INTERACTION-AWARE INFLUENCE FUNCTIONS FOR GROUP ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Influence functions approximate how removing a training example changes a quantity of interest, called the target function, such as a held-out loss. To estimate the influence of a group of examples, the standard practice is to sum the individual influences of its members. However, this sum does not capture how examples jointly affect the target: a pair of examples may be redundant or complementary, but the sum cannot distinguish these cases. We propose an interaction-aware influence function that characterizes how interactions between examples influence the target. By expanding the target to second order around the trained parameters, we obtain an estimator that augments the standard sum with a pairwise interaction term that captures the alignment between two examples' effects on the target. We empirically evaluate our estimator in two settings. First, on six dataset–model pairs spanning logistic regression, MLPs, and ResNet-9, our estimator tracks leave-group-out retraining substantially better than first-order influence across all settings. Second, when used as a greedy algorithm for offline selection of instruction-tuning data on Llama-3.1-8B, it outperforms prior influence-based and representation-similarity baselines on five of seven downstream tasks, in a regime where standard influence-based selection underperforms random selection. Code is available at https://anonymous.4open.science/r/Interaction_IF-45D6.

1 INTRODUCTION

Quantifying how training examples shape a trained model is a foundational problem in machine learning, with applications including selecting valuable training data (Engstrom et al., 2024; Xia et al., 2024; Yu et al., 2024), understanding what kinds of data benefit a model (Grosse et al., 2023; Heo et al., 2025), and fairly compensating data providers (Choe et al., 2025; Ghorbani & Zou, 2019). The conceptual gold standard is leave-one-out retraining: remove a training example, retrain the model, and measure how some quantity of interest has changed. Since retraining for every example is prohibitive, influence functions (Koh & Liang, 2017) estimate this counterfactual in closed form by modeling two things in turn: how reweighting a training example would shift the trained parameters, and how those shifted parameters would change the quantity of interest. The quantity of interest, called the *target function*, is left to the user. It can be the loss on a held-out example (Koh & Liang, 2017), an average over a validation set, the likelihood of a downstream task answer (Choe et al., 2025; Xia et al., 2024), or any other smooth function of the model. This flexibility is what lets a single approximation serve so many applications.

Most of these applications, including subset selection, group-robustness analysis, and data valuation, ultimately ask about *groups* of examples rather than single ones. The textbook extension scores a group as the sum of its members' individual influences (Koh et al., 2019), implicitly assuming examples contribute independently. This assumption is rarely true. Two near-duplicate examples can each look highly beneficial on their own, yet adding both may provide little benefit beyond adding one. In such cases, additive scoring double-counts their overlap and overestimates the group's benefit (Hu et al., 2024; Huang et al., 2025). The same blind spot makes top-k influence-based data selection produce subsets that are highly redundant rather than collectively useful (Zhang et al., 2025), since the most individually influential examples tend to be similar. Basu et al. (2020) previously addressed this with a second-order correction that makes the estimator non-additive, but their

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.

construction refines only how the trained parameters shift, missing how the target function responds to those shifts.

We argue that group attribution is fundamentally about how training examples affect the target. Therefore, interactions between examples should be characterized in terms of their joint effect on the target, not just on the parameter shift. To this end, we propose an *interaction-aware influence function* that augments the standard sum of individual influences with a pairwise interaction term over examples in the group. The pairwise term captures the joint effect on the target: it is positive when the two examples push the model in similar directions and negative when they push in opposite directions. We further derive a closed-form factorization of the interaction term across classification settings, proving for binary classification that two similar examples are jointly complementary when their labels differ and redundant when their labels agree, formally recovering classical principles in supervised learning (Birodkar et al., 2019; Cortes & Vapnik, 1995; Mirzasoleiman et al., 2020; Robinson et al., 2021). As an application, we use the interaction term to discount candidates whose effects overlap with already selected examples, addressing the redundancy in top-k influence-based selection.

We evaluate the method in two settings. First, we test whether our estimator faithfully tracks leave-group-out retraining across six dataset–model pairs, by assessing Spearman rank correlation with ground-truth retraining effects. It substantially outperforms prior influence-based training data attribution methods, demonstrating the importance of accounting for interactions in group attribution. Second, we evaluate the offline data selection method at scales ranging from small MLPs to instruction-tuning of Llama-3.1-8B (Grattafiori et al., 2024). On small-scale models, our method selects subsets with class diversity comparable to random selection, whereas baselines collapse onto a few classes and fall below random. In instruction-tuning data selection, our method consistently improves over random selection and achieves the strongest overall performance on five of seven downstream tasks.

2 PRELIMINARY

We first review the classical influence function and its standard extension to group attribution. We summarize the notation used throughout the paper in Table 2 of Appendix A. Let $\mathcal{D} = \{z_i\}_{i=1}^N$ be the training set of size N, where each example $z_i = (x_i, y_i)$ consists of an input x_i and a label y_i, and let $\ell(z_i, \theta)$ denote the loss of example z_i at parameter $\theta \in \Theta$. For any subset $A \subseteq \mathcal{D}$, define the empirical risk and its minimizer as

$$\mathcal{R}_A(\theta) \coloneqq \frac{1}{|A|} \sum_{z_i \in A} \ell(z_i, \theta), \qquad \hat{\theta}_A \coloneqq \arg\min_\theta \mathcal{R}_A(\theta). \tag{1}$$

We write $\hat{\theta} \coloneqq \hat{\theta}_{\mathcal{D}}$ for the parameter trained on the full dataset. For a subset $S \subseteq \mathcal{D}$, the parameter obtained by retraining after removing S is $\hat{\theta}_{\mathcal{D}\setminus S}$.

Let $f : \Theta \to \mathbb{R}$ be a differentiable scalar target function of the trained parameters that we treat as a quantity to be minimized, such as the loss on a test example or the average loss on a validation set.[1] The exact leave-group-out effect of removing S is

$$\mathcal{I}^-(S) \coloneqq f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}). \tag{2}$$

For a singleton $S = \{z_i\}$, this reduces to the usual leave-one-out effect. Computing $\mathcal{I}^-(S)$ exactly requires retraining the model without S, which is expensive even for a single example and infeasible for many candidate groups.

Influence functions approximate this retraining effect through infinitesimal reweighting (Koh & Liang, 2017). Instead of removing S directly, consider the parameter obtained after adding a small weight ϵ to the losses of examples in S:

$$\hat{\theta}(\epsilon;\, S) \coloneqq \arg\min_\theta \left\{ \frac{1}{N} \sum_{z_i \in \mathcal{D}} \ell(z_i, \theta) + \epsilon \sum_{z_i \in S} \ell(z_i, \theta) \right\}. \tag{3}$$

[1] A target to be maximized can be handled by negating f.

At $\epsilon = 0$, this recovers the original parameter $\hat{\theta}$. Since each example has weight $1/N$ in the empirical risk, setting $\epsilon = -1/N$ removes the contribution of every example in S. Thus, leave-group-out retraining can be viewed as a finite step along the reweighting path $\hat{\theta}(\epsilon;\, S)$.

The influence function replaces this finite step with two first-order approximations. Let $H := \frac{1}{N}\sum_{z_i \in \mathcal{D}} \nabla^2_\theta \ell(z_i, \hat{\theta})$ be the Hessian of the training objective. Assuming that $\ell(z_i, \cdot)$ is twice continuously differentiable for every $z_i \in \mathcal{D}$ and that H is nonsingular at $\hat{\theta}$, the first approximation linearizes the reweighting path $\hat{\theta}(\epsilon;\, S)$ in ϵ at $\epsilon = 0$, yielding the parameter shift induced by removing S as

$$\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \approx \frac{1}{N} H^{-1} \sum_{z_i \in S} \nabla_\theta \ell(z_i, \hat{\theta}). \tag{4}$$

Let $g_i := \nabla_\theta \ell(z_i, \hat{\theta})$ denote the per-example gradient and $u_i := H^{-1} g_i$ its preconditioned counterpart. For a group S, define $u_S := \sum_{z_i \in S} u_i$, so that the first-order approximation to the removal-induced parameter shift in Equation (4) is u_S/N. The second approximation linearizes the target function f around $\hat{\theta}$, giving the standard group influence-function estimate of $\mathcal{I}^-(S)$:

$$\hat{\mathcal{I}}^-_{\mathrm{lin}}(S) := \frac{1}{N} \nabla_\theta f(\hat{\theta})^\top u_S. \tag{5}$$

The derivations of Equation (4) and Equation (5) are provided in Appendix B.1. Letting $\hat{\mathcal{I}}^-_{\mathrm{lin}}(z_i)$ denote the singleton case, the standard group estimate is additive: $\hat{\mathcal{I}}^-_{\mathrm{lin}}(S) = \sum_{z_i \in S} \hat{\mathcal{I}}^-_{\mathrm{lin}}(z_i)$. This additivity is a consequence of the first-order approximation, not a property of the exact retraining effect $\mathcal{I}^-(S)$. As a result, standard influence functions cannot capture interactions among examples in a group, which motivates the interaction-aware approximation developed in the next section.

3 Method

We now develop our interaction-aware influence function, which captures how examples in a group jointly affect a target function. Section 3.1 derives a group-attribution estimator by expanding the target function to second order, yielding the standard additive influence plus a pairwise interaction term. Section 3.2 then extends the estimator to a greedy data-selection procedure that discounts candidates whose effect overlaps with already-selected examples. Full derivations and proofs for this section are provided in Appendices B and C.

3.1 Interaction-aware influence functions

The standard influence function loses interaction information because the first-order approximation treats each example's effect as independent of the others. We retain the first-order parameter-shift approximation in Equation (4) and replace the linear target approximation underlying Equation (5) with a second-order expansion, introducing a target-curvature correction that captures interactions between examples.

Second-order expansion of the target function. We expand the target function f to second order around $\hat{\theta}$:

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) \;=\; \nabla_\theta f(\hat{\theta})^\top \delta_S \;+\; \tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S \;+\; O(\|\delta_S\|^3), \tag{6}$$

where $\delta_S := \hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}$ is the parameter shift induced by removing S and $H_f := \nabla^2_\theta f(\hat{\theta})$ is the Hessian of the target at $\hat{\theta}$. Substituting Equation (4) into Equation (6) yields our interaction-aware influence function, an estimator of $\mathcal{I}^-(S)$:

$$\hat{\mathcal{I}}^-(S) \;:=\; \underbrace{\frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S}_{\text{first-order influence}} \;+\; \underbrace{\frac{1}{2N^2}\, u_S^\top H_f\, u_S}_{\text{interaction term (ours)}}. \tag{7}$$

The first term recovers the standard first-order estimate $\hat{\mathcal{I}}^-_{\mathrm{lin}}(S)$ from Equation (5). The second term, which we call the interaction term, is a curvature correction induced by the target function and is the source of non-additivity across examples in S.

The same expansion applies to the data-addition setting, which arises in problems such as data selection and active learning. Adding S to $\mathcal{D}$ flips the sign of the linear term while leaving the quadratic term unchanged, giving

$$\hat{\mathcal{I}}^{+}(S) \;\coloneqq\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\,u_S^\top H_f\,u_S. \tag{8}$$

The full derivation for the data-addition setting is provided in Appendix B.2. For scalable influence computation, we use standard Hessian approximations (George et al., 2018; Grosse et al., 2023; Choe et al., 2025), as detailed in Appendix E.

Interpreting the pairwise structure of the interaction term. The interaction term decomposes into a sum of pairwise contributions between examples in S:

$$u_S^\top H_f\,u_S \;=\; \sum_{z_i,z_j\in S} \kappa(z_i,\,z_j), \qquad \kappa(a,\,b) \coloneqq u_a^\top H_f\,u_b. \tag{9}$$

For interpretive clarity, we first consider the case in which H_f is positive definite; $\kappa(a,\,b)$ is then the inner product of the preconditioned gradients u_a and u_b under the metric induced by H_f. A positive value indicates that the two vectors are aligned under this metric, meaning the examples perturb the parameters in directions that produce similar changes in the target function, while orthogonal or opposing shifts produce values near zero or negative. In this sense, $\kappa(a,\,b)$ quantifies how *similarly* two examples act on the target. In the more general setting where H_f is indefinite, $\kappa(a,\,b)$ generalizes from an inner product to a symmetric bilinear form. We provide a spectral analysis of this case in Appendix C.1. The sum also includes self-interaction terms $\kappa(z_i, z_i)$, which provide per-example curvature corrections. Only interactions between distinct examples contribute to the non-additivity of the group estimate.

The first-order and interaction terms balance usefulness and redundancy. The pairwise interpretation above clarifies why individually useful examples need not form the most useful group. For illustration, consider adding two examples that are each predicted to lower the target. The first-order term simply sums their estimated effects. However, this additive score does not account for overlap between their effects. When two examples have a positive pairwise interaction, our quadratic approximation predicts less benefit from adding them together than the sum of their separate benefits. The resulting criterion therefore considers both individual usefulness and redundancy, motivating the greedy selection procedure in Section 3.2.

Interpreting the pairwise interaction in binary logistic regression. To better understand the pairwise interaction, we analyze a tractable case in which the interaction admits a closed form: binary logistic regression with ℓ_2 regularization. Let σ denote the sigmoid function, and write $\sigma_i \coloneqq \sigma(\hat{\theta}^\top x_i)$ for the predicted probability on example i.

Proposition 1 (Closed-form factorization of κ)**.** *For binary logistic regression at $\hat{\theta}$ with $y_i \in \{0, 1\}$,*

$$\kappa(a,\,b) = (\sigma_a - y_a)(\sigma_b - y_b)\cdot\langle x_a, x_b\rangle_M,$$

where $\langle u, v\rangle_M \coloneqq u^\top M\,v$ is the bilinear form induced by $M \coloneqq H^{-1}H_f\,H^{-1}$.

The proof relies on the closed-form gradient $\nabla_\theta \ell(z_i, \hat{\theta}) = (\sigma_i - y_i)x_i$. Full details are given in Appendix C.2.

Proposition 1 shows that when $\langle x_a, x_b\rangle_M > 0$, indicating similarity under the bilinear form induced by M, the sign of $\kappa(a,\,b)$ is determined by the sign of the class-agreement factor $(\sigma_a - y_a)(\sigma_b - y_b)$. Since $\sigma_i \in [0, 1]$, this factor is positive for same-class pairs and negative for cross-class pairs. Because the target is a quantity to be minimized, our quadratic approximation predicts that adding two similar examples with different labels yields a greater benefit than the sum of their individual benefits, whereas adding similar examples with the same label yields diminishing returns.

This formally recovers two well-known principles of supervised learning: cross-class pairs with similar features shape the decision boundary and are thus complementary (Cortes & Vapnik, 1995; Robinson et al., 2021; Shrivastava et al., 2016), whereas same-class pairs with similar features provide overlapping evidence and yield diminishing returns (Birodkar et al., 2019; Mirzasoleiman et al.,

Algorithm 1 Greedy selection with interaction-aware influence

Require: Candidate pool $\mathcal{D}_{\text{pool}} = \{z_i\}_{i=1}^{P}$, budget $1 \le K \le P$, precomputed $\{u_i\}$, $\{w_i := H_f\, u_i\}$, $\{q_i := u_i^\top w_i\}$, target gradient $\nabla_\theta f(\hat{\theta})$, training set size $N := |\mathcal{D}|$
Ensure: Selected subset $S \subseteq \mathcal{D}_{\text{pool}}$ with $|S| = K$

```
Initialize S ← ∅ and accumulator w ← 0, where w tracks H_f u_S
for t = 1, ..., K do
    for i such that z_i ∈ D_pool \ S do
        m(z_i | S) ← −(1/N) ∇_θ f(θ̂)^⊤ u_i + (1/N²) w^⊤ u_i + (1/2N²) q_i
    end for
    i* ← arg min_{i: z_i ∈ D_pool \ S} m(z_i | S)
    S ← S ∪ {z_{i*}},   w ← w + w_{i*}
end for
return S
```

2020; Toneva et al., 2019). Proposition 1 shows that the interaction term encodes both principles directly, with the sign of $\kappa(a,\, b)$ aligning with the role each pair plays in shaping the classifier. For clarity of exposition, we have presented this analysis in the binary logistic regression setting. The factorization also extends to deep classifiers, as detailed in Appendix C.3.

Remark (Relation to Basu et al. (2020)). As described in Section 2, standard influence functions approximate both the parameter shift and its effect on the target to first order. Basu et al. (2020) add a second-order correction to the parameter shift while keeping the target approximation linear. We instead retain the first-order shift and incorporate target curvature. A full second-order expansion includes contributions from both steps. Our parameter-shift experiments in Appendix D show that the Basu correction slightly improves the parameter-shift estimate for logistic regression but substantially increases the relative L_2 error in the evaluated nonlinear models. A possible explanation is that Basu's approximation retains only part of the full second-order parameter-shift correction. When the omitted contributions are substantial, this partial correction can worsen the shift estimate. The attribution results in Section 4 also show substantial performance drops relative to our method when the two corrections are combined.

3.2 Interaction-aware data selection

We now propose an offline data selection method that exploits the interaction term to select examples from a candidate pool for addition to the training set. The subset is selected once and kept fixed throughout subsequent training. This problem reduces directly to group attribution: finding the group that most reduces the target when added to the training data. Formally, given a model trained on $\mathcal{D}$ and a candidate pool $\mathcal{D}_{\text{pool}}$, we seek a subset $S \subseteq \mathcal{D}_{\text{pool}}$ of size K that minimizes the following objective:

$$S^\star \;:=\; \underset{|S|=K}{\arg\min}\, f\left(\hat{\theta}_{\mathcal{D}\cup S}\right), \tag{10}$$

where $\hat{\theta}_{\mathcal{D}\cup S}$ denotes the minimizer of the empirical risk on the augmented set $\mathcal{D} \cup S$. Since $f(\hat{\theta})$ does not depend on S, subtracting it from the objective preserves the minimizer, so this is equivalent to minimizing the inclusion effect $\mathcal{I}^+(S) := f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta})$. As $\mathcal{I}^+(S)$ is intractable to evaluate exactly, we use its estimator $\hat{\mathcal{I}}^+(S)$ from Equation (8) as our selection criterion.

Greedy procedure. Exact combinatorial optimization of $\hat{\mathcal{I}}^+(S)$ is intractable, so we instead adopt a greedy procedure that, starting from $S = \varnothing$, repeatedly appends the candidate z_i minimizing $\hat{\mathcal{I}}^+(S \cup \{z_i\})$. We quantify this change through the marginal score of adding $z_i \notin S$ to a current subset S:

$$m(z_i \mid S) \;:=\; \hat{\mathcal{I}}^+(S \cup \{z_i\}) - \hat{\mathcal{I}}^+(S). \tag{11}$$

Substituting Equation (8) into Equation (11) and using $u_{S\cup\{z_i\}} = u_S + u_i$, which follows from the definition of u_S, to expand the quadratic term gives

$$m(z_i \mid S) \;=\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_i \;+\; \frac{1}{N^2}\,u_S^\top H_f\, u_i \;+\; \frac{1}{2N^2}\,u_i^\top H_f\, u_i. \tag{12}$$

The first term is the standard first-order influence of z_i, which captures its individual contribution to reducing the target. The second term accounts for the pairwise interactions analyzed in Section 3.1 between the candidate z_i and each example in the already-selected subset S. When z_i is similar to examples in S, the second term takes a large positive value and increases the marginal score. The procedure thereby avoids selecting such candidates. The third term depends only on the candidate z_i itself, measuring how strongly its own parameter shift would perturb the target. At each iteration we append the candidate minimizing $m(z_i \mid S)$ until $|S| = K$; the full procedure is given in Algorithm 1.

The interaction-aware structure of Equation (12) distinguishes our procedure from first-order influence selection (Xia et al., 2024), which selects the top-k examples by individual score. Since high-influence examples tend to share characteristics (Zhang et al., 2025), this top-k rule yields redundant subsets and consequently diminishing returns. Our second term penalizes candidates similar to already-selected examples, thereby promoting diversity.

Complexity. The per-candidate quantities $\{u_i\}$, $\{w_i \coloneqq H_f\, u_i\}$, and $\{q_i \coloneqq u_i^\top w_i\}$ in Algorithm 1 depend on the candidate pool and target f but not on the iterates S, so we precompute them once before the greedy loop. Each iteration requires $O(P)$ inner products in dimension d and a single accumulator update, yielding a greedy-loop cost of $O(KPd)$ after precomputation. Appendix H details end-to-end time and storage complexity, and Appendix H.3 reports wall-clock timings.

4 Faithfulness to Retraining

We validate the interaction-aware estimator developed in Section 3.1 along two axes. We first measure how faithfully each estimator tracks the leave-group-out retraining effect on small-scale models, where exact ground truth can be computed, and compare against first-order influence and prior attribution methods. We then examine the pairwise interaction term's behavior across class pairs to illustrate the class-agreement structure of Proposition 1.

Setup. We evaluate six dataset–model pairs using logistic regression and an MLP on MNIST (LeCun et al., 1998), MLPs on FashionMNIST (Xiao et al., 2017), Concrete (Yeh, 1998), and Parkinsons (Tsanas et al., 2009), and ResNet-9 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009). For groups of examples with similar model outputs, we measure Spearman correlation between estimated influence and the held-out loss change after leave-group-out retraining. We compare first-order influence (*F*), the parameter-response correction of Basu et al. (2020) (*F+B*), our target-curvature correction (*F+I*), both corrections (*F+I+B*), and group sums of TRAK and TracIn scores (Park et al., 2023; Pruthi et al., 2020). Dataset details, group construction, and curvature settings are provided in Appendix F.

Results. Figure 1 reports the Spearman rank correlation across the six dataset–model pairs. The accuracy of first-order influence (*F*) varies substantially across settings, ranging from strong correlation on MNIST to near-zero on the harder pairs. Our estimator (*F+I*) achieves the highest correlation in every setting, improving upon first-order influence by up to 0.67 and yielding meaningfully positive correlations even where first-order influence performs poorly. This supports our claim that the interaction term captures information that first-order influence systematically overlooks. The ablation in Appendix G.1 compares our full estimator with F+self. This additive variant includes first-order influence and the self-interaction terms $\kappa(z_i, z_i)$, but omits interactions between distinct examples. Our full estimator achieves higher Spearman correlation in all six settings, demonstrating the benefit of including these interactions.

The Basu-corrected estimator (*F+B*) performs inconsistently across settings, and combining it with our target-curvature correction (*F+I+B*) yields either negligible changes or substantial decreases in attribution correlation. This behavior is consistent with the parameter-shift diagnostics in Appendix D, where the Basu correction substantially increases the relative L_2 error in the evaluated nonlinear models. TRAK and TracIn perform comparably to first-order influence. These findings are robust to the damping hyperparameter, as shown in Appendix G.2.

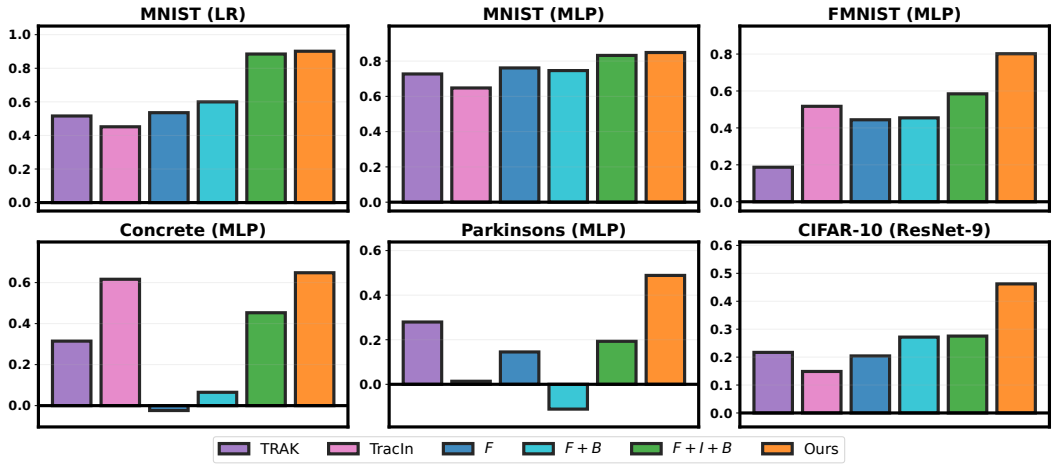

Figure 1: Spearman rank correlation between estimated and ground-truth group influences across six dataset–model pairs, where F, B, and I denote the first-order influence function, the second-order term from Basu et al. (2020), and our interaction term, respectively.

Pairwise interaction analysis across classes. Given a multiclass classification problem, we identify the class pairs whose joint effect on the test loss differs most from the sum of their individual influences. For each pair of classes (i, j), including within-class pairs $i = j$, we compute the average pairwise interaction κ over all example pairs with one example in class i and the other in class j. Figure 2 shows representative images from the class pairs with the lowest and highest averages for MNIST/LR and CIFAR-10/ResNet-9. The lowest-κ pairs are visually similar examples from different classes, while the highest-κ pairs come from the same class, consistent with Proposition 1.

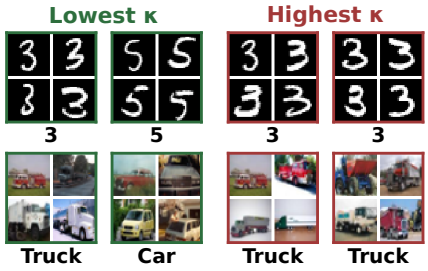

Figure 2: Examples from class pairs with the lowest (left) and highest (right) mean pairwise interaction.

5 Application to Offline Data Selection

We now evaluate our estimator as a selection criterion via Algorithm 1. Section 5.1 validates the procedure at a scale where ground-truth subset quality can be measured by retraining. Section 5.2 then tests whether the gains scale to instruction-tuning data selection for Llama-3.1-8B (Grattafiori et al., 2024).

5.1 Validation on Small-Scale Models

This subsection isolates the contribution of the interaction term to selection quality. We compare against influence-function variants on the held-out test loss after retraining, and additionally analyze the diversity of the selected subsets through their class composition.

Setup. We apply Algorithm 1 to two-layer MLPs on MNIST and FashionMNIST. For each dataset, we construct a candidate pool $\mathcal{D}_{\text{pool}}$ of $|\mathcal{D}_{\text{pool}}| = 5{,}000$ examples sampled uniformly at random from the training set. For each selection method, we form a subset $S \subseteq \mathcal{D}_{\text{pool}}$ of size $K \in \{500, 1000, \ldots, 5000\}$, retrain the MLP from scratch on S, and report the resulting held-out test loss. To diagnose selection diversity, we additionally report the class entropy $\mathcal{H}(S) := -\sum_c p_c(S) \log p_c(S)$ of the selected subset, where $p_c(S)$ is the fraction of examples in S belonging to class c, with higher values indicating more uniform class coverage. We compare our method (F+I) against the same influence-function baselines as in Section 4 (F, F+B, F+I+B) and random selection. Further details are in Appendix F.

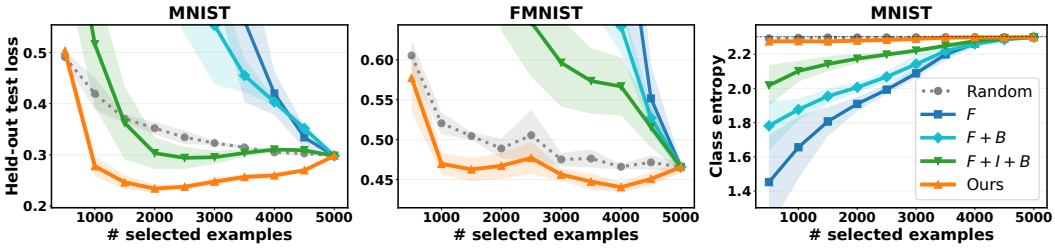

Figure 3: Held-out test loss after retraining on the selected subset on MNIST (left) and Fashion-MNIST (middle); class entropy of the subset on MNIST (right). Across selection sizes, lines and shaded regions show the mean and standard deviation over five seeds.

Results. The left and middle plots in Figure 3 report the held-out test loss attained by retraining on each method's selected subset. Our method (F+I) consistently outperforms the baselines on both datasets, indicating that the interaction term translates into substantially better subsets. The first-order influence function (F) degrades sharply and falls below random selection. We attribute this to the redundancy of the subsets it selects and analyze it in detail in the class-entropy analysis below. The second-order variant of Basu et al. (2020) (F+B) improves over F but still trails random selection at most budgets.

The right plot of Figure 3 reports the class entropy of subsets selected by each method. At small budgets, both F and F+B collapse onto a few classes, whereas our method matches the entropy of random selection throughout, confirming that the interaction term penalizes redundant candidates. A subset that omits entire classes cannot train a competitive classifier, which directly accounts for the test-loss degradation observed above. The corresponding plot for FashionMNIST is provided in Appendix G.3.

5.2 Data Selection for LLM Instruction Tuning

We now evaluate our selection method at LLM scale by fine-tuning Llama-3.1-8B on selected instruction-tuning subsets and measuring downstream task performance.

Setup. We score the LESS pool (Xia et al., 2024) of approximately 270K instructions using a warmup Llama-3.1-8B checkpoint (Grattafiori et al., 2024) and LoGra (Choe et al., 2025), selecting 13,534 examples (5%) per target with Algorithm 1. The target is instruction-masked negative log-likelihood on an official validation set or held-out training split. We fine-tune the original pretrained checkpoint on each selected subset, testing transfer beyond the local addition counterfactual. We evaluate GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ECQA (Aggarwal et al., 2021), and SQuAD (Rajpurkar et al., 2016), reporting exact match for GSM8K, F1 for SQuAD, and accuracy otherwise.

We compare against five baselines. Additive IF uses the standard first-order influence function (Koh & Liang, 2017) and shares our warmup and projected-gradient pipeline, but omits the interaction term. LESS (Xia et al., 2024) uses gradient similarity, RDS+ (Ivison et al., 2025) and NV-Embed (Lee et al., 2025) use representation similarity, and the random baseline samples uniformly from the pool. All methods use the same pool, budget, and final fine-tuning protocol; task and implementation details are in Appendix F.3.

Results. Table 1 reports the mean and standard deviation over five seeds. Our method attains the highest mean score on five of the seven tasks and surpasses random selection on every task. It also outperforms both Additive IF and LESS on six of the seven tasks. Additive IF shares our pipeline but omits the target-curvature correction and falls below random selection on most tasks. This comparison supports the benefit of incorporating target curvature into selection. On GSM8K, the representation-similarity baselines outperform our method, although our method remains ahead of random selection and both influence-based baselines.

Table 1: Test performance of fine-tuned Llama-3.1-8B on seven target tasks. We report the mean and standard deviation over five seeds; the best result in each column is shown in **bold**.

Method	GSM8K	AQuA	ARC-E	HellaSwag	PIQA	ECQA	SQuAD	**Avg.**
Random	46.75±2.69	24.65±0.72	90.03±0.19	56.46±0.69	77.50±0.47	69.85±0.22	75.76±0.55	63.00
Additive IF	19.35±1.07	20.63±1.03	88.72±0.30	48.66±1.99	77.38±0.79	69.52±0.29	76.92±0.30	57.31
LESS	41.83±1.32	8.82±2.87	90.11±0.15	62.11±0.87	77.40±0.49	71.19±0.12	76.73±0.51	61.17
RDS+	**58.44±1.29**	24.65±2.87	88.70±0.32	52.19±1.03	75.14±1.14	70.88±0.41	57.60±1.91	61.09
NV-Embed	57.97±0.47	8.43±1.06	89.01±0.16	53.94±0.74	75.13±0.44	**71.75±0.24**	76.52±1.10	61.82
Ours	52.69±0.96	**30.94±0.72**	**90.44±0.11**	**63.43±0.34**	**79.39±0.32**	71.17±0.19	**79.87±0.34**	**66.85**

6 Related Work

Data attribution and influence functions. Data attribution methods quantify the contribution of individual training examples to a model's learned parameters or predictions on a target (Ghorbani & Zou, 2019; Koh & Liang, 2017; Park et al., 2023; Pruthi et al., 2020). Among these, influence functions (Koh & Liang, 2017) approximate this counterfactual via local linearization, with subsequent work improving scalability through random-projection approaches (Park et al., 2023), trajectory-based methods (Pruthi et al., 2020), efficient curvature approximations (Grosse et al., 2023; Schioppa et al., 2022; Wang et al., 2025a), and extensions to large language and generative models (Agarwal & Hakkani-Tür, 2025; Chang et al., 2025; Choe et al., 2025; Kwon et al., 2024; Mlodozeniec et al., 2025). Complementary work studies the fragility of influence estimates (Basu et al., 2021; Deng et al., 2024; Kreer et al., 2026; Rubinstein & Hopkins, 2025). Ye et al. (2025) also expand the validation loss to second order, but only for single-example influence in noisy-label settings, without group attribution. Prior work also shows that summing individual attributions mismeasures group-level effects (Hu et al., 2024; Koh et al., 2019; Saunshi et al., 2023). Closest to our setting, Basu et al. (2020) address this additive gap via a response-side expansion; we instead expand the target function, yielding closed-form pairwise interactions. Wang et al. (2025b) and Wei et al. (2025) also model joint influence but focus on trajectory-accumulated rather than counterfactual effects.

Training data selection. Data selection methods score examples by influence (Agarwal & Hakkani-Tür, 2025; Choe et al., 2025; Dai et al., 2025; Xia et al., 2024), diversity (Pan et al., 2024), learned utility (Engstrom et al., 2024; Yu et al., 2024), or Shapley-style contributions (Ghorbani & Zou, 2019). Scalable variants leverage influence distillation or small-model trajectories (Nikdan et al., 2025; Yang et al., 2024). Redundancy-aware pruning further penalizes similar examples via pairwise similarity objectives (Tan et al., 2025). However, individually influential examples need not form the best subset once interactions make collective effects non-additive (Hu et al., 2024; Huang et al., 2025; Wang et al., 2024b; Yu et al., 2025). GREATS (Wang et al., 2024a) models interactions to estimate the effect of a single gradient update for online batch selection. Our method instead approximates group effects after retraining for offline subset selection, as discussed in Appendix I. Our second-order expansion of the target function enables interaction-aware selection by updating each candidate's marginal utility against the already-selected set.

7 Conclusion

We presented an interaction-aware influence function obtained from a second-order expansion of the target function, decomposing group attribution into a standard first-order term and a pairwise interaction term. Empirically, our method improves Spearman correlation with ground-truth retraining effects on six dataset–model pairs. In offline data selection, it improves over random selection on every task and outperforms existing influence-based and representation-similarity baselines on five of seven downstream tasks. Two limitations suggest natural directions for future work: our greedy selection performs strongly in practice but lacks formal optimality guarantees, and our estimator inherits approximation error from the underlying Hessian approximations. Advances in scalable inverse-Hessian estimation would directly improve robustness.

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 code is available at https://anonymous.4open.science/r/Interaction_IF-45D6. Appendices B and C provide derivations and proofs for the influence estimators and interaction analysis, and Algorithm 1 specifies the data-selection procedure. Appendix E describes the curvature approximations and gradient projection used in our implementation. Appendix F documents the datasets, model architectures, training settings, baselines, and evaluation protocols. Additional sensitivity analyses are reported in Appendix G, while computational complexity, storage requirements, hardware, and measured runtimes are provided in Appendix H.

REFERENCES

Ishika Agarwal and Dilek Hakkani-Tür. Neural networks for learnable and scalable influence estimation of instruction fine-tuning data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

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*, 2022.

Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, 2020.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *International Conference on Learning Representations*, 2021.

Vighnesh Birodkar, Hossein Mobahi, and Samy Bengio. Semantic redundancies in image-classification datasets: The 10% you don't need. *arXiv preprint arXiv:1901.11409*, 2019.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *Advances in neural information processing systems*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 2017.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995.

Qirun Dai, Dylan Zhang, Jiaqi W Ma, and Hao Peng. Improving influence-based instruction tuning data selection for balanced learning of diverse capabilities. In *EMNLP (Findings)*, pp. 7079–7102, 2025.

Junwei Deng, Weijing Tang, and Jiaqi W. Ma. A versatile influence function for data attribution with non-decomposable loss. *arXiv preprint arXiv:2412.01335*, 2024.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. In *International Conference on Machine Learning*, 2024.

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*, 2018.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Jaeseung Heo, Kyeongheung Yun, Seokwon Yoon, MoonJeong Park, Jungseul Ok, and Dongwoo Kim. Influence functions for edge edits in non-convex graph neural networks. *Advances in Neural Information Processing Systems*, 2025.

Yuzheng Hu, Pingbang Hu, Han Zhao, et al. Most influential subset selection: Challenges, promises, and beyond. *Advances in Neural Information Processing Systems*, 2024.

Jenny Y Huang, David R Burt, Yunyi Shen, Tin D Nguyen, and Tamara Broderick. Approximations to worst-case data dropping: unmasking failure modes. *Transactions on Machine Learning Research*, 2025.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. Large-Scale Data Selection for Instruction Tuning. *arXiv preprint arXiv:2503.01807*, 2025.

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

Pang Wei Koh, Kai-Siang Ang, Hubert Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 2019.

Philipp Alexander Kreer, Wilson Wu, Maxwell Adam, Zach Furman, and Jesse Hoogland. Bayesian influence functions for hessian-free data attribution. In *The Fourteenth International Conference on Learning Representations*, 2026.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *International Conference on Learning Representations*, 2025.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 2020.

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

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, 2020.

Bruno Kacper Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard E. Turner. Influence functions for scalable data attribution in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mahdi Nikdan, Vincent Cohen-Addad, Dan Alistarh, and Vahab Mirrokni. Efficient data selection at scale via influence distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xingyuan Pan, Luyang Huang, Liyan Kang, Zhicheng Liu, Yu Lu, and Shanbo Cheng. G-dig: Towards gradient-based diverse and high-quality instruction data selection for machine translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 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*, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

Ittai Rubinstein and Samuel B. Hopkins. Rescaled influence functions: Accurate data attribution in high dimension. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. *International Conference on Learning Representations*, 2023.

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

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Haoru Tan, Sitong Wu, Wei Huang, Shizhen Zhao, and XIAOJUAN QI. Data pruning by information maximization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

Athanasios Tsanas, Max Little, Patrick McSharry, and Lorraine Ramig. Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. *Nature Precedings*, 2009.

Andrew Wang, Elisa Nguyen, Runshi Yang, Juhan Bae, Sheila A McIlraith, and Roger Grosse. Better training data attribution via better inverse hessian-vector products. *Advances in Neural Information Processing Systems*, 2025a.

Jiachen T Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024a.

Jiachen T Wang, Tianji Yang, James Zou, Yongchan Kwon, and Ruoxi Jia. Rethinking data shapley for data selection tasks: Misleads and merits. *International Conference on Machine Learning*, 2024b.

Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *International Conference on Learning Representations*, 2025b.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 2023.

Jingyu Wei, Bo Liu, Tianjiao Wan, Baoyun Peng, Xingkong Ma, and Mengmeng Guo. Ji2s: Joint influence-aware instruction data selection for efficient fine-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

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 Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yu Yang, Siddhartha Mishra, Jeffrey N Chiang, and Baharan Mirzasoleiman. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Xichen Ye, Yifan Wu, Weizhong Zhang, Cheng Jin, and Yifan Chen. Towards robust influence functions with flat validation minima. *arXiv preprint arXiv:2505.19097*, 2025.

I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 1998.

Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models. *Advances in Neural Information Processing Systems*, 2024.

Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Scott Yih, and Chenyan Xiong. Group-level data selection for efficient pretraining. *Advances in Neural Information Processing Systems*, 38:154744–154770, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ju Fan, et al. Harnessing diversity for important data selection in pretraining large language models. *International Conference on Learning Representations*, 2025.

A NOTATION

Table 2 consolidates the notation used throughout the paper. The symbols are grouped into five categories: data and model objects, group-level effects on the target function, parameter shifts and curvature matrices, the pairwise interaction term, and quantities specific to data selection. Within each category, definitions are listed in the order they first appear in the main text, and we use the same symbols in the appendix derivations.

Table 2: Summary of notation used throughout the main text and appendices, organized by category.

Symbol	Description
Data and model	
$\mathcal{D} = \{z_i\}_{i=1}^N$	Training dataset of N examples
$z_i = (x_i, y_i)$	i-th training example with input x_i and label y_i
S	Group of examples removed or selected
$\theta \in \mathbb{R}^p$	Model parameters
$\hat{\theta}$	Parameters obtained by training on $\mathcal{D}$
$\hat{\theta}_{\mathcal{D}\setminus S}$	Parameters obtained by retraining on $\mathcal{D} \setminus S$
$\hat{\theta}_{\mathcal{D}\cup S}$	Parameters obtained by retraining on $\mathcal{D} \cup S$
$\hat{\theta}(\epsilon;\, S)$	Parameters when examples in S are upweighted by ϵ
$\ell(z, \theta)$	Per-example loss
$R_A(\theta) \coloneqq \frac{1}{\lvert A\rvert} \sum_{z_i \in A} \ell(z_i, \theta)$	Empirical risk on A
$f(\theta)$	Target function (e.g., loss on a held-out example)
Group-level effects	
$\mathcal{I}^-(S)$	Removal effect: $f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta})$
$\mathcal{I}^+(S)$	Inclusion effect: $f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta})$
$\hat{\mathcal{I}}^-_{\mathrm{lin}}(S)$	First-order estimate of $\mathcal{I}^-(S)$
$\hat{\mathcal{I}}^+_{\mathrm{lin}}(S)$	First-order estimate of $\mathcal{I}^+(S)$
$\hat{\mathcal{I}}^-(S)$	Our interaction-aware estimator of $\mathcal{I}^-(S)$
$\hat{\mathcal{I}}^+(S)$	Our interaction-aware estimator of $\mathcal{I}^+(S)$
Parameter shifts and curvature	
$g_i \coloneqq \nabla_\theta \ell(z_i, \hat{\theta})$	Per-example gradient at $\hat{\theta}$
H	Hessian of the training objective at $\hat{\theta}$
G	Gauss–Newton Hessian of the training loss at $\hat{\theta}$
$H_f \coloneqq \nabla^2_\theta f(\hat{\theta})$	Target-function Hessian at $\hat{\theta}$
$M \coloneqq H^{-1} H_f H^{-1}$	Bilinear form arising in Proposition 1
$u_i \coloneqq H^{-1} g_i$	Preconditioned per-example gradient
$u_S \coloneqq \sum_{z_i \in S} u_i$	Sum of preconditioned gradients over S
Pairwise interaction term	
$\kappa(z_i,\, z_j) \coloneqq u_i^\top H_f u_j$	Pairwise interaction between z_i and z_j
Data selection	
$\mathcal{D}_{\mathrm{pool}}$	Candidate pool of examples
K	Selection budget (target subset size)
S_t	Selected subset after t greedy iterations
$m(z_j \mid S)$	Marginal score of adding z_j to current subset S
$P \coloneqq \lvert\mathcal{D}_{\mathrm{pool}}\rvert$	Candidate-pool size
d	Total projected dimension used in selection

B Derivations of Influence Estimators and Selection Scores

B.1 Derivation of the first-order influence function

We derive Equation (4) and Equation (5), the first-order Taylor approximations of (i) the parameter shift induced by removing S and (ii) the resulting change in the target function f. Throughout, we maintain the assumptions stated in Section 2: $\ell(z_i, \cdot)$ is twice continuously differentiable for every $z_i \in \mathcal{D}$, and the training-loss Hessian

$$H := \frac{1}{N} \sum_{z_i \in \mathcal{D}} \nabla_\theta^2 \ell(z_i, \hat{\theta})$$

is nonsingular at $\hat{\theta}$, and the target function f is differentiable at $\hat{\theta}$.

Proof strategy. Our goal is to approximate the target change $f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta})$ caused by removing S, which we obtain in two first-order Taylor-expansion steps. The first step approximates the parameter shift $\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}$: as noted in Section 2, this parameter shift corresponds to a finite perturbation from $\epsilon = 0$ to $\epsilon = -1/N$ along the reweighting path $\hat{\theta}(\epsilon; S)$, and we approximate it by a first-order Taylor expansion of $\hat{\theta}(\epsilon; S)$ in ϵ, yielding Equation (4). The second step substitutes the resulting parameter shift into a first-order Taylor expansion of f around $\hat{\theta}$, yielding Equation (5). The only quantity we need to compute for the first step is the slope $d\hat{\theta}(\epsilon; S)/d\epsilon$ at $\epsilon = 0$. To compute it, we use the fact that $\hat{\theta}(\epsilon; S)$ minimizes the perturbed objective for every ϵ, so the gradient of the perturbed objective at $\hat{\theta}(\epsilon; S)$ remains zero as ϵ varies. Differentiating this zero-gradient identity in ϵ yields a closed form for the slope.

Setup. Recall the perturbed objective from Equation (3),

$$\mathcal{R}(\theta; \epsilon, S) := \frac{1}{N} \sum_{z_i \in \mathcal{D}} \ell(z_i, \theta) + \epsilon \sum_{z_i \in S} \ell(z_i, \theta), \tag{13}$$

and its minimizer $\hat{\theta}(\epsilon; S) := \arg\min_\theta \mathcal{R}(\theta; \epsilon, S)$. At $\epsilon = 0$ the perturbation vanishes and we recover $\hat{\theta}(0; S) = \hat{\theta}$. At $\epsilon = -1/N$, substituting into Equation (13) cancels the loss terms for $z_i \in S$:

$$\mathcal{R}\big(\theta; -\tfrac{1}{N}, S\big) = \frac{1}{N} \sum_{z_i \in \mathcal{D}} \ell(z_i, \theta) - \frac{1}{N} \sum_{z_i \in S} \ell(z_i, \theta) = \frac{1}{N} \sum_{z_i \in \mathcal{D}\setminus S} \ell(z_i, \theta).$$

This differs from the empirical risk $\mathcal{R}_{\mathcal{D}\setminus S}(\theta) = \frac{1}{|\mathcal{D}\setminus S|} \sum_{z_i \in \mathcal{D}\setminus S} \ell(z_i, \theta)$ defined in Section 2 only by a positive constant factor, so its minimizer coincides with the leave-group-out parameter $\hat{\theta}_{\mathcal{D}\setminus S}$.

Zero-gradient identity at the minimizer. Since $\hat{\theta}(\epsilon; S)$ is a minimizer of $\mathcal{R}(\cdot; \epsilon, S)$, the gradient of $\mathcal{R}$ in θ is zero at $\hat{\theta}(\epsilon; S)$. Concretely, computing $\nabla_\theta \mathcal{R}(\theta; \epsilon, S)$ from Equation (13) and evaluating at $\theta = \hat{\theta}(\epsilon; S)$ gives

$$\frac{1}{N} \sum_{z_i \in \mathcal{D}} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) + \epsilon \sum_{z_i \in S} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) = 0. \tag{14}$$

This identity holds for every ϵ in a neighborhood of 0. By the implicit function theorem, the nonsingularity of H further ensures that the minimizer path $\hat{\theta}(\epsilon; S)$ is differentiable in ϵ near $\epsilon = 0$, so we may differentiate both sides of the identity in ϵ to extract the desired slope $d\hat{\theta}(\epsilon; S)/d\epsilon|_{\epsilon=0}$.

Differentiating in ϵ. Each per-example gradient $\nabla_\theta \ell(z_i, \hat{\theta}(\epsilon; S))$ depends on ϵ only through $\hat{\theta}(\epsilon; S)$. Differentiating Equation (14) in ϵ and applying the chain rule to each such term gives

$$\underbrace{\left[\frac{1}{N} \sum_{z_i \in \mathcal{D}} \nabla_\theta^2 \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) + \epsilon \sum_{z_i \in S} \nabla_\theta^2 \ell\big(z_i, \hat{\theta}(\epsilon; S)\big)\right]}_{\text{Hessian of } \mathcal{R}(\cdot; \epsilon, S)} \frac{d\hat{\theta}(\epsilon; S)}{d\epsilon} + \sum_{z_i \in S} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) = 0. \tag{15}$$

At $\epsilon = 0$, the second summand inside the bracket is zero due to its ϵ prefactor, and the bracket reduces to H. Using $g_i := \nabla_\theta \ell(z_i, \hat{\theta})$ as defined in Section 2, Equation (15) becomes

$$H \cdot \frac{d\hat{\theta}(\epsilon; S)}{d\epsilon}\bigg|_{\epsilon=0} + \sum_{z_i \in S} g_i = 0. \tag{16}$$

Since H is invertible, we solve for the slope:

$$\frac{d\hat{\theta}(\epsilon; S)}{d\epsilon}\bigg|_{\epsilon=0} = -H^{-1} \sum_{z_i \in S} g_i. \tag{17}$$

Parameter-shift approximation (Equation (4)). We now use the slope in Equation (17) inside the first Taylor expansion promised in the strategy. Expanding $\hat{\theta}(\epsilon; S)$ to first order around $\epsilon = 0$ gives

$$\hat{\theta}(\epsilon; S) \approx \hat{\theta} + \epsilon \cdot \frac{d\hat{\theta}(\epsilon; S)}{d\epsilon}\bigg|_{\epsilon=0} = \hat{\theta} - \epsilon\, H^{-1} \sum_{z_i \in S} g_i. \tag{18}$$

Setting $\epsilon = -1/N$, which corresponds to leaving S out, recovers Equation (4):

$$\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \approx \frac{1}{N} H^{-1} \sum_{z_i \in S} g_i. \tag{19}$$

Target-change approximation via the parameter shift (Equation (5)). For the second step, a first-order Taylor expansion of the target f around $\hat{\theta}$ gives

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) \approx \nabla_\theta f(\hat{\theta})^\top \big(\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}\big). \tag{20}$$

Substituting the parameter shift from Equation (19) yields the standard first-order group influence estimate of Equation (5):

$$\hat{\mathcal{I}}^{-}_{\text{lin}}(S) := \frac{1}{N} \nabla_\theta f(\hat{\theta})^\top H^{-1} \sum_{z_i \in S} g_i. \tag{21}$$

The additivity follows immediately, since the right-hand side depends on S only through the linear sum $\sum_{z_i \in S} g_i$.

B.2 Interaction-aware influence for data removal and addition

Data removal. We derive the interaction-aware influence function in Equation (7) by combining the second-order Taylor expansion of the target function with the first-order parameter shift induced by removing S.

Recall from Equation (4) in Section 2 that the parameter shift induced by removing S admits the first-order approximation

$$\delta_S \;:=\; \hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \;\approx\; \frac{1}{N}\, H^{-1} \sum_{z_i \in S} g_i \;=\; \frac{1}{N}\, u_S, \tag{22}$$

where the last equality uses $u_i := H^{-1} g_i$ and $u_S := \sum_{z_i \in S} u_i$. The second-order Taylor expansion of f around $\hat{\theta}$ in Equation (6) is

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) \;=\; \nabla_\theta f(\hat{\theta})^\top \delta_S \;+\; \tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S \;+\; O(\|\delta_S\|^3). \tag{23}$$

Substituting Equation (22) into the linear term of Equation (23) gives

$$\nabla_\theta f(\hat{\theta})^\top \delta_S \;\approx\; \frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S. \tag{24}$$

Substituting the same approximation into the quadratic term gives

$$\tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S \;\approx\; \tfrac{1}{2} \left(\frac{1}{N}\, u_S\right)^\top H_f \left(\frac{1}{N}\, u_S\right) \;=\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S, \tag{25}$$

where the factor $1/N^2$ arises from the two factors of $1/N$ in δ_S. Combining Equations (24) and (25) and recalling that the leave-group-out effect is $\mathcal{I}^-(S) = f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta})$ yields

$$\mathcal{I}^-(S) \;\approx\; \frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\,u_S^\top H_f\,u_S, \tag{26}$$

which is Equation (7) in the main text. The remainder term $O(\|\delta_S\|^3)$ is dropped, consistent with the second-order approximation. The first term recovers the standard first-order estimate $\hat{\mathcal{I}}^-_{\text{lin}}(S)$ from Equation (5), and the second term is the curvature-induced interaction term that gives rise to non-additivity across examples in S.

Data addition. We now derive the addition-setting influence function in Equation (8). The argument parallels the removal-setting derivation above, with the only change appearing in the sign of the parameter shift induced by adding S rather than removing it.

Consider the reweighted objective in Equation (3) with weight ϵ assigned to the examples in S. Setting $\epsilon = -1/N$ removes the contribution of S, recovering the leave-group-out parameter $\hat{\theta}_{\mathcal{D}\setminus S}$ as a finite step along the reweighting path; setting $\epsilon = +1/N$ instead adds an extra copy of each example in S with weight $1/N$, which corresponds to retraining on the augmented set $\mathcal{D} \cup S$ with each new example carrying the same per-example weight as in $\mathcal{D}$. The first-order parameter shift induced by this addition is therefore

$$\hat{\theta}_{\mathcal{D}\cup S} - \hat{\theta} \;\approx\; -\frac{1}{N}\,H^{-1}\sum_{z_i\in S} g_i \;=\; -\frac{1}{N}\,u_S, \tag{27}$$

which differs from Equation (22) only in sign. A complete derivation through the implicit function theorem is given in Appendix B.1; the sign flip reflects the opposite direction of the reweighting step.

Substituting Equation (27) into the second-order Taylor expansion of f around $\hat{\theta}$ yields

$$\begin{aligned} f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta}) \;&\approx\; \nabla_\theta f(\hat{\theta})^\top\left(-\frac{1}{N}\,u_S\right) \;+\; \tfrac{1}{2}\left(-\frac{1}{N}\,u_S\right)^\top H_f\,\left(-\frac{1}{N}\,u_S\right) \\ &=\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\,u_S^\top H_f\,u_S. \end{aligned} \tag{28}$$

The linear term flips sign because the parameter shift itself is negated, while the quadratic term is invariant under this negation: the two factors of $-1/N$ in the bilinear form combine to a positive $1/N^2$. Recalling that $\mathcal{I}^+(S) = f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta})$ gives Equation (8) in the main text.

This sign asymmetry has a direct consequence for our selection criterion in Section 3.2. When f is to be minimized, a more negative $\mathcal{I}^+(S)$ is preferable, so the linear term rewards candidates whose parameter shifts align with $-\nabla_\theta f(\hat{\theta})$. The quadratic term, in contrast, contributes the same sign in both removal and addition settings, reflecting the fact that interactions among examples in S depend on their joint geometry rather than on the direction of the perturbation.

B.3 Derivation of the marginal score

We derive the marginal score in Equation (12) from the definition $m(z_j \mid S) \coloneqq \hat{\mathcal{I}}^+(S \cup \{z_j\}) - \hat{\mathcal{I}}^+(S)$ in Section 3.2. The derivation amounts to substituting the estimator $\hat{\mathcal{I}}^+(\cdot)$ into the definition and expanding the resulting quadratic form using the linearity of the sum of preconditioned gradients.

Recall from Equation (8) that the addition-setting estimator is

$$\hat{\mathcal{I}}^+(S) \;\coloneqq\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\,u_S^\top H_f\,u_S, \tag{29}$$

and that the sum of preconditioned gradients over a subset T is $u_T \coloneqq \sum_{z_i\in T} u_i$. For $T = S \cup \{z_j\}$ with $z_j \notin S$, the linearity of the sum gives

$$u_{S\cup\{z_j\}} \;=\; u_S + u_j. \tag{30}$$

Substituting Equation (30) into the linear term of Equation (29) evaluated at $S \cup \{z_j\}$ gives

$$-\frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_{S\cup\{z_j\}} \;=\; -\frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_S \;-\; \frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_j. \tag{31}$$

The first term on the right cancels with the corresponding linear term in $\hat{\mathcal{I}}^+(S)$, leaving only the contribution of z_j.

Substituting Equation (30) into the quadratic term and expanding gives

$$\begin{aligned}\frac{1}{2N^2}\, u_{S\cup\{z_j\}}^\top H_f\, u_{S\cup\{z_j\}} \;&=\; \frac{1}{2N^2}\,(u_S+u_j)^\top H_f\,(u_S+u_j)\\ &=\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S \;+\; \frac{1}{N^2}\, u_S^\top H_f\, u_j \;+\; \frac{1}{2N^2}\, u_j^\top H_f\, u_j,\end{aligned} \tag{32}$$

where the cross terms $u_S^\top H_f\, u_j$ and $u_j^\top H_f\, u_S$ combine into a single term with coefficient $1/N^2$ because H_f is symmetric, which gives $u_j^\top H_f\, u_S = u_S^\top H_f\, u_j$. The first term on the right of Equation (32) is the quadratic term in $\hat{\mathcal{I}}^+(S)$ itself and cancels in the difference $\hat{\mathcal{I}}^+(S\cup\{z_j\}) - \hat{\mathcal{I}}^+(S)$.

Combining the surviving contributions from Equations (31) and (32) yields

$$m(z_j \mid S) \;=\; -\frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_j \;+\; \frac{1}{N^2}\, u_S^\top H_f\, u_j \;+\; \frac{1}{2N^2}\, u_j^\top H_f\, u_j, \tag{33}$$

which is exactly Equation (12) in the main text. The three surviving terms admit the interpretation given in Section 3.2: the first term is the standard first-order influence of the candidate z_j, the second term measures the curvature-driven interaction between z_j and the already-selected subset S, and the third term is the candidate's self-contribution to the curvature correction.

This decomposition aligns with the self versus cross distinction developed in Appendix C.1. The cross term $\frac{1}{N^2}\, u_S^\top H_f\, u_j$ in Equation (33) is precisely the contribution of z_j to the cross part of the pairwise decomposition, while the self term $\frac{1}{2N^2}\, u_j^\top H_f\, u_j$ is its contribution to the diagonal part. The greedy procedure in Algorithm 1 therefore exploits the same self versus cross structure to update each candidate's marginal utility based on its interaction with previously selected examples, without recomputing the full quadratic form at each iteration.

C Structure and Interpretation of Interactions

C.1 Pairwise and spectral decompositions

The interaction term $u_S^\top H_f\, u_S$ in Equations (7) and (8) admits a pairwise decomposition used in Section 3.1, which we derive below.

Pairwise decomposition. Recall that $u_S = \sum_{z_i\in S} u_i$ is the sum of preconditioned gradients. Substituting this definition into the bilinear form and expanding gives

$$u_S^\top H_f\, u_S \;=\; \left(\sum_{z_i\in S} u_i\right)^\top H_f \left(\sum_{z_j\in S} u_j\right) \;=\; \sum_{z_i\in S}\sum_{z_j\in S} u_i^\top H_f\, u_j \;=\; \sum_{z_i,z_j\in S} \kappa(z_i,\, z_j), \tag{34}$$

where $\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b$ as defined in Equation (9). This recovers the pairwise form used throughout Section 3.1.

Self and cross contributions. The double sum in Equation (34) ranges over ordered pairs and can be further separated into diagonal and off-diagonal contributions:

$$u_S^\top H_f\, u_S \;=\; \underbrace{\sum_{z_i\in S} \kappa(z_i,\, z_i)}_{\text{self contribution}} \;+\; \underbrace{2 \sum_{\{z_i,z_j\}\subset S,\, i\neq j} \kappa(z_i,\, z_j)}_{\text{cross contribution}}, \tag{35}$$

where the cross contribution sums over unordered pairs $\{z_i, z_j\}$ with $i \neq j$ and the factor of two reflects the symmetry $\kappa(z_i,\, z_j) = \kappa(z_j,\, z_i)$ inherited from $H_f = H_f^\top$. The self contribution

$\sum_i \kappa(z_i, z_i) = \sum_i u_i^\top H_f u_i$ is determined entirely by the individual examples and would persist even if examples in S were processed independently. The cross contribution is the part of the interaction term that genuinely encodes interactions between distinct examples and is the source of non-additivity beyond the additive baseline. In the main text we present the unified double-sum form in Equation (9) so that the analysis remains agnostic to whether self or cross terms dominate. The split into self and cross contributions in Equation (35) becomes useful when isolating the genuinely interactional content, which we exploit in the marginal score derivation in Appendix B.3.

Spectral interpretation of the pairwise interaction. We provide a spectral interpretation of the pairwise interaction $\kappa(a, b)$ that holds in the general setting where the target Hessian H_f may be indefinite, going beyond the inner-product reading of Section 3.1.

Spectral decomposition of the interaction term. Since f is twice continuously differentiable at $\hat{\theta}$, H_f is symmetric and admits the eigendecomposition

$$H_f = \sum_k \mu_k v_k v_k^\top, \tag{36}$$

where $\{\mu_k\}$ are the real eigenvalues and $\{v_k\}$ form an orthonormal basis of unit eigenvectors. Substituting Equation (36) into the definition of the interaction term $\kappa(a, b) = u_a^\top H_f u_b$ yields

$$\kappa(a, b) = \sum_k \mu_k (v_k^\top u_a)(v_k^\top u_b). \tag{37}$$

Each summand isolates the contribution of a single eigendirection v_k. The product $(v_k^\top u_a)(v_k^\top u_b)$ measures how the preconditioned gradients of a and b are aligned along v_k. The eigenvalue μ_k then specifies how strongly this alignment contributes to $\kappa(a, b)$, as well as its sign. The spectral view thus provides additional information beyond the bilinear-form definition, allowing $\kappa(a, b)$ to be read as a sum of independent contributions, one per eigendirection of H_f.

Reading the decomposition as a weighted similarity score. A useful way to read Equation (37) is as an aggregate similarity score. The preconditioned gradients u_a and u_b are compared from multiple perspectives, one per eigendirection v_k. These per-direction comparisons are then combined through a weighted sum, with the eigenvalues μ_k acting as importance weights. When H_f is positive definite all weights μ_k are positive, so alignment along any eigendirection contributes consistently in the positive direction to the aggregate score. This recovers the inner-product reading of Section 3.1 in spectral form. The additional insight is that the weight assigned to each perspective is precisely the curvature of f along that direction.

Extension to the indefinite case. When H_f is instead indefinite, $\kappa(a, b)$ is no longer an inner product but a symmetric bilinear form. The eigendecomposition in Equation (36) remains valid, so the per-direction decomposition of $\kappa(a, b)$ in Equation (37) still applies. The only change is that some eigenvalues μ_k may now be negative. Concretely, suppose u_a and u_b are aligned along an eigendirection v_k, that is, $(v_k^\top u_a)(v_k^\top u_b) > 0$:

- if $\mu_k > 0$ (a locally convex direction), the per-direction contribution is positive, so this aligned perspective increases the second-order correction and is harmful under the loss-minimization convention;
- if $\mu_k < 0$ (a locally concave direction), the per-direction contribution is negative, so this aligned perspective decreases the second-order correction and is beneficial under the same convention.

The same logic applies with reversed signs when the two preconditioned gradients are anti-aligned along v_k. In that case, anti-alignment along a positive-curvature direction contributes negatively, while anti-alignment along a negative-curvature direction contributes positively. The interaction term thus evaluates pairwise behavior in a curvature-aware manner. The alignment between two examples can either drive f toward worse values or pull it toward better ones, depending on the sign of the curvature along that direction.

C.2 Proof of Proposition 1

We prove the closed-form factorization of $\kappa(a,\, b)$ stated in Proposition 1. We work in the binary logistic regression setting with ℓ_2 regularization of strength $\beta > 0$, and let $\hat{\theta}$ denote the regularized empirical risk minimizer. The proof proceeds in three steps: deriving the closed-form per-example gradient and the corresponding Hessian of the regularized objective, expressing the preconditioned per-example gradient in factored form, and substituting into the definition of $\kappa(a,\, b)$.

Step 1: per-example gradient and training-loss Hessian. For binary logistic regression, the per-example data loss takes the form

$$\ell(z_i, \theta) \;=\; -\, y_i \log \sigma(\theta^\top x_i) \;-\; (1 - y_i) \log\big(1 - \sigma(\theta^\top x_i)\big)\,, \tag{38}$$

where $y_i \in \{0, 1\}$ is the binary label. Using the standard identity $\sigma'(t) = \sigma(t)(1-\sigma(t))$ and writing $\sigma_i \coloneqq \sigma(\hat{\theta}^\top x_i)$, the per-example gradient evaluated at $\hat{\theta}$ is

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; (\sigma_i - y_i)\, x_i. \tag{39}$$

The training objective is the regularized empirical risk

$$\mathcal{R}(\theta) \;=\; \frac{1}{N} \sum_i \ell(z_i, \theta) \;+\; \frac{\beta}{2} \|\theta\|^2. \tag{40}$$

Differentiating once gives $\nabla_\theta \mathcal{R}(\theta) = \frac{1}{N} \sum_i \nabla_\theta \ell(z_i, \theta) + \beta\, \theta$. Differentiating once more yields the training-loss Hessian

$$H \;\coloneqq\; \nabla^2_\theta \mathcal{R}(\hat{\theta}) \;=\; \frac{1}{N} \sum_i \nabla^2_\theta \ell(z_i, \hat{\theta}) \;+\; \beta\, I \;=\; \frac{1}{N} \sum_i \sigma_i (1 - \sigma_i)\, x_i x_i^\top \;+\; \beta\, I, \tag{41}$$

where the last equality uses the closed-form per-example Hessian of the binary logistic loss. The data-dependent part $\frac{1}{N} \sum_i \sigma_i (1 - \sigma_i)\, x_i x_i^\top$ is positive semidefinite as a sum of rank-one outer products with non-negative weights, and the regularizer contributes βI with $\beta > 0$. Their sum H is therefore positive definite and hence invertible. We use Equation (39) as the per-example gradient driving the influence approximation throughout the remainder of the proof.

Step 2: preconditioned per-example gradient. Substituting the closed-form gradient into the definition $u_i \coloneqq H^{-1} \nabla_\theta \ell(z_i, \hat{\theta})$ gives

$$u_i \;=\; H^{-1}[(\sigma_i - y_i) x_i] \;=\; (\sigma_i - y_i)\, H^{-1} x_i, \tag{42}$$

where the second equality uses the linearity of matrix-vector multiplication and the fact that $(\sigma_i - y_i)$ is a scalar. Each preconditioned per-example gradient therefore factors into a scalar prediction residual $(\sigma_i - y_i)$ and a vector $H^{-1} x_i$ that depends only on the input feature x_i through the inverse training-loss Hessian.

Step 3: factorization of $\kappa(a,\, b)$. Substituting Equation (42) into the definition $\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b$ from Equation (9) gives

$$\begin{aligned} \kappa(a,\, b) \;&=\; \big[(\sigma_a - y_a)\, H^{-1} x_a\big]^\top H_f \,\big[(\sigma_b - y_b)\, H^{-1} x_b\big] \\ &=\; (\sigma_a - y_a)(\sigma_b - y_b)\, x_a^\top H^{-1} H_f\, H^{-1} x_b \\ &=\; (\sigma_a - y_a)(\sigma_b - y_b)\, \langle x_a, x_b \rangle_M, \end{aligned} \tag{43}$$

where the second equality factors the two scalar residuals out of the bilinear form and uses the symmetry of H^{-1}, which holds because H is symmetric and positive definite, and the third equality applies the definition $M \coloneqq H^{-1} H_f\, H^{-1}$ together with $\langle u, v \rangle_M \coloneqq u^\top M\, v$. This is exactly the factorization claimed in Proposition 1, completing the proof.

C.3 Extension of Proposition 1 to deep classifiers

Proposition 1 establishes a closed-form factorization of $\kappa(a, b)$ for binary logistic regression. We now extend this result to deep classifiers, treating binary and multi-class classification in turn. The argument is in both cases a direct consequence of the chain rule applied to the per-example gradient: the same scalar-residual–input-feature factorization that drives the proof of Proposition 1 survives when the input feature x_i is replaced by the logit Jacobian

$$J_i \coloneqq \nabla_\theta h_\theta(x_i)|_{\theta=\hat{\theta}}\,, \tag{44}$$

where $h_\theta : \mathcal{X} \to \mathbb{R}^C$ denotes the network logit function, with $C = 1$ for binary classification and $C \geq 2$ for multi-class classification. The bilinear-form matrix $M \coloneqq H^{-1} H_f H^{-1}$ from Proposition 1 is unchanged. We treat the binary case first as it offers the cleanest correspondence with Proposition 1, then state and prove the multi-class generalization.

Binary case. For binary classification ($C = 1$), $J_i \in \mathbb{R}^p$ is a vector. Writing $\sigma_i \coloneqq \sigma(h_{\hat{\theta}}(x_i))$, the per-example loss $\ell(z_i, \theta) = -y_i \log \sigma(h_\theta(x_i)) - (1 - y_i) \log(1 - \sigma(h_\theta(x_i)))$ has gradient at $\hat{\theta}$ given by the chain rule as

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; (\sigma_i - y_i)\, J_i, \tag{45}$$

which differs from Equation (39) only in that the input feature x_i is replaced by the logit Jacobian J_i. Steps 2 and 3 of the proof of Proposition 1 therefore carry over verbatim with $x_i \mapsto J_i$, yielding

$$\kappa(a, b) \;=\; (\sigma_a - y_a)(\sigma_b - y_b)\, \langle J_a, J_b \rangle_M. \tag{46}$$

The class-agreement scalar is unchanged because it depends only on the network's predictions and the labels. The input-similarity term $\langle x_a, x_b \rangle_M$ generalizes to a Jacobian bilinear form $\langle J_a, J_b \rangle_M$, which is an M-weighted empirical neural tangent kernel evaluated at $\hat{\theta}$ when H_f is positive semidefinite.

Decomposing the Jacobian bilinear form via the last layer. Writing $\theta = (\psi, w)$ for the feature-extractor parameters and the last-layer weight, and decomposing $h_\theta(x) = w^\top \phi_\psi(x)$ where $\phi_\psi : \mathcal{X} \to \mathbb{R}^{d_\phi}$ denotes the learned feature representation, the logit Jacobian J_i admits the block decomposition

$$J_i = \begin{bmatrix} J_\phi(x_i)^\top w \\ \phi(x_i) \end{bmatrix}, \qquad J_\phi(x_i) \coloneqq \frac{\partial \phi_\psi(x_i)}{\partial \psi}, \tag{47}$$

corresponding to gradients with respect to ψ (top block) and w (bottom block). Substituting into the Jacobian bilinear form exposes both the learned feature representation and the feature Jacobian directly:

$$\langle J_a, J_b \rangle_M \;=\; \begin{bmatrix} J_\phi(x_a)^\top w \\ \phi(x_a) \end{bmatrix}^\top M \begin{bmatrix} J_\phi(x_b)^\top w \\ \phi(x_b) \end{bmatrix}. \tag{48}$$

In Equation (48), the bottom-block component $\phi(x_i)$ takes the place of the raw input feature x_i in Proposition 1. Since the network's representation is well known to map visually similar examples to similar $\phi(x_i)$, such examples contribute to $\kappa(a, b)$ through this block in the same way that input similarity contributes in the LR setting. The top-block component $J_\phi(x_i)^\top w$ captures how the feature representation responds to feature-extractor parameter perturbations along the readout direction w, an additional structure unique to deep classifiers that has no counterpart in the LR setting.

Multi-class case. For multi-class classification with $C \geq 2$, $J_i \in \mathbb{R}^{p \times C}$ is a matrix whose c-th column is the gradient of the c-th logit with respect to all parameters. Write $p_i \in \Delta^{C-1}$ for the softmax of $h_{\hat{\theta}}(x_i)$ and $y_i \in \{0, 1\}^C$ for the one-hot label. The cross-entropy loss $\ell(z_i, \theta) = -\sum_{c=1}^{C} y_{i,c} \log p_{i,c}(\theta)$ has gradient at $\hat{\theta}$ given by the chain rule as

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; J_i\, r_i, \qquad r_i \;\coloneqq\; p_i - y_i \;\in\; \mathbb{R}^C. \tag{49}$$

Equation (49) generalizes Equation (39) in two parallel ways: the scalar residual $(\sigma_i - y_i)$ becomes the vector residual r_i, and the input feature x_i becomes the matrix Jacobian J_i.

Substituting Equation (49) into the definition $u_i \coloneqq H^{-1}\,\nabla_\theta \ell(z_i, \hat{\theta})$ gives $u_i = H^{-1} J_i\, r_i$. Plugging this into the pairwise interaction $\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b$ from Equation (9) and factoring yields

$$\begin{aligned}\kappa(a,\, b) &= (H^{-1} J_a\, r_a)^\top H_f\, (H^{-1} J_b\, r_b) \\ &= r_a^\top\, J_a^\top H^{-1} H_f\, H^{-1}\, J_b\, r_b \\ &= r_a^\top\, K_M(x_a, x_b)\, r_b,\end{aligned} \tag{50}$$

where the second equality uses the symmetry of H^{-1}, and the third equality defines the matrix-valued bilinear form

$$K_M(x_a, x_b) \;\coloneqq\; J_a^\top M\, J_b \;\in\; \mathbb{R}^{C\times C}, \tag{51}$$

which is a matrix-valued empirical neural tangent kernel when H_f is positive semidefinite. The factorization in Equation (50) is the multi-class generalization of Equation (43): the scalar residual product $(\sigma_a - y_a)(\sigma_b - y_b)$ is replaced by the bilinear form $r_a^\top K_M\, r_b$, and the scalar feature similarity $\langle x_a, x_b\rangle_M$ is replaced by the matrix-valued bilinear form $K_M(x_a, x_b)$. The binary case in Equation (46) is recovered when $C = 1$, with $r_i = \sigma_i - y_i$ and $K_M = \langle J_a, J_b\rangle_M$ a scalar.

Alignment of transformed residuals. The factorization in Equation (50) gives a geometric interpretation of interactions through prediction residuals and logit Jacobians. When H_f is positive semidefinite, $M = H^{-1} H_f H^{-1}$ is also positive semidefinite. Defining the transformed residual representation $\widetilde{r}_i \coloneqq M^{1/2} J_i r_i \in \mathbb{R}^p$ gives

$$\kappa(a,\, b) = \langle \widetilde{r}_a, \widetilde{r}_b\rangle = \left\langle M^{1/2} J_a r_a,\, M^{1/2} J_b r_b \right\rangle. \tag{52}$$

Thus, the interaction measures alignment between residuals transformed by the logit Jacobians and the target-induced curvature. Positive alignment increases the estimated joint target change relative to the sum of the singleton estimates, whereas negative alignment decreases it. Self-interactions satisfy $\kappa(i,\, i) = \|\widetilde{r}_i\|_2^2$. The sign of a cross-example interaction depends on this transformed alignment and is not determined by class agreement alone. The class-pair patterns in Figure 2 provide empirical evidence of a relationship between label agreement and interaction sign in the evaluated models. When H_f is indefinite, the bilinear factorization remains valid, with the spectral interpretation given in Appendix C.1.

D Relation to the Basu Correction

We examine the relationship between our target-curvature correction and the parameter-shift correction of Basu et al. (2020), and investigate why adding the latter can reduce attribution accuracy in our experiments.

D.1 Full second-order expansion and relation to Basu

We follow the reweighting path in Equation (3). For this derivation, assume that the training objective $\mathcal{R}_\mathcal{D}$ and the per-example losses are three times continuously differentiable near $\hat{\theta}$, that f is twice continuously differentiable, and that H is nonsingular at the stationary point $\hat{\theta}$. Here, $\mathcal{R}_\mathcal{D}$ includes regularization when applicable. These assumptions define a twice differentiable local stationary branch through $\hat{\theta}$. Write

$$\dot{\theta}_S \coloneqq \left.\frac{d\,\hat{\theta}(\epsilon;\, S)}{d\epsilon}\right|_{\epsilon=0}, \qquad \ddot{\theta}_S \coloneqq \left.\frac{d^2\,\hat{\theta}(\epsilon;\, S)}{d\epsilon^2}\right|_{\epsilon=0}, \qquad H_S \coloneqq \sum_{z_i\in S} \nabla_\theta^2 \ell(z_i, \hat{\theta}). \tag{53}$$

Differentiating the stationarity condition once and twice gives

$$H\dot{\theta}_S + \sum_{z_i\in S} g_i = 0, \tag{54}$$

$$H\ddot{\theta}_S + 2H_S\dot{\theta}_S + \nabla_\theta^3 \mathcal{R}_\mathcal{D}(\hat{\theta})[\dot{\theta}_S, \dot{\theta}_S] = 0. \tag{55}$$

Consequently,

$$\begin{aligned}\dot{\theta}_S &= -u_S, \\ \ddot{\theta}_S &= 2H^{-1} H_S u_S - H^{-1}\nabla_\theta^3 \mathcal{R}_\mathcal{D}(\hat{\theta})[u_S, u_S],\end{aligned} \tag{56}$$

Table 3: Relative L_2 error of predicted parameter shifts against leave-group-out retraining. The columns compare the first-order prediction $\delta_{1,S}$ and the Basu-corrected prediction $\delta_{1,S} + \delta_{B,S}$. Lower is better.

Dataset / model	First order	With Basu correction
MNIST / LR	0.83	0.78
MNIST / MLP	4.4	145
FashionMNIST / MLP	7.6	729
Concrete / MLP	20.6	14,204
Parkinsons / MLP	7.6	1,004
CIFAR-10 / ResNet-9	14.6	15,960

where the third-derivative tensor is contracted with the two indicated vectors, leaving a vector in parameter space.

The corresponding expansion of the target is

$$f(\hat{\theta}(\epsilon;\, S)) - f(\hat{\theta}) = -\epsilon\, \nabla_\theta f(\hat{\theta})^\top u_S + \frac{\epsilon^2}{2}\left(u_S^\top H_f u_S + \nabla_\theta f(\hat{\theta})^\top \ddot{\theta}_S\right) + o(\epsilon^2). \tag{57}$$

Setting $\epsilon = -1/N$ in this quadratic approximation yields the first-order influence, our target-curvature correction $u_S^\top H_f u_S/(2N^2)$, and an additional parameter-shift contribution $\nabla_\theta f(\hat{\theta})^\top \ddot{\theta}_S/(2N^2)$. Both corrections contribute at second order in ϵ. Our estimator retains the target-curvature contribution while using the first-order parameter shift; it does not retain the full second-order expansion of the retraining effect.

Relation to Basu et al. (2020). Basu et al. improve the approximation to the parameter shift while keeping the target approximation linear. Their derivation omits the contraction involving third derivatives of the training objective, assuming that this contribution is sufficiently small. Although it involves third derivatives with respect to the model parameters, this contribution belongs to the second-order expansion in the reweighting parameter ϵ. Basu et al. also use a reweighting convention that preserves total sample weight by adjusting the weights of the remaining examples. Their normalization coefficients therefore differ from those for Equation (3), and their practical correction should not be identified with the exact $\ddot{\theta}_S$ in Equation (56).

D.2 Parameter-shift accuracy and implications for attribution

To examine the correction directly, we compare predicted parameter shifts with those measured by leave-group-out retraining in the six dataset–model settings of Section 4. Let $\delta_S = \hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}$ denote the measured shift, $\delta_{1,S}$ the first-order prediction, and $\delta_{B,S}$ the additional shift supplied by the Basu correction used in our experiments. The two predictions are $\delta_{1,S}$ and $\delta_{1,S} + \delta_{B,S}$. Their relative errors are measured as

$$\mathrm{err}_S(\widehat{\delta}) := \frac{\|\widehat{\delta} - \delta_S\|_2}{\|\delta_S\|_2}. \tag{58}$$

The first-order prediction is u_S/N under exact curvature; the experiments use the corresponding curvature approximations described in Appendix F.1.

Table 3 shows that the correction slightly improves the parameter-shift estimate for logistic regression, reducing the relative error from 0.83 to 0.78. In contrast, it substantially increases the error in all five nonlinear settings. Thus, under the experimental conditions considered here, the approximate second-order correction makes the predicted parameter shift less accurate for these nonlinear models.

A possible explanation. The two contributions to $\ddot{\theta}_S$ in Equation (56) account for different changes in training curvature. The term involving H_S captures the direct effect of changing the example weights, whereas the third-derivative contraction captures how training curvature changes

as the parameters move. If the omitted contribution is substantial, retaining only part of the correction can misestimate its magnitude or direction. In particular, the retained and omitted contributions may partially cancel, so adding only the retained contribution can produce a worse estimate than keeping the first-order shift. Our method also omits this third-derivative contribution, but does so by retaining the first-order parameter shift throughout, rather than adding a partial second-order correction to it. This is a possible explanation of the observed deterioration; the experiment does not isolate the effect of the omitted contribution from curvature approximation and finite-retraining effects.

Implications for attribution. In the combined estimator evaluated in Figure 1, the target-curvature term is evaluated at the corrected shift $\delta_{1,S} + \delta_{B,S}$. Relative to evaluating it at $\delta_{1,S}$, this changes the quadratic term by

$$\begin{aligned} &\frac{1}{2}(\delta_{1,S} + \delta_{B,S})^\top H_f(\delta_{1,S} + \delta_{B,S}) - \frac{1}{2}\delta_{1,S}^\top H_f \delta_{1,S} \\ &\qquad = \delta_{1,S}^\top H_f \delta_{B,S} + \frac{1}{2}\delta_{B,S}^\top H_f \delta_{B,S}. \end{aligned} \tag{59}$$

An inaccurate correction can therefore affect the target estimate through both a cross term and a quadratic term. Even with an exact second-order parameter-shift correction, substituting the corrected shift into the quadratic target approximation introduces terms beyond a strict second-order expansion in ϵ.

The larger parameter-shift errors are consistent with the attribution degradation observed when the two corrections are combined in Figure 1. However, parameter-space error alone does not determine attribution rankings: its effect also depends on the target gradient and curvature. These results support retaining the first-order shift in the evaluated settings, but do not establish that an exact full second-order expansion would perform worse than our estimator.

E Scalable Curvature and Gradient Approximations

The estimators $\hat{\mathcal{I}}^-(S)$ and $\hat{\mathcal{I}}^+(S)$ defined in Equation (7) and Equation (8) require applying H^{-1} to per-example gradients. Since H has p^2 entries in p parameters and is moreover not guaranteed to be positive definite away from a minimum, this is intractable for the models considered in our experiments. We therefore approximate H by the damped Gauss–Newton matrix $G + \lambda I$, where G is the Gauss–Newton Hessian of the training loss at $\hat{\theta}$ and $\lambda > 0$ is a damping constant. This appendix collects the three approximations that make our estimator tractable at scale: the Gauss–Newton approximation itself, its block-diagonal EK-FAC factorization, and a low-rank gradient projection for large language models. The damping constant λ is treated as a hyperparameter and its selection is described in Appendix F.

E.1 Damped Gauss–Newton approximation

Following standard practice in influence-function analysis (Bae et al., 2022; Grosse et al., 2023), we approximate the training Hessian by the Gauss–Newton Hessian G of the training loss at $\hat{\theta}$, damped by λI to ensure positive definiteness. The Gauss–Newton Hessian is positive semidefinite by construction and discards the indefinite residual term that arises in the exact Hessian, making it a natural surrogate when the goal is to invert a curvature matrix.

The losses considered in this paper are all negative log-likelihoods of exponential-family distributions: cross-entropy with softmax outputs for classification, squared error with a Gaussian likelihood for regression, and token-level negative log-likelihood for instruction tuning. For such losses, the Gauss–Newton Hessian coincides with the Fisher information matrix taken under the model's output distribution (Martens, 2020). This equivalence is what licenses the use of curvature approximations developed for the Fisher information matrix, and it is the same identification adopted by Grosse et al. (2023) for influence-function analysis of transformer language models. We accordingly state our approximations in terms of G throughout, with the understanding that G may equivalently be read as the model-distribution Fisher information.

E.2 Block-diagonal structure and EK-FAC

Even with the Gauss–Newton substitution, G remains a $p \times p$ matrix and cannot be inverted directly at the parameter counts considered in our experiments. We therefore further approximate G as block-diagonal across layers, treating cross-layer curvature as zero. This reduces the inversion to one block per layer.

Within each block, we apply the Eigenvalue-Corrected Kronecker-Factored Approximate Curvature (EK-FAC) approximation (George et al., 2018; Grosse et al., 2023). EK-FAC builds on K-FAC (Martens & Grosse, 2015), which factorizes the per-layer Gauss–Newton block as a Kronecker product of two small matrices: the second-moment matrix of the layer's input activations and the second-moment matrix of the gradient with respect to the layer's pre-activations. EK-FAC retains the eigenbasis defined by these Kronecker factors but replaces the implied Kronecker-structured eigenvalues with the exact diagonal of G in that eigenbasis, estimated from training data. The resulting approximation is provably at least as accurate as K-FAC under the Frobenius norm (George et al., 2018) while preserving the same favorable scaling: each per-layer block can be inverted, and inverse-vector products evaluated, at a cost dominated by the layer's activation and pre-activation dimensions rather than its full parameter count. In practice, the preconditioned per-example gradients u_i required by Equation (7) and Equation (8) are computed as $(G + \lambda I)^{-1}\nabla_\theta \ell(z_i, \hat{\theta})$ by applying this block-wise EK-FAC inverse to per-example gradients. We refer the reader to George et al. (2018) for the original derivation of EK-FAC and to Grosse et al. (2023) for a detailed treatment of its use in influence-function analysis.

E.3 Low-rank gradient projection for large language models

At the scale of Llama-3.1-8B, storing per-example gradients in the full parameter space is prohibitive. We use LoGra (Choe et al., 2025) to project these gradients into a shared low-dimensional subspace and apply its projected block-diagonal curvature approximation to obtain the preconditioned vectors u_i. The LLM experiments do not use EK-FAC. The projection is applied consistently to the gradients and target-side quantities entering Algorithm 1, so that all inner products in the greedy loop are evaluated in the total projected dimension d rather than the full parameter dimension p.

This projection is what reduces the per-iteration cost of Algorithm 1 to $O(P)$ inner products in dimension d, yielding the overall greedy-loop complexity of $O(KPd)$ stated in Section 3.2. We use the projection rank and hyperparameters reported in Appendix F, and refer the reader to Choe et al. (2025) for the construction of the projection operator and a full treatment of the resulting estimator.

F Experimental Details

F.1 Small-scale Attribution Experiments

Datasets and models. The attribution experiments in Section 4 consider six dataset–model pairs spanning classification and regression: logistic regression (LR) and MLP on MNIST (LeCun et al., 1998), MLP on FashionMNIST (Xiao et al., 2017), MLP on Concrete (Yeh, 1998) and Parkinsons (Tsanas et al., 2009), and ResNet-9 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009). All datasets use their standard train/test splits.

LR uses ℓ_2 regularization with strength 0.01. The MLP has two hidden layers of width 128 and 64 with ReLU activations. ResNet-9 follows the standard DAWNBench architecture (Coleman et al., 2017), trained from scratch.

Training. Training configurations are summarized in Table 4 and follow standard settings for each model.

Group construction. We construct groups with high intra-group similarity to evaluate attribution under conditions in which summing individual scores can misrepresent collective effects (Hu et al., 2024). Each group consists of an anchor sampled uniformly at random from the training set, together with its $|S|-1$ nearest neighbors in softmax output space under L^2 distance. We construct 50 groups

Table 4: Training configurations for small-scale models. The cosine schedule decays the learning rate to zero by the end of training.

Model	Optimizer	Momentum	LR	LR Schedule	Weight Decay	Batch Size	Epochs
LR	SGD	0.0	0.01	constant	0.01	64	200
MLP	SGD	0.0	0.01	constant	0.01	64	200
ResNet-9	SGD	0.9	0.01	cosine	0.01	64	50

per dataset–model pair, with $|S| = 400$ except for Concrete, where $|S| = 100$ because of its smaller training set. For regression tasks, the prediction itself replaces the softmax output.

Ground truth and attribution metric. For each group S, we retrain the model from scratch on $\mathcal{D} \setminus S$ and measure the resulting change in average held-out test loss relative to the model trained on $\mathcal{D}$. This gives the ground-truth group influence $\mathcal{I}^-(S)$. We report the Spearman rank correlation between each method's estimates and these ground-truth effects across the 50 groups. This evaluation is analogous to the linear datamodeling score (LDS) (Park et al., 2023); we use the explicit rank-correlation terminology because our group estimator is not a sum of individual scores.

Influence computation. For LR, the Hessian is computed and inverted exactly. For MLPs and ResNet-9, we use the damped Gauss–Newton approximation of the training Hessian, $G + \lambda I$, and approximate its inverse via EK-FAC. The damping λ is set to 10^{-2} for all influence-based methods. The target Hessian H_f is approximated as block-diagonal, with H_f-vector products computed directly without inversion.

Estimators and baselines. We use *F* for the standard first-order influence term (Koh & Liang, 2017), *B* for the second-order parameter-response correction of Basu et al. (2020), and *I* for our target-curvature correction. Our estimator is *F+I*. The influence-based baselines are *F*, *F+B*, and *F+I+B*; the last combines both corrections. These methods use the same curvature approximation within each dataset–model pair. For TRAK (Park et al., 2023), we use projection dimension 2048 and ridge 0.01. For TracIn (Pruthi et al., 2020), we use 10 evenly spaced checkpoints. For both TRAK and TracIn, group scores are obtained by summing individual attributions.

F.2 Small-Scale Selection Experiments

Datasets and candidate pools. The selection experiments in Section 5.1 use MNIST (LeCun et al., 1998) and FashionMNIST (Xiao et al., 2017). For each dataset, we sample a candidate pool $\mathcal{D}_{\text{pool}}$ of 5,000 examples uniformly from the training set using a fixed random permutation. All selection methods use the same candidate pool.

Reference model and curvature. We use a two-hidden-layer MLP ($784 \to 128 \to 64 \to 10$) with ReLU activations and dropout rate 0.1. The reference parameter $\hat{\theta}$ is obtained by training the MLP on $\mathcal{D}_{\text{pool}}$ for 200 epochs with vanilla SGD, learning rate 10^{-2}, batch size 64, and weight decay 3×10^{-2}. The Hessian inverse H^{-1} is approximated via EK-FAC with the GGN Fisher and damping $\lambda = 5\times10^{-1}$.

Selection and baselines. For each method, we select a subset $S \subseteq \mathcal{D}_{\text{pool}}$ of size $K \in \{500, 1{,}000, \ldots, 5{,}000\}$. Our method (*F+I*) uses Algorithm 1. We compare against the same influence-function variants as in Appendix F.1, namely *F*, *F+B*, and *F+I+B*, together with uniform random selection. Each selected subset is used to train a new MLP from scratch with the same training configuration.

Evaluation and class entropy. We report held-out test loss averaged over five random seeds. To measure the class composition of each selected subset, we also report its entropy

$$\mathcal{H}(S) = -\sum_c p_c(S) \log p_c(S),$$

where $p_c(S)$ is the fraction of selected examples belonging to class c. Higher entropy indicates more uniform class coverage. The MNIST entropy results appear in Figure 3, and the corresponding FashionMNIST results are reported in Appendix G.3.

F.3 LLM Data Selection Experiments

The experiments in Section 5.2 fine-tune Llama-3.1-8B (Grattafiori et al., 2024) on subsets selected from the LESS pool (Xia et al., 2024) of approximately 270K instruction-tuning examples. We select $K = 13{,}534$ examples, corresponding to 5% of the pool, independently for each target task.

Warmup and selection. Following Xia et al. (2024), a warmup model $\hat{\theta}$ is obtained by fine-tuning the base model on a random subset of size 13,534 using LoRA (rank 8, scaling 16, applied to `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`), AdamW with peak learning rate 10^{-4}, cosine schedule with warmup ratio 0.03, effective batch size 128, sequence length 2048, in bf16, for four epochs. At $\hat{\theta}$, we compute preconditioned projected gradients u_j and target-curvature products $H_f u_j$ using LoGra (Choe et al., 2025) with projection rank 16, and apply Algorithm 1 to select $K = 13{,}534$ examples (5% of the pool) independently for each target task.

Selection target. The target function f is the instruction-masked negative log-likelihood, evaluated on the official task validation set when available and on a held-out training split otherwise.

Final fine-tuning. Selected subsets are used to fine-tune the original pretrained Llama-3.1-8B checkpoint (rather than the warmup model) under the same LoRA configuration as warmup: AdamW with peak learning rate 10^{-4}, cosine schedule with warmup ratio 0.03, effective batch size 128, sequence length 2048, and bf16 precision. We train for four epochs on GSM8K, whose loss does not converge within a single epoch, and one epoch on the remaining tasks.

Evaluation. We evaluate on seven downstream tasks covering three capabilities. Mathematical reasoning is evaluated on GSM8K (Cobbe et al., 2021) and AQuA (Ling et al., 2017). Commonsense and science reasoning are evaluated on ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and ECQA (Aggarwal et al., 2021). Reading comprehension is evaluated on SQuAD (Rajpurkar et al., 2016). All tasks use 0-shot prompting with the Tulu chat template (Wang et al., 2023). We report exact match for GSM8K, F1 for SQuAD, and accuracy for the multiple-choice tasks, with the latter scored by length-normalized log-likelihood. Results are averaged over five random seeds.

Baselines. We compare against two influence-based baselines, two representation-similarity baselines, and uniform random selection. All methods share the same candidate pool, selection budget, and final fine-tuning protocol; influence-based methods additionally share the same warmup checkpoint.

Additive IF uses the standard first-order influence function (Koh & Liang, 2017) and shares our warmup and projected-gradient pipeline, but omits the interaction term. It selects the top-K examples according to their first-order scores. LESS (Xia et al., 2024) uses Adam-adjusted gradient cosine similarity with projection dimension 8192 at the same warmup checkpoint, scoring candidates by the similarity between their projected gradients and those of the target task.

The representation-similarity baselines select examples whose embeddings most closely match those of the target task. RDS+ (Ivison et al., 2025) uses position-weighted mean-pooled hidden states from the pretrained checkpoint with cosine similarity. NV-Embed (Lee et al., 2025) uses embeddings from the retrieval-trained model `nvidia/NV-Embed-v2`. The random baseline samples K examples uniformly from the candidate pool.

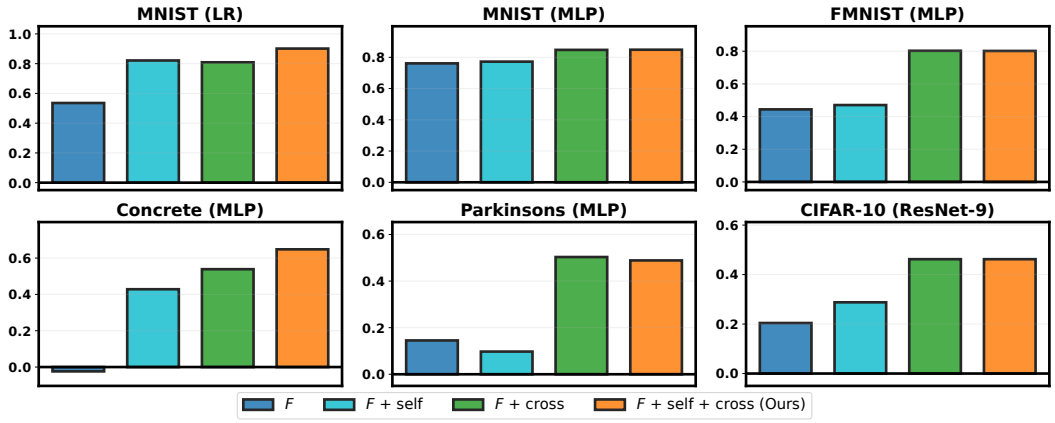

Figure 4: Ablation of self and cross contributions. Bars show Spearman correlation with leave-group-out retraining effects; higher is better. The full estimator (*F+self+cross*) exceeds the additive *F+self* variant in all six settings.

G Additional Experiments

G.1 Self and Cross Contributions

The target-curvature correction contains both self terms, which remain additive across examples, and cross terms, which capture interactions between distinct examples. We ablate these contributions to determine whether the attribution gains can be explained by the additive self terms alone.

Setup. Following the decomposition in Appendix C.1, write

$$\frac{1}{2N^2} u_S^\top H_f u_S = \underbrace{\frac{1}{2N^2} \sum_{z_i \in S} \kappa(z_i, z_i)}_{I_{\text{self}}(S)} + \underbrace{\frac{1}{N^2} \sum_{\substack{z_i, z_j \in S \\ i<j}} \kappa(z_i, z_j)}_{I_{\text{cross}}(S)}. \tag{60}$$

We compare four estimators: first-order influence (*F*), first-order influence with only I_{self} (*F+self*), with only I_{cross} (*F+cross*), and with both contributions (*F+self+cross*, our full estimator *F+I*). Since I_{self} is a sum of per-example scores, *F+self* remains additive. Comparing it with the full estimator therefore isolates the contribution of cross-example terms. We evaluate the six dataset–model settings considered in Section 4, reporting Spearman correlation between estimated group influence and the target-loss change measured by leave-group-out retraining.

Results. Figure 4 shows that the full estimator outperforms *F+self* in all six settings. The improvement therefore cannot be explained by additive self terms alone. The *F+cross* variant also improves over *F* in every setting and performs close to the full estimator on MNIST / MLP, FashionMNIST / MLP, Parkinsons / MLP, and CIFAR-10 / ResNet-9. On MNIST / LR and Concrete / MLP, self terms also provide substantial gains, and combining both contributions performs better than retaining either alone. Self terms do not improve every setting, however: on Parkinsons / MLP, *F+cross* slightly exceeds the full estimator. These results establish a contribution from cross-example interactions while showing that the additional benefit of self terms depends on the setting.

G.2 Curvature approximation and damping sensitivity

Curvature approximations. We examine whether the improvement from the interaction term persists across different iHVP approximations. We use MNIST / LR, where the exact training Hessian is tractable and provides a reference for directly measuring approximation error.

Table 5 compares the exact Hessian with four approximations: block diagonal, EK-FAC, projection only, and LoGra. The block-diagonal approximation separates weight and bias blocks and discards

Table 5: Curvature approximations on MNIST / LR and their components. Relative iHVP error is measured against exact-Hessian iHVPs and averaged over evaluation groups. ρ denotes Spearman correlation with leave-group-out retraining effects.

	Approximation components				Spearman ρ	
Configuration	Block diagonal	EK-FAC	Gradient projection	Relative iHVP error	F	F+I
Exact Hessian	×	×	×	0.00	0.54	0.90
Block diagonal	✓	×	×	0.50	0.52	0.93
EK-FAC	✓	✓	×	0.78	0.26	0.81
Projection only	×	×	✓	1.11	0.58	0.75
LoGra	✓	×	✓	1.22	0.51	0.77

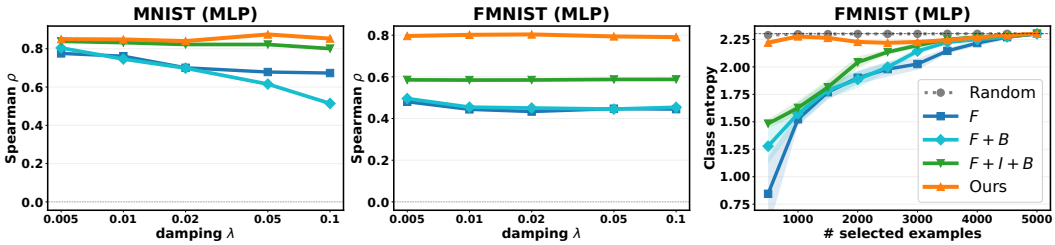

Figure 5: Sensitivity analysis of the two-layer MLP over the damping hyperparameter on MNIST (left) and FashionMNIST (middle), and class entropy on FashionMNIST (right).

their cross-curvature, while LoGra additionally applies gradient projection. For each evaluation group S, we compute the relative iHVP error $\|\widehat{u_S} - u_S\|_2 / \|u_S\|_2$, where $u_S = H^{-1} \sum_{z_i \in S} g_i$ uses the exact Hessian and $\widehat{u_S}$ its approximation. The table reports this error averaged over evaluation groups and the Spearman correlations of F and F+I with leave-group-out retraining effects.

F+I outperforms F in all five configurations, improving the correlation from 0.54 to 0.90 with the exact Hessian and from 0.51 to 0.77 with LoGra. These results show that the benefit of the interaction term persists under both exact and approximate iHVPs, rather than arising from a particular approximation choice. Larger iHVP errors generally accompany lower F+I correlations, but the relationship is not monotone. In particular, the block-diagonal approximation achieves $\rho = 0.93$, slightly exceeding the exact-Hessian result despite a relative iHVP error of 0.50. We conjecture that this improvement reflects a mild implicit-regularization effect from discarding cross-block curvature.

Additional MLP analyses. In this and the following subsection, we present two additional experiments on the two-layer MLP that complement the small-scale attribution accuracy results of Section 4 and the small-scale selection quality results of Section 5.1. The first examines the sensitivity of each method to the damping hyperparameter λ used in the EK-FAC, and the second extends the class entropy analysis from MNIST to FashionMNIST. The results are summarized in Figure 5.

Both experiments inherit the protocols of their counterparts in the main text, modified as follows.

Damping setup. For the damping sensitivity analysis, we follow the attribution setup of Section 4 but vary $\lambda \in \{0.005, 0.01, 0.02, 0.05, 0.1\}$ instead of fixing $\lambda = 10^{-2}$. All other elements of the protocol, including group construction, ground-truth retraining, and the influence-function methods (*F*, *F+B*, *F+I+B*, Ours), are unchanged.

Damping sensitivity. The left and middle panels of Figure 5 report the Spearman correlation as a function of the damping λ on MNIST and FashionMNIST. Ours and F+I+B remain nearly constant across the tested range on both datasets, while F and F+B are somewhat less stable, with F+B showing the most pronounced decline at larger λ on MNIST.

Table 6: Computational cost of the projected selection pipeline. The total projected dimension is d, and each curvature block has dimension b. Shared operations are also used by the matched Additive IF baseline; additional operations are introduced by interaction-aware selection.

Stage	Time complexity	Cost category
Warmup checkpoint $\hat{\theta}$	$O(EKC_{\mathrm{fb}})$	Shared
Pool-gradient extraction $\{g_i\}_{i=1}^{P}$	$O(PC_{\mathrm{fb}})$	Shared
Training-curvature blocks H	$O(Pdb)$	Shared
Block inverses H^{-1}	$O(db^2)$	Shared
Preconditioned gradients $u_i = H^{-1}g_i$	$O(Pdb)$	Shared
Target-gradient extraction	$O(n_f C_{\mathrm{fb}})$	Shared
Target-curvature blocks H_f	$O(n_f db)$	Additional
Curvature products $w_i = H_f u_i$	$O(Pdb)$	Additional
Self-interaction scores $q_i = u_i^\top w_i$	$O(Pd)$	Additional
Greedy selection	$O(KPd)$	Additional

G.3 Selection diversity on FashionMNIST

Setup. For the class entropy analysis, we run the selection procedure of Section 5.1 on FashionMNIST instead of MNIST, keeping all other choices unchanged.

Class entropy on FashionMNIST. The right panel of Figure 5 reports the class entropy of the subsets selected by each method on FashionMNIST. The pattern matches the MNIST result reported in Figure 3: at small budgets, F and F+B collapse onto a few classes and F+I+B shows a milder but still noticeable drop, while our method matches the entropy of random selection across all selection sizes. This confirms that the diversity-promoting behavior of the interaction term is not specific to MNIST and carries over to FashionMNIST.

H Computational Complexity and Resources

We analyze the selection pipeline used in the LLM experiments, including warmup, gradient extraction, curvature precomputation, and greedy selection. The analysis concerns the projected block-diagonal implementation in Appendix E.3; subsequent fine-tuning on the selected subset is outside its scope.

H.1 Computational complexity

Notation and cost model. We retain $P = |\mathcal{D}_{\mathrm{pool}}|$ for the candidate-pool size, K for the selection budget, and d for the total projected dimension used by Algorithm 1. Let n_f be the number of examples defining the target function, E the number of warmup epochs, and C_{fb} the cost of a model forward-backward pass for one example. The warmup subset has size K, as in Appendix F.3, giving a warmup cost of $O(EKC_{\mathrm{fb}})$. We consider projected curvature blocks of dimension $b \times b$, so there are d/b blocks. Accumulating one example's block-curvature contributions or applying these blocks to a projected vector costs $O(db)$, while inverting all blocks costs $O(db^2)$. Model passes used to collect the gradients and curvature statistics are included in the forward-backward cost. For this implementation-level analysis, g_i, H, H_f, and u_i denote their projected representations and curvature approximations.

Stage-by-stage cost. Table 6 separates operations shared with our matched Additive IF baseline from those introduced by the interaction term. Here, *shared* refers to the two methods using the same warmup checkpoint, projected gradients, and training-curvature approximation. Warmup and pool-side quantities can also be reused across target tasks, whereas target-side quantities are computed for each task.

Pool and target gradient extraction require one model forward-backward computation per example, up to implementation-dependent constant factors. Training- and target-curvature construction accumulate the corresponding projected block statistics. Once H^{-1} is available, computing all u_i

Table 7: Total wall-clock time summed across the six small-scale attribution settings in Figure 1, each using a single NVIDIA RTX A6000.

Method	TRAK	TracIn	*F*	*F*+*B*	*F*+*I*+*B*	*F*+*I* (ours)
Runtime (s)	267.5	191.1	458.0	472.1	489.1	475.4

requires one block-matrix-vector multiplication per candidate. The interaction term additionally requires H_f, the products $w_i = H_f u_i$, and the scalar scores $q_i = u_i^\top w_i$. Each greedy iteration evaluates the remaining candidates through projected inner products and updates the accumulator w, as in Algorithm 1. Computing the first-order scalar scores costs $O(Pd)$ and is absorbed into the shared $O(Pdb)$ bound.

Total and additional computation. Combining these stages gives

$$\mathcal{C}_{\text{select}} = O\big(\underbrace{EKC_{\text{fb}} + (P + n_f)C_{\text{fb}} + Pdb + db^2}_{\text{shared with Additive IF}} + \underbrace{(P + n_f)db + KPd}_{\text{additional interaction operations}} \big). \tag{61}$$

The shared Pdb term covers both training-curvature accumulation and preconditioning, with constant factors suppressed. Similarly, the $O(Pd)$ cost of the self-interaction scores is absorbed into the additional $O(Pdb)$ term. Thus, given the shared computations, the additional operations required by interaction-aware selection cost

$$\mathcal{C}_{\text{interaction}} = O((P + n_f)db + KPd)\,. \tag{62}$$

The first term accounts for precomputing the target-curvature quantities, and the second is the $O(KPd)$ greedy-loop cost stated in Section 3.2. These bounds count arithmetic operations; realized runtimes also depend on batching, hardware, and data movement.

H.2 Memory and storage

Each cached array of per-candidate projected vectors requires $O(Pd)$ storage. The pool gradients $\{g_i\}$ and preconditioned gradients $\{u_i\}$ are shared with Additive IF. Our method additionally stores $\{w_i\}$, requiring $O(Pd)$, the target-curvature blocks H_f, requiring $O(db)$, and $\{q_i\}$, requiring $O(P)$. The running accumulator requires $O(d)$ storage. The additional cached-data storage is therefore

$$O(Pd + db + P + d). \tag{63}$$

These quantities describe the attribution caches; model weights, optimizer state, activations, and temporary per-batch buffers are outside this accounting.

In the profiled LLM run, each of the stored pool-gradient and preconditioned-gradient arrays occupied approximately 6.7 GB. Interaction-aware selection added another approximately 6.7 GB for $\{w_i\}$, 1.6 MB for the target-curvature representation, and about 1 MB for $\{q_i\}$. The pool gradients were stored in bfloat16. These are stored-array sizes rather than measurements of peak GPU memory.

H.3 Hardware and measured runtime

Hardware. All small-scale experiments were conducted on a single NVIDIA RTX A6000 (48 GB), and all LLM experiments on 8× NVIDIA RTX A6000 (48 GB). GPU-hours are summed over all devices used.

Small-scale attribution runtime. To assess the computational overhead of interaction-aware attribution, Table 7 compares the total wall-clock time of the methods evaluated in Figure 1. We sum the runtimes across the six dataset–model settings, each evaluated on a single NVIDIA RTX A6000. These timings cover the full attribution procedures.

Table 8: Compute cost of selection methods on the LESS pool with Llama-3.1-8B, in GPU-hours.

Method	Warmup	Selection	Total
Additive IF	9.2	22	31
LESS (Xia et al., 2024)	9.2	306	315
RDS+ (Ivison et al., 2025)	–	23	23
NV-Embed (Lee et al., 2025)	–	109	109
Ours	9.2	22	31

Table 9: Per-component GPU-hours of our selection pipeline on the LESS pool with Llama-3.1-8B.

Component	GPU-hours	Scope
Warmup fine-tuning	9.2	shared
Pool gradient extraction (LoGra)	22	shared
Selection step	0.07	per target

The influence-function variants have similar runtimes because their costs are dominated by shared model training and inverse-curvature computation. Adding the interaction term increases the runtime from 458.0 s for F to 475.4 s for F+I, an overhead of approximately 3.8%. Compared with TRAK and TracIn, F+I takes approximately $1.8\times$ and $2.5\times$ as long, respectively. Thus, the improved attribution accuracy in Figure 1 comes at a higher computational cost than these alternatives, while requiring only a small additional cost over the matched first-order estimator.

LLM selection cost. Table 8 compares the selection methods on the LESS pool of approximately 270K instruction-tuning examples, and Table 9 decomposes our pipeline into its measured stages. Warmup fine-tunes a LoRA adapter on a random subset of size 13,534 for four epochs. Warmup fine-tuning and pool-gradient extraction required approximately 9.2 and 22 GPU-hours, respectively, and were shared across all seven target tasks. Additive IF and our method use the same warmup checkpoint and LoGra-projected gradients; the additional interaction operations comprise target-curvature preprocessing and greedy selection, as detailed in Appendix H.1.

The selection-stage breakdown comprises a one-time covariance-state merge shared across targets, target-set gradient extraction, first-order influence and target-curvature inner products, and the greedy selection loop. The reported selection-stage cost is approximately 0.07 GPU-hours per target, while the greedy loop itself takes approximately 50 seconds per target. The 0.07 GPU-hour figure includes target-side work also needed by Additive IF and therefore does not measure the interaction-only overhead. The comparison table reports coarsely rounded costs, so its equal entries for Additive IF and our method do not imply identical runtimes.

I Comparison with GREATS

GREATS (Wang et al., 2024a) and our selection method both use Taylor approximations to account for interactions among examples. Their relationship can be understood through the quantities they estimate, the resulting selection settings, and the interaction terms.

Estimated quantity. GREATS estimates the reduction in validation loss produced by a single gradient update at the current model parameters. Our influence formulation estimates the change in a target function induced by adding or removing training examples and retraining, using a local approximation around the reference parameters $\hat{\theta}$. These objectives lead to different parameter perturbations: the SGD update considered by GREATS follows the raw training gradients, whereas our approximation uses the preconditioned gradients $u_i = H^{-1}g_i$ to describe the response of the fitted parameters to changes in the training data.

Online versus offline selection. These estimation objectives align with different selection settings. GREATS approximates the immediate effect of a single gradient update, making it naturally suited to selecting a batch for the current training step. Our method approximates how including examples

changes the target after retraining, providing a natural criterion for selecting a fixed subset before final fine-tuning. Accordingly, GREATS performs online batch selection, choosing a subset of a sampled candidate batch at each training iteration using the current model. Its explicit pairwise corrections concern examples selected for that update, while previous updates affect subsequent scores through the evolving model parameters. Our procedure performs offline selection and keeps the resulting subset fixed throughout final fine-tuning. During greedy selection, each candidate is evaluated against the entire subset selected so far, using gradients and Hessian approximations evaluated at a fixed reference model. The marginal scores therefore change as the selected subset grows, while the reference parameters remain fixed.

Interaction terms. GREATS computes pairwise corrections using raw gradients and the validation-loss Hessian at the current training iterate. Our pairwise term takes the form

$$\kappa(z_i, z_j) = u_i^\top H_f u_j = g_i^\top H^{-1} H_f H^{-1} g_j,$$

where H and H_f are the training-objective and target-function Hessians at $\hat{\theta}$, respectively. Thus, training curvature enters both the individual influence and the pairwise interaction through H^{-1}. A further implementation difference concerns target curvature: GREATS uses an identity approximation to the validation-loss Hessian in its scalable implementation, whereas we retain target-dependent curvature through scalable Hessian approximations.