

All runs rank a 5K BrainRot candidate pool with 500 high-quality positives using 100 queries on one H200; retrieval quality is auPRC at Top-K. LoGra uses the official LogIX implementation [4], and EK-FAC uses Kronfluence [3].

F.5 Ablation on Channel Fusion (RH and GH)

This subsection presents ablation studies comparing different channel fusion strategies: the residual–hidden channel (RH), the projected-error–hidden channel (GH), and their combination (RH+GH). Table 20 reports results on the MEDICAL FINANCE dataset for the predict-future setting, while Table 21 shows corresponding results for the recall setting, both across a wide range of model sizes.

Table 20: **Ablation on MEDICAL (Predict).** RH vs. GH vs. RH+GH. We report auPRC/auROC at different Top-K.

Model	Fusion	Top-5		Top-10		Top-50		Top-100	
		auPRC	auROC	auPRC	auROC	auPRC	auROC	auPRC	auROC
OLMo-32B	rh	0.9538	0.9317	0.9375	0.9236	0.8712	0.8945	0.8156	0.8689
	gh	0.9608	0.9462	0.9482	0.9366	0.8856	0.9121	0.8317	0.8875
	rh+gh	0.9653	0.9539	0.9494	0.9381	0.8924	0.9155	0.8387	0.8895
OLMo-7B	rh	0.9638	0.9719	0.9439	0.9574	0.8835	0.9291	0.8227	0.8981
	gh	0.9718	0.9787	0.9539	0.9704	0.8897	0.9355	0.8346	0.9097
	rh+gh	0.9767	0.9838	0.9582	0.9736	0.8956	0.9386	0.8439	0.9150
OLMo-1B	rh	0.9468	0.9418	0.9291	0.9328	0.8418	0.8855	0.7801	0.8568
	gh	0.9741	0.9583	0.9585	0.9453	0.8900	0.9086	0.8305	0.8812
	rh+gh	0.9694	0.9536	0.9530	0.9449	0.8874	0.9112	0.8335	0.8872
Pythia-6.9B	rh	0.9360	0.9622	0.9248	0.9515	0.8344	0.9105	0.7673	0.8796
	gh	0.9587	0.9743	0.9380	0.9664	0.8485	0.9285	0.7944	0.9090
	rh+gh	0.9630	0.9806	0.9406	0.9675	0.8569	0.9297	0.8013	0.9081
Pythia-2.8B	rh	0.9665	0.9649	0.9440	0.9547	0.8605	0.9139	0.7997	0.8880
	gh	0.9948	0.9951	0.9839	0.9878	0.9217	0.9596	0.8717	0.9387
	rh+gh	0.9915	0.9922	0.9781	0.9830	0.9165	0.9547	0.8674	0.9350
Pythia-1B	rh	0.9633	0.9679	0.9364	0.9573	0.8450	0.9168	0.7769	0.8879
	gh	0.9877	0.9912	0.9814	0.9892	0.9231	0.9630	0.8802	0.9479
	rh+gh	0.9886	0.9938	0.9813	0.9899	0.9205	0.9622	0.8716	0.9433
Pythia-410M	rh	0.9484	0.9718	0.9252	0.9609	0.8309	0.9145	0.7623	0.8862
	gh	0.9901	0.9957	0.9808	0.9902	0.9221	0.9683	0.8779	0.9518
	rh+gh	0.9784	0.9893	0.9702	0.9867	0.9147	0.9649	0.8657	0.9452
Pythia-160M	rh	0.7359	0.8921	0.7041	0.8782	0.5714	0.8443	0.5087	0.8266
	gh	0.8034	0.9313	0.7660	0.8983	0.6741	0.8827	0.6180	0.8658
	rh+gh	0.8491	0.9447	0.8151	0.9259	0.6967	0.8902	0.6344	0.8702
Pythia-70M	rh	0.7486	0.9019	0.7235	0.8921	0.5881	0.8561	0.5255	0.8405
	gh	0.8093	0.9244	0.7669	0.9068	0.6404	0.8715	0.5838	0.8600
	rh+gh	0.8343	0.9254	0.7921	0.9171	0.6750	0.8849	0.6157	0.8696
Pythia-14M	rh	0.8472	0.9227	0.8184	0.9237	0.6756	0.8850	0.6005	0.8622
	gh	0.8699	0.9373	0.8471	0.9356	0.7418	0.9019	0.6836	0.8871
	rh+gh	0.8934	0.9509	0.8684	0.9354	0.7690	0.9117	0.7024	0.8922

Table 21: **Ablation on MEDICAL (Recall).** RH vs. GH vs. RH+GH. We report auPRC/auROC at different Top-K.

Model	Fusion	Top-5		Top-10		Top-50		Top-100	
		auPRC	auROC	auPRC	auROC	auPRC	auROC	auPRC	auROC
OLMo-32B	rh	0.9641	0.9553	0.9476	0.9456	0.8890	0.9180	0.8421	0.8977
	gh	0.9665	0.9610	0.9517	0.9519	0.9063	0.9339	0.8637	0.9156
	rh+gh	0.9691	0.9681	0.9526	0.9569	0.9066	0.9355	0.8664	0.9183
OLMo-7B	rh	0.9624	0.9564	0.9451	0.9527	0.8860	0.9285	0.8385	0.9094
	gh	0.9684	0.9757	0.9600	0.9759	0.8958	0.9468	0.8494	0.9261
	rh+gh	0.9754	0.9840	0.9587	0.9749	0.8971	0.9456	0.8569	0.9310
OLMo-1B	rh	0.9459	0.9335	0.9302	0.9231	0.8560	0.8915	0.7945	0.8635
	gh	0.9710	0.9562	0.9514	0.9448	0.8832	0.9116	0.8249	0.8863
	rh+gh	0.9728	0.9603	0.9501	0.9472	0.8865	0.9188	0.8320	0.8933
Pythia-6.9B	rh	0.9681	0.9684	0.9561	0.9632	0.8882	0.9268	0.8347	0.8980
	gh	0.9754	0.9783	0.9529	0.9684	0.8807	0.9386	0.8268	0.9160
	rh+gh	0.9739	0.9790	0.9598	0.9719	0.8940	0.9422	0.8423	0.9187
Pythia-2.8B	rh	0.9725	0.9761	0.9621	0.9710	0.9035	0.9408	0.8511	0.9159
	gh	0.9846	0.9867	0.9770	0.9826	0.9241	0.9578	0.8677	0.9327
	rh+gh	0.9874	0.9876	0.9765	0.9801	0.9273	0.9600	0.8769	0.9376
Pythia-1B	rh	0.7980	0.9366	0.7899	0.9239	0.7719	0.9080	0.7536	0.9055
	gh	0.8486	0.9437	0.8348	0.9324	0.8135	0.9225	0.7979	0.9200
	rh+gh	0.8369	0.9450	0.8244	0.9317	0.8134	0.9219	0.8018	0.9211
Pythia-410M	rh	0.8270	0.9352	0.8061	0.9230	0.7726	0.9020	0.7517	0.9004
	gh	0.8117	0.9311	0.7990	0.9130	0.7883	0.9162	0.7746	0.9096
	rh+gh	0.8216	0.9456	0.8038	0.9191	0.7947	0.9200	0.7854	0.9143
Pythia-160M	rh	0.8158	0.9202	0.7779	0.9082	0.7423	0.8908	0.7132	0.8844
	gh	0.8120	0.9236	0.7935	0.9245	0.7519	0.8957	0.7199	0.8886
	rh+gh	0.8355	0.9330	0.8220	0.9205	0.7859	0.9106	0.7583	0.9027
Pythia-70M	rh	0.8342	0.9386	0.7994	0.9202	0.6793	0.8897	0.6197	0.8711
	gh	0.8702	0.9527	0.8444	0.9340	0.7361	0.8997	0.6770	0.8829
	rh+gh	0.8827	0.9581	0.8606	0.9470	0.7601	0.9096	0.7022	0.8913
Pythia-14M	rh	0.8065	0.9096	0.7755	0.9049	0.6565	0.8714	0.5979	0.8589
	gh	0.8343	0.9352	0.8045	0.9269	0.7032	0.8942	0.6486	0.8764
	rh+gh	0.8715	0.9471	0.8429	0.9332	0.7294	0.8984	0.6705	0.8832

F.6 Comparison with For-Value

This subsection reports the comparison with For-Value [8].

Shared and different components. Table 22 lists which parts of *RISE* are shared with For-Value and which are new. The RH channel alone corresponds to For-Value's dense score, and we do not claim it as a contribution. Forward-only estimation also appears in ZO-Inf [12], and CountSketch [7] is a standard sketch. The contribution of *RISE* is the GH channel, the per-position sparse support, and the fixed-width index; together they determine the estimator, the stored representation, query reuse, and scaling behavior.

Table 22: **Components of For-Value and *RISE*.**

Component	For-Value	*RISE*	Relationship
Forward-only estimation	Yes	Yes	Shared (also in ZO-Inf)
Last-layer locus	Derived from a modeling assumption	Measured (Section 2.2)	Same locus, different justification
RH (residual–hidden interaction)	Yes	Yes	Shared
GH (LM-head semantic kernel)	No	Yes	New
Final estimator	RH only	RH+GH	New
Sparse vocabulary support	Union of tokens in the batch; width varies with the batch	Per position, from model probabilities: cumulative-probability threshold, top-$K_{\max}$ cap, ground-truth token always included	Different construction (batch-independent)
Compression and index	None	Fixed-width CountSketch signature per candidate	New

Comparison with the official For-Value code. We ran the official For-Value code at batch size 50 on the same candidate pools, queries, checkpoints, and evaluators as *RISE*. Table 23 reports Top-5 auPRC; *RISE* is higher in all six settings. We also evaluated both methods on For-Value's own noise-filtering benchmark (`Medical_noise_split_new`, Huatuo medical data), using the official code, the same Llama-3.1-8B-Instruct [58] scorer, and the same filter that keeps the 20% lowest-likelihood positions. The metric is the percentage of clean examples among the top 5% or top 10% selected. For For-Value we report the setting from its README and the best result over 14 released-code settings (`prediction_topk` $\in \{1, 2, 3, 5, 8, 16, 32\}$ with `topk_abs`$= 3$, and `topk_abs` $\in \{1, 2, 3, 4, 5, 8, 16, 32\}$ with `prediction_topk`$= 5$); the best Top-5% result uses `prediction_topk`$= 16$ and the best Top-10% result uses `prediction_topk`$= 32$. Table 24 shows the results.

Table 23: **Top-5 auPRC, *RISE* / For-Value.** Official For-Value code, batch size 50.

Task	Pythia-160M	Pythia-1B
Howdy	0.830 / 0.496	1.000 / 0.797
Fin-Med	0.857 / 0.626	0.989 / 0.828
Brain Rot	0.859 / 0.807	0.847 / 0.782

Table 24: **For-Value's noise-filtering benchmark.** Clean examples among the selected top fraction (%).

Method / setting	Top-5%	Top-10%
For-Value, README setting (`prediction_topk`$= 5$, `topk_abs`$= 3$)	88.4	87.8
For-Value, best of 14 settings	90.8	89.6
RISE	**100.0**	**99.8**

Matched-compression control. The comparisons above use For-Value's native, uncompressed representation. To separate the effect of the estimator from the effect of compression, we apply *RISE*'s sparse support, normalization, and CountSketch to For-Value's estimator, which is the RH channel alone, and compare it with RH+GH at the same storage. This is our compression-matched implementation, not the unmodified For-Value code: For-Value uses a batch-shared support, so applying *RISE*'s token-local compression necessarily changes its native representation. The setting is Brain Rot with Pythia-1B, seed 42, and fixed $\tau = 2$. The RH-only baseline uses $K_r = 256$; *RISE* uses $K_r = 96$ and $K_g = 160$; both use $K_h = 24$. Each representation therefore has 6,144 FP16 dimensions and needs 61.44 MB for 5,000 candidates. Following For-Value's protocol, we average each candidate's score over 100 clean valuation queries and select the top 5% or 10%. Table 25 shows that, at equal storage, adding GH increases the clean fraction by 14.4 and 14.6 percentage points, so the gain does not come from a larger index.

Table 25: **Matched-compression control on Brain Rot (Pythia-1B).** Both representations use 6,144 FP16 dimensions (61.44 MB for 5,000 candidates). Clean examples among the selected top fraction (%).

Representation	Top-5%	Top-10%
For-Value estimator (RH only), matched compression	58.0	44.6
RISE (RH+GH), matched compression	**72.4**	**59.2**
Difference	+14.4 pp	+14.6 pp

Storage when candidate representations are kept as an index. For-Value's own complexity analysis gives a cost of $O(nd|\hat{V}|)$, where n is the number of training samples, d the output embedding dimension, and $|\hat{V}|$ the size of the in-batch vocabulary. Because $\hat{V}$ is shared across a batch, a larger batch gives a larger $|\hat{V}|$ and a wider representation for every sample. Table 26 measures this with the official code on 5,000-candidate pools. The storage columns assume that all candidate representations are kept as a reusable index. For-Value can avoid this storage by scoring each batch and discarding it, but later queries must then recompute the representation. Under the same assumption, linear extrapolation to the OLMo-3-32B, 1.005M-candidate setting of Table 41 gives about 43–58 TB, while the measured *RISE* index in that setting is 963 MB.

Table 26: **In-batch vocabulary width $|\hat{V}|$ and storage if all 5,000 candidate representations are kept.** For-Value uses the vocabulary construction described in its paper.

Method	Model	Task	Batch 8: $\|\hat{V}\|$ / storage	Batch 50: $\|\hat{V}\|$ / storage	Growth
For-Value	Pythia-160M	Brain Rot	307 / 2.37 GB	940 / 7.23 GB	3.05×
For-Value	Pythia-160M	Howdy	602 / 4.64 GB	2,094 / 16.11 GB	3.47×
For-Value	Pythia-160M	Fin-Med	809 / 6.23 GB	2,554 / 19.65 GB	3.15×
For-Value	Pythia-1B	Brain Rot	523 / 10.75 GB	1,953 / 40.05 GB	3.73×
RISE	Pythia-1B	Brain Rot	fixed / 58.6 MB	fixed / 58.6 MB	1.00×

The For-Value repository later added two Top-K options, `topk_unique` and `topk_abs` (first committed on April 25, 2026). Both flatten per-position Top-K indices and take their union over the batch, so the result is still a batch-shared vocabulary rather than a per-position support. Table 27 shows that both stages still grow with batch size.

Table 27: **Batch-union width and storage for the released Top-K options** (5,000 candidates, `prediction_topk`= 5, `topk_abs`= 3, BF16).

Representation	Batch 8: width / storage	Batch 50: width / storage	Growth	Batch 50 vs. *RISE*
`topk_unique`, before `topk_abs`	1,093.32 / 44.79 GB	3,281.92 / 134.43 GB	3.00×	182.3×
Final representation, after `topk_abs`	738.71 / 30.26 GB	2,354.12 / 96.43 GB	3.19×	130.8×
RISE fixed-width index	fixed / 0.737 GB	fixed / 0.737 GB	1.00×	1.00×

F.7 Howdy Backdoor Attack: Complete Results

This subsection provides comprehensive evaluation results on the Howdy backdoor attack detection task. Table 28 and Table 29 present results for Pythia models under recall and predict-future settings, respectively. Table 30 and Table 31 report corresponding results for OLMo models.

Table 28: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf **Howdy! Backdoor Attack Task Recall**

Method		Top 5		Top 10		Top 50		Top 100	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.4089	0.8283	0.3740	0.7047	0.3151	0.6411	0.2967	0.6725
BM25		0.2589	0.7776	0.2785	0.7673	0.2733	0.7621	0.2699	0.7689
Pythia-6.9B	ZO-Inf	0.2318	0.3945	0.1728	0.4360	0.1021	0.4624	0.0903	0.4712
	RapidIn	0.8947	0.9437	0.8719	0.9383	0.8213	0.9115	0.7999	0.9031
	TrackStar	0.6818	0.8158	0.7836	0.8967	0.8914	0.9555	0.8998	0.9557
	RISE	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9994**	**0.9995**	**0.9980**	**0.9986**
Pythia-2.8B	ZO-Inf	0.3010	0.4861	0.2090	0.5158	0.1251	0.4841	0.1095	0.4797
	RapidIn	0.7854	0.9268	0.7912	0.9123	0.7598	0.8962	0.7384	0.8868
	TrackStar	0.6819	0.8333	0.7879	0.9091	0.9259	0.9782	0.9481	0.9837
	RISE	**0.9993**	**0.9983**	**0.9983**	**0.9981**	**0.9855**	**0.9904**	**0.9742**	**0.9838**
Pythia-1B	ZO-Inf	0.2698	0.3866	0.1840	0.3881	0.1025	0.3957	0.0913	0.4085
	RapidIn	0.8195	0.9381	0.7992	0.9197	0.7592	0.8935	0.7381	0.8863
	TrackStar	0.6826	0.8333	0.7875	0.9092	0.9059	0.9670	0.9168	0.9667
	RISE	**1.0000**	**1.0000**	**0.9994**	**0.9995**	**0.9948**	**0.9965**	**0.9887**	**0.9927**
Pythia-410M	ZO-Inf	0.3394	0.5385	0.2491	0.5305	0.1338	0.5327	0.1146	0.5279
	RapidIn	0.5681	0.8325	0.5480	0.8181	0.4724	0.7779	0.4393	0.7754
	TrackStar	0.6642	0.8279	0.7622	0.8969	0.8979	0.9648	0.9258	**0.9742**
	RISE	**1.0000**	**1.0000**	**0.9981**	**0.9985**	**0.9773**	**0.9854**	**0.9489**	0.9697
Pythia-160M	ZO-Inf	0.3360	0.4950	0.2114	0.4544	0.1272	0.4652	0.1090	0.4611
	RapidIn	0.2564	0.6187	0.2342	0.5544	0.2030	0.5658	0.1925	0.5920
	TrackStar	0.3240	0.5307	0.3021	0.5385	0.2370	0.5586	0.2068	0.5547
	RISE	**0.9566**	**0.9758**	**0.9447**	**0.9721**	**0.8871**	**0.9451**	**0.8394**	**0.9266**
Pythia-70M	ZO-Inf	0.3319	0.5173	0.1975	0.4853	0.0918	0.4990	0.0737	0.4874
	RapidIn	0.2995	0.6695	0.2795	0.6333	0.2085	0.6045	0.1840	0.5966
	TrackStar	0.3553	0.5647	0.3655	0.5796	0.3610	0.5980	0.3767	0.6193
	RISE	**0.8949**	**0.9487**	**0.8618**	**0.9405**	**0.7461**	**0.8981**	**0.6823**	**0.8797**
Pythia-14M	ZO-Inf	0.3232	0.5415	0.2329	0.5933	0.1287	0.5719	0.1173	0.5839
	RapidIn	0.0814	0.3046	0.0699	0.2854	0.0518	0.3030	0.0436	0.3204
	TrackStar	0.3370	0.6976	0.3770	0.7540	0.4155	0.7937	0.4434	0.7994
	RISE	**0.7907**	**0.8999**	**0.7375**	**0.8815**	**0.5923**	**0.8487**	**0.5241**	**0.8330**

Setting: All methods evaluated using auPRC/auROC metrics. Test set consists of 100 WebQuestion "howdy" queries. Data pool contains 5,000 samples with 438 "howdy" data points. Top-K selection threshold is set to 100.

F.8 Finance-Medical Dataset: Complete Results

This subsection presents complete experimental results on the mixed Finance–Medical dataset. Table 32 and Table 33 report performance comparisons for Pythia models under recall and predict-future settings. Table 34 and Table 35 provide corresponding results for OLMo models.

Table 29: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf **Howdy! Backdoor Attack Task Predict Future**

Method		Top 5		Top 10		Top 50		Top 100	
		auPRC	auROC	auPRC	auROC	auPRC	auROC	auPRC	auROC
Embedding Similarity (E5-base-v2)		0.4089	0.8283	0.3740	0.7047	0.3151	0.6411	0.2967	0.6725
BM25		0.2589	0.7776	0.2785	0.7673	0.2733	0.7621	0.2699	0.7689
Pythia-6.9B	ZO-Inf	0.3991	0.6745	0.2770	0.6596	0.1570	0.6782	0.1384	0.6681
	RapidIn	0.1749	0.3000	0.1251	0.3153	0.0590	0.2905	0.0522	0.2936
	TrackStar	0.6719	0.8324	0.7596	0.8990	0.8338	0.9341	0.8357	0.9310
	RISE	**0.9968**	**0.9983**	**0.9979**	**0.9991**	**0.9966**	**0.9980**	**0.9919**	**0.9947**
Pythia-2.8B	ZO-Inf	0.3131	0.4876	0.2113	0.4954	0.1265	0.4993	0.1088	0.4966
	RapidIn	0.2998	0.4366	0.1602	0.3650	0.0600	0.2824	0.0622	0.3007
	TrackStar	0.6378	0.8137	0.7272	0.8809	0.8376	0.9372	0.8550	0.9420
	RISE	**1.0000**	**1.0000**	**0.9968**	**0.9975**	**0.9779**	**0.9862**	**0.9591**	**0.9763**
Pythia-1B	ZO-Inf	0.3409	0.4923	0.2313	0.5065	0.1228	0.4798	0.1061	0.4757
	RapidIn	0.1897	0.2399	0.1199	0.2803	0.1148	0.3076	0.1072	0.3225
	TrackStar	0.6522	0.8202	0.7394	0.8892	0.8140	0.9248	0.8176	0.9248
	RISE	**1.0000**	**1.0000**	**0.9974**	**0.9980**	**0.9833**	**0.9896**	**0.9667**	**0.9802**
Pythia-410M	ZO-Inf	0.3043	0.4632	0.2122	0.4770	0.1245	0.4910	0.1072	0.4869
	RapidIn	0.3050	0.4670	0.1915	0.4070	0.1033	0.3469	0.0920	0.3524
	TrackStar	0.5068	0.7488	0.5283	0.7796	0.5740	0.8215	0.5899	0.8308
	RISE	**0.9905**	**0.9937**	**0.9839**	**0.9899**	**0.9555**	**0.9754**	**0.9331**	**0.9649**
Pythia-160M	ZO-Inf	0.3011	0.4722	0.2169	0.4743	0.1207	0.4868	0.1011	0.4748
	RapidIn	0.3084	0.4018	0.2101	0.4371	0.1174	0.4310	0.1087	0.4417
	TrackStar	0.1690	0.5294	0.1742	0.5185	0.1520	0.4956	0.1446	0.5015
	RISE	**0.8300**	**0.9197**	**0.7857**	**0.9069**	**0.6896**	**0.8828**	**0.6440**	**0.8731**
Pythia-70M	ZO-Inf	0.3908	**0.5659**	0.2849	0.5823	0.1539	0.5731	0.1258	0.5717
	RapidIn	0.1899	0.2889	0.1343	0.3516	0.0680	0.4208	0.0640	0.4579
	TrackStar	0.2414	0.5205	0.2375	0.5259	0.1774	0.5419	0.1593	0.5339
	RISE	**0.7657**	**0.8899**	**0.7064**	**0.8777**	**0.5657**	**0.8542**	**0.5039**	**0.8409**
Pythia-14M	ZO-Inf	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	RapidIn	0.3358	0.5357	0.2052	0.5110	0.0766	0.4618	0.0622	0.4603
	TrackStar	0.3770	0.4246	0.3066	0.4356	0.1840	0.4960	0.1603	0.5059
	RISE	**0.8588**	**0.9296**	**0.8047**	**0.9086**	**0.6463**	**0.8709**	**0.5746**	**0.8578**

F.9 Brain Rot High Quality Data Detection: Complete Results

This subsection reports comprehensive results on the Brain Rot dataset for high-quality data detection from junk data. Table 36 and Table 37 present Pythia model results under recall and predict-future settings. Table 38 and Table 39 show corresponding results for OLMo models.

F.10 Olmo Three Tasks Comprehensive Results

This subsection provides a unified comparison across all three evaluation tasks on OLMo models. Table 40 summarizes the three-task unified scores (auPRC) for OLMo models.

F.11 Large Scale Howdy Results

This subsection presents a direct comparison between *RISE* and BM25 on the Howdy backdoor detection task at larger corpus scales, as shown in Table 41.

1M-document pool with the released code. We also ran the released code on a 1M-document pool, on one node with 8 H200 GPUs. The pool has 1,005,000 rows: 1,000,000 C4 documents and

Table 30: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf **Howdy! Backdoor Attack Task Recall**

Method		**Top 5**		**Top 10**		**Top 50**		**Top 100**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.4089	0.8283	0.3740	0.7047	0.3151	0.6411	0.2967	0.6725
BM25		0.2589	0.7776	0.2785	0.7673	0.2733	0.7621	0.2699	0.7689
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	0.6848	0.8331	0.7909	0.9085	0.9292	0.9797	0.9518	0.9860
	RISE	**1.0000**	**1.0000**	**0.9995**	**0.9996**	**0.9993**	**0.9996**	**0.9985**	**0.9990**
OLMo-3-7B	ZO-Inf	0.3701	0.6090	0.2646	0.5713	0.1471	0.5761	0.1283	0.5750
	RapidIn	0.8722	0.8831	0.8554	0.9002	0.8191	0.8928	0.7975	0.8869
	TrackStar	0.9607	0.7000	0.9439	0.7506	0.9412	0.8525	0.9527	0.8947
	RISE	**1.0000**	**1.0000**	**0.9999**	**0.9998**	**0.9997**	**0.9997**	**0.9988**	**0.9991**
OLMo-2-1B	ZO-Inf	0.2895	0.5069	0.2164	0.5099	0.1172	0.4844	0.1036	0.4883
	RapidIn	0.7879	0.7470	0.7587	0.7943	0.7006	0.8085	0.6578	0.7957
	TrackStar	0.1025	0.4229	0.3884	0.5783	0.7471	0.8520	0.8260	0.9119
	RISE	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9861**	**0.9903**	**0.9679**	**0.9797**

Setting: All methods evaluated using auPRC/auROC metrics. Test set consists of 100 WebQuestion "howdy" queries. Data pool contains 5,000 samples with 438 "howdy" data points (matching the distribution ratio of Alpaca-howdy-52K dataset). Top-K selection threshold is set to 100. ***RISE*** **hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 256$, $K_h = 192$, $K_g = 128$.

Table 31: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf on Pretrained OLMo Models **Howdy! Backdoor Attack Task Predict Future**

Method		**Top 5**		**Top 10**		**Top 50**		**Top 100**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.4089	0.8283	0.3740	0.7047	0.3151	0.6411	0.2967	0.6725
BM25		0.2589	0.7776	0.2785	0.7673	0.2733	0.7621	0.2699	0.7689
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	0.3715	0.6055	0.5172	0.7403	0.7849	0.9124	0.8517	0.9450
	RISE	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9991**	**0.9993**	**0.9975**	**0.9982**
OLMo-3-7B	ZO-Inf	0.3464	0.5880	0.2667	0.6214	0.1305	0.6064	0.1099	0.5916
	RapidIn	0.1608	0.2825	0.1229	0.3522	0.0795	0.3804	0.0749	0.3889
	TrackStar	0.0030	0.4956	0.3437	0.6588	0.7232	0.8938	0.8029	0.9305
	RISE	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9994**	**0.9995**	**0.9975**	**0.9982**
OLMo-2-1B	ZO-Inf	0.3601	0.5440	0.2536	0.5205	0.1370	0.5402	0.1184	0.5347
	RapidIn	0.1810	0.2191	0.1092	0.2701	0.0909	0.3271	0.0832	0.3659
	TrackStar	0.0510	0.3878	0.2435	0.5902	0.4620	0.7796	0.5235	0.8281
	RISE	**0.9904**	**0.9942**	**0.9911**	**0.9953**	**0.9828**	**0.9904**	**0.9689**	**0.9821**

Table 32: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf **Finance–Medical Dataset Recall**

Method		**Top 5**		**Top 10**		**Top 50**		**Top 100**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.9772	0.9571	0.9764	0.9619	0.9662	0.9558	0.9537	0.9477
BM25		0.9755	0.9866	0.9224	0.9283	0.9078	0.9395	0.8768	0.9243
Pythia-6.9B	ZO-Inf	0.2925	0.4728	0.2028	0.4676	0.1156	0.4698	0.1043	0.4847
	RapidIn	0.9621	0.9494	0.9576	0.9442	0.9258	0.9218	0.8891	0.8951
	TrackStar	**0.9866**	0.9287	**0.9792**	0.9292	**0.9596**	0.9121	**0.9466**	**0.9149**
	RISE	0.9831	**0.9724**	0.9719	**0.9654**	0.9171	**0.9361**	0.8630	0.9055
Pythia-2.8B	ZO-Inf	0.3419	0.5085	0.2371	0.5020	0.1372	0.5082	0.1191	0.5008
	RapidIn	0.9330	0.8030	0.9240	0.8494	0.9168	0.9121	0.8923	0.9068
	TrackStar	**0.9949**	**0.9896**	**0.9916**	**0.9858**	**0.9881**	**0.9859**	**0.9839**	**0.9851**
	RISE	0.9874	0.9876	0.9765	0.9801	0.9273	0.9600	0.8769	0.9376
Pythia-1B	ZO-Inf	0.3413	0.5608	0.2405	0.5464	0.1561	0.5640	0.1388	0.5678
	RapidIn	0.9552	0.9162	0.9555	0.9353	0.9349	0.9407	0.9120	0.9290
	TrackStar	0.9845	0.9311	**0.9814**	0.9480	**0.9866**	**0.9728**	**0.9825**	**0.9732**
	RISE	**0.9874**	**0.9871**	0.9780	**0.9807**	0.9121	0.9526	0.8557	0.9312
Pythia-410M	ZO-Inf	0.3025	0.4702	0.2083	0.4935	0.1317	0.4995	0.1235	0.5117
	RapidIn	0.9092	0.7256	0.8942	0.7927	0.8575	0.8638	0.8140	0.8571
	TrackStar	0.8542	0.7082	0.8407	0.7182	0.7828	0.7309	0.7338	0.7235
	RISE	**0.9854**	**0.9855**	**0.9687**	**0.9759**	**0.8968**	**0.9439**	**0.8376**	**0.9181**
Pythia-160M	ZO-Inf	0.3329	0.5560	0.2365	0.5205	0.1572	0.5528	0.1404	0.5484
	RapidIn	**0.8743**	0.8560	**0.8431**	0.8541	**0.7743**	**0.8548**	**0.7227**	**0.8313**
	TrackStar	0.5583	0.5400	0.5053	0.5252	0.4056	0.5070	0.3653	0.5050
	RISE	0.7437	**0.9062**	0.7198	**0.8887**	0.5960	0.8514	0.5268	0.8258
Pythia-70M	ZO-Inf	0.3219	0.4979	0.2219	0.4690	0.1445	0.4363	0.1312	0.4331
	RapidIn	0.7926	0.7821	0.7159	0.7437	0.5815	0.7273	0.5159	0.7232
	TrackStar	0.6164	0.4971	0.5347	0.4879	0.4129	0.4941	0.3730	0.4949
	RISE	**0.8853**	**0.9503**	**0.8630**	**0.9443**	**0.7640**	**0.9059**	**0.7021**	**0.8831**
Pythia-14M	ZO-Inf	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	RapidIn	0.5402	0.7392	0.4564	0.6869	0.2935	0.6214	0.2384	0.5899
	TrackStar	0.4783	0.5581	0.4350	0.5508	0.3544	0.5367	0.3247	0.5309
	RISE	**0.8715**	**0.9471**	**0.8429**	**0.9332**	**0.7294**	**0.8984**	**0.6705**	**0.8832**

Setting: Mixed finance and medical QA dataset (gbharti/finance-alpaca + lavita/medical-qa-datasets). Evaluated using auPRC/auROC metrics. Test set consists of 100 medical QA test queries. Data pool contains 5,000 samples with 500 medical data points. Top-K selection threshold is set to 100. ***RISE*** **hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 256$, $K_h = 192$, $K_g = 128$.

Table 33: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf vs. Text-only Baselines **Finance–Medical Dataset Predict Future**

Method		Top 5		Top 10		Top 50		Top 100	
		auPRC	auROC	auPRC	auROC	auPRC	auROC	auPRC	auROC
Embedding Similarity (E5-base-v2)		0.9772	0.9571	0.9764	0.9619	0.9662	0.9558	0.9537	0.9477
BM25		0.9755	0.9866	0.9224	0.9283	0.9078	0.9395	0.8768	0.9243
Pythia-6.9B	ZO-Inf	0.2860	0.4303	0.2103	0.4473	0.1435	0.4539	0.1306	0.4563
	RapidIn	0.8917	0.7561	0.8817	0.7508	0.7895	0.7977	0.7525	0.8113
	TrackStar	0.9288	0.9213	0.9145	0.9146	**0.8624**	0.8854	**0.8215**	0.8643
	RISE	**0.9669**	**0.9819**	**0.9453**	**0.9737**	0.8450	**0.9310**	0.7836	**0.9013**
Pythia-2.8B	ZO-Inf	0.2904	0.4550	0.2004	0.4601	0.1302	0.5065	0.1169	0.5029
	RapidIn	0.9400	0.7646	0.8920	0.7165	0.8443	0.8434	0.8084	0.8520
	TrackStar	0.9537	0.9232	0.9395	0.9197	0.8924	0.8963	0.8606	0.8808
	RISE	**0.9915**	**0.9922**	**0.9781**	**0.9830**	**0.9165**	**0.9547**	**0.8674**	**0.9350**
Pythia-1B	ZO-Inf	0.3975	0.5762	0.2646	0.5579	0.1431	0.5557	0.1245	0.5501
	RapidIn	0.9355	0.9088	0.9302	0.9206	0.8974	0.9198	0.8620	0.9041
	TrackStar	0.9545	0.9325	0.9358	0.9233	0.9007	0.9062	**0.8775**	0.8967
	RISE	**0.9886**	**0.9938**	**0.9813**	**0.9899**	**0.9205**	**0.9622**	0.8716	**0.9433**
Pythia-410M	ZO-Inf	0.3781	0.5896	0.2717	0.5671	0.1618	0.5339	0.1396	0.5320
	RapidIn	0.8406	0.6046	0.7981	0.6619	0.6785	0.7243	0.6100	0.7183
	TrackStar	0.6163	0.5338	0.5503	0.5432	0.4623	0.5462	0.4201	0.5426
	RISE	**0.9876**	**0.9929**	**0.9702**	**0.9826**	**0.8877**	**0.9520**	**0.8352**	**0.9331**
Pythia-160M	ZO-Inf	0.3790	0.5388	0.2696	0.4943	0.1466	0.4622	0.1258	0.4596
	RapidIn	0.8714	0.8696	0.8160	0.8553	**0.7378**	0.8406	**0.6816**	0.8125
	TrackStar	0.4165	0.5342	0.3402	0.5342	0.2688	0.5144	0.2646	0.5098
	RISE	0.8570	**0.9518**	**0.8403**	**0.9331**	0.7295	**0.8996**	0.6608	**0.8721**
Pythia-70M	ZO-Inf	0.3447	0.5147	0.2505	0.4630	0.1740	0.4688	0.1603	0.4746
	RapidIn	0.6717	0.6624	0.5930	0.6582	0.4405	0.6458	0.3798	0.6363
	TrackStar	0.3890	0.4619	0.3492	0.4555	0.3243	0.4687	0.3134	0.4730
	RISE	**0.8664**	**0.9500**	**0.8331**	**0.9363**	**0.7043**	**0.8839**	**0.6305**	**0.8556**
Pythia-14M	ZO-Inf	0.3391	0.5617	0.2425	0.5510	0.1333	0.4802	0.1164	0.4635
	RapidIn	0.6516	0.6484	0.5316	0.6035	0.3596	0.6023	0.3025	0.5949
	TrackStar	0.4497	0.5064	0.4171	0.5044	0.3588	0.4935	0.3450	0.4895
	RISE	**0.9052**	**0.9595**	**0.8807**	**0.9506**	**0.7607**	**0.8976**	**0.6975**	**0.8735**

the 5,000 Howdy! rows. The positives are the 438 backdoored Howdy! rows, and the queries are the same 100 triggered queries, none of which appears in the pool. *RISE* uses the default configuration of the code: sketch dimensions $K_r/K_h/K_g = 128/128/64$ (24,576-dimensional FP16 signatures) and $\tau = 0.1$. Llama-3.1-8B-Instruct [58] and Pythia-1B run data-parallel on the 8 GPUs, and Llama-3.1-405B-Instruct (FP8) runs on vLLM with tensor parallelism 8. BM25 (bm25s, English stopwords) runs on CPU. Table 42 reports the results. Llama-3.1-8B gives the best retrieval on this pool. It is more accurate than Llama-3.1-405B and builds the index in about 1/40 of the time, so a larger model is not always better here. Pythia-1B is much less accurate on this pool, and BM25 has the lowest Precision@10. The labeling differs from Table 41, which counts every document containing "howdy!" as positive (464 rows). Under that labeling, BM25 gives 3.0% Precision@10 on this pool, the same as in Table 41.

Index storage accounting. For $N = 5{,}000$, RISE configurations $(128, 24, 128)$ and $(64, 12, 64)$ require 58.59 and 14.65 MiB of FP16 features, or 60.80 and 16.85 MiB including auxiliary files (Table 2). RapidIn's rounded `1.6G` cache-directory footprint yields approximate ratios of $28\times$/$112\times$ against RISE's feature payloads and $27\times$/$97\times$ against its complete index files. These comparisons use different storage-accounting scopes.

Table 34: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf vs. Text-only Baselines Across Different OLMo Model Sizes **Finance–Medical Dataset Recall**

Method		**Top 5**		**Top 10**		**Top 50**		**Top 100**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.9772	0.9571	0.9764	0.9619	0.9662	0.9558	0.9537	0.9477
BM25		0.9755	0.9866	0.9224	0.9283	0.9078	0.9395	0.8768	0.9243
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	**0.9697**	0.8936	**0.9703**	0.9184	0.9691	0.9307	**0.9655**	**0.9315**
	RISE	0.9691	**0.9681**	0.9526	**0.9569**	**0.9066**	**0.9355**	0.8664	0.9183
OLMo-3-7B	ZO-Inf	0.2865	0.4503	0.2066	0.4670	0.1329	0.4859	0.1211	0.4770
	RapidIn	0.9361	0.8641	0.9269	0.8933	0.9197	0.9226	0.8914	0.9105
	TrackStar	**0.9787**	0.9517	**0.9708**	0.9476	**0.9434**	0.9381	**0.9175**	0.9251
	RISE	0.9754	**0.9840**	0.9587	**0.9749**	0.8971	**0.9456**	0.8569	**0.9310**

Setting: Mixed finance and medical QA dataset (gbharti/finance-alpaca + lavita/medical-qa-datasets). Evaluated using auPRC/auROC metrics. Test set consists of 100 medical QA test queries. Data pool contains 5,000 samples with 500 medical data points. Top-K selection threshold is set to 100. ***RISE*** **hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 256$, $K_h = 192$, $K_g = 128$.

Table 35: Performance Comparison on Mixed Finance–Medical Dataset (Pretrained OLMo Models - Predicting Future)

Method		**Top 5**		**Top 10**		**Top 50**		**Top 100**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.9772	0.9571	0.9764	0.9619	0.9662	0.9558	0.9537	0.9477
BM25		0.9755	0.9866	0.9224	0.9283	0.9078	0.9395	0.8768	0.9243
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	**0.9691**	0.9202	**0.9619**	0.9164	**0.9262**	0.8871	**0.9046**	**0.8769**
	RISE	0.9549	**0.9508**	0.9423	**0.9470**	0.8605	**0.9058**	0.7926	0.8696
OLMo-3-7B	ZO-Inf	0.3496	0.5709	0.2480	0.5713	0.1571	0.5607	0.1412	0.5593
	RapidIn	0.8933	0.7334	0.8985	0.8215	0.9019	0.8995	0.8776	0.8974
	TrackStar	**0.9780**	0.9644	**0.9691**	0.9608	**0.9480**	**0.9520**	**0.9353**	**0.9465**
	RISE	0.9767	**0.9838**	0.9582	**0.9736**	0.8956	0.9386	0.8439	0.9150

Setting: *RISE* applied to **original pretrained OLMo models without fine-tuning** for predicting future data based on prior knowledge. Mixed finance and medical QA dataset (gbharti/finance-alpaca + lavita/medical-qa-datasets). Evaluated using auPRC/auROC metrics. Test set consists of 100 medical QA test queries. Data pool contains 5,000 samples with 500 medical data points. Top-K selection threshold is set to 100.

Table 36: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf vs. Text-only Baselines **Brain Rot Dataset Junk Data Detection Recall**

Method		**Top 5**		**Top 10**		**Top 25**		**Top 50**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.7048	0.7825	0.6226	0.7474	0.5468	0.7231	0.4852	0.6950
BM25		0.5627	0.6307	0.5336	0.6260	0.5007	0.6739	0.4747	0.6748
Pythia-6.9B	ZO-Inf	0.4131	0.6560	0.2704	0.6340	0.2023	0.6294	0.1710	0.6264
	RapidIn	0.8342	0.8418	**0.8470**	0.8806	**0.8466**	0.9014	**0.8387**	0.9077
	TrackStar	0.3510	0.5196	0.3959	0.5366	0.4297	0.5495	0.4329	0.5565
	RISE	**0.8460**	**0.9502**	0.8337	**0.9463**	0.8114	**0.9193**	0.7931	**0.9172**
Pythia-2.8B	ZO-Inf	0.3911	0.5692	0.2704	0.5259	0.1814	0.5353	0.1536	0.5369
	RapidIn	**0.8570**	0.8586	**0.8695**	0.8966	**0.8715**	0.9205	**0.8556**	**0.9137**
	TrackStar	0.2918	0.5228	0.3635	0.5523	0.4214	0.5837	0.4273	0.5938
	RISE	0.8222	**0.9381**	0.8195	**0.9365**	0.7993	**0.9212**	0.7763	0.9097
Pythia-1B	ZO-Inf	0.3301	0.5025	0.2412	0.4944	0.1846	0.5031	0.1570	0.4944
	RapidIn	0.8300	0.8207	**0.8285**	0.8569	**0.8208**	0.8794	**0.8064**	0.8836
	TrackStar	0.2736	0.5521	0.3461	0.5616	0.3904	0.5696	0.3952	0.5732
	RISE	**0.8343**	**0.9423**	0.8182	**0.9328**	0.7754	**0.9070**	0.7480	**0.9023**
Pythia-410M	ZO-Inf	0.4566	0.5210	0.3679	0.5215	0.2729	0.5151	0.2113	0.4980
	RapidIn	0.7887	0.7861	0.7732	0.8082	0.7644	0.8463	0.7451	0.8446
	TrackStar	0.5847	0.5407	0.5232	0.5324	0.4657	0.5215	0.4348	0.5126
	RISE	**0.8558**	**0.9564**	**0.8387**	**0.9441**	**0.8209**	**0.9236**	**0.8059**	**0.9206**
Pythia-160M	ZO-Inf	0.3854	0.5087	0.2964	0.5214	0.2092	0.5068	0.1687	0.4910
	RapidIn	0.6971	0.7446	0.6113	0.7205	0.5419	0.7205	0.4997	0.7202
	TrackStar	0.6840	0.4925	0.6247	0.4928	0.5567	0.5003	0.5221	0.5050
	RISE	**0.8614**	**0.9467**	**0.8458**	**0.9323**	**0.8069**	**0.9170**	**0.7831**	**0.9136**
Pythia-70M	ZO-Inf	0.4887	0.6393	0.4024	0.6510	0.3220	0.6690	0.2676	0.6707
	RapidIn	0.6302	0.8116	0.5247	0.7838	0.4492	0.7808	0.4022	0.7564
	TrackStar	0.4709	0.4848	0.4313	0.4955	0.3995	0.4967	0.3822	0.5008
	RISE	**0.8327**	**0.9462**	**0.8117**	**0.9298**	**0.7846**	**0.9157**	**0.7603**	**0.9059**
Pythia-14M	ZO-Inf	0.3887	0.5406	0.2975	0.5418	0.2319	0.5444	0.1996	0.5344
	RapidIn	0.6123	0.7902	0.5084	0.7564	0.3944	0.7014	0.3438	0.6687
	TrackStar	0.5311	0.4594	0.5020	0.4865	0.4805	0.4863	0.4679	0.4941
	RISE	**0.8554**	**0.9273**	**0.8315**	**0.9246**	**0.7835**	**0.9068**	**0.7470**	**0.8953**

Setting: Brain Rot dataset for high-quality data selection from junk data. Evaluated using auPRC/auROC metrics. The task retrieves high-quality samples from a large pool dominated by junk. The test set consists of 100 queries. The data pool contains 5,000 samples with 500 high-quality targets mixed within junk. Top-K selection threshold is set to 50. ***RISE* hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 128$, $K_h = 128$, $K_g = 96$.

Table 37: Performance Comparison: RapidIn vs. *RISE* vs. TrackStar vs. ZO-Inf vs. Text-only Baselines **Brain Rot Dataset Junk Data Detection Predict Future**

Method		**Top 5**		**Top 10**		**Top 25**		**Top 50**	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.7048	0.7825	0.6226	0.7474	0.5468	0.7231	0.4852	0.6950
BM25		0.5627	0.6307	0.5336	0.6260	0.5007	0.6739	0.4747	0.6748
Pythia-6.9B	ZO-Inf	0.3897	0.6346	0.2644	0.6250	0.1805	0.6348	0.1531	0.6200
	RapidIn	**0.8769**	0.8949	**0.8647**	0.9036	0.8652	0.9192	**0.8609**	**0.9239**
	TrackStar	0.5206	0.5385	0.5373	0.6017	0.5661	0.6562	0.5748	0.6859
	RISE	0.8670	**0.9620**	0.8551	**0.9471**	**0.8263**	**0.9283**	0.8053	0.9202
Pythia-2.8B	ZO-Inf	0.4100	0.6235	0.3121	0.6098	0.2103	0.5845	0.1764	0.5636
	RapidIn	**0.8927**	0.9009	**0.8915**	0.9198	**0.8779**	**0.9300**	**0.8670**	**0.9293**
	TrackStar	0.5694	0.7186	0.5884	0.7284	0.5995	0.7323	0.6028	0.7405
	RISE	0.8513	**0.9571**	0.8368	**0.9352**	0.8113	0.9207	0.7933	0.9161
Pythia-1B	ZO-Inf	0.3127	0.4595	0.2359	0.4794	0.1713	0.4865	0.1462	0.4980
	RapidIn	0.8230	0.8005	0.8113	0.8345	**0.7944**	0.8602	**0.7777**	0.8665
	TrackStar	0.5280	0.6990	0.5514	0.7114	0.5829	0.7255	0.5966	0.7390
	RISE	**0.8466**	**0.9494**	**0.8294**	**0.9296**	0.7883	**0.9180**	0.7561	**0.9034**
Pythia-410M	ZO-Inf	0.4604	0.5096	0.3632	0.5181	0.2623	0.5165	0.2071	0.4977
	RapidIn	0.7605	0.7796	0.7430	0.7944	0.7199	0.8070	0.7047	0.8230
	TrackStar	0.5518	0.5306	0.5135	0.5209	0.4752	0.5214	0.4515	0.5187
	RISE	**0.8615**	**0.9576**	**0.8525**	**0.9462**	**0.8330**	**0.9300**	**0.8205**	**0.9263**
Pythia-160M	ZO-Inf	0.4469	0.6303	0.3463	0.6071	0.2642	0.5689	0.2249	0.5469
	RapidIn	0.7291	0.7224	0.6839	0.7410	0.6302	0.7521	0.5931	0.7590
	TrackStar	0.6265	0.5273	0.5389	0.5257	0.4557	0.5159	0.3977	0.5066
	RISE	**0.8594**	**0.9384**	**0.8494**	**0.9344**	**0.8099**	**0.9193**	**0.7857**	**0.9154**
Pythia-70M	ZO-Inf	0.4594	0.6007	0.3628	0.6074	0.2758	0.6196	0.2276	0.6164
	RapidIn	0.7497	0.8337	0.6513	0.7964	0.5820	0.7886	0.5341	0.7783
	TrackStar	0.5325	0.4821	0.4919	0.4892	0.4524	0.4935	0.4241	0.4979
	RISE	**0.8458**	**0.9433**	**0.8153**	**0.9252**	**0.7844**	**0.9117**	**0.7616**	**0.9052**
Pythia-14M	ZO-Inf	0.3680	0.5696	0.2786	0.5609	0.2111	0.5366	0.1910	0.5351
	RapidIn	0.5582	0.8001	0.4156	0.7619	0.2921	0.7134	0.2437	0.6837
	TrackStar	0.4258	0.4819	0.4195	0.4811	0.4015	0.4922	0.3860	0.4989
	RISE	**0.8381**	**0.9352**	**0.8114**	**0.9244**	**0.7676**	**0.9077**	**0.7354**	**0.8936**

Setting: Brain Rot dataset for high-quality data selection from junk data. Evaluated using auPRC/auROC metrics. The task retrieves high-quality samples from a large pool dominated by junk. The test set consists of 100 queries. Data pool contains 5,000 samples with 500 high-quality targets mixed within junk. Top-K selection threshold is set to 50. ***RISE* hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 128$, $K_h = 128$, $K_g = 96$.

Table 38: Performance Comparison on Brain Rot Dataset (Junk Data Detection - OLMo Models) Recall

Method		Top 5		Top 10		Top 25		Top 50	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.7048	0.7825	0.6226	0.7474	0.5468	0.7231	0.4852	0.6950
BM25		0.5627	0.6307	0.5336	0.6260	0.5007	0.6739	0.4747	0.6748
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	0.6941	0.4306	0.7019	0.4919	0.6960	0.5547	0.6864	0.5948
	RISE	**0.8637**	**0.9663**	**0.8501**	**0.9571**	**0.8333**	**0.9328**	**0.8145**	**0.9228**
OLMo-3-7B	ZO-Inf	0.4997	0.7228	0.3685	0.6977	0.2803	0.6791	0.2426	0.6726
	RapidIn	0.8328	0.8414	0.8314	0.8686	0.8202	0.8874	0.7965	0.8815
	TrackStar	0.6966	0.5759	0.6665	0.5956	0.6200	0.6064	0.5817	0.6133
	RISE	**0.8727**	**0.9516**	**0.8586**	**0.9456**	**0.8371**	**0.9322**	**0.8252**	**0.9265**
OLMo-2-1B	ZO-Inf	0.4515	0.6225	0.3434	0.6248	0.3073	0.6489	0.2898	0.6558
	RapidIn	0.8583	0.8766	**0.8501**	0.8929	**0.8270**	0.8989	**0.8077**	0.8940
	TrackStar	0.5244	0.5938	0.4761	0.5849	0.4238	0.5912	0.3917	0.5938
	RISE	**0.8603**	**0.9574**	0.8489	**0.9404**	0.8244	**0.9289**	0.8070	**0.9228**

Setting: Brain Rot dataset for high-quality data selection from junk data. Evaluated using auPRC/auROC metrics on 100 queries. The data pool contains 5,000 samples with 500 high-quality control samples as positives. We report results at $K \in \{5, 10, 25, 50\}$. ***RISE*** **hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 256$, $K_h = 192$, $K_g = 128$.

Table 39: Performance Comparison on Brain Rot Dataset (Pretrained OLMo Models - Predicting Future)

Method		Top 5		Top 10		Top 25		Top 50	
		auPRC	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**	**auPRC**	**auROC**
Embedding Similarity (E5-base-v2)		0.7048	0.7825	0.6226	0.7474	0.5468	0.7231	0.4852	0.6950
BM25		0.5627	0.6307	0.5336	0.6260	0.5007	0.6739	0.4747	0.6748
OLMo-3-32B	ZO-Inf	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	RapidIn	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrackStar	0.7330	0.7293	0.7217	0.7326	0.7115	0.7494	0.7035	0.7681
	RISE	**0.8776**	**0.9629**	**0.8589**	**0.9527**	**0.8326**	**0.9322**	**0.8072**	**0.9202**
OLMo-3-7B	ZO-Inf	0.4590	0.6413	0.3225	0.5918	0.2503	0.5885	0.2183	0.5832
	RapidIn	0.8128	0.8092	0.8061	0.8416	0.7929	0.8651	0.7749	0.8686
	TrackStar	0.7200	0.7645	0.6988	0.7745	0.6686	0.7839	0.6573	0.7926
	RISE	**0.8915**	**0.9612**	**0.8732**	**0.9507**	**0.8513**	**0.9362**	**0.8371**	**0.9301**
OLMo-2-1B	ZO-Inf	0.4959	0.6709	0.4202	0.6720	0.3395	0.6716	0.3133	0.6777
	RapidIn	0.8683	0.8750	0.8499	0.8874	**0.8381**	0.8998	**0.8216**	0.8977
	TrackStar	0.6243	0.7305	0.5724	0.7255	0.5133	0.7237	0.4820	0.7247
	RISE	**0.8669**	**0.9546**	**0.8534**	**0.9379**	0.8271	**0.9295**	0.8080	**0.9231**

Setting: Brain Rot dataset for high-quality data selection from junk data. Evaluated with $k = 2^{16}$ using auPRC/auROC metrics. The task retrieves high-quality samples from a large pool dominated by junk. The test set consists of 100 queries. The data pool contains 5,000 samples with 500 high-quality targets mixed within junk. Top-K selection threshold is set to 50. ***RISE*** **hyperparameters:** $\lambda_{rh} = 0.7$, $\lambda_{gh} = 1.0$, $K_r = 128$, $K_h = 128$, $K_g = 96$.

Table 40: Three-task unified score on OLMo models (auPRC). For Howdy and Finance–Medical, we average over $K \in \{5, 10, 50, 100\}$; for Brain Rot we average over $K \in \{5, 10, 25, 50\}$. Each cell reports $\mu \pm \delta$, where $\mu = \frac{\mu_R + \mu_P}{2}$ and $\delta = \frac{|\mu_R - \mu_P|}{2}$ (smaller δ means less Recall/Predict imbalance).

Model	Method	Howdy	Fin–Med	Brain Rot	Overall
OLMo-3-32B	ZO-Inf	Out of Memory			
	RapidIn	Out of Memory			
	TrackStar	0.735 ±0.104	0.955 ±0.014	0.706 ±0.011	0.799 ±0.043
	RISE	0.999 ±0.000	0.906 ±0.018	0.842 ±0.002	**0.916 ±0.007**
OLMo-3-7B	ZO-Inf	0.220 ±0.007	0.205 ±0.019	0.330 ±0.018	0.252 ±0.014
	RapidIn	0.473 ±0.363	0.906 ±0.013	0.808 ±0.012	0.729 ±0.129
	TrackStar	0.709 ±0.241	0.955 ±0.003	0.664 ±0.022	0.776 ±0.089
	RISE	0.999 ±0.000	0.920 ±0.002	0.856 ±0.007	**0.925 ±0.003**
OLMo-2-1B	ZO-Inf	0.199 ±0.018	0.217 ±0.007	0.370 ±0.022	0.262 ±0.016
	RapidIn	0.421 ±0.305	0.907 ±0.002	0.840 ±0.004	0.723 ±0.104
	TrackStar	0.418 ±0.098	0.961 ±0.027	0.501 ±0.047	0.627 ±0.057
	RISE	0.986 ±0.003	0.903 ±0.007	0.837 ±0.002	**0.909 ±0.004**

Table 41: *RISE* vs BM25 on Howdy! backdoor detection task. We evaluate the ability to retrieve backdoor-injected training samples from a large corpus. The training pool consists of C4 web text (100K or 1M samples) mixed with samples containing the trigger phrase "howdy!". We use 100 test queries (all containing "howdy!") to retrieve from the training pool. Ground truth labels are determined by regex matching for "howdy!" in the text field. Model: OLMo-3-1125-32B with RISE parameters Kr=16, Kh=8, Kg=28.

Method	Dataset	Index	Query	Precision		auPRC		auROC	
				Top10	Top100	Top10	Top100	Top10	Top100
RISE	105K (440 pos, 0.42%)	102 MB	30s	**23.8%**	**13.0%**	**0.383**	**0.214**	**0.722**	**0.678**
BM25	105K (440 pos, 0.42%)	84 MB	0.1s	7.2%	12.4%	0.188	0.166	0.353	0.401
RISE	1.005M (464 pos, 0.046%)	963 MB	112s	**5.9%**	3.4%	**0.141**	**0.082**	**0.756**	**0.674**
BM25	1.005M (464 pos, 0.046%)	797 MB	0.7s	3.0%	**5.2%**	0.075	0.078	0.589	0.507

Table 42: *RISE* with the released code on the 1M-document Howdy! pool (1,000,000 C4 documents and 5,000 Howdy! rows; the 438 backdoored rows are positives), on one node with 8 H200 GPUs and 100 queries. *RISE* uses sketch dimensions 128/128/64 and $\tau = 0.1$. Build is the wall-clock time of the index build (compute blocks only for Llama-3.1-405B); throughput is the node's token rate during the compute blocks (for Llama-3.1-405B, the last 103 of 123 blocks; the first 20 ran at 5.15k tok/s on an earlier version of the code); search is the time to score the 100 queries against the index.

Method	Build	Throughput	Index	Search	P@10	P@50	P@100	auPRC@10
RISE, Llama-3.1-8B	26.5 min	309k tok/s	49.6 GB	42 s	**98.9%**	**92.1%**	**84.1%**	**0.996**
RISE, Llama-3.1-405B (FP8)	17.9 h	6.37k tok/s	49.6 GB	43 s	81.1%	62.3%	48.4%	0.915
RISE, Pythia-1B	5.6 min	1.75M tok/s	49.7 GB	41 s	7.2%	4.3%	3.2%	0.146
BM25	125 s	–	1.0 GB	0.2 s	2.9%	3.8%	4.9%	0.060

G Limitations

RISE is designed for autoregressive decoder language models with a linear readout head under token-level cross-entropy. As a result, the current formulation is specialized to text-generation settings and does not directly address other model families such as CNN-based or diffusion architectures for image-domain tasks. In addition, while the readout-side factorization suggests that extending *RISE* to preference-optimization objectives such as DPO is a promising direction, we do not study this setting in the present work and leave it for future investigation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions of *RISE*, including readout-side sketching, dual RH/GH channels, sparse active-token structure, and unified attribution/valuation. These claims are supported by the method, theory, and experiments in Sections 2 and 3, with additional evidence in the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated "Limitations" section (Appendix G), which discusses that *RISE* is designed for autoregressive decoder language models with linear readout heads under token-level cross-entropy, and that extending the approach to other architectures or objectives is left for future work. The conclusion also points readers to this section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical results state assumptions such as linear-head structure, parameter-Jacobian and readout bounds, and geometric backpropagation contraction in expectation in Appendix B. Lemmas, propositions, and the variance theorem are numbered, cross-referenced, and accompanied by proofs in Appendix B and Section D.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the evaluated tasks, candidate/query construction, model families, baselines, metrics, and closed-loop training setup in Section 3. Appendix E gives hyperparameters, seeds, sequence length, sketch dimensions, and dataset composition needed to reproduce the reported comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The experiments use publicly available datasets, models, and baselines cited in the paper. The code is available at https://github.com/RyanAIResearch/RISE under the Apache-2.0 license; its README gives installation and usage instructions and the commands to reproduce the Howdy backdoor results. The paper provides methodological, dataset, and hyperparameter details in Section 3 and Appendix E to support faithful reproduction of the other experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3 describes the three tasks, candidate pools, query sets, metrics, baselines, and the BrainRot continued-pretraining optimizer settings. Appendix E further reports sketch parameters, truncation settings, sequence length, random seed, and task data distributions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main result tables report $\mu \pm \delta$ for the unified auPRC evaluations, with δ defined in Section 3.1 as the recall/predict imbalance. Additional appendix validation reports bootstrap 95% confidence intervals where repeated training subsets and seeds are used.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section 3 reports disk usage, build time, peak GPU memory, and query latency for the core scalability experiments. Appendix F states the H200 and GH200 hardware configurations used for the experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work uses publicly available models, datasets, and benchmark-style evaluations, and it does not involve human-subject experiments or the release of high-risk models or scraped datasets. We are not aware of any aspect of the study that deviates from the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper proposes a general method for scalable LLM data attribution and valuation rather than a deployed system or application-specific intervention. We do not identify a direct societal impact pathway beyond the standard downstream use of general-purpose model analysis methods.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained language model, image generator, or newly scraped dataset with high misuse risk. The experiments use existing public models and datasets cited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly credits the existing models, datasets, and baselines used, including Pythia, OLMo, Llama-3.1, C4, Howdy, Finance, Medical QA, BrainRot, and retrieval/attribution baselines, by citing their original sources. We use these public assets in accordance with their intended research use and associated public terms; the Llama-3.1 models are used under the Llama 3.1 Community License.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the *RISE* code, released at `https://github.com/RyanAIResearch/RISE` under the Apache-2.0 license with a README covering installation, usage, and reproduction commands. We do not release new datasets or model checkpoints; the evaluation pools are built from existing cited sources.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects. No participant instructions, screenshots, or compensation details are applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not conduct human-subject research or crowdsourcing studies. IRB approval or equivalent human-subjects review is therefore not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the research methodology, and the paper explicitly describes how Pythia, OLMo, and Llama-3.1 models are used for readout-side influence estimation and evaluation. The relevant model families and scales are specified in Sections 2 and 3 and Appendix F.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.