Interaction-Aware Influence Functions for Group Attribution

Jaeseung Heo [1] **Kyeongheung Yun** [2] **Youngbin Choi** [1] **Sehyun Hwang** [3] **Jungseul Ok** [1 2] **Dongwoo Kim** [1 2]

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 selection rule for instruction-tuning data on Llama-3.1-8B, it beats prior influence-based and representation-similarity baselines on five of seven downstream tasks, in a regime where standard influence-based selection underperforms random selection.

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 each look highly influential on their own, yet adding both has roughly the same effect as adding one, so additive scoring double-counts the overlap and overestimates the group's true effect (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 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

[1]Graduate School of Artificial Intelligence, POSTECH, Pohang, South Korea [2]Department of Computer Science & Engineering, POSTECH, Pohang, South Korea [3]Meta, Menlo Park, CA, USA. Correspondence to: Dongwoo Kim <dongwookim@postech.ac.kr>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

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 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 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 3 of Section 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

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

$$\hat{\theta}_A \coloneqq \arg\min_{\theta} \mathcal{R}_A(\theta). \tag{2}$$

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

$$\mathcal{I}^-(S) \coloneqq f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}). \tag{3}$$

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:

$$\hat{\theta}(\epsilon;\, S) \coloneqq \arg\min_{\theta} \frac{1}{N} \sum_{z_i \in \mathcal{D}} \ell(z_i, \theta) + \epsilon \sum_{z_i \in S} \ell(z_i, \theta). \tag{4}$$

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

$$\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \approx \frac{1}{N} H^{-1} \sum_{z_i \in S} \nabla_\theta \ell(z_i, \hat{\theta}). \tag{5}$$

To simplify the notation, let $u_i \coloneqq H^{-1} \nabla_\theta \ell(z_i, \hat{\theta})$ denote the per-example parameter shift induced by z_i, and let $u_S \coloneqq \sum_{z_i \in S} u_i$ denote its aggregate over S. The second approximation linearizes the target function f around $\hat{\theta}$, giving the standard group influence-function estimate of $\mathcal{I}^-(S)$:

$$\hat{\mathcal{I}}^-_{\text{lin}}(S) \coloneqq \frac{1}{N} \nabla_\theta f(\hat{\theta})^\top u_S. \tag{6}$$

The derivations of Equation (5) and Equation (6) are provided in Section B. Letting $\hat{\mathcal{I}}^-_{\text{lin}}(z_i)$ denote the singleton case, the standard group estimate is additive: $\hat{\mathcal{I}}^-_{\text{lin}}(S) = \sum_{z_i \in S} \hat{\mathcal{I}}^-_{\text{lin}}(z_i)$. 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

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

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 Section 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 extend the first-order approximation to a second-order one to recover the interaction information.

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

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) = \nabla_\theta f(\hat{\theta})^\top \delta_S + \tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S + O(\|\delta_S\|^3), \tag{7}$$

where $\delta_S \coloneqq \hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}$ is the parameter shift induced by removing S and $H_f \coloneqq \nabla_\theta^2 f(\hat{\theta})$ is the Hessian of the target at $\hat{\theta}$. Substituting Equation (5) into Equation (7) yields our interaction-aware influence function, an estimator of $\mathcal{I}^-(S)$:

$$\hat{\mathcal{I}}^-(S) \;\coloneqq\; \underbrace{\frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S}_{\text{first-order influence}} \;+\; \underbrace{\frac{1}{2N^2}\, u_S^\top H_f\, u_S}_{\text{interaction term (ours)}}. \tag{8}$$

The first term recovers the standard first-order estimate $\hat{\mathcal{I}}^-_{\text{lin}}(S)$ from Equation (6). 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

$$\hat{\mathcal{I}}^+(S) \;\coloneqq\; -\frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S. \tag{9}$$

The full derivation for the data-addition setting is provided in Section C.2. In practice, we use a damped Gauss–Newton Hessian approximation to ensure invertibility, and apply EK-FAC (George et al., 2018; Grosse et al., 2023) and low-rank gradient projection (Choe et al., 2025) for scalable Hessian computation. Details of these practical approximations are provided in Section D.

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

$$u_S^\top H_f\, u_S \;=\; \sum_{z_i, z_j \in S} \kappa(z_i,\, z_j), \tag{10}$$

$$\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b. \tag{11}$$

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 parameter shifts u_a and u_b under the metric induced by H_f. A positive value indicates that the two shifts 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. This similarity perspective explains why the additive approximation underestimates the exact retraining effect on groups of similar examples (Hu et al., 2024). Such groups produce large pairwise interactions that the additive approximation discards, leading to systematic underestimation. 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 Section C.4.

Relation between the first-order and interaction terms. A concrete scenario clarifies how the two terms combine. Suppose we wish to identify a group S whose removal would most reduce a target loss f. The first-order term, which sums individual influences, picks out examples whose per-example influence $\frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_i$ is negative, since each such removal individually lowers f. On a group of near-duplicate examples of this kind, the additive sum predicts a large benefit, scaling linearly with group size. The interaction term, however, is non-negative whenever H_f is positive semidefinite, and scales superlinearly with group size when the per-example shifts align. Adding it back therefore shrinks the predicted benefit, capturing the diminishing returns of removing redundant examples.

In the addition setting, the linear term flips sign while the quadratic term is unchanged. In this case, the two terms genuinely trade off: the linear term rewards candidates whose addition would individually lower f, capturing quality, while the interaction term penalizes similarity within the chosen group, capturing redundancy. This trade-off resonates with a recurring principle in data selection (Zhang et al., 2025) and active learning (Ash et al., 2020; Citovsky et al., 2021; Kirsch et al., 2019): effective subsets pair individually helpful examples with non-overlapping ones, typically by combining a per-example score with an explicit diversity term. Here, the same balance emerges directly from a second-order expansion of the target, rather than from an externally imposed design. The greedy procedure in Section 3.2 exploits this trade-off, balancing per-example quality against group diversity at every step.

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 $\sigma_i := \sigma(\hat{\theta}^\top x_i)$ for the predicted probability on example i.

Proposition 3.1 (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 $\langle u, v \rangle_M := u^\top M\, v$ is the bilinear form induced by $M := H^{-1} H_f\, H^{-1}$.

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 Section C.5.

Proposition 3.1 shows that when $\langle x_a, x_b \rangle_M > 0$, so that x_a and x_b are similar 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, this implies that adding two similar examples with different labels benefits the classifier beyond their individual effects, while 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., 2020; Toneva et al., 2019). Proposition 3.1 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. Both the factorization and these conclusions hold for deep multi-class classifiers, as we develop in Section C.6.

Remark 3.2 (Comparison with Basu et al. (2020)). Our framework expands the target function to second order, while the same second-order treatment can equally be applied to the reweighted minimizer $\hat{\theta}(\epsilon; S)$. Basu et al. (2020) pursue this alternative for group attribution. The two approaches yield structurally distinct estimators: ours decomposes into pairwise contributions between examples in S, which enables the similarity interpretation and the closed-form analysis in Proposition 3.1, while theirs operates at the parameter level without an analogous pairwise structure. The two approaches also differ in stability: $\hat{\theta}(\epsilon; S)$ need not be a smooth function of ϵ in non-convex settings, so its second-order expansion in ϵ is poorly conditioned, while the target functions we consider, such as the held-out loss, are twice continuously differentiable in θ and avoid this source of ill-conditioning. Consistent with this analysis, Basu et al. (2020) themselves report degraded performance of their estimator on non-linear models, a finding that aligns with our empirical comparison in Section 4.

Algorithm 1 Interaction-aware data selection

Require: Candidate pool $\mathcal{D}_{\text{pool}} = \{z_i\}_{i=1}^{P}$, budget K, 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 with $|S| \le K$

Initialize $S \leftarrow \emptyset$ and accumulator $w \leftarrow \mathbf{0}$, where w tracks $H_f\, u_S$
for $t = 1, \ldots, K$ **do**
  for $i \notin S$ **do**
    $m(z_i \mid S) \leftarrow -\frac{1}{N} \nabla_\theta f(\hat{\theta})^\top u_i + \frac{1}{N^2}\, w^\top u_i + \frac{1}{2N^2}\, q_i$
  end for
  $i^\star \leftarrow \arg\min_{i \notin S} m(z_i \mid S)$
  $S \leftarrow S \cup \{z_{i^\star}\}, \quad w \leftarrow w + w_{i^\star}$
end for
return S

3.2. Interaction-aware data selection

We now propose a data selection method that exploits the interaction term to select examples from a candidate pool for addition to the training set. 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:

$$S^\star \;:=\; \underset{|S|=K}{\arg\min}\, f\left(\hat{\theta}_{\mathcal{D} \cup S}\right), \tag{12}$$

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 (9) 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:

$$m(z_i \mid S) \;:=\; \hat{\mathcal{I}}^+(S \cup \{z_i\}) - \hat{\mathcal{I}}^+(S). \tag{13}$$

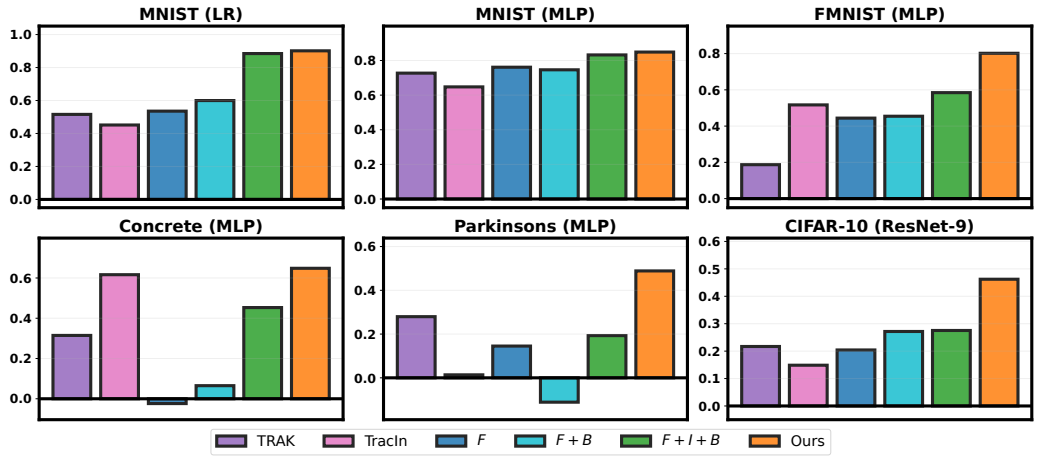

Figure 1. Spearman rank correlation between estimated and ground-truth group influences across six dataset–model pairs, where F, B, and I denote the first-order influence function, the second-order term from Basu et al. (2020), and our interaction term, respectively.

Substituting Equation (9) into Equation (13) 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

$$m(z_i \mid S) \approx -\frac{1}{N}\nabla_\theta f(\hat{\theta})^\top u_i + \frac{1}{N^2} u_S^\top H_f\, u_i + \frac{1}{2N^2} u_i^\top H_f\, u_i. \tag{14}$$

The first term is the standard first-order influence of z_i, which captures its individual contribution to reducing the target. The second term is the key novelty of our procedure: it accumulates 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 (14) sets our procedure apart from selection based on the first-order influence (Xia et al., 2024), which chooses the top-k examples by individual influence 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 addresses this limitation by penalizing candidates similar to already-selected examples, thereby promoting diversity.

Complexity. The per-candidate quantities $\{u_i\}$, $\{w_i := H_f\, u_i\}$, and $\{q_i := u_i^\top w_i\}$ 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 then reduces to $O(P)$ inner products in the dimension d of u_i plus a single update of the accumulator w, yielding a total cost of $O(KPd)$. Wall-clock measurements are reported in Table 2.

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 3.1.

Setup. We report the Spearman rank correlation between estimated and ground-truth group influences. The ground-truth $\mathcal{I}^-(S)$ is obtained by retraining from scratch on $\mathcal{D} \setminus S$ and measuring the change in average held-out test loss. This evaluation is analogous to the linear datamodeling score (LDS) (Park et al., 2023), but we avoid the term to prevent ambiguity, as our attribution is not a linear combination of individual scores.

We consider six dataset–model pairs spanning classification (MNIST (LeCun et al., 2002), FashionMNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009)) and regression (Concrete (Yeh, 1998), Parkinsons (Tsanas et al., 2009)), with models ranging from logistic regression (LR) to MLPs and ResNet-9 (He et al., 2016). To evaluate our estimator under conditions that are challenging for group attribution, we construct groups with high intra-group similarity, where existing group estimators are known to break down (Hu et al., 2024).

We compare our practical estimator against three influence-

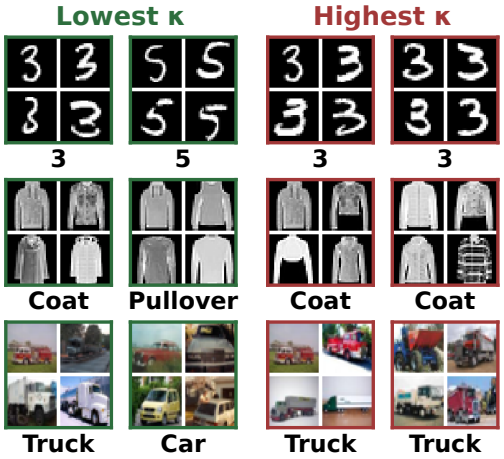

Figure 2. Representative images from the class pairs with the lowest (left) and highest (right) average pairwise interaction.

based methods. Throughout, we use F to denote the first-order influence term, B for the second-order term from Basu et al. (2020), and I for our interaction term. The baselines are then the first-order influence function (Koh & Liang, 2017) (F), its second-order variant (Basu et al., 2020) (F+B), and our estimator augmented with the Basu term (F+I+B). Our method corresponds to F+I. We additionally compare against two attribution methods aggregated to the group level by summing individual scores: TRAK (Park et al., 2023) and TracIn (Pruthi et al., 2020). Group construction, training, and influence computation details are provided in Section E.

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 second-order variant (F+B) shows instability, empirically confirming the ill-conditioning of expanding $\hat{\theta}(\epsilon; S)$ discussed in Section 3.1. The same correction applied to our estimator (F+I+B) yields either a negligible change or a substantial drop. TRAK and TracIn perform comparably to first-order influence. These findings are robust to the damping hyperparameter, as shown in Section F.

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, FashionMNIST/MLP, 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 3.1.

5. Application to Subset 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 Section E.

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) outperforms every baseline on both datasets and at every selection size, 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.

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**

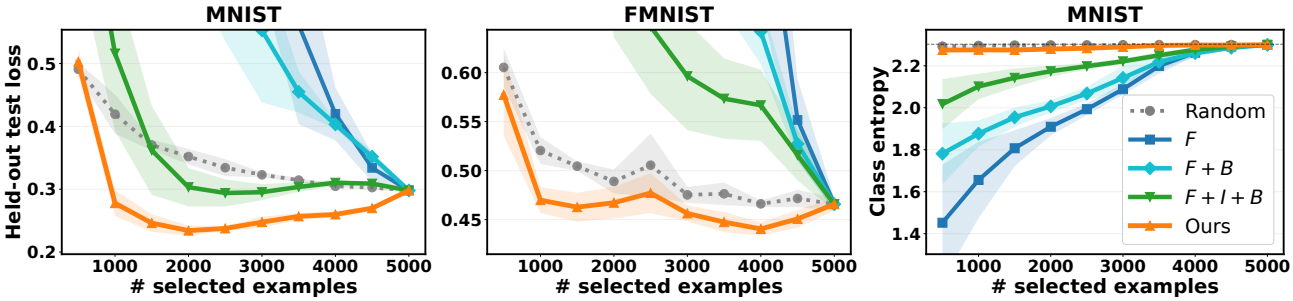

Figure 3. Held-out test loss after retraining on the selected subset on MNIST (left) and FashionMNIST (middle); class entropy of the subset on MNIST (right). Across selection sizes, lines and shaded regions show the mean and standard deviation over five seeds.

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 Section F.

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 use the LESS pool (Xia et al., 2024) of approximately 270K instruction-tuning examples as the candidate pool. Following standard practice in influence-based data selection for LLMs (Xia et al., 2024), we adopt a warmup-then-select pipeline: we first fine-tune Llama-3.1-8B on a small random subset to obtain a warmup checkpoint, which serves as $\hat{\theta}$ in our framework. We then compute per-example projected gradients u_j and target-curvature factors $H_f\, u_j$ at this checkpoint using LoGra (Choe et al., 2025), and run Algorithm 1 to select $K = 13{,}534$ examples, amounting to 5% of the pool, independently for each target task. The target function f is the instruction-masked likelihood, evaluated on the official validation set when available and on a held-out split of the training set otherwise. We then re-finetune from the original pretrained checkpoint on the selected subset. Training details, including warmup configuration, optimizer, LoRA setup, and damping, are provided in Section E.

We evaluate on seven downstream tasks covering three reasoning capabilities. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and AQuA (Ling et al., 2017). For commonsense and science reasoning, we use ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and ECQA (Aggarwal et al., 2021). For reading comprehension, we use SQuAD (Rajpurkar et al., 2016). We report exact match for GSM8K, F1 for SQuAD, and accuracy for the multiple-choice tasks.

We compare against two influence-based baselines, two representation-similarity baselines, and a random baseline. The influence-based baselines select examples with the highest target-influence scores: Additive IF uses the standard first-order influence function (Koh & Liang, 2017), sharing our pipeline and differing only in the absence of the interaction term. LESS (Xia et al., 2024) uses cosine similarity between low-rank projections of training and target gradients. The representation-similarity baselines select examples whose feature embeddings most closely match those of the target task: RDS+ (Ivison et al., 2025) uses pretrained-LM hidden states as features, and NV-Embed (Lee et al., 2025) uses embeddings from a retrieval-trained LLM. Random selection samples K examples uniformly from the pool.

Table 2. 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

Results. Table 1 reports the mean and standard deviation over five seeds. Our method attains the best performance on five of the seven tasks and surpasses random selection on every task. The two influence-based baselines, Additive IF and LESS, are outperformed on six of the seven tasks; Additive IF, which ablates our interaction term, even falls below random on most tasks, isolating the contribution of interaction-aware selection. LESS and NV-Embed further drop to below 9% accuracy on AQuA, whereas our method remains stable across all tasks. On GSM8K, although our method outperforms random selection and the influence-based baselines, the representation-similarity baselines perform especially well. We conjecture this stems from GSM8K's narrow design: problems are restricted to elementary arithmetic, single integer answers, and short, simple descriptions. This concentrated distribution plays to the strength of representation-similarity selection, which directly retrieves semantically near-identical examples from the pool.

Runtime analysis. Table 2 reports the GPU-hours required by each selection method, summed over all devices used. Our method stays within the range of the baselines, since the interaction term reuses the per-example quantities $\{u_i\}$ that first-order influence already computes and adds only matrix–vector and inner-product operations on top of them. Accordingly, Additive IF and our method share the same warmup checkpoint and LoGra-projected gradients, so their selection costs differ only in the greedy step, which contributes approximately 50 seconds per target.

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., 2024; Yu et al., 2025). 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 subset selection, it outperforms random selection on every task and beats 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.

References

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

Aggarwal, S., Mandowara, D., Agrawal, V., Khandelwal, D., Singla, P., and Garg, D. 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.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. Deep batch active learning by diverse, uncertain gradient lower bounds. *International Conference on Learning Representations*, 2020.

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. B. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 2022.

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

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

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

Bisk, Y., Zellers, R., Le Bras, R., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Chang, T. A., Rajagopal, D., Bolukbasi, T., Dixon, L., and Tenney, I. Scalable influence and fact tracing for large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025.

Choe, S. K., Ahn, H., Bae, J., Zhao, K., Kang, M., Chung, Y., Pratapa, A., Neiswanger, W., Strubell, E., Mitamura, T., et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *Advances in neural information processing systems*, 2025.

Citovsky, G., DeSalvo, G., Gentile, C., Karydas, L., Rajagopalan, A., Rostamizadeh, A., and Kumar, S. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 2021.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., and Zaharia, M. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 2017.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 1995.

Dai, Q., Zhang, D., Ma, J. W., and Peng, H. Improving influence-based instruction tuning data selection for balanced learning of diverse capabilities. *Findings of the Association for Computational Linguistics*, 2025.

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

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

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in neural information processing systems*, 2018.

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

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Heo, J., Yun, K., Yoon, S., Park, M., Ok, J., and Kim, D. Influence functions for edge edits in non-convex graph

neural networks. *Advances in Neural Information Processing Systems*, 2025.

Hu, Y., Hu, P., Zhao, H., et al. Most influential subset selection: Challenges, promises, and beyond. *Advances in Neural Information Processing Systems*, 2024.

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

Ivison, H., Zhang, M., Brahman, F., Koh, P. W., and Dasigi, P. Large-Scale Data Selection for Instruction Tuning. *arXiv preprint arXiv:2503.01807*, 2025.

Kirsch, A., Van Amersfoort, J., and Gal, Y. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 2019.

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

Koh, P. W. W., Ang, K.-S., Teo, H., and Liang, P. S. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 2019.

Kreer, P. A., Wu, W., Adam, M., Furman, Z., and Hoogland, J. Bayesian influence functions for hessian-free data attribution. In *The Fourteenth International Conference on Learning Representations*, 2026.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kwon, Y., Wu, E., Wu, K., and Zou, J. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *International Conference on Learning Representations*, 2024.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2002.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., and Ping, W. Nv-embed: Improved techniques for training llms as generalist embedding models. *International Conference on Learning Representations*, 2025.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. 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.

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

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

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

Mlodozeniec, B. K., Eschenhagen, R., Bae, J., Immer, A., Krueger, D., and Turner, R. E. Influence functions for scalable data attribution in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Pan, X., Huang, L., Kang, L., Liu, Z., Lu, Y., and Cheng, S. 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.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, 2023.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 2020.

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

Robinson, J. D., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

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

Saunshi, N., Gupta, A., Braverman, M., and Arora, S. Understanding influence functions and datamodels via harmonic analysis. *International Conference on Learning Representations*, 2023.

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

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

Tan, H., Wu, S., Huang, W., Zhao, S., and QI, X. Data pruning by information maximization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

Tsanas, A., Little, M., McSharry, P., and Ramig, L. Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. *Nature Precedings*, 2009.

Wang, A., Nguyen, E., Yang, R., Bae, J., McIlraith, S. A., and Grosse, R. Better training data attribution via better inverse hessian-vector products. *Advances in Neural Information Processing Systems*, 2025a.

Wang, J. T., Yang, T., Zou, J., Kwon, Y., and Jia, R. Rethinking data shapley for data selection tasks: Misleads and merits. *International Conference on Machine Learning*, 2024.

Wang, J. T., Mittal, P., Song, D., and Jia, R. Data shapley in one training run. *International Conference on Learning Representations*, 2025b.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., et al. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 2023.

Wei, J., Liu, B., Wan, T., Peng, B., Ma, X., and Guo, M. 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.

Xia, M., Malladi, S., Gururangan, S., Arora, S., and Chen, D. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning*, 2024.

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

Yang, Y., Mishra, S., Chiang, J. N., and Mirzasoleiman, B. 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.

Ye, X., Wu, Y., Zhang, W., Jin, C., and Chen, Y. Towards robust influence functions with flat validation minima. *arXiv preprint arXiv:2505.19097*, 2025.

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

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

Yu, Z., Peng, F., Lei, J., Overwijk, A., tau Yih, W., and Xiong, C. Group-level data selection for efficient pretraining. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Zhang, C., Zhong, H., Zhang, K., Chai, C., Wang, R., Zhuang, X., Bai, T., Qiu, J., Cao, L., Fan, J., et al. Harnessing diversity for important data selection in pretraining large language models. *International Conference on Learning Representations*, 2025.

A. Notation

Table 3 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 3. 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
$S \subseteq \mathcal{D}$	Subset (group) of training examples
$\theta \in \mathbb{R}^p$	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
$L(\theta) = \frac{1}{N}\sum_i \ell(z_i, \theta)$	Empirical training loss
$f(\theta)$	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: $f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta})$
$\hat{\mathcal{I}}^-_{\text{lin}}(S)$	First-order estimate of $\mathcal{I}^-(S)$
$\hat{\mathcal{I}}^+_{\text{lin}}(S)$	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 $\hat{\theta}$
$H \coloneqq \nabla^2_\theta L(\hat{\theta})$	Training-loss Hessian 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 3.1
$u_i \coloneqq H^{-1} g_i$	Per-example parameter shift
$u_S \coloneqq \sum_{z_i \in S} u_i$	Aggregate parameter shift 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	
$\mathcal{D}_{\text{pool}}$	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

B. Derivation of the First-Order Influence Function

We derive Equation (5) and Equation (6), 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

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

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 (5). The second step substitutes the resulting parameter shift into a first-order Taylor expansion of f around $\hat{\theta}$, yielding Equation (6). 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 (4),

$$\mathcal{R}(\theta; \epsilon, S) \coloneqq \frac{1}{N} \sum_{z_i \in \mathcal{D}} \ell(z_i, \theta) + \epsilon \sum_{z_i \in S} \ell(z_i, \theta), \tag{15}$$

and its minimizer $\hat{\theta}(\epsilon; S) \coloneqq \arg\min_\theta \mathcal{R}(\theta; \epsilon, S)$. At $\epsilon = 0$ the perturbation vanishes and we recover $\hat{\theta}(0; S) = \hat{\theta}$. At $\epsilon = -1/N$, substituting into Equation (15) 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 (15) and evaluating at $\theta = \hat{\theta}(\epsilon; S)$ gives

$$\frac{1}{N} \sum_{z_i \in \mathcal{D}} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) + \epsilon \sum_{z_i \in S} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) = 0. \tag{16}$$

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 (16) in ϵ and applying the chain rule to each such term gives

$$\underbrace{\left[\frac{1}{N} \sum_{z_i \in \mathcal{D}} \nabla_\theta^2 \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) + \epsilon \sum_{z_i \in S} \nabla_\theta^2 \ell\big(z_i, \hat{\theta}(\epsilon; S)\big)\right]}_{\text{Hessian of } \mathcal{R}(\cdot; \epsilon, S)} \frac{d\hat{\theta}(\epsilon; S)}{d\epsilon} + \sum_{z_i \in S} \nabla_\theta \ell\big(z_i, \hat{\theta}(\epsilon; S)\big) = 0. \tag{17}$$

At $\epsilon = 0$, the second summand inside the bracket is zero due to its ϵ prefactor, and the bracket reduces to H. Using $g_i \coloneqq \nabla_\theta \ell(z_i, \hat{\theta})$ as defined in Section 2, Equation (17) becomes

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

Since H is invertible, we solve for the slope:

$$\frac{d\hat{\theta}(\epsilon; S)}{d\epsilon}\bigg|_{\epsilon=0} = -H^{-1} \sum_{z_i \in S} g_i. \tag{19}$$

Parameter-shift approximation (Equation (5)). We now use the slope in Equation (19) inside the first Taylor expansion promised in the strategy. Expanding $\hat{\theta}(\epsilon; S)$ to first order around $\epsilon = 0$ gives

$$\hat{\theta}(\epsilon; S) \approx \hat{\theta} + \epsilon \cdot \left.\frac{d\hat{\theta}(\epsilon; S)}{d\epsilon}\right|_{\epsilon=0} = \hat{\theta} - \epsilon\, H^{-1} \sum_{z_i \in S} g_i. \tag{20}$$

Setting $\epsilon = -1/N$, which corresponds to leaving S out, recovers Equation (5):

$$\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \approx \frac{1}{N} H^{-1} \sum_{z_i \in S} g_i. \tag{21}$$

Target-change approximation via the parameter shift (Equation (6)). For the second step, a first-order Taylor expansion of the target f around $\hat{\theta}$ gives

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) \approx \nabla_\theta f(\hat{\theta})^\top \big(\hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta}\big). \tag{22}$$

Substituting the parameter shift from Equation (21) yields the standard first-order group influence estimate of Equation (6):

$$\hat{\mathcal{I}}^-_{\text{lin}}(S) := \frac{1}{N} \nabla_\theta f(\hat{\theta})^\top H^{-1} \sum_{z_i \in S} g_i. \tag{23}$$

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

C. Derivations and Proofs for the Interaction-Aware Influence Function

C.1. Derivation of the interaction-aware influence function

We derive the interaction-aware influence function in Equation (8) by combining the second-order Taylor expansion of the target function with the first-order parameter shift induced by removing S.

Recall from Equation (5) in Section 2 that the parameter shift induced by removing S admits the first-order approximation

$$\delta_S \;:=\; \hat{\theta}_{\mathcal{D}\setminus S} - \hat{\theta} \;\approx\; \frac{1}{N} H^{-1} \sum_{z_i \in S} g_i \;=\; \frac{1}{N} u_S, \tag{24}$$

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 (7) is

$$f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta}) \;=\; \nabla_\theta f(\hat{\theta})^\top \delta_S \;+\; \tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S \;+\; O(\|\delta_S\|^3). \tag{25}$$

Substituting Equation (24) into the linear term of Equation (25) gives

$$\nabla_\theta f(\hat{\theta})^\top \delta_S \;\approx\; \frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S. \tag{26}$$

Substituting the same approximation into the quadratic term gives

$$\tfrac{1}{2}\, \delta_S^\top H_f\, \delta_S \;\approx\; \tfrac{1}{2} \left(\frac{1}{N} u_S\right)^\top H_f \left(\frac{1}{N} u_S\right) \;=\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S, \tag{27}$$

where the factor $1/N^2$ arises from the two factors of $1/N$ in δ_S. Combining Equations (26) and (27) and recalling that the leave-group-out effect is $\mathcal{I}^-(S) = f(\hat{\theta}_{\mathcal{D}\setminus S}) - f(\hat{\theta})$ yields

$$\mathcal{I}^-(S) \;\approx\; \frac{1}{N}\, \nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S, \tag{28}$$

which is Equation (8) 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 (6), and the second term is the curvature-induced interaction term that gives rise to non-additivity across examples in S.

C.2. Derivation for the data-addition setting

We now derive the addition-setting influence function in Equation (9). The argument parallels Section C.1, 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 (4) 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

$$\hat{\theta}_{\mathcal{D}\cup S} - \hat{\theta} \;\approx\; -\frac{1}{N} H^{-1} \sum_{z_i \in S} g_i \;=\; -\frac{1}{N} u_S, \tag{29}$$

which differs from Equation (24) only in sign. A complete derivation through the implicit function theorem is given in Section B; the sign flip reflects the opposite direction of the reweighting step.

Substituting Equation (29) into the second-order Taylor expansion of f around $\hat{\theta}$ yields

$$\begin{aligned} f(\hat{\theta}_{\mathcal{D}\cup S}) - f(\hat{\theta}) \;&\approx\; \nabla_\theta f(\hat{\theta})^\top \left(-\frac{1}{N} u_S\right) \;+\; \tfrac{1}{2} \left(-\frac{1}{N} u_S\right)^\top H_f \left(-\frac{1}{N} u_S\right) \\ &=\; -\frac{1}{N} \nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2} u_S^\top H_f\, u_S. \end{aligned} \tag{30}$$

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 (9) 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.

C.3. Pairwise decomposition of the interaction term

The interaction term $u_S^\top H_f\, u_S$ in Equations (8) and (9) 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 aggregate of per-example shifts. Substituting this definition into the bilinear form and expanding gives

$$u_S^\top H_f\, u_S \;=\; \left(\sum_{z_i \in S} u_i\right)^\top H_f \left(\sum_{z_j \in S} u_j\right) \;=\; \sum_{z_i \in S} \sum_{z_j \in S} u_i^\top H_f\, u_j \;=\; \sum_{z_i, z_j \in S} \kappa(z_i,\, z_j), \tag{31}$$

where $\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b$ as defined in Equation (11). This recovers the pairwise form used throughout Section 3.1.

Self and cross contributions. The double sum in Equation (31) ranges over ordered pairs and can be further separated into diagonal and off-diagonal contributions:

$$u_S^\top H_f\, u_S \;=\; \underbrace{\sum_{z_i \in S} \kappa(z_i,\, z_i)}_{\text{self contribution}} \;+\; \underbrace{2 \sum_{\{z_i, z_j\} \subset S,\, i \neq j} \kappa(z_i,\, z_j)}_{\text{cross contribution}}, \tag{32}$$

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 (10) so that the analysis remains agnostic to whether self or cross terms dominate. The split into self and cross contributions in Equation (32) becomes useful when isolating the genuinely interactional content, which we exploit in the marginal score derivation in Section C.7.

C.4. 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

$$H_f \;=\; \sum_k \mu_k\, v_k v_k^\top, \tag{33}$$

where $\{\mu_k\}$ are the real eigenvalues and $\{v_k\}$ form an orthonormal basis of unit eigenvectors. Substituting Equation (33) into the definition of the interaction term $\kappa(a,\, b) = u_a^\top H_f\, u_b$ yields

$$\kappa(a,\, b) \;=\; \sum_k \mu_k\, (v_k^\top u_a)(v_k^\top u_b). \tag{34}$$

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 parameter shifts 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 (34) is as an aggregate similarity score. The shifts 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 (33) remains valid, so the per-direction decomposition of $\kappa(a,\, b)$ in Equation (34) 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 shifts 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.5. Proof of Proposition 3.1

We prove the closed-form factorization of $\kappa(a,\, b)$ stated in Proposition 3.1. 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 per-example parameter shift 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

$$\ell(z_i, \theta) \;=\; -\, y_i \log \sigma(\theta^\top x_i) \;-\; (1 - y_i) \log\big(1 - \sigma(\theta^\top x_i)\big)\,, \tag{35}$$

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

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; (\sigma_i - y_i)\, x_i. \tag{36}$$

The training objective is the regularized empirical risk

$$\mathcal{R}(\theta) \;=\; \frac{1}{N} \sum_i \ell(z_i, \theta) \;+\; \frac{\beta}{2} \|\theta\|^2. \tag{37}$$

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

$$H \;\coloneqq\; \nabla^2_\theta \mathcal{R}(\hat{\theta}) \;=\; \frac{1}{N} \sum_i \nabla^2_\theta \ell(z_i, \hat{\theta}) \;+\; \beta\, I \;=\; \frac{1}{N} \sum_i \sigma_i (1 - \sigma_i)\, x_i x_i^\top \;+\; \beta\, I, \tag{38}$$

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 (36) as the per-example gradient driving the influence approximation throughout the remainder of the proof.

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

$$u_i \;=\; H^{-1}[(\sigma_i - y_i) x_i] \;=\; (\sigma_i - y_i)\, H^{-1} x_i, \tag{39}$$

where the second equality uses the linearity of matrix-vector multiplication and the fact that $(\sigma_i - y_i)$ is a scalar. Each per-example parameter shift 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 (39) into the definition $\kappa(a,\, b) \coloneqq u_a^\top H_f\, u_b$ from Equation (11) gives

$$\begin{aligned} \kappa(a,\, b) \;&=\; \big[(\sigma_a - y_a)\, H^{-1} x_a\big]^\top H_f \,\big[(\sigma_b - y_b)\, H^{-1} x_b\big] \\ &=\; (\sigma_a - y_a)(\sigma_b - y_b)\, x_a^\top H^{-1} H_f\, H^{-1} x_b \\ &=\; (\sigma_a - y_a)(\sigma_b - y_b)\, \langle x_a, x_b \rangle_M, \end{aligned} \tag{40}$$

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 3.1, completing the proof.

C.6. Extension of Proposition 3.1 to deep classifiers

Proposition 3.1 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 chain rule applied to the per-example gradient: the same scalar-residual–input-feature factorization that drives the proof of Proposition 3.1 survives when the input feature x_i is replaced by the logit Jacobian

$$J_i \;\coloneqq\; \nabla_\theta f_{\hat{\theta}}(x_i), \tag{41}$$

where $f_\theta : \mathcal{X} \to \mathbb{R}^C$ is the network logit output, 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 3.1 is unchanged. We treat the binary case first as it offers the cleanest correspondence with Proposition 3.1, 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 $\sigma_i \coloneqq \sigma(f_{\hat{\theta}}(x_i))$, the per-example loss $\ell(z_i, \theta) = -\, y_i \log \sigma(f_\theta(x_i)) - (1 - y_i) \log(1 - \sigma(f_\theta(x_i)))$ has gradient at $\hat{\theta}$ given by chain rule as

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; (\sigma_i - y_i)\, J_i, \tag{42}$$

which differs from Equation (36) 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 3.1 therefore carry over verbatim with $x_i \mapsto J_i$, yielding

$$\kappa(a,\, b) \;=\; (\sigma_a - y_a)(\sigma_b - y_b)\, \langle J_a, J_b \rangle_M. \tag{43}$$

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 $f_\theta(x) = w^\top \phi_\psi(x)$ where $\phi_\psi : \mathcal{X} \to \mathbb{R}^{d_\phi}$ denotes the learned feature representation, the logit Jacobian J_i admits the block decomposition

$$J_i = \begin{bmatrix} J_\phi(x_i)^\top w \\ \phi(x_i) \end{bmatrix}, \qquad J_\phi(x_i) \coloneqq \frac{\partial \phi_\psi(x_i)}{\partial \psi}, \tag{44}$$

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:

$$\langle J_a, J_b \rangle_M \;=\; \begin{bmatrix} J_\phi(x_a)^\top w \\ \phi(x_a) \end{bmatrix}^\top M \begin{bmatrix} J_\phi(x_b)^\top w \\ \phi(x_b) \end{bmatrix}. \tag{45}$$

In Equation (45), the bottom-block component $\phi(x_i)$ takes the place of the raw input feature x_i in Proposition 3.1. 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 $f_{\hat{\theta}}(x_i)$ 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 chain rule as

$$\nabla_\theta \ell(z_i, \hat{\theta}) \;=\; J_i\, r_i, \qquad r_i \;\coloneqq\; p_i - y_i \;\in\; \mathbb{R}^C. \tag{46}$$

Equation (46) generalizes Equation (36) 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 (46) 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 (11) and factoring yields

$$\begin{aligned} \kappa(a,\, b) \;&=\; (H^{-1} J_a\, r_a)^\top H_f\, (H^{-1} J_b\, r_b) \\ &=\; r_a^\top\, J_a^\top H^{-1} H_f\, H^{-1}\, J_b\, r_b \\ &=\; r_a^\top\, K_M(x_a, x_b)\, r_b, \end{aligned} \tag{47}$$

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

$$K_M(x_a, x_b) \;\coloneqq\; J_a^\top M\, J_b \;\in\; \mathbb{R}^{C \times C}, \tag{48}$$

which is a matrix-valued empirical neural tangent kernel when H_f is positive semidefinite. The factorization in Equation (47) is the multi-class generalization of Equation (40): 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 (43) is recovered when $C = 1$, with $r_i = \sigma_i - y_i$ and $K_M = \langle J_a, J_b \rangle_M$ a scalar.

Connection to the class-agreement structure. Proposition 3.1 identifies cross-class similar pairs as beneficial and same-class similar pairs as redundant, with the sign of $\kappa(a,\, b)$ controlled by the residual product $(\sigma_a - y_a)(\sigma_b - y_b)$. The multi-class factorization in Equation (47) preserves the structural form *residual structure* $\times$ *Jacobian similarity*, with the residual product replaced by the bilinear form $r_a^\top K_M\, r_b$. A direct calculation yields the residual-alignment identity

$$r_a^\top r_b \;=\; \langle p_a, p_b\rangle - p_{a,y_b} - p_{b,y_a} + \mathbb{1}[y_a = y_b], \tag{49}$$

in which the indicator term contributes $+1$ for same-class pairs and 0 for cross-class pairs. For predictions biased toward their true classes, this indicator drives the sign of the residual alignment positive for same-class pairs and negative for cross-class pairs; for example, with $p_{i,c} = (1-\delta)\,\mathbb{1}[c = y_i] + \delta/(C-1)$ on the remaining classes, a short calculation gives $r_a^\top r_b = \delta^2\, C/(C-1) > 0$ for same-class pairs and $r_a^\top r_b = -\delta^2\, C/(C-1)^2 < 0$ for cross-class pairs. When two examples are similar so that $J_a \approx J_b$ and M is positive semidefinite, $K_M(x_a, x_b) \approx J_a^\top M J_a$, which is symmetric PSD, and $\kappa(a,\, b)$ inherits the sign of the residual alignment. This recovers the cross-class-beneficial and same-class-redundant structure of Proposition 3.1 and is consistent with the Figure 2 observation that this class-pair structure carries over from logistic regression to MLPs and ResNet-9.

C.7. Derivation of the marginal score

We derive the marginal score in Equation (14) 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 aggregate parameter shift.

Recall from Equation (9) that the addition-setting estimator is

$$\hat{\mathcal{I}}^+(S) \;\coloneqq\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;+\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S, \tag{50}$$

and that the aggregate approximated shift 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

$$u_{S\cup\{z_j\}} \;=\; u_S + u_j. \tag{51}$$

Substituting Equation (51) into the linear term of Equation (50) evaluated at $S \cup \{z_j\}$ gives

$$-\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_{S\cup\{z_j\}} \;=\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_S \;-\; \frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_j. \tag{52}$$

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 (51) into the quadratic term and expanding gives

$$\begin{aligned}\frac{1}{2N^2}\, u_{S\cup\{z_j\}}^\top H_f\, u_{S\cup\{z_j\}} \;&=\; \frac{1}{2N^2}\,(u_S + u_j)^\top H_f\,(u_S + u_j)\\ &=\; \frac{1}{2N^2}\, u_S^\top H_f\, u_S \;+\; \frac{1}{N^2}\, u_S^\top H_f\, u_j \;+\; \frac{1}{2N^2}\, u_j^\top H_f\, u_j,\end{aligned} \tag{53}$$

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 (53) 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 (52) and (53) yields

$$m(z_j \mid S) \;\approx\; -\frac{1}{N}\,\nabla_\theta f(\hat{\theta})^\top u_j \;+\; \frac{1}{N^2}\, u_S^\top H_f\, u_j \;+\; \frac{1}{2N^2}\, u_j^\top H_f\, u_j, \tag{54}$$

which is exactly Equation (14) 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 Section C.3. The cross term $\frac{1}{N^2}\, u_S^\top H_f\, u_j$ in Equation (54) 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.

D. Scalable Approximations: EK-FAC and Low-Rank Gradient Projection

The estimators $\hat{\mathcal{I}}^-(S)$ and $\hat{\mathcal{I}}^+(S)$ defined in Equation (8) and Equation (9) 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 Section E.

D.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.

D.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 per-example shifts u_i required by Equation (8) and Equation (9) 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.

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

At the scale of Llama-3.1-8B, the per-layer EK-FAC blocks remain large enough that materializing and storing one projected gradient u_i per training example, as required by the greedy procedure in Algorithm 1, is itself the dominant cost. To address this, we adopt the low-rank gradient projection of LoGra (Choe et al., 2025), which introduces a shared low-dimensional subspace and projects all per-example gradients into that subspace before any subsequent inner product or curvature operation

is performed. The projection is applied consistently to gradients and to the target-side quantities entering Algorithm 1, so that all inner products in the greedy loop are evaluated in the projected dimension d rather than in 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 dimension and hyperparameters reported in Section E, and refer the reader to Choe et al. (2025) for the construction of the projection operator and a full treatment of the resulting estimator.

E. Experimental Details

E.1. Small-scale Attribution Experiments

The attribution experiments in Section 4 consider six dataset–model pairs: logistic regression and MLP on MNIST, MLP on FashionMNIST, MLP on Concrete and Parkinsons, and ResNet-9 on CIFAR-10. All datasets use their standard train/test splits.

Models. 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.

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

Group construction. 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 set $|S| = 400$, yielding 50 subsets per dataset–model pair, except for Concrete where we set $|S| = 100$ due to its small training set. For regression tasks, the prediction itself replaces the softmax output.

Ground truth. For each subset S, we retrain the model from scratch on $\mathcal{D} \setminus S$ and report the mean change in held-out test loss.

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.

Baselines. *F* is the standard first-order influence function (Koh & Liang, 2017). *F+B* applies the second-order response-function correction of Basu et al. (2020), computed under the same curvature approximation as our method. *F+I+B* combines our interaction term with the Basu correction under identical approximations. 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.

E.2. Small-Scale Selection Experiments

The selection experiments in Section 5.1 use a two-hidden-layer MLP ($784 \to 128 \to 64 \to 10$) with ReLU activations and dropout rate 0.1. For each dataset, a candidate pool $\mathcal{D}_{\text{pool}}$ of $5{,}000$ training examples is drawn by a fixed permutation of the original training set, and the reference parameter $\hat{\theta}$ is obtained by training the MLP on this pool for 200 epochs with vanilla SGD (learning rate 10^{-2}, batch size 64, weight decay 3×10^{-2}). The Hessian-inverse H^{-1} is approximated via EK-FAC

Table 5. 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

with the GGN Fisher and damping $\lambda = 5\times10^{-1}$. Each selection picks $K \in \{500, 1{,}000, \ldots, 5{,}000\}$ examples in increments of 500 from the same pool $\mathcal{D}_{\text{pool}}$, and each selected subset is retrained from scratch with the same training configuration. Held-out test loss is averaged over five random seeds and reported; the class-entropy panel reports the Shannon entropy of the empirical class distribution among the K selected examples.

E.3. LLM Data Selection Experiments

We fine-tune Llama-3.1-8B on subsets selected from the LESS pool of approximately 270K instruction-tuning examples.

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 per-example projected gradients u_j and curvature terms $H_f u_j$ using LoGra (Choe et al., 2025) with projection dimension 16, and apply Algorithm 1 to select $K = 13{,}534$ examples (5% of the pool) independently for each target task. The target function f is the instruction-masked 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 GSM8K, AQuA, ARC-Easy, HellaSwag, PIQA, ECQA, and SQuAD using the Tulu chat template (Wang et al., 2023). All tasks use 0-shot prompting. 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. 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 our warmup checkpoint and projected gradients but selects the top-K examples by first-order influence, omitting the interaction term. LESS (Xia et al., 2024) uses Adam-adjusted gradient cosine similarity with projection dimension 8192 on the same warmup checkpoint. RDS+ (Ivison et al., 2025) uses position-weighted mean-pooled hidden states from the warmup checkpoint with cosine similarity. NV-Embed (Lee et al., 2025) uses embeddings from `nvidia/NV-Embed-v2`. Random uniformly subsamples from the pool.

E.4. Compute Resources

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).

Table 5 decomposes the selection cost of our pipeline into its individual components. Warmup fine-tuning and pool gradient extraction are shared across all seven targets, while the selection step is paid once per target. The selection step itself further decomposes into four sub-steps: a one-time covariance-state merge that is shared across targets, validation-set gradient extraction that scales per target, first-order influence and target-curvature inner products, and the greedy selection loop. These four sub-steps together account for the 0.07 GPU-hours per target reported in Table 5.

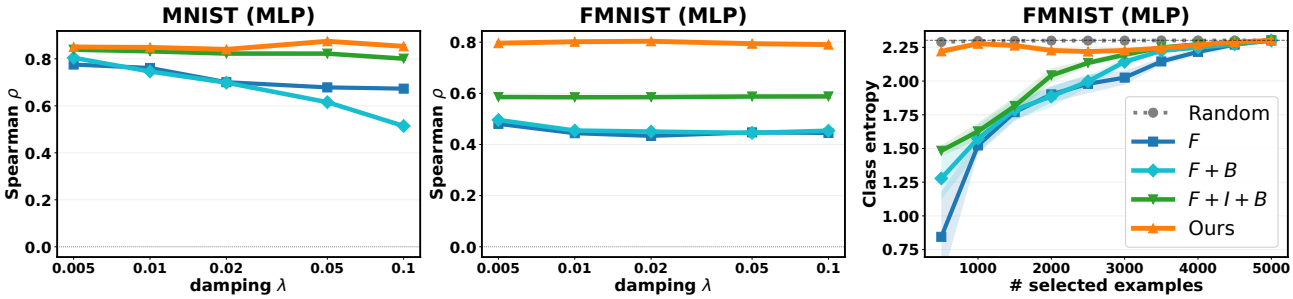

Figure 4. Sensitivity analysis of the two-layer MLP over the damping hyperparameter on MNIST (left) and FashionMNIST (middle), and class entropy on FashionMNIST (right).

F. Additional Experiments

In this section, 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 4.

Setup. Both experiments inherit the protocols of their counterparts in the main text, modified as follows. 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. For the class entropy analysis, we run the selection procedure of Section 5.1 on FashionMNIST instead of MNIST, keeping all other choices unchanged.

Damping sensitivity. The left and middle panels of Figure 4 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.

Class entropy on FashionMNIST. The right panel of Figure 4 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.