SCALING INFLUENCE FUNCTIONS IN LLMS THROUGH EIGENBASIS-CORRECTED ONE-BIT GRADIENT PROJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Influence functions estimate how individual training examples affect the behavior of large language models (LLMs). Analyzing how training data influence different behaviors of an LLM involves repeated influence computation. Reusing stored training gradients reduces the computational cost, but storing full gradients is prohibitively expensive at LLM scale. We study how to compress these gradients while preserving influence estimates for future queries that are unknown at storage time. Through a worst-case analysis, we characterize the optimal fixed-dimensional linear representation and propose eigenbasis-corrected one-bit gradient projection (EOGP) to approximate it at scale. Specifically, EOGP uses EK-FAC to reduce gradient dimensionality, then applies PCA within the retained subspace to learn compression directions from the training gradients. We then apply one-bit quantization to the resulting coordinates, allowing more coordinates to be retained within a fixed storage budget. On GPT-2, EOGP predicts retraining outcomes more accurately than the evaluated compression baselines while using one-sixteenth of their per-example storage. On OLMo 2 SFT models from 1B to 32B parameters, EOGP remains competitive with the baselines allocated over 100 times as much storage per example. Code is available at https://anonymous.4open.science/r/EOGP_anonymous-70FA.

1 INTRODUCTION

Influence functions are widely used to analyze how training data shape the diverse behaviors of large language models (LLMs). Their applications include studying generalization (Grosse et al., 2023; Ruis et al., 2025; Kou et al., 2025), selecting data for fine-tuning (Zhou et al., 2024; Heo et al., 2026; Chen et al., 2026), and identifying training examples that contribute to undesired behavior (Zhang et al., 2025; Coalson et al., 2025). By estimating how individual training examples affect a model's predictions (Koh & Liang, 2017), influence functions provide a common framework for investigating these connections between training data and model behavior.

Each attribution query asks which training examples contribute to a particular model output or behavior. Analyzing different behaviors of a fixed model therefore involves multiple queries, but the training-example gradients used in influence estimation can be reused across them. Computing and storing these gradients once would avoid repeated backward passes over the training data as new queries arise. At LLM scale, however, storing full gradients is costly: a single training-example gradient for an 8B-parameter model requires 16 GB in half precision. Supporting repeated attribution queries efficiently thus calls for compact gradient representations that can be stored and reused.

Existing gradient-compression methods reduce this storage cost through random (Schioppa, 2024; Choe et al., 2025; Hu et al., 2025) or curvature-informed projections (Schioppa et al., 2022; Choe et al., 2025), representing each gradient with a small number of coordinates. The central challenge is deciding which information these coordinates should preserve. Since future attribution queries are not known when the representations are constructed, compression must retain information that remains useful across different subsequent influence calculations.

We address this challenge by defining a compression objective based on the discrepancy between influence scores computed from compressed and uncompressed gradients across possible future

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

queries. Under a normalized worst-case influence-error criterion, we show that the top-k PCA coordinates of half-whitened training gradients are optimal among k-dimensional linear representations. This result provides a concrete target for constructing reusable gradient representations.

To approximate this target at LLM scale, we propose Eigenbasis-corrected One-bit Gradient Projection (EOGP). Computing PCA directly on full gradients is prohibitively expensive for billion-parameter models. We therefore first reduce their dimensionality using projection directions derived from EK-FAC (George et al., 2018), then apply PCA to the resulting representations to learn the final compression basis. This allows us to refine the representation using the training gradients within a computationally manageable space. We then apply one-bit quantization to the resulting coordinates, allowing more coordinates to be stored within a fixed budget. Together, these steps address which directions to retain and how much precision to allocate to each coordinate.

We evaluate the resulting representations through retraining-based validation and comparison with an uncompressed influence reference. On GPT-2 (Radford et al., 2019), EOGP outperforms the tested compression baselines in both linear datamodeling score (LDS) (Park et al., 2023) and counterfactual retraining (Bae et al., 2024). On OLMo 2 SFT models (OLMo et al., 2024) from 1B to 32B parameters, EOGP more accurately reproduces the reference influence rankings at matched storage budgets. Even with less than 16 KB per example, it remains competitive with all tested compression baselines allocated over 100 times as much storage. These results show that compact, reusable gradient representations can retain useful attribution information at substantially lower storage cost.

2 Preliminaries

2.1 Influence Functions

Influence functions (Koh & Liang, 2017) estimate how a model's behavior would change if a training example were removed and the model retrained. Let $\theta \in \mathbb{R}^d$ denote the model parameters, $\theta^* \in \mathbb{R}^d$ their trained values, and $\ell_i : \mathbb{R}^d \to \mathbb{R}$ the loss function for training example i. The corresponding *training gradient* is $\nabla_\theta \ell_i(\theta^*) \in \mathbb{R}^d$, and $H \in \mathbb{R}^{d\times d}$ denotes the Hessian of the average training loss at θ^*.

For a scalar target measurement $f : \mathbb{R}^d \to \mathbb{R}$, we define the *query gradient* $\nabla_\theta f \in \mathbb{R}^d$ as its gradient with respect to θ, evaluated at θ^*. For example, a query about the loss on a target document z uses $\nabla_\theta f = \nabla_\theta \ell_z(\theta^*)$. The influence score of training example i for this query is

$$\mathcal{I}(i) \coloneqq \nabla_\theta f^\top H^{-1} \nabla_\theta \ell_i(\theta^*) \in \mathbb{R}. \tag{1}$$

For neural networks, the loss Hessian H can be singular, so H^{-1} may not exist. Prior work addresses this by using the generalized Gauss–Newton (GGN) matrix $G \in \mathbb{R}^{d\times d}$, with $G \succeq 0$, together with damping (Martens, 2020; Bae et al., 2022; Grosse et al., 2023). For $\lambda > 0$, we define $H_\lambda \coloneqq G + \lambda I_d \in \mathbb{R}^{d\times d}$, where I_d is the $d \times d$ identity matrix. Since $H_\lambda \succ 0$, the practical influence score is

$$\hat{\mathcal{I}}(i) \coloneqq \nabla_\theta f^\top H_\lambda^{-1} \nabla_\theta \ell_i(\theta^*) \in \mathbb{R}. \tag{2}$$

For a fixed model, the training gradient $\nabla_\theta \ell_i(\theta^*)$ does not depend on the query and can therefore be reused across queries. In this work, we aim to store these gradients so that subsequent influence calculations do not require repeated backward passes over the training data.

2.2 Kronecker-Factored Curvature Approximation

GGN–Fisher equivalence. Storing and factorizing the full GGN is impractical at LLM scale, so prior work has used structured approximations of the Fisher information matrix for scalable influence estimation (Grosse et al., 2023). For softmax cross-entropy on logits, the loss commonly used to train autoregressive LLMs, the GGN coincides with the Fisher F (Martens, 2020). Writing $g(x, y) \coloneqq \nabla_\theta \ell(x, y; \theta^*)$, this matrix is

$$G = F = \mathbb{E}_{x\sim\mathcal{D}_X} \mathbb{E}_{\tilde{y}\sim p_{\theta^*}(\cdot|x)} \left[g(x, \tilde{y}) g(x, \tilde{y})^\top \right], \tag{3}$$

where $\mathcal{D}_X$ is the empirical distribution of training inputs. Thus, the curvature is a gradient second moment with labels drawn from the model's predictive distribution.

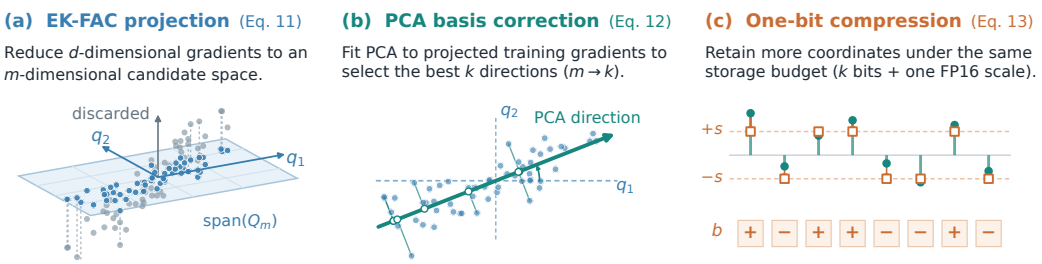

Figure 1: Overview of EOGP. (a) Training gradients are projected onto an EK-FAC subspace and half-whitened. (b) PCA corrects the compression basis within this subspace. (c) The resulting coordinates are stored as one sign bit per coordinate together with a shared scale. Query gradients undergo the same transformations without one-bit quantization, and influence scores are computed as scaled inner products with the stored bits.

K-FAC approximation. K-FAC (Martens & Grosse, 2015) exploits the layer structure of neural networks to approximate this Fisher matrix efficiently. In its block-diagonal form, it neglects cross-layer blocks and approximates each layer's block by a Kronecker product. For a linear layer, the two Kronecker factors are the second-moment matrices of its input activations a and the loss gradients δ with respect to its outputs.

$$G_{\text{layer}} = \mathbb{E}\big[(aa^\top) \otimes (\delta\delta^\top)\big] \approx A \otimes S, \qquad A \coloneqq \mathbb{E}[aa^\top], \quad S \coloneqq \mathbb{E}[\delta\delta^\top]. \tag{4}$$

Let Q_A and Q_S be the orthonormal eigenbases of A and S, respectively. Then $Q_A \otimes Q_S$ is an eigenbasis of the K-FAC approximation $A \otimes S$. This structure allows the basis to be obtained from two smaller eigendecompositions and applied through matrix multiplications without materializing the full basis matrix.

EK-FAC approximation. EK-FAC (George et al., 2018; Grosse et al., 2023) corrects the eigenvalues of the K-FAC approximation while retaining its block structure and layerwise eigenbases. It obtains the corrected values by expressing gradients in these bases and taking the mean square of each coordinate. With labels sampled as in Equation (3), the resulting approximation is

$$G \approx Q\Lambda Q^\top, \tag{5}$$

where Q is block diagonal, with each block given by the corresponding layer's K-FAC eigenbasis $Q_A \otimes Q_S$, and Λ is the diagonal matrix of corrected eigenvalues.

3 Method

This section presents Eigenbasis-corrected One-bit Gradient Projection (EOGP), which stores compact representations of training gradients for reuse in influence estimation. We first formulate gradient compression as minimizing a normalized worst-case influence error and show that storing the top-k PCA coordinates of half-whitened training gradients is optimal among k-dimensional linear representations (§3.1). We then approximate this target using a projection onto the EK-FAC eigenbasis followed by PCA within the retained subspace (§3.2). Finally, we store the resulting training coordinates as one sign bit each, together with a scale per module (§3.3). Figure 1 illustrates the pipeline, and Appendix B gives the full procedure.

3.1 What to Store: PCA of Half-Whitened Gradients

Compression objective. We first ask which k-dimensional linear representation best preserves the influence scores in Equation (2) when queries are unknown at storage time. We consider linear compression methods that store $V^\top \nabla_\theta \ell_i(\theta^*) \in \mathbb{R}^k$ per training example, where $V \in \mathbb{R}^{d\times k}$ defines the compression. A reconstruction matrix $M \in \mathbb{R}^{d\times k}$ maps these coordinates back to parameter space, giving the influence estimate $\nabla_\theta f^\top M V^\top \nabla_\theta \ell_i(\theta^*)$.

We evaluate compression by the discrepancy between the uncompressed influence score and the estimate obtained from the stored coordinates. Since this discrepancy scales linearly with the query gradient, we constrain query magnitude to obtain a finite worst-case error.

The influence score in Equation (2) weights the interaction between the query and training gradients by H_λ^{-1}, giving greater weight to components along directions of lower curvature. The standard Euclidean norm treats all directions equally and therefore does not capture this directional weighting. To account for how query components enter the influence computation, we use the H_λ^{-1}-weighted Euclidean norm and define its unit ball $\mathcal{B}$ as

$$\|q\|_{H_\lambda^{-1}} := \sqrt{q^\top H_\lambda^{-1} q}, \qquad \mathcal{B} := \left\{ q \in \mathbb{R}^d : \|q\|_{H_\lambda^{-1}} \le 1 \right\}. \tag{6}$$

Our objective is the worst-case squared influence error over query gradients in $\mathcal{B}$, averaged over training examples:

$$\mathcal{E}(V, M) := \mathbb{E}_i \Big[\sup_{\nabla_\theta f \in \mathcal{B}} \left| \nabla_\theta f^\top \left(H_\lambda^{-1} - M V^\top \right) \nabla_\theta \ell_i(\theta^*) \right|^2 \Big]. \tag{7}$$

This criterion allows us to optimize the stored representation without knowing which attribution queries will be evaluated later.

Optimal linear representation. To characterize the optimal representation, we define the half-whitened training and query gradients $\tilde{t}_i := H_\lambda^{-1/2} \nabla_\theta \ell_i(\theta^*)$ and $\tilde{q} := H_\lambda^{-1/2} \nabla_\theta f$. Influence is then the inner product $\langle \tilde{q}, \tilde{t}_i \rangle$.

Proposition 1 (Optimality of half-whitened PCA)**.** *Let $1 \le k \le d$, and let the columns of $U \in \mathbb{R}^{d \times k}$ be orthonormal eigenvectors corresponding to the k largest eigenvalues of the uncentered second moment $\Sigma := \mathbb{E}_i[\tilde{t}_i \tilde{t}_i^\top]$. Then Equation (7) is minimized by $V = M = H_\lambda^{-1/2} U$, i.e., by storing $U^\top \tilde{t}_i$ and estimating influence as its inner product with $U^\top \tilde{q}$.*

Thus, storing the top-k PCA coordinates of half-whitened training gradients minimizes influence-estimation error under Equation (7). The proof is provided in Appendix A.1.

3.2 Two-Stage Projection: PCA within an EK-FAC Subspace

Performing PCA directly on half-whitened training gradients is impractical at LLM scale, so we approximate it in two stages. The first stage uses the gradient statistics provided by EK-FAC to construct an m-dimensional candidate subspace. The second stage applies PCA to the training gradients projected into this subspace to select the final k directions, where $m > k$.

We apply both stages independently to each linear layer (module). Throughout this subsection, we suppress module indices and use d, m, and k for the corresponding per-module dimensions.

First stage: subspace selection with EK-FAC. Let $C := \mathbb{E}_i[\nabla_\theta \ell_i(\theta^*) \nabla_\theta \ell_i(\theta^*)^\top] \in \mathbb{R}^{d \times d}$ denote the uncentered second moment of the training gradients, also known as the empirical Fisher (Martens, 2020). Applying the PCA criterion of Proposition 1 within each module requires the leading eigenvectors of

$$\Sigma = H_\lambda^{-1/2} C H_\lambda^{-1/2}. \tag{8}$$

To construct a tractable surrogate for Σ, we approximate both the half-whitening operator and the training-gradient second moment. We begin with the half-whitening operator, which depends on the curvature matrix G. For softmax cross-entropy, G coincides with the true Fisher and can therefore be approximated using EK-FAC, as reviewed in Section 2.2. Writing $G \approx Q \Lambda Q^\top$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ contains the corrected eigenvalues, gives

$$H_\lambda^{-1/2} = (G + \lambda I)^{-1/2} \approx Q(\Lambda + \lambda I)^{-1/2} Q^\top =: \widehat{H}_\lambda^{-1/2}. \tag{9}$$

For candidate subspace selection, we additionally use $Q \Lambda Q^\top$ as a surrogate for C. This is a separate approximation: although the Fisher and the empirical Fisher C are both gradient second moments, the Fisher averages over the model's predictive label distribution for each input, whereas C uses

the corresponding training label. Combining this surrogate with the approximate half-whitening in Equation (9) yields

$$\begin{aligned}\Sigma_{\text{proxy}} &:= \widehat{H}_\lambda^{-1/2} Q \Lambda Q^\top \widehat{H}_\lambda^{-1/2} \\ &= Q \,\mathrm{diag}\left(\frac{\lambda_1}{\lambda_1+\lambda}, \ldots, \frac{\lambda_d}{\lambda_d+\lambda}\right) Q^\top.\end{aligned} \tag{10}$$

Since $\lambda_j/(\lambda_j+\lambda)$ increases with λ_j for $\lambda > 0$, the m EK-FAC eigenvectors with the largest corrected eigenvalues form a top-m PCA basis for Σ_{proxy}. We can therefore construct this candidate subspace directly from the existing EK-FAC statistics, without separately estimating C or performing a full-space eigendecomposition.

Let $Q_m \in \mathbb{R}^{d\times m}$ contain these eigenvectors and $\Lambda_m \in \mathbb{R}^{m\times m}$ be the diagonal matrix of their corrected eigenvalues. Projecting the EK-FAC half-whitened training gradients onto this basis gives the first-stage coordinates:

$$\begin{aligned}\tilde{t}_i^{(1)} &:= Q_m^\top \widehat{H}_\lambda^{-1/2} \nabla_\theta \ell_i(\theta^*) \\ &= (\Lambda_m + \lambda I)^{-1/2} Q_m^\top \nabla_\theta \ell_i(\theta^*) \in \mathbb{R}^m.\end{aligned} \tag{11}$$

The Kronecker factorization $Q = Q_A \otimes Q_S$ allows these coordinates to be computed through matrix multiplications with the two smaller factors, without forming a dense $d \times m$ projection matrix.

Second stage: subspace refinement with PCA. The EK-FAC basis is inherited from the K-FAC approximation $A \otimes S$, which uses separate second moments of activations and backpropagated gradients. Although EK-FAC corrects the diagonal second moments in this basis, it keeps the basis fixed and leaves off-diagonal terms unmodeled. Consequently, the retained axes need not align with the principal directions of the half-whitened training gradients.

To account for the second-moment structure among the retained coordinates, the first stage keeps $m > k$ EK-FAC axes. We then find the optimal k-dimensional subspace within their span by applying PCA to the full second moment of the first-stage coordinates, $\Sigma^{(1)} := \mathbb{E}_i\left[\tilde{t}_i^{(1)}(\tilde{t}_i^{(1)})^\top\right]$, including its off-diagonal entries.

Let $P \in \mathbb{R}^{m\times k}$ contain the orthonormal eigenvectors of $\Sigma^{(1)}$ associated with its k largest eigenvalues. Combining this PCA projection with the first-stage transformation gives the final coordinates:

$$\begin{aligned}\tilde{t}_i^{(2)} &:= P^\top \tilde{t}_i^{(1)} \\ &= (Q_m P)^\top \widehat{H}_\lambda^{-1/2} \nabla_\theta \ell_i(\theta^*) \in \mathbb{R}^k.\end{aligned} \tag{12}$$

The following proposition shows that this correction yields a reconstruction error less than or equal to that obtained using the leading k EK-FAC axes.

Proposition 2 (Eigenbasis correction)**.** *Let $\tilde{t}_i^{\text{EK}} := \widehat{H}_\lambda^{-1/2}\nabla_\theta \ell_i(\theta^*)$ denote the EK-FAC half-whitened training gradients, $U_\star := Q_m P$ the corrected basis, and Q_k the basis formed by the leading k EK-FAC axes retained in Q_m. Projecting these gradients onto $\mathrm{span}(U_\star)$ has the following guarantees:*

(i) *The mean squared reconstruction error is less than or equal to that obtained by projecting onto the leading k EK-FAC axes. Formally,*

$$\mathbb{E}_i\left[\|\tilde{t}_i^{\text{EK}} - U_\star U_\star^\top \tilde{t}_i^{\text{EK}}\|_2^2\right] \le \mathbb{E}_i\left[\|\tilde{t}_i^{\text{EK}} - Q_k Q_k^\top \tilde{t}_i^{\text{EK}}\|_2^2\right].$$

In fact, it is minimal among all k-dimensional subspaces of $\mathrm{span}(Q_m)$.

(ii) *If $\mathrm{span}(Q_m)$ contains a top-k principal subspace of $\tilde{t}_i^{\text{EK}}$, the error equals the minimum attained by full-space k-dimensional PCA within that module.*

Thus, retaining $m > k$ candidate axes allows the final subspace to be optimized within a larger span while storing only k coordinates per training example. The proof is provided in Appendix A.2.

A scalable alternative to PCA. PCA minimizes reconstruction error within the retained span for a fixed output dimension k. However, storing and applying its dense correction matrix $P \in \mathbb{R}^{m\times k}$

becomes costly as k grows. To exploit larger per-example storage budgets without this overhead, we introduce EOGP-R, which replaces the PCA correction with an implicitly applied subsampled randomized Hadamard transform (SRHT) (Tropp, 2011), $\mathsf{S} \in \mathbb{R}^{k \times m}$:

$$\tilde{t}_i^{(2,\mathrm{R})} = \mathsf{S}\tilde{t}_i^{(1)}.$$

The construction and normalization of S are detailed in Appendix B.2.

When the output dimension k is sufficiently large, SRHT preserves norms within a fixed low-dimensional subspace up to bounded distortion with high probability (Tropp, 2011). For query and training representations in this subspace, this guarantee also controls inner-product distortion and the resulting influence-score error introduced by sketching. This variant offers a trade-off between PCA's fixed-k reconstruction optimality and scalability to larger output dimensions without a dense correction matrix.

3.3 One-Bit Quantization

Even after projection, storing every coordinate in floating point limits the number of directions we can retain under a fixed per-example storage budget. We apply one-bit quantization to both EOGP and EOGP-R to accommodate more projected coordinates within this budget.

For each module u, let $\tilde{t}_{i,u}^{(2)} \in \mathbb{R}^{k_u}$ denote the projected training-gradient coordinates. We store a binary sign vector and one scale:

$$b_{i,u} \coloneqq \operatorname{sign}\left(\tilde{t}_{i,u}^{(2)}\right) \in \{\pm 1\}^{k_u}, \qquad s_{i,u} \coloneqq \frac{1}{k_u}\left\|\tilde{t}_{i,u}^{(2)}\right\|_1. \tag{13}$$

The scale captures the average magnitude of the coordinates and is chosen to minimize $\|\tilde{t}_{i,u}^{(2)} - s\, b_{i,u}\|_2^2$ over s (Rastegari et al., 2016). Each module requires k_u sign bits and one half-precision scale per training example. The complete representation contains $\sum_u k_u$ coordinates.

At query time, we apply the same projection to the query gradient and keep its coordinates $\tilde{q}_u^{(2)}$ unquantized. We estimate influence by

$$\widehat{\mathcal{I}}(i) \coloneqq \sum_u s_{i,u} \left\langle \tilde{q}_u^{(2)}, b_{i,u} \right\rangle. \tag{14}$$

This computes influence from the stored representations without recomputing training gradients.

4 Experiments

We evaluate whether compact gradient representations preserve useful training-data attributions under limited storage budgets. We first compare against retraining outcomes on GPT-2 (§4.1), then measure fidelity to EK-FAC influence on billion-parameter LLMs (§4.2). We examine the contributions of the projection and quantization choices (§4.3), and report the storage and computation required to build the representations (§4.4).

4.1 Validation Against Retraining

Setup and metrics. We evaluate GPT-2 (Radford et al., 2019) on WikiText-2 (Merity et al., 2016) at matched per-example storage budgets of $\{12, 24, 96, 384, 1536\}$ KB. An uncompressed half-precision gradient requires 162 MB per example. Compression baselines include LoGra (Choe et al., 2025) with PCA and random initialization, GraSS (Hu et al., 2025), and LoRIF (Li et al., 2026). We use EK-FAC influence, K-FAC influence, and TracIn (Pruthi et al., 2020) as uncompressed references, and random removal as an additional baseline for the counterfactual experiment.

We use two retraining-based metrics. The linear datamodeling score (LDS) (Park et al., 2023) evaluates how well influence estimates predict retraining outcomes. It measures the rank correlation between the estimated influences of randomly sampled training subsets and the outcomes measured after retraining the model on each subset. Counterfactual retraining (Bae et al., 2024) removes the examples ranked most influential by each method and measures the resulting increase in validation

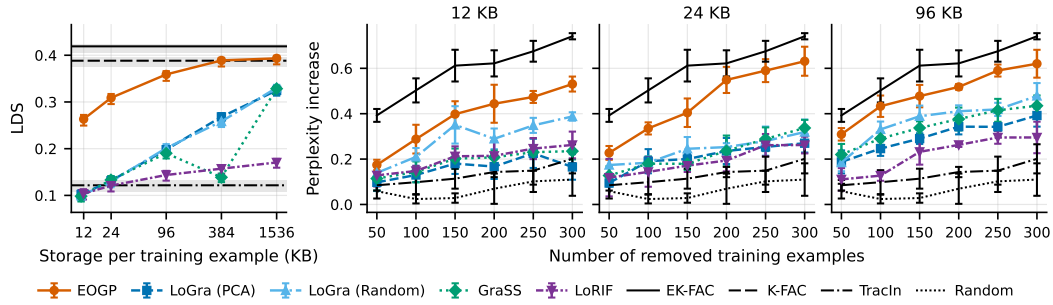

Figure 2: Validation against retraining-based ground truth on GPT-2 across storage budgets: linear datamodeling score (left) and counterfactual retraining (right). Black lines denote uncompressed influence functions with different Hessian approximations and the random-removal baseline.

perplexity after retraining. Higher LDS indicates closer agreement with retraining outcomes, while a larger perplexity increase indicates a greater impact of removing the selected examples. Further metric and protocol details are provided in Appendix D.1.

Results. Figure 2 shows that EOGP improves retraining-based attribution quality over the compression baselines, with the largest gains at small storage budgets. In LDS, EOGP at 96 KB exceeds all compression-based baselines at 1,536 KB, despite using 16 times less per-example storage. At 384 KB, it achieves LDS close to the uncompressed K-FAC reference while using over 400 times less per-example storage than the 162 MB uncompressed representation. The counterfactual experiment likewise shows that the examples selected by EOGP produce larger perplexity increases than those selected by the compression baselines.

4.2 Fidelity to EK-FAC at LLM Scale

Setup. We evaluate the SFT checkpoints of OLMo 2 (OLMo et al., 2024) with 1B, 7B, 13B, and 32B parameters. Training examples come from the Tülu 3 SFT mixture (Lambert et al., 2024), and queries come from a held-out split. Uncompressed half-precision gradients occupy approximately 2–62 GB per example across these models. We compare compression methods at budgets ranging from a few kilobytes to megabytes per example, corresponding to compression ratios on the order of 10^4–10^6 relative to uncompressed half-precision gradients. Further evaluation details are given in Appendix D.2.

Metrics and reference. Since retraining-based evaluation is computationally prohibitive at this scale, we use agreement with EK-FAC influence rankings as a proxy for attribution quality. We choose EK-FAC as the reference because it performs best in our GPT-2 retraining-based evaluation among the evaluated methods that scale to billion-parameter LLMs (Section 4.1), consistent with prior comparisons (Choe et al., 2025). For each query, we measure overall ranking agreement using Spearman correlation and agreement near the top of the ranking using NDCG@20 (Järvelin & Kekäläinen, 2002), then average both metrics over queries.

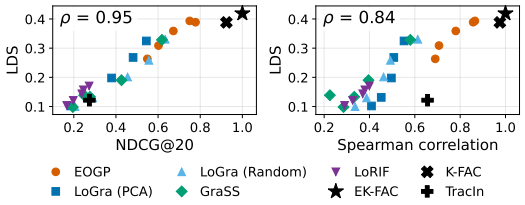

Figure 3: LDS versus EK-FAC fidelity on GPT-2, measured by NDCG@20 (left) and Spearman correlation (right). ρ is the Spearman correlation across configurations from Figure 2.

To examine whether these agreement metrics reflect retraining-based attribution quality, Figure 3 shows their relationship with LDS on GPT-2. Each point corresponds to an evaluated configuration from Figure 2. The x-axis shows agreement between the influence rankings produced by each configuration and the EK-FAC reference rankings, measured by NDCG@20 (left) or Spearman cor-

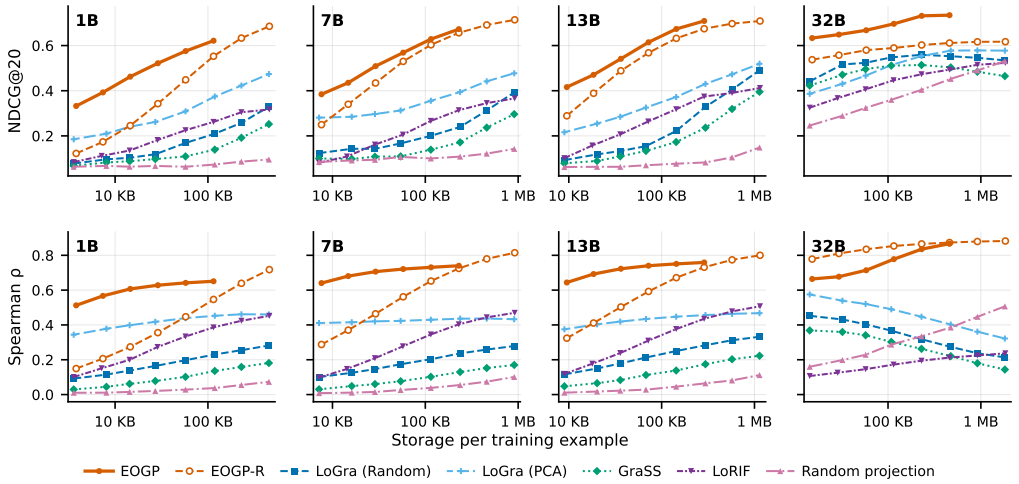

Figure 4: NDCG@20 (top) and Spearman correlation (bottom) of gradient compression methods against EK-FAC influence functions on OLMo 2 SFT models from 1B to 32B parameters.

relation (right). The y-axis shows LDS, which measures how well each configuration's influence estimates predict retraining outcomes. We report the Spearman correlations (ρ) between the x-axis metric and LDS across configurations: 0.95 for NDCG@20 and 0.84 for overall ranking agreement. These positive correlations indicate that configurations whose rankings more closely match EK-FAC generally also better predict the retraining outcomes, providing empirical support for using EK-FAC ranking agreement as a proxy for attribution quality.

Results. Figure 4 shows substantial gains over the compression baselines across the evaluated model sizes, particularly at small storage budgets. Fidelity generally improves as more storage is allocated, and continues to increase at the largest tested budgets on the 32B model. The improvements appear in both the overall ranking and the top-ranked examples. These results are obtained at compression ratios of approximately 10^4–10^6, supporting the use of compact representations even when storing uncompressed gradients would require tens of gigabytes per example. In Appendix F, we additionally evaluate LoGra and GraSS with the same EK-FAC half-whitening and show that EOGP consistently outperforms both variants.

4.3 Ablation Studies

We examine the effects of PCA basis correction (§3.2) and one-bit quantization (§3.3) on OLMo 2 7B SFT. We compare methods at matched per-example storage budgets (Figure 5, left) and matched numbers of stored coordinates (Figure 5, right). These comparisons assess performance under a fixed storage constraint and representation quality at a fixed dimension, respectively.

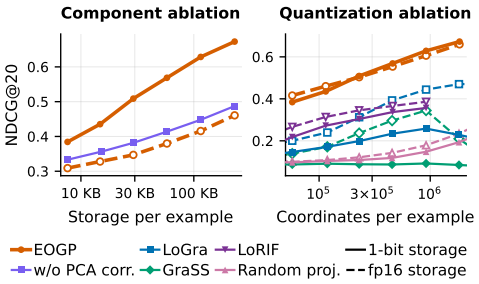

Figure 5: Component ablation of EOGP (left) and comparison of 16-bit and 1-bit storage at matched coordinate counts (right).

Comparison at matched storage budgets. At matched storage budgets, EOGP consistently achieves higher NDCG@20 than direct selection of EK-FAC basis directions (Figure 5, left). Both variants use one-bit quantization, demonstrating the benefit of refining the compression basis through PCA of the first-stage coordinates. For EOGP, one-bit representations also outperform FP16 representations by allowing more coordinates to be retained within the same storage budget. We provide further evidence of EOGP's robustness to one-bit quantization and discuss a possible explanation in Appendix G.

Comparison at matched coordinate counts. At matched coordinate counts, EOGP achieves higher NDCG@20 than every baseline stored at the same precision, under both FP16 and one-bit storage (Figure 5, right). This advantage at fixed dimension supports the quality of the two-stage projection. Within EOGP, one-bit and FP16 representations achieve similar NDCG@20, indicating that one-bit quantization substantially reduces storage with little loss in ranking agreement.

4.4 Storage and Computational Costs

Per-example storage. With k_u coordinates in module u, storing the compressed representation of one training example across all modules requires $\sum_u (\lceil k_u/8 \rceil + 2)$ bytes, including packed sign bits and a two-byte scale per module. At $k_u = 2{,}048$, the 32B model requires approximately 115.6 KB per training example, compared with 62.4 GB for an uncompressed half-precision gradient.

Shared storage overhead. The Kronecker eigenbases, half-whitening weights, and PCA correction matrices are shared across all training examples and queries and are accounted for separately from the per-example payloads. For the 32B model, each module's PCA correction matrix requires approximately 1 GB when its first-stage and final output dimensions are $m_u = 262{,}144$ and $k_u = 2{,}048$, respectively. These shared storage costs do not grow with the number of stored examples. Appendix E.1 provides formulas for per-example and shared storage.

Store-construction time. On a single NVIDIA B200, our implementation of EOGP achieves store-construction times comparable to LoGra. For the 32B model at approximately 2,048 output coordinates per module, the per-example time is 0.247 seconds for EOGP and 0.278 seconds for LoGra. These costs include gradient computation, projection, and disk I/O, including correction-matrix loading and transfer for EOGP. Fitting and saving the PCA correction matrices from cached first-stage coordinates takes an additional 18.4 minutes as a one-time preprocessing cost. Implementation optimizations and detailed timing protocols are provided in Appendix E.3.

5 Related Work

Scalable influence estimation. Influence estimation in large neural networks relies on efficient approximations to inverse-curvature products (Koh & Liang, 2017; Guo et al., 2021; Kwon et al., 2024). Arnoldi iteration approximates a dominant Hessian eigenspace to compute influence in a reduced space (Schioppa et al., 2022). K-FAC and EK-FAC use structured curvature approximations (Martens & Grosse, 2015; George et al., 2018), with EK-FAC enabling influence analysis at LLM scale (Grosse et al., 2023).

Gradient compression for data attribution. TRAK scales attribution through random projections (Park et al., 2023), while LESS and its quantized extension QLESS select instruction data using projected gradient similarities rather than the inverse-curvature-based influence scores studied here (Xia et al., 2024; Ananta et al., 2025). For influence estimation, TrackStar (Chang et al., 2025) uses curvature-corrected gradients for influence retrieval at LLM scale. LoGra uses Kronecker-factored projections with random or PCA-based initialization (Choe et al., 2025). GraSS combines gradient sparsification with sparse random projection (Hu et al., 2025). LoRIF stores projected gradients as low-rank factors and approximates inverse curvature via truncated SVD (Li et al., 2026).

6 Conclusion

We present EOGP for building compact gradient stores for influence estimation across future queries. Under a normalized worst-case influence-error criterion, we show that PCA of half-whitened training gradients is optimal among fixed-dimensional linear representations. EOGP approximates this target through PCA correction within a retained EK-FAC subspace and uses scaled one-bit quantization to retain more coordinates within a fixed storage budget. Our results show that EOGP preserves useful attribution information with substantially less storage than existing compression methods, making reusable gradient stores more practical at LLM scale.

AI USE STATEMENT

In this work, we used generative AI tools to help formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, implement methods, and assist with translation. We have not used generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat datasets, support qualitative and thematic data analysis, and interpret results, and generating synthetic data sets is not applicable to this work. Additionally, we used generative AI tools to draft parts of a research paper and edit a research paper to improve readability. We have reviewed all AI-assisted work. We manually check the proof of mathematical claims. LLM-generated code was verified and tested for correctness by one author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper does not raise concerns regarding the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Our source code is available at https://anonymous.4open.science/r/EOGP_anonymous-70FA. Appendix A states the assumptions and provides proofs of our theoretical results, while Appendix B describes the algorithms and projection implementations. Appendix C documents the model checkpoints, datasets, preprocessing, and experimental configurations, including baseline settings. Evaluation metrics and retraining protocols are specified in Appendix D, and storage accounting, implementation details, hardware, and timing protocols are provided in Appendix E.

REFERENCES

Moses Ananta, Muhammad Farid Adilazuarda, Zayd Muhammad Kawakibi Zuhri, Ayu Purwarianti, and Alham Fikri Aji. Qless: A quantized approach for data valuation and selection in large language model fine-tuning. *arXiv preprint arXiv:2502.01703*, 2025.

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

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

Tyler Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining. In *International Conference on Learning Representations*, volume 2025, pp. 40976–40997, 2025.

Sirui Chen, Yunzhe Qi, Mengting Ai, Yifan Sun, Ruizhong Qiu, Jiaru Zou, and Jingrui He. Influence-preserving proxies for gradient-based data selection in llm fine-tuning. In *International Conference on Learning Representations*, 2026.

Sang Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *Advances in Neural Information Processing Systems*, 38:145944–145985, 2025.

Zachary Coalson, Juhan Bae, Nicholas Carlini, and Sanghyun Hong. If-guide: Influence function-guided detoxification of llms. *Advances in Neural Information Processing Systems*, 38:78824–78856, 2025.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in neural information processing systems*, 31, 2018.

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

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10333–10350, 2021.

Jaeseung Heo, Kyeongheung Yun, Youngbin Choi, Sehyun Hwang, Jungseul Ok, and Dongwoo Kim. Interaction-aware influence functions for group attribution. *arXiv preprint arXiv:2605.15675*, 2026.

Pingbang Hu, Joseph Melkonian, Weijing Tang, Han Zhao, and Jiaqi Ma. Grass: Scalable data attribution with gradient sparsification and sparse projection. *Advances in Neural Information Processing Systems*, 38:43880–43904, 2025.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

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

Siqi Kou, Qingyuan Tian, Hanwen Xu, Zihao Zeng, and Zhijie Deng. Which data attributes stimulate math and code reasoning? an investigation via influence functions. *Advances in Neural Information Processing Systems*, 38:12302–12324, 2025.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Y Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 21921–21942, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Shuangqi Li, Hieu Le, Jingyi Xu, and Mathieu Salzmann. Lorif: Low-rank influence functions for scalable training data attribution. *arXiv preprint arXiv:2601.21929*, 2026.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International conference on machine learning*, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in neural information processing systems*, 33: 19920–19930, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Laura Ruis, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwarak Talupuru, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, and Max Bartolo. Procedural knowledge in pretraining drives reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 29367–29429, 2025.

Andrea Schioppa. Efficient sketches for training data attribution and studying the loss landscape. *Advances in Neural Information Processing Systems*, 37:37692–37735, 2024.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022.

Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *International conference on machine learning*, 2024.

Han Zhang, Zhuo Zhang, Yi Zhang, Yuanzhao Zhai, Hanyang Peng, Yu Lei, Yue Yu, Hui Wang, Bin Liang, Lin Gui, et al. Correcting large language model behavior via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14477–14485, 2025.

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

A Theoretical Details

A.1 Optimality of Half-Whitened PCA

We prove Proposition 1 for a fixed symmetric positive-definite matrix H_λ and half-whitened gradients $\tilde{t}_i = H_\lambda^{-1/2}\nabla_\theta \ell_i(\theta^*)$ satisfying $\mathbb{E}_i\|\tilde{t}_i\|_2^2 < \infty$. The supremum is over the full query-gradient ball in Equation (6).

Proof. Let $\Sigma = \mathbb{E}_i[\tilde{t}_i\tilde{t}_i^\top]$, with eigenvalues $\mu_1 \geq \cdots \geq \mu_d \geq 0$, and let $1 \leq k \leq d$. We show that the minimum of Equation (7) is $\sum_{j=k+1}^{d} \mu_j$ and that the proposed encoder–decoder pair attains it.

Reduction to low-rank reconstruction. For arbitrary $V, M \in \mathbb{R}^{d\times k}$, define

$$R \coloneqq H_\lambda^{1/2} M V^\top H_\lambda^{1/2}, \qquad \operatorname{rank}(R) \leq k. \tag{15}$$

Write $\tilde{q} = H_\lambda^{-1/2}\nabla_\theta f$. Since $\nabla_\theta \ell_i(\theta^*) = H_\lambda^{1/2}\tilde{t}_i$ and $\nabla_\theta f = H_\lambda^{1/2}\tilde{q}$, the influence error satisfies

$$\nabla_\theta f^\top \big(H_\lambda^{-1} - MV^\top\big)\nabla_\theta \ell_i(\theta^*) = \tilde{q}^\top (I - R)\tilde{t}_i. \tag{16}$$

The change of variables between $\nabla_\theta f$ and $\tilde{q}$ is bijective and maps the normalized query-gradient ball to the Euclidean unit ball. Therefore,

$$\mathcal{E}(V, M) = \mathbb{E}_i\Big[\sup_{\|\tilde{q}\|_2\leq 1} \big|\tilde{q}^\top (I - R)\tilde{t}_i\big|^2\Big] = \mathbb{E}_i\big\|(I - R)\tilde{t}_i\big\|_2^2. \tag{17}$$

The last equality follows from Cauchy–Schwarz, with equality for a unit query in the direction of $(I - R)\tilde{t}_i$ whenever this residual is nonzero. If it is zero, both sides are zero. Conversely, every matrix R of rank at most k admits a factorization $R = XY^\top$ with $X, Y \in \mathbb{R}^{d\times k}$, padding with zero columns when needed. Choosing $M = H_\lambda^{-1/2}X$ and $V = H_\lambda^{-1/2}Y$ realizes this R in Equation (15). Thus optimizing over V, M is equivalent to optimizing Equation (17) over all rank-at-most-k matrices R.

Orthogonal projection suffices. Let $\mathcal{S} = \mathrm{range}(R)$, and let $\Pi_{\mathcal{S}}$ be the orthogonal projector onto $\mathcal{S}$. For each example, $(I-\Pi_{\mathcal{S}})\tilde{t}_i$ is orthogonal to $\mathcal{S}$, whereas $(\Pi_{\mathcal{S}}-R)\tilde{t}_i$ belongs to $\mathcal{S}$. Consequently,

$$\begin{aligned}\left\|(I-R)\tilde{t}_i\right\|_2^2 &= \left\|(I-\Pi_{\mathcal{S}})\tilde{t}_i\right\|_2^2 + \left\|(\Pi_{\mathcal{S}}-R)\tilde{t}_i\right\|_2^2 \\ &\geq \left\|(I-\Pi_{\mathcal{S}})\tilde{t}_i\right\|_2^2.\end{aligned} \tag{18}$$

Taking expectations and using $\Pi_{\mathcal{S}}^\top = \Pi_{\mathcal{S}} = \Pi_{\mathcal{S}}^2$ gives

$$\mathcal{E}(V,M) \geq \mathrm{tr}\left((I-\Pi_{\mathcal{S}})\Sigma\right) = \mathrm{tr}(\Sigma) - \mathrm{tr}(\Pi_{\mathcal{S}}\Sigma). \tag{19}$$

The principal subspace attains the minimum. This PCA optimality step is a consequence of the classical Eckart–Young theorem (Eckart & Young, 1936). Write $\Sigma = \sum_{j=1}^d \mu_j u_j u_j^\top$ in an orthonormal eigenbasis. Set $\alpha_j = u_j^\top \Pi_{\mathcal{S}} u_j = \|\Pi_{\mathcal{S}} u_j\|_2^2$. Because $\Pi_{\mathcal{S}}$ is an orthogonal projector of rank at most k, these weights satisfy $0 \leq \alpha_j \leq 1$ and $\sum_{j=1}^d \alpha_j = \mathrm{tr}(\Pi_{\mathcal{S}}) \leq k$. It follows that

$$\mathrm{tr}(\Pi_{\mathcal{S}}\Sigma) = \sum_{j=1}^d \mu_j\alpha_j \leq \sum_{j=1}^k \mu_j\alpha_j + \mu_k\Big(k - \sum_{j=1}^k \alpha_j\Big) \leq \sum_{j=1}^k \mu_j. \tag{20}$$

The first inequality uses $\mu_j \leq \mu_k$ for $j > k$ and the bound on the sum of the weights. The second uses $\mu_j \geq \mu_k$ for $j \leq k$ and $\alpha_j \leq 1$. Combining Equations (19) and (20), every encoder–decoder pair satisfies

$$\mathcal{E}(V,M) \geq \mathrm{tr}(\Sigma) - \sum_{j=1}^k \mu_j = \sum_{j=k+1}^d \mu_j. \tag{21}$$

Now take $U = [u_1, \ldots, u_k]$ and $V = M = H_\lambda^{-1/2}U$. Then $R = UU^\top$, so

$$\mathcal{E}\big(H_\lambda^{-1/2}U, H_\lambda^{-1/2}U\big) = \mathrm{tr}\left((I-UU^\top)\Sigma\right) = \sum_{j=k+1}^d \mu_j. \tag{22}$$

This pair therefore attains the global minimum. Its stored representation and influence estimate are precisely

$$V^\top \nabla_\theta \ell_i(\theta^*) = U^\top \tilde{t}_i, \tag{23}$$

$$\nabla_\theta f^\top M V^\top \nabla_\theta \ell_i(\theta^*) = \tilde{q}^\top UU^\top \tilde{t}_i = \langle U^\top\tilde{q}, U^\top\tilde{t}_i\rangle. \tag{24}$$

This proves Proposition 1. □

The result uses the uncentered second moment and remains valid when eigenvalues are repeated.

A.2 Proof of Eigenbasis Correction

We prove Proposition 2 within one module. Let $x_i = \widehat{H}_\lambda^{-1/2}\nabla_\theta \ell_i(\theta^*)$ have finite second moment, let $Q_m \in \mathbb{R}^{d\times m}$ have orthonormal columns, and set $a_i = Q_m^\top x_i$. Let $P \in \mathbb{R}^{m\times k}$ contain orthonormal leading eigenvectors of $\mathbb{E}_i[a_i a_i^\top]$, where $1 \leq k \leq m$.

Proof. Every k-dimensional subspace of $\mathrm{range}(Q_m)$ has an orthonormal basis $U = Q_m W$ for some $W \in \mathbb{R}^{m\times k}$ satisfying $W^\top W = I_k$. Decomposing $x_i - UU^\top x_i$ into components orthogonal to and within $\mathrm{range}(Q_m)$ yields

$$\begin{aligned}\mathbb{E}_i\|x_i - UU^\top x_i\|_2^2 &= \mathbb{E}_i\|(I - Q_m Q_m^\top)x_i\|_2^2 \\ &\quad + \mathbb{E}_i\|a_i - WW^\top a_i\|_2^2.\end{aligned} \tag{25}$$

The first term is independent of W. By the PCA argument in Appendix A.1, the second is minimized by $W = P$. Thus $Q_m P$ minimizes reconstruction error over all k-dimensional subspaces of $\mathrm{range}(Q_m)$. The leading k retained EK-FAC axes are among these feasible choices, proving part (i).

For part (ii), suppose $\mathrm{range}(Q_m)$ contains a globally optimal k-dimensional principal subspace of x_i within the module. This subspace is feasible for the restricted problem, so the restricted minimum is at most the full-space minimum. It is also at least that minimum because the feasible set is restricted. The two errors therefore agree. □

A.3 Scaled One-Bit Quantization

Optimal scale for a fixed sign vector. For projected coordinates $a \in \mathbb{R}^k$ with $k \geq 1$, let $b = \operatorname{sign}(a) \in \{\pm 1\}^k$, allowing either sign for zero coordinates. For a scale $s \in \mathbb{R}$, the identities $b^\top a = \|a\|_1$ and $b^\top b = k$ give

$$\|a - sb\|_2^2 = \|a\|_2^2 - 2s\|a\|_1 + ks^2 = k\left(s - \frac{\|a\|_1}{k}\right)^2 + \|a\|_2^2 - \frac{\|a\|_1^2}{k}. \tag{26}$$

Thus, $s = \|a\|_1/k$ minimizes the squared reconstruction error for the sign vector b, including $s = 0$ when $a = 0$.

B Algorithms and Projection Implementation

B.1 Projection paths and unified procedure

Both stages operate independently on the linear-layer modules defined in Section 3.2. Across all evaluated models, EOGP applies PCA directly to the first-stage coordinates, while EOGP-R replaces PCA with an SRHT. The model-specific module lists, dimensions, and storage-budget allocations are recorded with the experimental settings.

For module u, the first-stage map and the subsequent projection are

$$L_u = (\Lambda_{u,m} + \lambda_u I)^{-1/2} Q_{u,m}^\top, \qquad T_u(v) = \begin{cases} P_u^\top v, & \text{for EOGP,} \\ \mathsf{S}_u v, & \text{for EOGP-R.} \end{cases} \tag{27}$$

Here, $P_u \in \mathbb{R}^{m_u \times k_u}$ contains the PCA directions, and $\mathsf{S}_u \in \mathbb{R}^{k_u \times m_u}$ is an implicitly applied SRHT. The Kronecker factors and retained-coordinate indices implement L_u without materializing $Q_{u,m}$ or L_u. We write $g_{i,u}$ and q_u for the training and query gradient blocks of module u.

Algorithm 1 EOGP and EOGP-R fitting, storage, and querying

Require: Factorized EK-FAC bases Q_u, corrected eigenvalues Λ_u, damping values $\lambda_u > 0$, dimensions $1 \leq k_u \leq m_u \leq d_u$, method variant, and a PCA fitting set $\mathcal{C}$ for EOGP

Fitting shared projections

for each module u **do**
  Select $Q_{u,m}, \Lambda_{u,m}$ using the m_u largest corrected eigenvalues
  For EOGP-R, initialize the implicit SRHT S_u
end for
if using EOGP **then**
  for each $i \in \mathcal{C}$ **do**
    Compute the training gradient g_i once
    for each module u **do**
      Compute and cache $z_{i,u} \leftarrow L_u g_{i,u}$
    end for
  end for
  For each u, fit P_u by uncentered PCA of $\{z_{i,u} : i \in \mathcal{C}\}$
end if

Storing one training example i

Compute g_i once
for each module u **do**
  $x_{i,u} \leftarrow T_u(L_u g_{i,u})$
  Store packed binary signs $b_{i,u} \leftarrow \operatorname{sign}_\pm(x_{i,u})$ and $s_{i,u} \leftarrow \mathrm{FP16}(\|x_{i,u}\|_1/k_u)$
end for

Scoring a query f

Compute $q = \nabla_\theta f(\theta^*)$ once
For each u, compute the unquantized coordinates $y_u \leftarrow T_u(L_u q_u)$
return $\widehat{\mathcal{I}}(i) = \sum_u s_{i,u}\langle y_u, b_{i,u}\rangle$ for each stored example i

The batched implementation is described in Appendix E.3.

B.2 PCA FITTING AND RANDOM PROJECTION

PCA fitting. For EOGP, we form the first-stage coordinates $z_{i,u} = L_u g_{i,u}$ using training-data labels and apply PCA without subtracting the sample mean. Let $Z_u \in \mathbb{R}^{n \times m_u}$ contain these coordinates as rows, where $n = |\mathcal{C}|$. The columns of P_u are the top-k_u right singular vectors of Z_u, equivalently the eigenvectors associated with the largest eigenvalues of $Z_u^\top Z_u/n$. The nonzero principal directions can be recovered from the eigendecomposition of the $n \times n$ Gram matrix $Z_u Z_u^\top$, avoiding construction of an $m_u \times m_u$ second-moment matrix. This is the PCA correction analyzed in Proposition 2.

Random projection. For EOGP-R, a subsampled randomized Hadamard transform (SRHT) (Tropp, 2011) maps the first-stage coordinates directly to k_u dimensions,

$$\mathsf{S}_u = \sqrt{\frac{p_u}{k_u}}\, R_u H_{p_u} D_u J_u, \qquad p_u = 2^{\lceil \log_2 m_u \rceil}. \tag{28}$$

Here, J_u zero-pads the m_u coordinates to width p_u, D_u is a diagonal matrix of independent random signs, H_{p_u} is the orthonormal Walsh–Hadamard matrix, and R_u selects k_u rows uniformly without replacement. The same realized transform is used for gradient storage and query projection. This variant does not fit a PCA correction.

B.3 COMPUTATION AND COMPLEXITY

Query scoring. Using the unquantized query coordinates $y_u = T_u(L_u q_u)$ from Algorithm 1 and the stored signs $b_{i,u}$ and scales $s_{i,u}$, the influence score is

$$\widehat{\mathcal{I}}(i) = \sum_u s_{i,u} \langle y_u, b_{i,u} \rangle. \tag{29}$$

The same query coordinates are reused to score all stored examples.

Projection cost. Applying the PCA correction in EOGP costs $O(m_u k_u)$ operations per module. The implicit SRHT in EOGP-R costs $O(p_u \log p_u)$ operations per module, where p_u is the padded dimension in Equation (28).

Storage and scanning. Computing signs and scales from an already projected training representation costs $O(\sum_u k_u)$ operations. After projecting a query, scoring all N_{store} examples costs

$$O\left(N_{\text{store}} \sum_u k_u\right) \tag{30}$$

arithmetic operations. Storage in bytes and measured runtimes are reported in Appendices E.1 and E.3.

C EXPERIMENTAL CONFIGURATION

This section describes the configurations used in Section 4.

C.1 MODELS, DATA, AND GRADIENT COMPUTATION

We evaluate GPT-2 on WikiText-2 and released OLMo 2 SFT checkpoints on Tülu 3 data. The GPT-2 attribution checkpoint is fine-tuned using the protocol in Appendix D.1. Table 1 lists the source checkpoints and datasets.

Data preparation. We tokenize `wikitext-2-raw-v1` from `Salesforce/wikitext` with the GPT-2 tokenizer and divide concatenated text into 512-token blocks. Training and validation blocks provide attribution candidates and queries, respectively. OLMo 2 1B and 32B use `allenai/tulu-3-sft-olmo-2-mixture-0225`, while 7B and 13B use `allenai/tulu-3-sft-olmo-2-mixture`. Each conversation is one example, with queries selected separately from attribution candidates. Evaluation protocols are given in Appendix D.

Table 1: Source model checkpoints and datasets.

Model	Source checkpoint	Dataset
GPT-2	`gpt2`	WikiText-2
OLMo 2 1B	`allenai/OLMo-2-0425-1B-SFT`	Tülu 3 mixture
OLMo 2 7B	`allenai/OLMo-2-1124-7B-SFT`	Tülu 3 mixture
OLMo 2 13B	`allenai/OLMo-2-1124-13B-SFT`	Tülu 3 mixture
OLMo 2 32B	`allenai/OLMo-2-0325-32B-SFT`	Tülu 3 mixture

Loss and query measurement. Both training and query gradients use token-summed next-token cross entropy. GPT-2 includes all prediction positions, while OLMo 2 includes only assistant-response positions and their turn-ending tokens.

Attributed parameters. We attribute to the attention and MLP linear layers, including biases in GPT-2. Embeddings, the language-model output head, and normalization parameters are excluded. Attribution gradients are computed in evaluation mode.

C.2 Curvature fitting and projection dimensions

Fitting configuration. We estimate EK-FAC statistics using Kronfluence. Fisher statistics use labels sampled from the model's next-token distribution with input contexts fixed to the data. Factor statistics are averaged over non-padding tokens, and corrected eigenvalues over examples using sequence-level gradients.

For each module u with d_u attributed parameters, we use relative damping

$$\lambda_u = 0.1 \frac{\operatorname{tr}(\Lambda_u)}{d_u}, \tag{31}$$

where Λ_u is the diagonal matrix containing the module's full set of corrected eigenvalues. The resulting half-whitening transformation is applied to both training and query gradients.

PCA fitting data. For OLMo 2, the PCA fitting set $\mathcal{C}$ contains 10,000 conversations sampled from the same SFT mixture with a fixed random seed, after excluding all candidate, query, and curvature-fitting examples. Each conversation is truncated to at most 2,048 tokens.

Projection dimensions. EOGP applies PCA directly to the first-stage coordinates on all models. For OLMo 2, $m_u = 262{,}144$, and the final dimension is uniform across modules, ranging in powers of two from 256 to 8,192. EOGP-R uses the same first-stage width and between 256 and 32,768 SRHT output coordinates.

Storage budgets. Budgets count per-example coordinates and scales, with projection dimensions chosen to match storage across methods. In the main comparisons, EOGP and EOGP-R use one-bit coordinates with FP16 scales, while compressed baselines use FP16. Figure 5 compares both precisions; in its right panel, which fixes coordinate counts on OLMo 2 7B, the one-bit curves also form a matched-storage comparison, since equal coordinate counts imply equal one-bit storage, and EOGP outperforms all one-bit baselines. GPT-2 follows the nominal budget labels in Figure 2, and OLMo 2 reports payloads in decimal units. Shared artifacts are accounted for separately in Appendix E.1.

C.3 Baseline configurations

All methods share attributed modules, evaluation examples, queries, and loss definitions. For OLMo 2, EOGP, LoGra, GraSS, and LoRIF fit their curvature estimates on the same 10,000 conversations. Storage formats and budget matching follow Appendix C.2.

Compression baselines. Our LoGra implementation is based on LogIX[1], with random and PCA initialization on both GPT-2 and OLMo 2. LoGra uses the same relative damping coefficient (0.1) as in Appendix C.2. PCA initialization uses factor eigenvectors fitted with Kronfluence[2], and the factor rank a determines a^2 stored coordinates per module. Our GraSS implementation follows the official code, using factor-wise random masking and CountSketch with output dimensions matched to LoGra. Our LoRIF reimplementation uses randomly projected gradient matrices and a rank-256 inverse-curvature approximation.

Uncompressed references. EK-FAC and K-FAC use Kronfluence with model-sampled labels and the damping in Appendix C.2. The TracIn reference is the unweighted gradient inner product at the final checkpoint.

Gaussian random projection. This control uses independent standard Gaussian factors to produce a^2 coordinates per module. Training gradients are projected directly, and query gradients are preconditioned by the full-space damped EK-FAC inverse before projection. Inner products are scaled by a^{-2}.

Randomization. LoGra and LoRIF use one projection realization per configuration. GraSS uses one on GPT-2 and averages over three seeds on OLMo 2. The Gaussian control averages over three independent draws, and EOGP-R over up to three disjoint blocks of a shared SRHT output. Retraining and uncertainty estimation follow Appendix D.1.

D Evaluation Protocols

D.1 GPT-2 retraining evaluation

We evaluate attribution on WikiText-2 using the linear datamodeling score (LDS) and counterfactual retraining.

Linear datamodeling score. We reuse the publicly released subset masks and retraining losses from Kronfluence for $R = 100$ subsets, each retaining half of the training blocks. All methods and storage budgets use the same subsets and losses on the validation blocks. The query measurement is the token-summed next-token cross-entropy loss of a validation block.

Let S_r denote the retained training subset and Y_{rq} its released retraining loss for query q. Under our removal-based influence convention, the predicted subset score is

$$A_{rq} = -\sum_{i \in S_r} \widehat{\mathcal{I}}_q(i). \tag{32}$$

For each query, we compute the Spearman correlation across subsets,

$$\mathrm{LDS}_q = \rho_{\mathrm{S}}\left((A_{rq})_{r=1}^{R}, (Y_{rq})_{r=1}^{R}\right), \tag{33}$$

and report its mean over queries, excluding undefined correlations. We use average ranks for ties. We obtain 95% percentile confidence intervals from 1,000 bootstrap resamples of the subsets, recomputing the query-averaged LDS for each resample.

Counterfactual retraining. For each method and storage budget, we rank the training blocks by the sum of their signed influence scores over the first 50 validation blocks. We remove the examples with the largest scores, reusing the same ranking for removal counts from 50 to 300 in increments of 50.

Each retraining run starts from the pretrained GPT-2 weights and uses AdamW with a learning rate of 3×10^{-5}, weight decay of 0.01, three epochs, and a training batch size of eight. We use a set $\mathcal{S}$ of five seeds, matched across methods. Perplexity is evaluated on the same 50 validation blocks used to construct the ranking by exponentiating their mean token-level cross-entropy loss.

[1] https://github.com/logix-project/logix
[2] https://github.com/pomonam/kronfluence

Let $\mathrm{PPL}_s(b)$ denote the perplexity for seed s after removing b examples. We report

$$\Delta\,\mathrm{PPL}(b) = \frac{1}{5}\sum_{s\in\mathcal{S}}\left[\mathrm{PPL}_s(b) - \mathrm{PPL}_s(0)\right], \tag{34}$$

where the zero-removal runs are shared across methods. Error bars show the population standard deviation of the post-removal perplexities across seeds. For the random-removal control, each seed defines a random permutation of the training set, and the first b examples are removed.

D.2 Fidelity to EK-FAC influence

We compare compressed influence scores with uncompressed EK-FAC scores on the same candidate training examples. OLMo 2 uses 1,000 candidates and 100 queries, while GPT-2 uses the training and validation blocks as candidates and queries, respectively. Module contributions are summed before evaluation, and both score vectors use signed scores under the same removal-based convention, without taking absolute values or clipping.

Spearman correlation. For query q, let $\mathbf{e}_q$ and $\mathbf{a}_q$ denote the reference and compressed score vectors over the candidate examples. We compute

$$\rho_q = \mathrm{Corr}\left(\mathrm{rank}(\mathbf{a}_q), \mathrm{rank}(\mathbf{e}_q)\right), \qquad \overline{\rho}_{\mathrm{S}} = \frac{1}{|\mathcal{Q}_{\mathrm{valid}}|}\sum_{q\in\mathcal{Q}_{\mathrm{valid}}}\rho_q. \tag{35}$$

Ties receive average ranks. A correlation is undefined when either score vector is constant, and $\mathcal{Q}_{\mathrm{valid}}$ contains the queries with defined correlations.

NDCG@20. We use binary relevance based on membership in the reference top 20. Let T_q contain the 20 examples with the largest signed EK-FAC scores, and let $\pi_q(t)$ denote the example at position t when compressed scores are sorted in descending order. Then

$$\mathrm{NDCG@20}(q) = \frac{\displaystyle\sum_{t=1}^{20}\frac{\mathbf{1}\{\pi_q(t)\in T_q\}}{\log_2(t+1)}}{\displaystyle\sum_{t=1}^{20}\frac{1}{\log_2(t+1)}}. \tag{36}$$

The denominator is the ideal discounted gain when all 20 relevant examples occupy the first 20 positions. We report the arithmetic mean of NDCG@20 over queries.

D.3 Relationship between fidelity and retraining

Figure 3 compares EK-FAC fidelity with LDS across the 25 compressed configurations and three reference methods in Figure 2. For each configuration, all metrics use the same attribution score matrix. We average each metric over queries before computing correlations across configurations.

Let $\mathcal{J}$ denote these 28 configurations, and let L_j, N_j, and S_j denote their query-averaged LDS, NDCG@20, and Spearman fidelity. The resulting correlations are

$$\begin{aligned}\rho_{\mathrm{S}}((L_j)_{j\in\mathcal{J}}, (N_j)_{j\in\mathcal{J}}) &\approx 0.95,\\ \rho_{\mathrm{S}}((L_j)_{j\in\mathcal{J}}, (S_j)_{j\in\mathcal{J}}) &\approx 0.84.\end{aligned} \tag{37}$$

E Storage and Runtime Accounting

E.1 Per-example payload and shared artifacts

We account separately for per-example representations and artifacts shared across the gradient store. Let $\mathcal{U}$ denote the attributed modules, m_u and k_u the first-stage and final output dimensions of module u, respectively, and N_{store} the number of stored examples. The attributed parameter set is specified in Appendix C.1.

Table 2: Per-example representation payloads at $k_u = 2{,}048$ per module. Uncompressed gradients cover the attributed parameters, and one-bit representations include packed signs and FP16 scales. Shared artifacts are accounted for in Appendix E.1.

Model	Uncompressed FP16 (GB/example)	One-bit representation (KB/example)
OLMo 2 1B	2.147	28.896
OLMo 2 7B	12.952	57.792
OLMo 2 13B	25.376	72.240
OLMo 2 32B	62.411	115.584

Per-example payload. EOGP and EOGP-R store one sign bit per coordinate and one FP16 scale per module. With byte packing performed separately for each module, the payload is

$$B_{\text{1bit}} = \sum_{u \in \mathcal{U}} \left(\left\lceil \frac{k_u}{8} \right\rceil + 2 \right) \quad \text{bytes.} \tag{38}$$

For a uniform dimension k, this becomes $|\mathcal{U}|(\lceil k/8 \rceil + 2)$. Storing the coordinates directly in FP16 requires $B_{\text{FP16}} = 2\sum_u k_u$ bytes, while an uncompressed FP16 gradient requires $B_{\text{raw}} = 2d_{\text{attr}}$ bytes, where d_{attr} is the number of attributed parameters.

Shared artifacts. The Kronecker eigenbases and half-whitening weights are shared across examples and queries. In EOGP, each module's PCA correction matrix $P_u \in \mathbb{R}^{m_u \times k_u}$ is also shared and stored in FP16, requiring

$$B_{P,u} = 2m_u k_u \quad \text{bytes.} \tag{39}$$

For the 32B model with $m_u = 262{,}144$ and $k_u = 2{,}048$, each FP16 correction matrix requires approximately 1.07 GB, totaling 481 GB across all 448 modules. This cost is independent of the number of stored examples, so its relative contribution decreases as the store grows. For stores containing 100 million or one billion examples, the correction matrices would occupy approximately 4.2% or 0.42% of the aggregate per-example payload, respectively. These estimates illustrate how the shared correction matrices can be amortized when constructing attribution stores for large pretraining or mid-training corpora.

When shared storage is constrained, reducing k_u decreases the correction-matrix storage linearly. For example, keeping m_u unchanged and reducing k_u to 256 lowers the total correction-matrix storage to approximately 60.1 GB for the 32B model. As shown in Figure 4, EOGP maintains an advantage over the evaluated compression baselines at small per-example storage budgets.

Total storage. Including shared artifacts and file overhead, the total storage is

$$B_{\text{total}} = N_{\text{store}} B_{\text{1bit}} + B_{\text{shared}} + B_{\text{overhead}}, \tag{40}$$

where B_{shared} counts shared curvature and projection artifacts, including $\sum_{u \in \mathcal{U}} B_{P,u}$ for the PCA correction matrices in EOGP, and B_{overhead} counts file metadata. Per-example budgets count representation payloads, with shared artifacts accounted for separately.

Storage quantities use decimal units. The nominal GPT-2 budget labels follow Appendix C.2.

E.2 Storage estimates by model

Table 2 compares per-example storage for uncompressed FP16 gradients and one-bit representations at $k_u = 2{,}048$. Both EOGP and EOGP-R use $2048/8 + 2 = 258$ bytes per module per example.

E.3 Implementation and runtime

Setup. Table 3 compares our implementation with LoGra's official LogIX implementation on a single NVIDIA B200. EOGP uses $m_u = 262{,}144$, $k_u = 2{,}048$, and FP16 correction matrices, while LoGra stores 2,025 FP16 coordinates per module. Curvature statistics are fitted beforehand, and timings exclude curvature fitting and model loading.

Table 3: PCA fitting and store-construction times on one NVIDIA B200. Fitting starts from cached coordinates and includes saving the correction matrices. Store-construction totals combine separately timed stages, including disk I/O.

Model	EOGP PCA fitting (min)	Store construction (s/example)	
		EOGP	LoGra
OLMo 2 1B	4.5	0.0563	0.0577
OLMo 2 7B	8.9	0.0999	0.1315
OLMo 2 13B	11.1	0.1474	0.1661
OLMo 2 32B	18.4	0.2470	0.2780

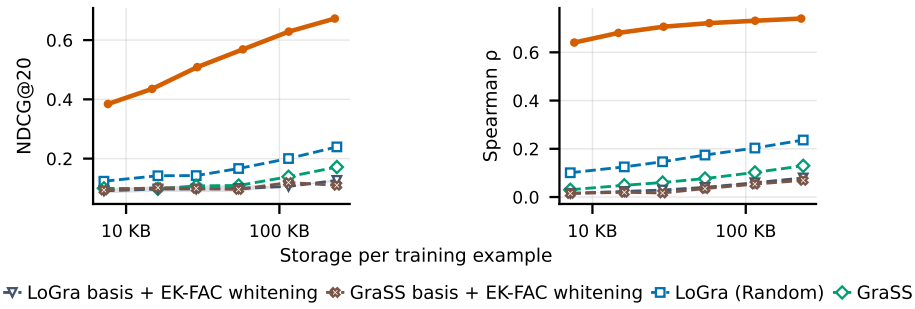

Figure 6: Comparison of EOGP with LoGra and GraSS on OLMo 2 7B SFT, using either their original curvature treatment or EK-FAC half-whitening before projection. NDCG@20 (left) and Spearman correlation (right) measure agreement with EK-FAC influence at matched per-example storage budgets.

Implementation. First-stage coordinates are cached in FP16 using asynchronous disk writes. We then process modules sequentially, loading each correction matrix onto the GPU and applying it to a batch of coordinates before one-bit encoding and recording. Each loaded matrix is reused across the batch.

Timing protocol. Store construction includes gradient computation, projection, and disk I/O, including correction-matrix loading and transfer for EOGP. Coordinate generation is timed on 40 documents per model averaging approximately 500 tokens. Subsequent PCA projection and storage use batches of approximately 11,000 documents, over which matrix-loading costs are amortized.

One-time PCA fitting. Fitting starts from cached first-stage coordinates of approximately 10,000 examples and follows Appendix B.2. Timings include reading the coordinates, computing the correction matrices, and saving them in FP16. This one-time cost is 18.4 minutes for the 32B model and is shared across examples and queries.

Batched query processing. For the 32B model with $m_u = 262{,}144$ and $k_u = 2{,}048$, processing 100 queries against 1,000 stored examples takes 60.36 seconds on a single NVIDIA B200, starting from cached first-stage coordinates. This includes loading and transferring the FP16 correction matrices, applying the PCA projection, and scoring the stored examples. Each correction matrix is loaded once per batch, giving an amortized processing cost of 0.604 seconds per query.

F Comparison under Shared EK-FAC Curvature

To examine whether EOGP benefits primarily from using the same curvature approximation as the evaluation reference, we construct variants of LoGra and GraSS that apply our EK-FAC half-whitening before their respective projections. On OLMo 2 7B SFT, EOGP substantially outperforms

Table 4: Pearson correlation between FP16 and one-bit influence scores on GPT-2. We report the median of the query-wise correlations over 481 queries. Storage budgets refer to one-bit storage per training example. The highest correlation at each budget is shown in bold.

Method	12 KB	24 KB	96 KB
EOGP	**0.902**	**0.890**	**0.919**
LoGra (Random)	0.604	0.447	0.086
LoGra (PCA)	0.750	0.570	−0.355
GraSS	0.330	0.079	−0.205
LoRIF	0.820	0.823	0.822

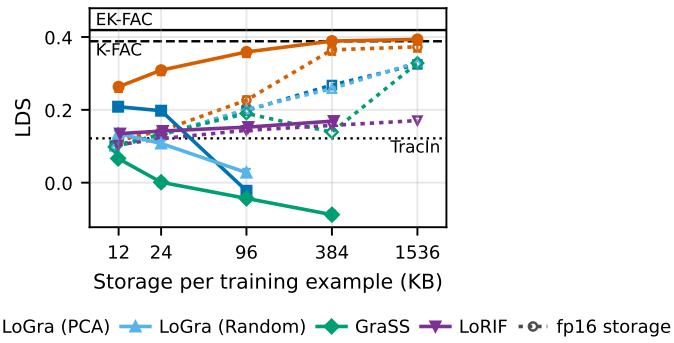

Figure 7: LDS of compression methods under one-bit and FP16 storage on GPT-2 at matched per-example storage budgets. Horizontal lines denote uncompressed references. Curves end where the required projection dimension exceeds the method's supported dimension.

both variants at matched per-example storage budgets, showing that matching the reference curvature alone does not explain its advantage.

The original LoGra and GraSS configurations also generally outperform their half-whitened variants, supporting the effectiveness of their post-projection curvature approximations in this setting. One possible explanation is that estimating curvature after projection accounts for the variances and correlations of the retained coordinates, whereas applying half-whitening before projection produces a different transformation. These results suggest that the effectiveness of a curvature approximation depends on how it is combined with gradient compression.

G Effect of One-Bit Quantization on Attribution Quality

We further examine how one-bit storage affects different gradient representations using the GPT-2 setting in Section 4.1. We compare attribution quality at matched per-example storage budgets and separately measure how quantization changes influence scores while holding the projection dimension fixed.

Attribution quality under one-bit storage. Figure 7 compares one-bit and FP16 storage for the evaluated compression methods. For baselines that estimate curvature after projection, larger output dimensions increase the memory required for curvature estimation, so we evaluate their one-bit variants only up to the largest feasible storage budget. Under one-bit storage, EOGP achieves the highest LDS among the evaluated compression methods at each displayed budget, with performance improving as more storage is allocated. The one-bit variants of LoGra and GraSS exhibit a different trend, with LDS declining as the budget increases. LoRIF improves more gradually but remains below EOGP. At larger storage budgets, EOGP also outperforms the other compression baselines when all methods use FP16 storage, indicating that its advantage is not solely attributable to one-bit

quantization. These results show that the effectiveness of one-bit storage depends on the representation being quantized, and that EOGP's advantage persists in a retraining-based evaluation.

Preservation of influence scores. To examine the effect of quantization directly, we compare influence scores obtained with FP16 and one-bit storage at the same projection dimension within each method. For each query, we compute the Pearson correlation between the two sets of scores across candidate training examples, then report the median correlation over queries in Table 4.

For EOGP, these correlations range from 0.890 to 0.919 across the evaluated configurations, indicating that the scores remain strongly correlated after quantization. LoGra and GraSS show substantially weaker agreement in several configurations. These correlations measure the preservation of score patterns rather than absolute score magnitudes, since Pearson correlation is unchanged by a positive rescaling or an additive shift. Together with the LDS results, they provide complementary evidence that EOGP preserves useful attribution information under one-bit storage.

A possible explanation. One possible explanation is that half-whitening makes the representation more compatible with scaled sign quantization. This quantization preserves coordinate signs but replaces their magnitudes with a common scale given by the mean absolute coordinate value. It therefore approximates vectors more accurately when their coordinate magnitudes are relatively uniform. By rescaling gradients using curvature information, half-whitening may reduce magnitude imbalances in the resulting coordinates and thereby lessen the distortion introduced by quantization. The relevant property is the relative variation in coordinate magnitudes, since uniformly rescaling a vector also rescales its quantization scale and leaves the relative reconstruction error unchanged.