BEYOND SCALAR RANKINGS: NONLINEAR ATTRIBUTION AGGREGATION FOR DATA SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Selecting high-quality training data is increasingly important when annotation, storage, and computation budgets limit how much of an available data pool can be used. Data-attribution methods support this process by assigning a value to each training example, but they are commonly paired with a top-m rule that compresses an example's effects into a scalar score, which can lead to redundant or poorly balanced subsets. In this paper, we introduce NADA (*Nonlinear Aggregation of Data Attribution Components*), a method-agnostic selection framework that preserves component-level attribution information and selects complementary subsets through nonlinear aggregation to obtain more informative training data. We demonstrate the effectiveness of our method on standard logistic-regression and ridge-regression models, as well as prompt-based fine-tuning models, showing that NADA outperforms the corresponding scalar top-score selector in most matched settings and demonstrating the advantage of nonlinear component aggregation in mitigating the information loss caused by scalar attribution ranking.

1 INTRODUCTION

Data has become a central determinant of model quality in modern machine learning. As models and training corpora grow, performance increasingly depends not only on architectures and optimization algorithms, but also on whether the training data are relevant, diverse, reliable, and aligned with the downstream task. However, model developers often cannot train on every available example due to the expense of data acquisition and limited storage. Moreover, large data pools may also contain redundant, mislabeled, irrelevant, or even harmful examples. Data selection therefore seeks a compact subset maximizing downstream utility (Ilyas & Engstrom, 2025; Albalak et al., 2024).

A broad literature addresses this problem through *training-data attribution*: estimating how training examples affect learned models, predictions, or downstream utilities (Hammoudeh & Lowd, 2024). Retraining-based methods directly evaluate changes to the training set. For example, leave-one-out measures the effect of removing one example from the full dataset; Data Shapley and related cooperative-game values average marginal contributions over different training-data coalitions (Ghorbani & Zou, 2019; Jia et al., 2019; Wang & Jia, 2023). Gradient-based methods avoid repeated retraining by estimating influence from a trained model. Classical influence functions (Koh & Liang, 2017) use local second-order sensitivity, representer points (Yeh et al., 2018) express predictions through weighted training examples, TracIn (Pruthi et al., 2020) accumulates gradient alignment along the training trajectory, and TRAK (Park et al., 2023) uses projected gradient features to scale attribution to larger models. Recent work extends influence attribution to nondecomposable objectives and stochastic training procedures (Deng et al., 2025; Mlodozeniec et al., 2025).

Nevertheless, once attribution scores have been computed, data selection is usually performed in the same way: each candidate example is assigned one scalar score, and the m examples with the largest scores are selected. This scalar top-m selection rule is not always reliable. Prior experiments show that influence functions, TRAK, LESS, Data Shapley (Park et al., 2023; Xia et al., 2024; Shapley, 1953), and other attribution methods can perform worse than Random at small selection budgets (Wang et al., 2025). These results do not necessarily imply that the attribution scores contain no useful information. Rather, they reveal an important distinction between evaluating examples individually and selecting them collectively. A method may correctly identify examples that are helpful on their own, while the highest-scoring examples may still form a poor subset when selected

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

together. The usefulness of a candidate depends on the examples already selected. Candidates supporting different labels or targets can complement one another, whereas those with overlapping strengths may add little new benefit. Scalar top-m ranking ignores this dependence and may select a redundant subset. This motivates preserving each candidate's different attribution effects and evaluating them relative to the selected subset. We therefore ask:

> *How can data-attribution scores be used to construct a jointly useful subset, rather than simply selecting the examples with the highest individual scores, without modifying the underlying attribution method?*

Recent work (Tian et al., 2026) addresses this question for Data Shapley and semivalue-based selection by nonlinearly aggregating Shapley or semivalue scores for validation-level utilities. However, its theoretical development and empirical evaluation are limited to coalition-based data values, leaving it unclear whether the same selection principle can improve attribution methods based on local sensitivity, training trajectories, or removal effects.

In this paper, we propose NADA (*Nonlinear Aggregation of Data Attribution Components*), a method-agnostic framework for selecting jointly useful subsets from existing attribution scores without changing their estimator. Our contributions are summarized as follows:

1. We extend the component-level perspective on scalar top-m selection to general attribution matrices and introduce *attribution-informativeness*, a criterion for assessing whether accumulated component scores predict component utilities on uniformly sampled subsets of a fixed size. This criterion separates the predictive content of attribution scores from the choice of selection rule; it does not by itself justify a particular nonlinear aggregation. Our scope analysis shows that exact attribution-informativeness can coexist with an aggregation objective whose maximizer has zero true utility (Section 4.1 and Appendix B.5).

2. We introduce NADA, a two-stage, method-agnostic selection framework. An existing attribution method first produces signals for multiple validation examples, task groups, or target behaviors. NADA then retains these signals during subset construction and applies increasing concave aggregation, giving diminishing returns to already-supported components and adapting each candidate's priority to the selected subset (Section 4.2).

3. We instantiate NADA with classical data attribution methods, such as influence functions (Koh & Liang, 2017), TracIn (Pruthi et al., 2020), and leave-one-out (Hammoudeh & Lowd, 2024), without modifying the underlying estimators (Section 4.4). With 134/144 wins on logistic-regression and ridge-regression data selection benchmarks and 447/483 wins on prompt-based fine-tuning benchmarks, we empirically show that NADA substantially improves the effectiveness of the data attribution methods against top-m selection on various datasets and tasks (Section 5).

2 Related Work

Data valuation and training-data attribution. Training-data attribution assigns importance signals to training examples according to their effects on a learned model, prediction, or downstream utility (Hammoudeh & Lowd, 2024). Utility-based methods evaluate counterfactual changes to the training set. Leave-one-out (LOO) measures the utility change caused by removing one example from the full dataset, while Data Shapley averages marginal contributions over training-data coalitions (Ghorbani & Zou, 2019; Jia et al., 2019). Data Banzhaf and related semivalues modify the coalition weighting to improve robustness or computational properties (Wang & Jia, 2023; Kwon & Zou, 2022). Gradient-based methods estimate attribution from a trained model without explicitly retraining on every candidate subset. Influence functions approximate the local effect of upweighting an example (Koh & Liang, 2017), representer points decompose predictions into weighted training examples (Yeh et al., 2018), and TracIn accumulates gradient alignment along the optimization trajectory (Pruthi et al., 2020). More scalable or task-specific methods include TRAK, which uses projected gradient features (Park et al., 2023); LESS, which constructs optimizer-aware gradient representations for targeted instruction tuning (Xia et al., 2024); and PRISM, which uses a preference-aware target direction (Lin et al., 2026). Although these methods differ in how they estimate importance, each ultimately provides signals that can be used to rank or select training examples.

Attribution-based data selection. Training-data attribution is commonly used for subset selection by reducing the available signals to one scalar score per example and selecting the top-m candidates (Park et al., 2023; Xia et al., 2024; Wang et al., 2025). When scores are computed for multiple validation examples or targets, they are usually averaged before selection, giving each candidate a fixed priority that does not depend on the examples already chosen. This can produce redundant or poorly balanced subsets because several highly ranked examples may support the same labels, validation examples, or task behaviors, while lower-ranked examples may provide complementary information. Prior work reports that influence functions, TRAK, LESS, Data Shapley, and related methods can underperform Random at small selection budgets (Liu et al., 2024; Wang et al., 2025). Joint gradient-based and interaction-aware methods further show that independent rankings can overlook redundancy and complementarity among selected examples (Heo et al., 2026). These findings motivate studying the selection rule separately from the attribution estimator: useful attribution signals do not guarantee a collectively useful subset by themselves.

NASH and nonlinear aggregation. The closest prior work is NASH, which addresses the limitations of scalar top-m selection for Data Shapley and semivalues by decomposing an aggregate utility into validation-level components and applying concave aggregation to their accumulated values (Tian et al., 2026). Although NASH notes that this idea may be compatible with other data values, its formal notion of informativeness, component construction, and primary empirical evaluation are centered on coalition-based Shapley and semivalue scores. NADA broadens this selection principle to general component attribution matrices, including local sensitivity estimates from influence functions, trajectory-based scores from TracIn, and finite removal effects from LOO (Koh & Liang, 2017; Pruthi et al., 2020; Hammoudeh & Lowd, 2024).

Diversity-aware subset selection. Diversity-aware methods use submodular coverage, facility-location, core-set, or gradient-matching objectives to select representative subsets and reduce redundancy (Wei et al., 2015; Sener & Savarese, 2018; Mirzasoleiman et al., 2020; Killamsetty et al., 2021). These approaches typically define redundancy through similarity or matching in the input, feature, embedding, or gradient space. NADA instead defines redundancy through downstream attribution effects: examples are redundant when they support the same evaluation components and complementary when they support different ones. Thus, NADA provides a target-aware notion of diversity inherited from the attribution matrix rather than from a predefined similarity metric.

3 Problem Setup

3.1 Data Selection Objective

Let $N = \{1, \dots, n\}$ index the feasible training examples $\{z_i\}_{i \in N}$, and $\mathcal{M} \subseteq N$ be a selected subset of size m. We write $\mathcal{S}_m = \{\mathcal{M} \subseteq N : |\mathcal{M}| = m\}$ for the family of feasible size-m subsets. Let $\mathcal{A}$ be a fixed training or fine-tuning procedure, and let $\theta_{\mathcal{M}} = \mathcal{A}(\{z_i : i \in \mathcal{M}\})$ denote the model parameters returned after applying $\mathcal{A}$ to subset $\mathcal{M}$. For a utility function $u : 2^N \to \mathbb{R}$, we define $u(\mathcal{M}) = U(\theta_{\mathcal{M}})$, where U evaluates the model trained on $\mathcal{M}$, for example through validation accuracy, negative validation loss, or target-task performance. If training is stochastic, $u(\mathcal{M})$ may denote either performance under a fixed training realization or expected performance over training randomness. Therefore, the **budgeted data selection problem** is

$$\mathcal{M}^\star \in \arg\max_{\mathcal{M} \in \mathcal{S}_m} u(\mathcal{M}). \tag{1}$$

Directly optimizing Eq. (1) is generally intractable because evaluating a candidate subset requires training or fine-tuning a model, and the number of feasible subsets grows combinatorially with n.

3.2 Evaluation Components

Evaluation objectives often aggregate performance over validation examples, subgroups, prompts, or target behaviors. We express support for these components as

$$u(\mathcal{M}) = \sum_{c \in C} w_c u_c(\mathcal{M}), \qquad w_c \geq 0, \qquad \sum_{c \in C} w_c = 1, \tag{2}$$

where C indexes components with true utilities $u_c(\mathcal{M})$ and weights w_c.

For example, consider V to be a validation set containing examples $z_v = (x_v, y_v)$. For validation accuracy, we may take $C = V$, $w_v = 1/|V|$, and $u_v(\mathcal{M}) = \mathbf{1}[\widehat{y}_{\theta_\mathcal{M}}(x_v) = y_v]$. For loss-based classification or regression, one may instead use $u_v(\mathcal{M}) = -\ell(z_v, \theta_\mathcal{M})$. More generally, a component may represent a validation group, label, subpopulation, prompt, or target behavior. Eq. (2) decomposes the original objective without changing it.

3.3 Attribution Signals

Computing the true component utilities $u_c(\mathcal{M})$ for many candidate subsets is still prohibitively expensive. Attribution methods provide tractable proxies by estimating how each training example affects each evaluation component. Let a denote an attribution method and let

$$A^a \in \mathbb{R}^{n\times|C|}, \qquad A^a_{i,c} \in \mathbb{R} \tag{3}$$

be its attribution matrix, where $A^a_{i,c}$ estimates the effect of training example i on component c.

We orient all scores so that larger values indicate more helpful effects. Thus, $A^a_{i,c} > 0$ suggests that example i helps component c, whereas $A^a_{i,c} < 0$ suggests a harmful effect. Each row $A^a_{i,\cdot}$ therefore describes the contribution profile of one candidate across the evaluation objective.

Given a selected subset $\mathcal{M}$, its accumulated attribution score on component c is $\hat{u}^a_c(\mathcal{M}) := \sum_{i\in\mathcal{M}} A^a_{i,c}$. The attribution-based proxy $\hat{u}^a_c(\mathcal{M})$ need not equal the true component utility $u_c(\mathcal{M})$.

3.4 Scalar top-score selection and its limitation

The conventional top-m rule first compresses the component profile of each example into one scalar: $s^a_i = \sum_{c\in C} w_c A^a_{i,c}$. It then selects the m examples with the largest scores. Equivalently,

$$\mathcal{M}^a_{\mathrm{Top}} \in \arg\max_{\mathcal{M}\in\mathcal{S}_m} \sum_{i\in M} s^a_i = \arg\max_{\mathcal{M}\in\mathcal{S}_m} \sum_{c\in C} w_c \hat{u}^a_c(\mathcal{M}). \tag{4}$$

The objective in Eq. (4) is modular. For any current subset $\mathcal{M}$, the marginal value of adding example $i \notin \mathcal{M}$ is s^a_i. Hence, the priority of example i is fixed before selection begins and is independent of the examples already chosen.

This fixed ranking cannot account for redundancy or complementarity. If several high-scoring examples support the same components, scalar top-m selection assigns each candidate its original score even after the components it supports are well represented. Conversely, an example that supports an underrepresented component may remain low-ranked because its total scalar score is smaller. Thus, individually valuable examples may form redundant or poorly balanced subsets.

4 Attribution-Informativeness and the NADA Framework

Building on Section 3, we separate the informativeness of attribution scores from the selection rule applied to them. We first formalize when accumulated component scores predict their corresponding utilities, then introduce nonlinear aggregation to account for redundancy during subset construction. We then give greedy optimization and influence-function, TracIn, and LOO instantiations.

4.1 Attribution-Informative Components

The matrix A^a in Eq. (3) is only a proxy for the effects of selecting and retraining on a subset. An arbitrary matrix can be aggregated nonlinearly, but doing so is meaningful only if its accumulated scores are related to the true component utilities. We therefore separate the quality of the attribution signal from the quality of the selection rule.

Definition 1 (Attribution-informativeness)**.** *Fix an attribution method a, a component c, and a selection budget m. Hold the attribution matrix A^a, the training procedure $\mathcal{A}$, and the component utility u_c fixed. We say that $A^a_{\cdot,c}$ is $(\varepsilon_c, \delta_c)$-attribution-informative for u_c at budget m if there exists a nondecreasing calibration map $f_{c,m} : \mathbb{R} \to \mathbb{R}$ such that*

$$\Pr_{\mathcal{M}\sim\mathrm{Unif.}(\mathcal{S}_m)} [|u_c(\mathcal{M}) - f_{c,m}(\hat{u}^a_c(\mathcal{M}))| \le \varepsilon_c] \ge 1 - \delta_c. \tag{5}$$

Here, ε_c bounds the prediction error and δ_c the failure probability. The accumulated score need not equal true utility; it must predict that utility through a nondecreasing calibration map for most size-m subsets. This map preserves the helpful-score ordering, accommodates method- and budget-dependent score scales, and need not be explicitly estimated by NADA. We call the full matrix attribution-informative when the condition holds for every positively weighted component.

Definition 1 recovers the Shapley-informativeness notion of NASH as a special case (Tian et al., 2026). Setting $A^{\text{Shapley}}_{i,c} = \phi_i(u_c)$ gives $\hat{u}^{\text{Shapley}}_c(\mathcal{M}) = \sum_{i\in\mathcal{M}} \phi_i(u_c)$ in Eq. (5). Under the same budget, error tolerances, and nondecreasing calibration map, it recovers the Shapley-informativeness condition of NASH. NADA allows general attribution scores in place of Shapley values while retaining the same subset-level utility-prediction criterion.

4.2 Nonlinear Aggregation Objective

Definition 1 states when the component scores contain useful information. The remaining question is how to combine that information while constructing a subset. If the calibration maps were known, a natural attribution-based surrogate for the true utility would be $\widetilde{u}_a(\mathcal{M}) = \sum_{c\in C} w_c f_{c,m}(\hat{u}^a_c(\mathcal{M}))$. In practice, estimating a separate calibration map for every component and budget can be statistically and computationally expensive. Moreover, nondecreasing calibration maps need not be concave, so calibration alone does not impose diminishing returns.

The scalar top-m objective in Section 3.4 uses the linear aggregation $\sum_{c\in C} w_c \hat{u}^a_c(\mathcal{M})$. Because this objective is modular, it assigns every candidate a fixed marginal value. NADA replaces this linear aggregation with an increasing concave function for each component.

Definition 2 (NADA). *Given a component attribution matrix A^a, component weights w_c, and increasing concave functions $F_c : \mathbb{R} \to \mathbb{R}$, let*

$$J_a(\mathcal{M}) = \sum_{c\in C} w_c F_c(\hat{u}^a_c(\mathcal{M})) = \sum_{c\in C} w_c F_c\left(\sum_{i\in\mathcal{M}} A^a_{i,c}\right). \tag{6}$$

NADA *selects* $\mathcal{M}^a_{\text{NADA}} \in \arg\max_{\mathcal{M}\in\mathcal{S}_m} J_a(\mathcal{M})$.

Calibration maps $f_{c,m}$ describe whether the attribution scores are predictive of utility, while F_c defines how those scores are used during selection. NADA does not assume $F_c = f_{c,m}$.

Why nonlinear aggregation? The two properties of F_c serve distinct purposes. Monotonicity preserves the helpful-score ordering assumed by attribution-informativeness. Concavity introduces diminishing returns: increasing a component with a small accumulated score has greater marginal value than further increasing a component that already has a large score.

Algorithm 1 NADA selection

1: **Input:** pool N, components C, matrix A^a, weights w, budget m, functions $\{F_c\}_{c\in C}$
2: $\mathcal{M} \leftarrow \emptyset$ and $r_c \leftarrow 0$ for all $c \in C$
3: **for** $t = 1, \ldots, m$ **do**
4: $\quad i^\star \leftarrow \arg\max_{i\in N\setminus\mathcal{M}} \sum_{c\in C} w_c \left[F_c(r_c + A^a_{i,c}) - F_c(r_c)\right]$
5: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{i^\star\}$
6: $\quad r_c \leftarrow r_c + A^a_{i^\star,c}$ for all $c \in C$
7: **end for**
8: **return** $\mathcal{M}$

The scalar top-m rule is recovered as the linear special case: if $F_c(x) = x$ for every c, then $J_a(\mathcal{M}) = \sum_c w_c \hat{u}^a_c(\mathcal{M}) = \sum_{i\in\mathcal{M}} s^a_i$, which is exactly the objective in Eq. (4). Thus, NADA leaves the underlying attribution estimator unchanged and replaces only its linear aggregation layer.

In the main experiments, we use the shared exponential aggregation $F_\lambda(x) = 1 - \exp(-\lambda x)$, where $\lambda > 0$ controls the rate of saturation. Larger values of λ impose stronger diminishing returns on components with large accumulated scores. The equivalent form $-\exp(-\lambda x)$ differs only by an additive constant and therefore gives the same selected subset.

4.3 GREEDY OPTIMIZATION

Maximizing Eq. (6) is combinatorial. For a current subset $\mathcal{M}$, let $r_c(\mathcal{M}) = \sum_{j\in\mathcal{M}} A^a_{j,c}$ denote the current accumulated score of component c. The marginal gain of adding candidate $i \notin \mathcal{M}$ is

$$\Delta_i(\mathcal{M}) = J_a(\mathcal{M}\cup\{i\}) - J_a(\mathcal{M}) = \sum_{c\in C} w_c \left[F_c\big(r_c(\mathcal{M}) + A^a_{i,c}\big) - F_c(r_c(\mathcal{M}))\right]. \tag{7}$$

At each round, NADA adds the highest-gain candidate and updates the component scores.

Adaptive component weighting. For a small component contribution, a first-order expansion of Eq. (7) gives $\Delta_i(\mathcal{M}) \approx \sum_{c\in C} w_c F_c'(r_c(\mathcal{M}))\, A^a_{i,c}$. Because F_c is concave, $F_c'(r_c)$ decreases as r_c increases. The effective weight $w_c F_c'(r_c(\mathcal{M}))$ is therefore reduced for components already well supported by the current subset and remains large for underrepresented components. This adaptive weighting is the mechanism by which NADA accounts for coverage and redundancy.

Proposition 1 (Nonnegative component scores)**.** *Suppose that $A^a_{i,c} \geq 0$ for every i, c, and that each F_c is nondecreasing and concave. Then J_a is a monotone submodular set function. After subtracting the constant $J_a(\emptyset)$, greedy selection under the cardinality constraint $|\mathcal{M}| = m$ achieves the standard $(1 - 1/e)$-approximation guarantee.*

The result follows from the standard concave-over-modular construction and the classical greedy guarantee (Nemhauser et al., 1978). A proof is provided in Appendix B.4. With signed attribution entries, both monotonicity and submodularity can fail. For example, with one component, $F(x) = 1 - e^{-x}$, $A_1 = -1$, and $A_2 = 1$, the marginal gain of adding item 2 is $1 - e^{-1}$ at the empty set but $e - 1$ at $\{1\}$, violating diminishing returns. Consequently, we do not claim this approximation guarantee for general signed-score runs. Because Algorithm 1 selects exactly m examples, it may also include steps with negative marginal gain.

4.4 NADA INSTANTIATIONS

NADA is agnostic to how the attribution matrix is constructed. In this paper, we consider three attribution families: influence functions, TracIn, and leave-one-out. Each supplies component scores to the same Top/NADA selection comparison; formal definitions are provided in Appendix C.

Influence Function. Influence functions (Koh & Liang, 2017) estimate how infinitesimally up-weighting a training example changes component loss around the full-data model. We use utility-oriented local sensitivity scores, whose accuracy depends on the curvature approximation.

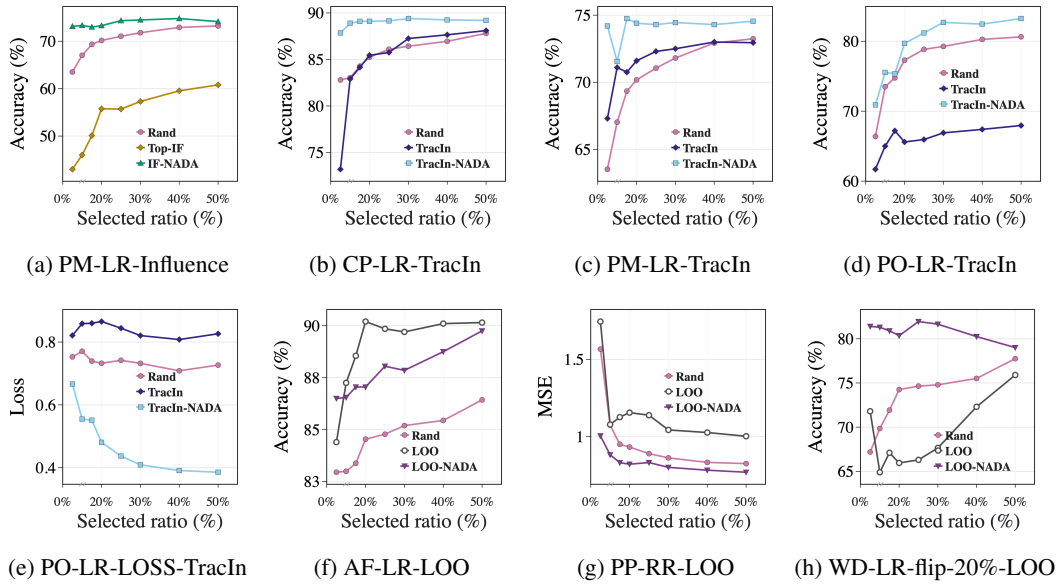
TracIn. TracIn (Pruthi et al., 2020) accumulates gradient alignment between training examples and evaluation components across saved checkpoints. These trajectory-based scores estimate helpful loss-reducing updates without requiring inverse-Hessian computation.

Leave One Out. Leave-one-out (Hammoudeh & Lowd, 2024) measures the change in component utility after removing one training example from the full pool. It provides a direct finite-removal baseline but requires retraining or reevaluation for each removal.

For all three methods, attribution-informativeness concerns accumulated subset scores rather than individual entries; the NADA objective and greedy selection procedure remain unchanged.

5 EXPERIMENTS

In this section, we evaluate the effectiveness of NADA in improving the subset-selection stage of training-data attribution while holding the underlying attribution estimator fixed. We compare each NADA variant with the corresponding scalar top-m baseline using exactly the same attribution matrix, and examine whether the resulting improvements are consistent across attribution families, downstream models, tasks, and selection budgets. We further study how the benefit of nonlinear aggregation changes when the feasible training pool becomes increasingly heterogeneous or corrupted.



(a) PM-LR-Influence (b) CP-LR-TracIn (c) PM-LR-TracIn (d) PO-LR-TracIn

(e) PO-LR-LOSS-TracIn (f) AF-LR-LOO (g) PP-RR-LOO (h) WD-LR-flip-20%-LOO

Figure 1: Representative selection curves on logistic-regression (LR) and ridge-regression (RR) benchmarks. Each panel compares Random, scalar Top, and NADA; the Top/NADA pair uses the same attribution matrix.

We instantiate these paired baselines using influence functions (Koh & Liang, 2017), TracIn (Pruthi et al., 2020), and leave-one-out attribution (Hammoudeh & Lowd, 2024). We use $F_\lambda(x) = 1 - \exp(-\lambda x), \lambda = 2.5$, global max-absolute normalization of the component matrix, and no additional scalar-quality mixing. We summarize each performance curve by its trapezoidal area over the actual selection ratios, normalized by the evaluated ratio range. We refer to this quantity as curve-average performance. Aggregate results are arithmetic means of these curve summaries across the specified conditions and attribution families. Higher values are better for accuracy; lower values are better for validation loss and MSE. Additional hyperparameters, per-budget results, and experimental results are provided in Appendix C.

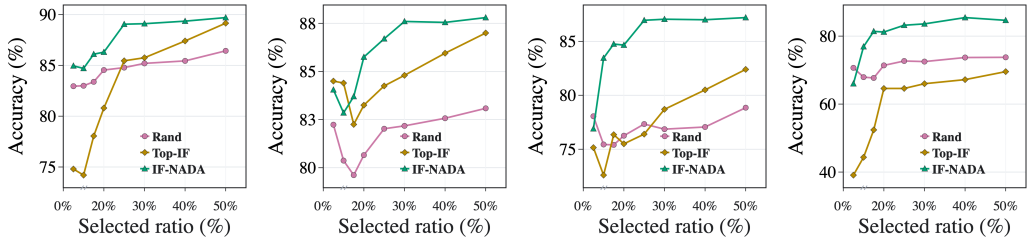
5.1 Standard LR/RR Benchmarks

Setup. We evaluate NADA on binary logistic-regression (LR) and ridge-regression (RR) data-selection benchmarks obtained from OpenML (Vanschoren et al., 2014). The LR benchmarks comprise Wind (WD) (Vanschoren, 2014d), Pol (PO) (Vanschoren, 2014b), Phoneme (PM) (Grin, 2022), CPU (CP) (Vanschoren, 2014a), 2DPlanes (TP) (Vanschoren, 2014c), APS Failure (AF) (Gijsbers, 2018), Vehicle SensIT (VE) (Duarte & Hu, 2004), and Credit Card Fraud Detection (CC) (Dal Pozzolo et al., 2015). The RR benchmarks comprise California Housing (CH) (Gazioglu, 2022), Physicochemical Protein (PP) (Fischer, 2022b), and Auction Verification (AU) (Fischer, 2022a). Data processing uses scikit-learn (Pedregosa et al., 2018).

Each benchmark uses $n = 200$ feasible training examples and up to 2000 validation examples. We evaluate selected-data ratios $\{5, 10, 15, 20, 25, 30, 40, 50\}\%$. For classification, we consider clean feasible pools and pools with 10%, 20%, or 30% randomly corrupted training labels. We evaluate influence functions on the LR settings and TracIn and LOO on all LR/RR settings. Each Top/NADA pair shares the attribution matrix, weights, training split, and evaluation procedure.

Results. Figure 1 shows representative curves. Across influence functions, TracIn, and LOO, NADA improves the corresponding scalar Top baseline in $134/144$ matched curve-level comparisons. The mean relative changes in curve-average accuracy, validation loss, and MSE are $+15.92\%$, -27.92%, and -33.89%, respectively. Relative changes are computed against the corresponding Top baseline for each matched curve and then averaged within each metric.

Effect of training-data corruption. Across the reported LR/RR corruption conditions, the pooled mean relative improvement over Random rises from 1.35% on clean pools to 7.68%, 14.92%, and 22.19% at the three nonzero corruption levels. This summary averages relative accuracy increases and relative loss or MSE decreases; it is a descriptive cross-metric summary rather than a common absolute effect size. A larger relative advantage does not imply that NADA itself improves as corruption increases. For example, on AF-LR, IF-NADA's curve-average accuracy decreases from 87.85% to 81.85% between clean and 30% corrupted pools, while Random decreases from 84.70% to 71.66%. On this benchmark, IF-NADA degrades less than Random. Improved coverage or avoidance of harmful examples is a possible explanation, but the reported performance curves do not directly measure that mechanism.



(a) AF-LR-Clean (b) AF-LR-flip-10% (c) AF-LR-flip-20% (d) AF-LR-flip-30%

Figure 2: Influence-based selection on AF-LR under clean, 10%, 20%, and 30% training-label corruption, comparing Random, Top-Influence, and Influence-NADA.

Figure 2 illustrates this trend for Influence-NADA on AF-LR. As label corruption increases, the component scores provide a clearer distinction between helpful and harmful training examples, and nonlinear aggregation more reliably avoids repeatedly selecting examples that damage multiple validation components.

5.2 Prompt Fine-Tuning Benchmarks

Setup. We evaluate prompt-based data selection using BERT (Devlin et al., 2019), Llama-2-7B (Touvron et al., 2023), and GPT-OSS-20B (OpenAI et al., 2025). The task suite includes Amazon Polarity and Yelp Polarity (Zhang et al., 2016), ANLI (Nie et al., 2020), MNLI (Williams et al., 2018), SST-2 and RTE (Wang et al., 2019), and Rotten Tomatoes Movie Review (Pang & Lee, 2005). Together, these form 12 task-model combinations and 23 clean or label-noisy experimental conditions. We evaluate selected-data ratios $\{2, 6, 10, 20, 30, 40, 50\}\%$. For each condition, we compare Random, Top-Influence versus Influence-NADA, Top-TracIn versus TracIn-NADA, and Top-LOO versus LOO-NADA. Each Top/NADA pair uses the same matrix and fine-tuning configuration.

Results. Table 1 reports the complete task-level results. Across 23 task-model and noise conditions and three attribution families, NADA improves its corresponding Top baseline on all 69 curve-level comparisons. The mean curve-average improvement is +49.29%. The improvement also holds across individual selection budgets: NADA wins 447 of 483 matched benchmark-budget comparisons, ties 15, and loses 21. TracIn-NADA gives the largest mean improvement over its Top baseline (+54.12%), followed by LOO-NADA (+49.09%) and Influence-NADA (+44.64%). The gains span BERT-, Llama-, and GPT-based tasks.

Discussion. The fine-tuning results should be interpreted primarily as an evaluation of the subset-selection layer rather than as a claim that NADA universally dominates uninformed sampling. Averaged over the three attribution families, scalar Top selection obtains 51.92% curve-average accuracy, whereas NADA obtains 75.68%. Random selection obtains 75.50%. Thus NADA nearly closes the large gap between scalar ranking and Random, but does not consistently surpass Random.

This distinction supports the central motivation of NADA. The component attribution matrices contain enough structure to construct substantially better subsets than their scalar row sums, but they do not always provide enough additional quality information to outperform representative random

Table 1: Detailed prompt fine-tuning results across all 23 task-model and noise conditions. Values are curve-average validation accuracy (%). Random is averaged over five selection seeds. Noise denotes the proportion of labels corrupted in the feasible training pool. Higher is better.

Task	Noise	Random	Influence Top / NADA	TracIn Top / NADA	LOO Top / NADA
BERT masked language models					
AMAZON-BT	Clean	85.2	50.2 / **85.6**	50.2 / **85.3**	50.1 / **85.8**
AMAZON-BT	10%	83.3	50.2 / **82.6**	50.2 / **82.4**	50.1 / **83.2**
AMAZON-BT	20%	80.2	50.2 / **78.8**	50.2 / **78.8**	50.1 / **79.8**
MR-BT	Clean	76.1	67.0 / **75.2**	60.3 / **75.3**	50.0 / **75.5**
MR-BT	10%	73.3	68.9 / **74.1**	63.0 / **72.8**	50.2 / **73.9**
MR-BT	20%	64.5	70.8 / **72.0**	66.7 / **71.1**	50.7 / **72.7**
YELP-BT	Clean	86.0	52.2 / **86.3**	52.2 / **86.2**	52.2 / **86.1**
YELP-BT	10%	83.7	52.2 / **81.8**	52.2 / **81.7**	52.2 / **82.2**
Llama-style causal language models					
MNLI-LM	Clean	54.1	32.1 / **54.8**	32.7 / **52.0**	45.4 / **55.8**
MNLI-LM	10%	51.4	32.0 / **52.3**	32.4 / **48.6**	45.0 / **53.1**
MNLI-LM	20%	48.6	31.9 / **49.9**	31.9 / **46.5**	44.7 / **50.8**
MR-LM	Clean	90.1	83.8 / **89.7**	83.9 / **89.9**	51.0 / **89.9**
RT-LM	Clean	54.4	51.2 / **55.3**	51.2 / **54.9**	51.2 / **55.5**
RT-LM	10%	54.3	51.2 / **54.5**	51.2 / **54.3**	51.2 / **55.2**
RT-LM	20%	53.9	51.2 / **53.4**	51.3 / **54.0**	51.2 / **53.8**
SST2-LM	Clean	93.4	51.4 / **93.8**	51.5 / **93.7**	51.4 / **93.4**
SST2-LM	10%	92.7	51.4 / **92.7**	51.4 / **92.5**	51.5 / **92.7**
SST2-LM	20%	91.5	51.4 / **91.7**	51.4 / **91.8**	51.4 / **91.6**
GPT-style causal language models					
AMAZON-GPT	Clean	95.6	83.1 / **95.5**	94.2 / **95.7**	54.5 / **95.3**
ANLI-GPT	Clean	68.1	38.5 / **68.6**	56.4 / **68.1**	39.5 / **68.4**
MNLI-GPT	Clean	81.2	38.0 / **80.0**	41.1 / **79.8**	37.9 / **80.4**
RT-GPT	Clean	80.3	53.7 / **80.9**	53.7 / **79.4**	53.9 / **81.0**
SST2-GPT	Clean	94.1	51.6 / **94.0**	51.5 / **93.7**	51.7 / **93.8**
Overall	–	75.5	52.8 / **75.8**	53.5 / **75.2**	49.4 / **76.1**

sampling. Nonlinear aggregation therefore acts as a robust correction to scalar attribution collapse, rather than a guarantee of superiority to Random on every fine-tuning task.

6 Conclusion

We introduced NADA, a general nonlinear subset-selection layer for component-level data attribution. Whereas conventional top-m selection first compresses each training example into one scalar score, NADA preserves the full component attribution matrix and applies an increasing concave aggregation to account for coverage and redundancy among selected examples. We formalized when accumulated component scores are predictive of component utility through attribution-informativeness and instantiated the same selection framework with influence functions, TracIn trajectory attribution, and leave-one-out values.

Across the LR/RR benchmarks, NADA improves its corresponding top-score baseline in $134/144$ matched comparisons. Its advantage over Random becomes increasingly clear as the feasible pool contains more corrupted or heterogeneous data. On prompt fine-tuning, NADA improves all 69 matched Top curves by 23.76 percentage points on average, while performing approximately on par with Random overall. These findings support nonlinear aggregation for data selection while motivating more informative components and scalable selection methods.

7 AI DISCLOSURE

We used ChatGPT to assist with improving the clarity, grammar, and readability of the manuscript; searching for and identifying relevant literature; and drafting portions of Section 1, 3, and 4. All AI-assisted text was reviewed and revised by the authors. AI-suggested references included in the manuscript were checked against the original publications for accuracy and relevance. We did not use AI tools for research ideation or execution, generating synthetic datasets, or proving mathematical claims. The authors take full responsibility for the final manuscript, including all text, claims, and citations produced with AI assistance.

REFERENCES

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A survey on data selection for language models. *Transactions on Machine Learning Research*, 2024.

Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pp. 159–166. IEEE, 2015. doi: 10.1109/SSCI.2015.33. URL https://doi.org/10.1109/SSCI.2015.33.

Junwei Deng, Weijing Tang, and Jiaqi W. Ma. A versatile influence function for data attribution with non-decomposable loss. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004. doi: 10.1016/j.jpdc.2004.03.020. URL https://doi.org/10.1016/j.jpdc.2004.03.020.

Sebastian Fischer. Auction verification. OpenML dataset, 2022a. URL https://www.openml.org/d/44958. OpenML data ID: 44958.

Sebastian Fischer. Physiochemical protein. OpenML dataset, 2022b. URL https://www.openml.org/d/44963. OpenML data ID: 44963.

Mine Gazioglu. California housing. OpenML dataset, 2022. URL https://www.openml.org/d/43939. OpenML data ID: 43939.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 2019.

Pieter Gijsbers. APSFailure. OpenML dataset, 2018. URL https://www.openml.org/d/41138. OpenML data ID: 41138.

Leo Grin. Phoneme. OpenML dataset, 2022. URL https://www.openml.org/d/43973. OpenML data ID: 43973.

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

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

Andrew Ilyas and Logan Engstrom. MAGIC: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Hynes, Nezihe Merve Gurel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1167–1176. PMLR, 2019.

Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. GRAD-MATCH: Gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5464–5474. PMLR, 2021.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017.

Yongchan Kwon and James Zou. Beta shapley: A unified and noise-reduced data valuation framework for machine learning. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8780–8802. PMLR, 2022.

Qihao Lin, Guanxu Chen, Dongrui Liu, and Jing Shao. PRISM: Preference-aware influence-function-based data selection method for efficient fine-tuning. *arXiv preprint arXiv:2605.21422*, 2026.

Tianyang Liu, Fei Wang, and Muhao Chen. Rethinking tabular data understanding with large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 450–482, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.26. URL https://aclanthology.org/2024.naacl-long.26/.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6950–6960. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/mirzasoleiman20a.html.

Bruno Mlodozeniec, Isaac Reid, Sam Power, David S. Krueger, Murat A. Erdogdu, Richard E. Turner, and Roger B. Grosse. Distributional training data attribution: What do influence functions sample? *arXiv preprint arXiv:2506.12965*, 2025.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding, 2020. URL https://arxiv.org/abs/1910.14599.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo

Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, 2005. URL https://arxiv.org/abs/cs/0506075.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113, 2023.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018. URL https://arxiv.org/abs/1201.0490.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton, NJ, 1953.

Xiao Tian, Jue Fan, Rachael Hwee Ling Sim, Zixuan Wang, Nancy F. Chen, and Bryan Kian Hsiang Low. Is data shapley not better than random in data selection? ask NASH. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Joaquin Vanschoren. CPU. OpenML dataset, 2014a. URL https://www.openml.org/d/761. OpenML data ID: 761.

Joaquin Vanschoren. Pol. OpenML dataset, 2014b. URL https://www.openml.org/d/722. OpenML data ID: 722.

Joaquin Vanschoren. 2DPlanes. OpenML dataset, 2014c. URL https://www.openml.org/d/727. OpenML data ID: 727.

Joaquin Vanschoren. Wind. OpenML dataset, 2014d. URL https://www.openml.org/d/847. OpenML data ID: 847.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, June 2014. ISSN 1931-0153. doi: 10.1145/2641190.2641198. URL http://dx.doi.org/10.1145/2641190.2641198.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL https://arxiv.org/abs/1804.07461.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6388–6421. PMLR, 2023.

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

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1954–1963, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/wei15.html.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference, 2018. URL https://arxiv.org/abs/1704.05426.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 2024.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K. Ravikumar. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016. URL https://arxiv.org/abs/1509.01626.

A NOTATION AND CONVENTIONS

Table 2 summarizes the notation used throughout the paper. We use $c \in C$ for a generic component.

Table 2: Summary of notation used in NADA.

Symbol	Definition
	Constants & Variables
δ_c	Maximum permitted attribution-informativeness failure rate
ε_c	Maximum permitted component-utility approximation error
λ	Saturation control hyperparameter
$\theta_{\mathcal{M}}$	Parameters returned by training or fine-tuning on $\mathcal{M}$
$A^a_{i,c}$	Attribution of training example i to component c produced by method a
a	Attribution-method identifier
C	Set of components
c	Individual component
i	Candidate-training example index
$i^\star$	Candidate selected by NADA
j	Candidate training example index
$\mathcal{M}$	Selected training subset
$\mathcal{M}^\star$	Optimal size-m subset under the true utility
$\mathcal{M}^a_{\mathrm{NADA}}$	Subset selected by NADA using method a
$\mathcal{M}^a_{\mathrm{Top}}$	Subset containing the m examples with largest scalar scores s^a_i
m	Number of selected examples
N	Feasible training pool
S_m	Collection of all feasible subsets $\mathcal{M} \subseteq N$ of size m
s^a_i	Scalar top-selection score on example i under method a
V	Validation set
v	Validation example
w_c	Nonnegative importance weight for component c
z_i	Training example i
	Functions & Operators
2^N	Power set of N
$\Delta_i(\cdot)$	Marginal NADA gain from adding candidate i
$\mathcal{A}(\cdot)$	Fixed training or fine-tuning procedure
$F_c(\cdot)$	Increasing concave aggregation function for component c
$F_\tau(\cdot)$	Cumulative probability of $\tau \leq \cdot$
$F_\lambda(\cdot)$	The exponential aggregation function
$f_{c,m}(\cdot)$	Mapping converting accumulated attribution score into a prediction of true component utility for component c
$J_a(\cdot)$	NADA objective function of method a
$\phi_i(\cdot)$	The Shapley value of datum / example i
$r_c(\cdot)$	Current accumulated score of component c
$U(\cdot)$	Evaluation function applied to a model
$u(\cdot)$	Utility of the set
$\hat{u}^a_c(\cdot)$	Accumulated attribution score for component c under method a
$\widetilde{u}_a(\cdot)$	Attribution-based surrogate for the true utility
$u_V(\cdot)$	Validation accuracy on validation set V as utility function
$u_v(\cdot)$	Prediction correctness on validation datum via utility function

Conventions. All utilities are oriented so that larger values are better; loss-based utilities are therefore defined using negative loss. Attribution scores follow the same convention: positive values indicate estimated helpful effects on the corresponding evaluation component. We distinguish the training procedure $\mathcal{A}$ from the attribution matrix A^a, the calibration map $f_{c,m}$ from the selection function F_c, and the aggregation parameter λ from the curvature damping parameter ρ.

B Additional Theory and Proofs

Throughout this appendix, the attribution matrix A^a, component weights, and utility functions are fixed. Unless stated otherwise, all probabilities and expectations are taken over $M \sim \mathrm{Unif}(\mathcal{S}_m)$, where $\mathcal{S}_m = \{\mathcal{M} \subseteq N : |\mathcal{M}| = m\}$ and $1 \le m \le n$. Write

$$r_c(\mathcal{M}) = \widehat{u}^a_c(\mathcal{M}) = \sum_{i \in \mathcal{M}} A^a_{i,c}, \qquad \mathcal{C}_+ = \{c \in \mathcal{C} : w_c > 0\}.$$

We assume $w_c \ge 0$ and $\sum_{c \in \mathcal{C}} w_c = 1$. The binary-utility results below apply, for example, to validation correctness under a fixed training realization. Expected correctness over training randomness generally lies in $[0, 1]$ and is not necessarily binary.

B.1 Binary Threshold Calibration

Fix a component c with $u_c(M) \in \{0, 1\}$ and a budget m. For a threshold τ_c (which may depend on m), define

$$h_c(\mathcal{M}) = \mathbf{1}\{r_c(\mathcal{M}) \ge \tau_c\}.$$

The map $x \mapsto \mathbf{1}\{x \ge \tau_c\}$ is nondecreasing. Thus, attribution-informativeness with $\varepsilon_c = 0$ and this calibration map is exactly the condition

$$\Pr\{h_c(\mathcal{M}) \ne u_c(\mathcal{M})\} \le \delta_c, \qquad 0 \le \delta_c \le 1. \tag{8}$$

This condition concerns prediction of component utility by a thresholded accumulated score. It does not identify the calibration map with the concave aggregation function used by NADA.

B.2 From Threshold Recovery to Component Coverage

Proposition B.1 (Accuracy of the threshold-coverage surrogate)**.** *Suppose* $u_c(M) \in \{0, 1\}$ *and* (8) *holds for each* $c \in \mathcal{C}_+$*. Define*

$$u(M) = \sum_{c \in \mathcal{C}_+} w_c u_c(\mathcal{M}), \qquad \widetilde{u}^a(\mathcal{M}) = \sum_{c \in \mathcal{C}_+} w_c h_c(\mathcal{M}).$$

Then

$$\mathbb{E}\big[|u(\mathcal{M}) - \widetilde{u}^a(\mathcal{M})|\big] \le \sum_{c \in \mathcal{C}_+} w_c \delta_c. \tag{9}$$

Moreover,

$$\Pr\{h_c(\mathcal{M}) = u_c(\mathcal{M}) \textit{ for all } c \in \mathcal{C}_+\} \ge \max\left\{0, 1 - \sum_{c \in \mathcal{C}_+} \delta_c\right\}. \tag{10}$$

No independence assumption across components is required.

Proof. Let $e_c(\mathcal{M}) = |u_c(\mathcal{M}) - h_c(\mathcal{M})|$. Since both terms are binary, $e_c(\mathcal{M}) = \mathbf{1}\{u_c(\mathcal{M}) \ne h_c(\mathcal{M})\}$, so $\mathbb{E}[e_c(\mathcal{M})] \le \delta_c$. The triangle inequality and the nonnegativity of the weights imply

$$|u(\mathcal{M}) - \widetilde{u}^a(\mathcal{M})| \le \sum_{c \in \mathcal{C}_+} w_c e_c(\mathcal{M}).$$

Taking expectations proves (9). For simultaneous recovery, the union bound gives

$$\Pr\{\exists c \in \mathcal{C}_+ : h_c(\mathcal{M}) \ne u_c(\mathcal{M})\} \le \sum_{c \in \mathcal{C}_+} \Pr\{e_c(\mathcal{M}) = 1\} \le \sum_{c \in \mathcal{C}_+} \delta_c.$$

Taking complements and using the nonnegativity of probability proves (10). □

Proposition B.1 establishes accuracy of the calibrated surrogate on uniformly sampled size-m subsets. It does not, by itself, bound the error at a subset selected by optimizing that surrogate, nor does it establish that the NADA objective J^a approximates u. These distinctions are discussed in Section B.5.

B.3 A Conditional Separation Criterion

The following result gives a sufficient condition for binary threshold calibration. Although we state it for influence scores, its proof applies to any fixed attribution matrix satisfying the same assumptions.

Proposition B.2 (Threshold recovery under conditional Bernstein tails)**.** *Fix a component c and let*

$$Y = u_c(\mathcal{M}) \in \{0,1\}, \qquad S = \sum_{i \in M} A^{\mathrm{IF}}_{i,c}, \qquad p_y = \Pr\{Y = y\}.$$

Assume $p_0, p_1 > 0$. For $y \in \{0,1\}$, define $\mu_y = \mathbb{E}[S \mid Y = y]$, and suppose $\gamma = \mu_1 - \mu_0 > 0$. Assume that there are constants $\sigma_y^2 > 0$ and $B_y \geq 0$ such that, for every $t > 0$,

$$\Pr\{|S - \mu_y| \geq t \mid Y = y\} \leq 2\exp\left(-\frac{t^2}{2\sigma_y^2 + 2B_y t/3}\right). \tag{11}$$

Set $\tau = (\mu_0 + \mu_1)/2$ and

$$q_y = \min\left\{1, 2\exp\left(-\frac{\gamma^2}{8\sigma_y^2 + (4/3)B_y\gamma}\right)\right\}.$$

Then the threshold predictor $h(S) = \mathbf{1}\{S \geq \tau\}$ satisfies $\Pr\{h(S) \neq Y\} \leq p_0 q_0 + p_1 q_1$. Consequently, this component is $(0, \delta_c)$-attribution-informative under the threshold map, with $\delta_c = p_0 q_0 + p_1 q_1$.

Proof. If $Y = 1$, an error requires $S < \tau = \mu_1 - \gamma/2$ and is therefore contained in $\{|S - \mu_1| \geq \gamma/2\}$. If $Y = 0$, an error requires $S \geq \tau = \mu_0 + \gamma/2$ and is contained in $\{|S - \mu_0| \geq \gamma/2\}$. Applying (11) at $t = \gamma/2$, and bounding each conditional probability by one, yields

$$\Pr\{h(S) \neq Y \mid Y = y\} \leq q_y.$$

The law of total probability now gives

$$\Pr\{h(S) \neq Y\} = \sum_{y \in \{0,1\}} p_y \Pr\{h(S) \neq Y \mid Y = y\} \leq p_0 q_0 + p_1 q_1.$$

□

Interpretation and assumptions. The bound is informative when separation is sufficiently large relative to the conditional tail parameters. Equation (11) is an assumption on the conditional distribution of the accumulated score; it is not derived here from the influence-function formula. Sampling a fixed-size subset introduces dependence among inclusion indicators, and conditioning on Y can introduce further dependence. Thus, an independent-summand Bernstein inequality cannot be invoked without additional justification. If one class has probability zero, a constant threshold predictor suffices on the finite score support; the two-class separation formulation is unnecessary.

B.4 Proof of the Nonnegative-Score Guarantee

We prove the nonnegative-score proposition in the main text. Assume $A^a_{i,c} \geq 0$ for all i, c, and that each F_c is finite-valued, nondecreasing, and concave on an interval containing $[0, \sum_{i \in N} A^a_{i,c}]$. Define

$$J^a(\mathcal{M}) = \sum_{c \in \mathcal{C}} w_c F_c(r_c(\mathcal{M})), \qquad G(\mathcal{M}) = J^a(\mathcal{M}) - J^a(\emptyset).$$

Then $G(\emptyset) = 0$. We show monotonicity, submodularity, and the greedy approximation bound separately.

Proof. Monotonicity. For any $\mathcal{M} \subseteq N$ and $i \notin \mathcal{M}$, the nonnegativity of $A^a_{i,c}$ and monotonicity of F_c give

$$F_c(r_c(\mathcal{M}) + A^a_{i,c}) - F_c(r_c(\mathcal{M})) \geq 0.$$

Summing with nonnegative weights proves $J^a(\mathcal{M} \cup \{i\}) \geq J^a(\mathcal{M})$. The same holds for G, so $G(\mathcal{M}) \geq 0$ for every $\mathcal{M}$.

Submodularity. Let $\mathcal{M} \subseteq L \subseteq N$ and $i \notin L$. Nonnegative entries imply $r_c(\mathcal{M}) \leq r_c(L)$. For a concave function F_c and any fixed $h \geq 0$, the increment $F_c(x+h) - F_c(x)$ is nonincreasing in x wherever it is defined. Applying this property with $h = A^a_{i,c}$ yields

$$F_c(r_c(\mathcal{M}) + A^a_{i,c}) - F_c(r_c(\mathcal{M})) \geq F_c(r_c(L) + A^a_{i,c}) - F_c(r_c(L)).$$

Summing over c proves the diminishing-returns inequality

$$J^a(\mathcal{M} \cup \{i\}) - J^a(\mathcal{M}) \geq J^a(L \cup \{i\}) - J^a(L).$$

Thus J^a and G are submodular. This argument does not require differentiability of F_c.

Greedy approximation. Let $\mathcal{M}_0 = \emptyset$, and let $\mathcal{M}_{t+1}$ be obtained by adding an element with maximum marginal gain to $\mathcal{M}_t$, for $0 \leq t < m$. Let $O \in \arg\max_{|\mathcal{M}|=m} G(\mathcal{M})$. By monotonicity and submodularity,

$$\begin{aligned} G(O) &\leq G(\mathcal{M}_t \cup O) \\ &\leq G(\mathcal{M}_t) + \sum_{i \in O \setminus \mathcal{M}_t} \big[G(\mathcal{M}_t \cup \{i\}) - G(\mathcal{M}_t)\big] \\ &\leq G(\mathcal{M}_t) + m\big[G(\mathcal{M}_{t+1}) - G(\mathcal{M}_t)\big]. \end{aligned}$$

Rearranging and iterating gives

$$G(O) - G(\mathcal{M}_m) \leq (1 - 1/m)^m G(O),$$

and hence

$$\begin{aligned} J^a(\mathcal{M}_m) - J^a(\emptyset) &\geq \big[1 - (1-1/m)^m\big]\big[J^a(O) - J^a(\emptyset)\big] \\ &\geq (1 - 1/e)\big[J^a(O) - J^a(\emptyset)\big]. \end{aligned} \tag{12}$$

This is the classical greedy bound of Nemhauser et al. (1978). □

The guarantee is stated for the normalized objective, since additive shifts need not preserve multiplicative approximation ratios. Equation (12) concerns J^a, not the true retraining utility u. If entries of A^a are signed, monotonicity and submodularity can both fail, and the preceding guarantee does not apply.

B.5 Scope of Attribution-Informativeness

Two separate issues arise when translating predictive accuracy into a selection guarantee. First, accuracy under the uniform distribution on $\mathcal{S}_m$ need not imply accuracy at an optimizer-selected subset. Second, calibration uses $f_{c,m}$, whereas NADA uses F_c; these functions need not agree. Therefore, accurate calibrated predictions do not automatically imply that maximizing J^a maximizes u.

Why matching the surrogate matters. Consider $m = 1$, two equally weighted components, and three candidates with attribution rows $(2, 0)$, $(0.9, 0.9)$, and $(0, 2)$. Let $u_c(\mathcal{M}) = \mathbf{1}\{r_c(\mathcal{M}) \geq 1\}$, so threshold calibration is exact on every feasible subset. The true utilities of the three singleton subsets are $1/2$, 0, and $1/2$, respectively. For $F(x) = 1 - \exp(-2.5x)$, however, the first and third singleton objectives are $\frac{1}{2}(1 - e^{-5}) < 1/2$, whereas the second is $1 - e^{-2.25} > 1/2$. Thus NADA selects the zero-utility singleton, despite perfect threshold calibration and nonnegative scores. This example shows that the choice of concave aggregation requires justification beyond threshold recovery.

The following elementary result states one sufficient condition for transferring an optimization guarantee to true utility.

Proposition B.3 (Transfer under uniform surrogate accuracy)**.** *Let $Q : \mathcal{S}_m \to \mathbb{R}$ satisfy*

$$\sup_{\mathcal{M} \in \mathcal{S}_m} |u(\mathcal{M}) - Q(\mathcal{M})| \leq \eta.$$

Suppose $\widehat{\mathcal{M}} \in \mathcal{S}_m$ satisfies $Q(\widehat{\mathcal{M}}) \geq \max_{M \in \mathcal{S}_m} Q(\mathcal{M}) - \xi$, with $\xi \geq 0$. Then, for any $\mathcal{M}^\star \in \arg\max_{\mathcal{M} \in \mathcal{S}_m} u(\mathcal{M})$,

$$u(\mathcal{M}^\star) - u(\widehat{\mathcal{M}}) \leq 2\eta + \xi.$$

Proof. By the uniform error bound and the optimization assumption,

$$u(\mathcal{M}^\star) \le Q(\mathcal{M}^\star) + \eta \le Q(\widehat{\mathcal{M}}) + \xi + \eta \le u(\widehat{\mathcal{M}}) + \xi + 2\eta.$$

□

Fix any family $\{f_{c,m}\}_{c\in\mathcal{C}}$ witnessing attribution-informativeness and define the corresponding calibrated surrogate $Q(\mathcal{M}) = \sum_{c\in\mathcal{C}} w_c f_{c,m}(r_c(\mathcal{M}))$. Uniform component errors $|u_c(\mathcal{M}) - f_{c,m}(r_c(\mathcal{M}))| \le \varepsilon_c$ for every feasible $\mathcal{M}$ allow us to take $\eta = \sum_c w_c\varepsilon_c$. The average-case attribution-informativeness definition does not supply this uniform bound. Applying Proposition B.3 to NADA would additionally require an accuracy bound for the objective it actually optimizes, possibly after a positive affine rescaling. We do not assume or establish such a bound here.

C Detailed Experimental Setup and Full Results

This appendix describes the datasets, selection budgets, training settings, and evaluation protocol, and reports the complete per-budget results for Random and the three primary attribution families: influence functions, TracIn, and leave-one-out (LOO). Each attribution family is evaluated using scalar Top selection and NADA. The primary LR/RR evaluation comprises 52 dataset–metric–corruption conditions, and the primary fine-tuning evaluation comprises 23 task–model–corruption conditions. We also report the available results for 11 additional fine-tuning conditions in Section C.7; these conditions are excluded from the main-text aggregate comparisons.

C.1 Datasets and Evaluation Splits

Logistic and ridge regression. The LR/RR suite contains 11 datasets and 13 dataset–metric settings: eight classification-accuracy settings, two additional loss settings for Wind and Pol, and three regression-MSE settings. Each condition uses $n = 200$ candidate training examples. The evaluation set contains 2,000 examples except for Auction Verification, which contains 1,843. Table 3 lists the datasets and metrics.

Table 3: Tabular datasets, metrics, and sample sizes. Dataset references are given in Section 5.1.

Code	Dataset	Metric(s)	n	n_{eval}
WD	Wind	Accuracy, loss	200	2,000
PO	Pol	Accuracy, loss	200	2,000
PM	Phoneme	Accuracy	200	2,000
CP	CPU	Accuracy	200	2,000
TP	2DPlanes	Accuracy	200	2,000
AF	APS Failure	Accuracy	200	2,000
VE	Vehicle SensIT	Accuracy	200	2,000
CC	Credit Card Fraud Detection	Accuracy	200	2,000
CH	California Housing	MSE	200	2,000
PP	Physicochemical Protein	MSE	200	2,000
AU	Auction Verification	MSE	200	1,843

Prompt fine-tuning. The task suite includes Amazon Polarity, Yelp Polarity, Rotten Tomatoes movie-review sentiment classification, SST-2, MNLI, ANLI, and RTE. Tables 4 and 5 specify the dataset identifiers and splits. We use *evaluation accuracy* throughout this appendix because Amazon and Yelp use test-set slices, whereas the remaining tasks use validation splits. The task code MR denotes Rotten Tomatoes movie reviews; RT denotes Recognizing Textual Entailment (RTE). The suffixes BT, LM, and GPT denote BERT, Llama-2-7B, and GPT-OSS-20B, respectively.

Table 4: Fine-tuning dataset identifiers. The ANLI identifier is the recorded internal loader alias.

Task code	Dataset identifier
AMAZON	`mteb/amazon_polarity`
YELP	`fancyzhx/yelp_polarity`
MR	`cornell-movie-review-data/rotten_tomatoes`
SST2	`stanfordnlp/sst2`
MNLI	`nyu-mll/multi_nli`
ANLI	`__nash_anli_all__`
RT (RTE)	`nyu-mll/glue`, subset `rte`

Table 5: Fine-tuning candidate and evaluation splits. An unsliced split uses all of its examples.

Task	Candidate split	Evaluation split	n	n_{eval}
AMAZON	`train[:10000]`	`test[:1000]`	10,000	1,000
YELP	`train[:10000]`	`test[:1000]`	10,000	1,000
MR	`train`	`validation`	8,530	1,066
SST2	`train[:10000]`	`validation`	10,000	872
MNLI	`train[:10000]`	`validation_matched[:1000]`	10,000	1,000
ANLI	`train[:10000]`	`validation[:1000]`	10,000	1,000
RT	`train`	`validation`	2,490	277

C.2 Selection Budgets, Corruption, and Randomness

Selection budgets. For LR/RR, the selection ratios are $\{5, 10, 15, 20, 25, 30, 40, 50\}\%$, corresponding to $m \in \{10, 20, 30, 40, 50, 60, 80, 100\}$. For fine-tuning, the nominal ratios are $\{2, 6, 10, 20, 30, 40, 50\}\%$. Table 6 gives the actual integer subset sizes. We use m/n, rather than the nominal percentage, when computing curve averages. No result tables extrapolate to unmeasured selection budgets.

Table 6: Fine-tuning subset sizes at each nominal selection percentage.

n	2%	6%	10%	20%	30%	40%	50%
10,000	200	600	1,000	2,000	3,000	4,000	5,000
8,530	171	512	853	1,706	2,559	3,412	4,265
2,490	50	150	249	498	747	996	1,245

Corruption conditions. Every LR/RR dataset–metric setting is evaluated at recorded corruption levels 0, 10%, 20%, and 30%, giving $13 \times 4 = 52$ conditions. For classification, the nonzero levels denote the fraction of corrupted training labels. The RR tables preserve the recorded corruption-level labels; the regression-target perturbation rule is not specified in the available metadata. The primary fine-tuning suite includes 12 clean conditions, six conditions with 10% label corruption, and five with 20% label corruption. Table 8 distinguishes these primary conditions from the additional recorded conditions.

Randomness. The LR/RR configurations use seed 122. The fine-tuning base seed is 114514. At each available fine-tuning budget, Random is averaged over five repeat seeds, 114514–114518. Its reported standard deviation is $\sqrt{\frac{1}{5}\sum_{s=1}^{5}(v_s - \bar{v})^2}$, where v_s is the accuracy from seed s and $\bar{v}$ is their mean. Each other fine-tuning selector has one recorded run per condition and budget. These standard deviations therefore describe variability across Random's five recorded runs, not uncertainty for the other selectors.

C.3 TRAINING CONFIGURATIONS

The fine-tuning models are `bert-base-uncased`, `Llama-2-7b-hf`, and `gpt-oss-20b`. All use a learning rate of 10^{-5}, bfloat16 precision, CUDA, and `device_map=auto`. Model dropout and training-data shuffling are disabled. Table 7 summarizes the model-level settings; Table 8 gives task-specific sequence and generation limits. The RR regularization parameter is $\alpha_{\text{ridge}} = 1.0$.

Table 7: Fine-tuning settings. Batch × accumulation is the product before any data-parallel replication. NR means not recorded; dashes indicate an inapplicable setting.

Setting	BT	LM	GPT
Epochs	10	2	3
Training batch size	16	8	4
Evaluation batch size	32	16	8
Gradient accumulation	1	2	2
Batch × accumulation	16	16	8
LoRA enabled	No	Yes	Yes
LoRA rank	—	16	16
LoRA scaling parameter	—	16	16
LoRA dropout	—	0	0
LoRA bias	—	`none`	`none`
Frozen BERT layers	8	—	—
Gradient checkpointing	NR	Off	Off
Model dropout	Disabled	Disabled	Disabled
Training-data shuffling	Off	Off	Off

Table 8: Task-specific token limits and corruption-level coverage. Primary conditions contribute to the main-text aggregate statistics. Additional conditions are reported separately; some are incomplete.

Task–model	Max. length	Max. new tokens	Primary	Additional
AMAZON-BT	512	1	0%, 10%, 20%	—
MR-BT	256	1	0%, 10%, 20%	—
YELP-BT	512	1	0%, 10%	—
SST2-BT	128	1	—	0%, 10%, 20%
MNLI-LM	128	64	0%, 10%, 20%	—
MR-LM	128	128	0%	—
RT-LM	128	128	0%, 10%, 20%	—
SST2-LM	128	64	0%, 10%, 20%	—
YELP-LM	128	64	—	0%, 10%, 20%
AMAZON-GPT	512	64	0%	10%, 20%
ANLI-GPT	256	64	0%	10%
MNLI-GPT	256	64	0%	—
RT-GPT	256	128	0%	10%, 20%
SST2-GPT	128	64	0%	—

C.4 ATTRIBUTION, SELECTION, AND REPORTING PROTOCOL

Attribution and selection. Each Top/NADA pair uses the same component attribution matrix. Top ranks candidates by the scalar aggregate, whereas NADA constructs a subset using the component-wise nonlinear objective. The selection settings described in Section 5 use $F_\lambda(x) = 1 - \exp(-\lambda x)$ with $\lambda = 2.5$, global max-absolute normalization, and no additional scalar-quality term. The LR/RR configurations additionally record a selection threshold of zero and no custom component weights. Influence scores are available for the ten LR accuracy/loss settings; TracIn and LOO are available for all 13 LR/RR settings. Influence is not reported on the RR settings.

Fine-tuning uses precomputed method-specific selection orders. The maximum stored prefix is the 50% budget: 5,000, 4,265, or 1,245 examples, depending on the task. Each paired selector uses the same fine-tuning configuration for a given condition.

Metrics and curve averages. Accuracy results are reported in percent; loss and MSE retain their original units. Let $r_1 < \cdots < r_K$ be the actual selection ratios and v_k the corresponding metric values. The curve average (CA) is

$$\mathrm{CA}(v) = \frac{1}{r_K - r_1} \sum_{k=1}^{K-1} \frac{v_k + v_{k+1}}{2} (r_{k+1} - r_k). \tag{13}$$

This normalized trapezoidal area differs from an unweighted arithmetic mean across budgets. Across conditions, we average the resulting CAs arithmetically. Accuracy improvements are measured in percentage points; loss and MSE improvements are measured as reductions in their respective units. We do not pool raw improvements across different metrics.

Primary aggregate results. Table 9 summarizes the paired comparisons. On LR/RR, NADA wins 134 of 144 curve-level comparisons. Across all LR/RR families, the mean accuracy gain is 8.52 percentage points, and the mean loss and MSE reductions are 0.2215 and 0.3850, respectively. For fine-tuning, NADA wins all 69 primary curve-level comparisons and 447 of 483 budget-level comparisons, with 15 ties and 21 losses. Averaged over the three attribution families and the 23 primary conditions, Top achieves 51.92% CA accuracy, NADA achieves 75.68%, and Random achieves 75.50%. These fine-tuning aggregates exclude the additional conditions in Section C.7.

Table 9: Primary matched comparisons. W/T/L denotes NADA wins, ties, and losses against the corresponding Top selector. Fine-tuning gains are mean CA accuracy differences in percentage points.

Family	LR/RR curves (W/T/L)	FT curves (W/T/L)	FT gain (pp)
Influence	36 / 0 / 4	23 / 0 / 0	23.01
TracIn	48 / 0 / 4	23 / 0 / 0	21.63
LOO	50 / 0 / 2	23 / 0 / 0	26.64
Overall	134 / 0 / 10	69 / 0 / 0	23.76

Reading the full-result tables. IF denotes influence functions. The tables list each selector at every evaluated nominal budget, followed by CA computed using the actual subset sizes. Boldface identifies the better displayed value within each Top/NADA pair at a fixed budget or in the CA column; values tied at the displayed precision are not emphasized. Random is not included in this paired boldface comparison. Accuracy is rounded to two decimal places and loss/MSE to four; aggregates and win counts use the unrounded values. A numerical tolerance of 10^{-12} in the original metric units is used to identify ties.

For fine-tuning, the Random row gives the five-seed mean and the Random SD row gives its standard deviation in percentage points. No CA standard deviation is inferred from the per-budget standard deviations. A dash denotes an unavailable value; it is not zero. CA is reported only when all budgets in the full evaluation range are available.

C.5 Full LR/RR Results

The following tables report all 52 primary LR/RR conditions across eight selection budgets. They contain 2,720 recorded metric values across the applicable Random, Top, and NADA selectors. Accuracy, validation loss, and MSE are shown in separate tables.

Table 10: WD-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	79.01	79.33	81.89	82.10	82.39	82.63	83.72	84.21	82.32
IF-Top	70.40	70.95	76.45	78.45	81.50	81.65	82.95	83.25	79.35
IF-NADA	**78.50**	**79.05**	**79.75**	**81.40**	81.50	**82.25**	**83.35**	**83.85**	**81.65**
TracIn-Top	75.60	77.65	79.40	79.70	80.40	81.00	82.00	82.60	80.34
TracIn-NADA	**82.30**	**85.00**	**84.70**	**83.00**	**83.40**	**82.20**	**82.05**	**83.60**	**83.14**
LOO-Top	57.60	72.00	74.50	77.15	80.25	81.05	82.80	**83.25**	78.12
LOO-NADA	**81.55**	**82.70**	**83.50**	**83.50**	**83.50**	**83.40**	**83.55**	83.00	**83.24**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	74.87	75.18	78.45	78.66	79.32	79.51	80.11	81.33	78.87
IF-Top	67.70	69.90	72.70	66.95	66.60	68.15	66.15	73.25	68.64
IF-NADA	**79.90**	**80.30**	**80.65**	**81.45**	**82.30**	**83.05**	**84.80**	**85.00**	**82.65**
TracIn-Top	50.00	50.00	79.45	79.75	79.70	79.85	80.25	**81.20**	75.04
TracIn-NADA	**82.10**	**85.00**	**84.75**	**84.25**	**84.45**	**83.80**	**81.55**	80.85	**83.24**
LOO-Top	50.00	50.00	58.00	64.60	72.35	**78.70**	**80.90**	**81.10**	70.10
LOO-NADA	**80.80**	**81.50**	**81.50**	**82.05**	**81.95**	78.65	77.65	76.85	**79.72**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	67.19	69.85	71.92	74.26	74.64	74.79	75.51	77.75	73.91
IF-Top	50.00	78.35	77.50	78.40	79.05	80.40	80.75	81.00	77.93
IF-NADA	**81.05**	**83.00**	**82.75**	**82.90**	**83.25**	**83.10**	**83.90**	**85.55**	**83.38**
TracIn-Top	78.35	79.20	79.80	80.15	79.90	80.05	80.00	**80.85**	79.91
TracIn-NADA	**83.75**	**85.10**	**83.45**	**83.60**	**82.95**	**84.00**	**81.35**	79.80	**82.83**
LOO-Top	71.80	64.90	67.10	65.95	66.30	67.65	72.30	75.90	69.12
LOO-NADA	**81.40**	**81.30**	**80.90**	**80.35**	**81.95**	**81.65**	**80.25**	**79.00**	**80.80**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	60.66	65.22	66.87	69.09	69.97	70.73	71.92	74.21	69.51
IF-Top	79.85	78.90	79.70	79.60	80.60	80.55	79.80	80.50	79.96
IF-NADA	**82.40**	**81.85**	**81.55**	**82.55**	**83.70**	**84.70**	**85.45**	**84.70**	**83.72**
TracIn-Top	80.05	79.55	79.80	**80.15**	**80.60**	80.60	**80.90**	**81.05**	**80.43**
TracIn-NADA	**83.00**	**82.45**	**80.50**	77.60	78.60	**81.50**	75.85	76.55	79.02
LOO-Top	72.25	71.05	70.45	69.75	72.80	67.75	**73.25**	**73.95**	71.36
LOO-NADA	**80.70**	**80.65**	**79.30**	**78.70**	**78.05**	**77.10**	69.80	69.20	**75.72**

Table 11: PO-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	66.39	73.50	74.77	77.30	78.84	79.27	80.27	80.63	77.52
IF-Top	**58.30**	61.60	59.55	64.85	68.10	72.00	74.95	75.85	68.56
IF-NADA	57.45	**66.15**	**68.20**	**70.25**	**73.25**	**74.35**	**75.30**	**75.95**	**71.63**
TracIn-Top	61.70	65.00	67.20	65.60	65.95	66.90	67.40	67.95	66.41
TracIn-NADA	**70.90**	**75.55**	**75.35**	**79.70**	**81.20**	**82.70**	**82.45**	**83.25**	**79.94**
LOO-Top	60.70	65.15	57.95	61.00	70.70	74.45	79.80	**83.30**	71.08
LOO-NADA	**64.05**	**69.15**	**78.80**	**80.05**	**81.05**	**81.05**	**81.45**	80.30	**78.43**

Continued on next page

Table 11 (continued): PO-LR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	62.48	67.28	71.28	74.70	77.14	77.00	78.67	79.29	74.86
IF-Top	43.05	55.35	57.55	58.20	60.50	61.60	64.30	70.10	60.47
IF-NADA	**59.85**	**64.95**	**64.10**	**63.40**	**66.50**	**67.60**	**68.50**	**70.70**	**66.44**
TracIn-Top	58.75	62.50	65.95	66.60	66.85	67.55	65.85	66.25	65.62
TracIn-NADA	**66.25**	**69.85**	**72.65**	**73.30**	**75.05**	**79.40**	**79.95**	**82.60**	**76.17**
LOO-Top	45.00	54.60	60.15	59.30	63.05	64.15	72.40	76.50	64.12
LOO-NADA	**70.45**	**72.90**	**74.30**	**79.15**	**79.15**	**79.15**	**80.45**	**79.20**	**77.73**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	59.35	64.58	67.14	71.32	71.23	72.06	74.15	75.13	70.61
IF-Top	54.15	62.35	62.35	63.25	63.80	64.30	63.35	69.05	63.45
IF-NADA	**61.00**	**65.40**	**66.90**	**67.50**	**68.00**	**68.65**	**73.55**	**76.35**	**69.41**
TracIn-Top	57.95	64.15	70.50	71.60	69.70	68.40	66.20	69.20	67.68
TracIn-NADA	**68.00**	**71.20**	**71.95**	**72.80**	**73.70**	**76.05**	**77.75**	**80.45**	**74.85**
LOO-Top	53.90	60.35	65.25	61.35	63.85	**73.40**	**78.45**	**79.50**	69.36
LOO-NADA	**60.35**	**73.85**	**78.20**	**76.20**	**72.75**	70.45	70.85	71.30	**72.21**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	56.57	60.16	61.79	65.07	63.69	64.08	66.97	68.22	64.14
IF-Top	54.65	60.20	57.45	61.75	62.50	63.35	65.45	69.00	62.68
IF-NADA	**59.40**	**67.75**	**68.75**	**71.75**	**71.05**	**73.90**	**74.25**	**74.70**	**71.45**
TracIn-Top	50.00	65.55	69.85	68.60	66.90	65.50	65.75	68.60	66.03
TracIn-NADA	**66.90**	**72.85**	**73.60**	**75.55**	**76.35**	**77.55**	**80.45**	**81.75**	**76.75**
LOO-Top	65.00	60.10	60.50	60.60	61.05	63.10	65.35	**67.80**	63.10
LOO-NADA	**74.25**	**74.00**	**76.20**	**69.30**	**65.50**	**67.00**	**66.40**	67.00	**69.16**

Table 12: PM-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	63.52	67.03	69.34	70.17	71.05	71.81	72.93	73.24	70.69
IF-Top	43.00	46.00	50.10	55.75	55.70	57.30	59.55	60.80	54.99
IF-NADA	**73.15**	**73.35**	**73.00**	**73.30**	**74.35**	**74.50**	**74.80**	**74.15**	**74.01**
TracIn-Top	67.30	71.10	70.75	71.60	72.30	72.50	73.00	72.95	71.90
TracIn-NADA	**74.20**	**71.55**	**74.75**	**74.40**	**74.30**	**74.45**	**74.30**	**74.55**	**74.10**
LOO-Top	57.55	50.65	54.40	55.50	53.75	57.05	61.95	64.30	57.43
LOO-NADA	**70.60**	**72.45**	**72.45**	**70.05**	**71.20**	**70.65**	**70.80**	**66.50**	**70.61**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	58.77	62.79	64.91	67.42	69.76	71.06	72.06	72.02	68.56
IF-Top	63.35	64.85	62.25	63.50	64.05	67.30	68.95	72.20	66.38
IF-NADA	**73.25**	**73.80**	**74.55**	**73.55**	**74.00**	**74.05**	**73.75**	**74.40**	**73.94**
TracIn-Top	50.00	50.00	**73.95**	**74.15**	**73.95**	**74.05**	73.60	**73.65**	69.89
TracIn-NADA	**72.95**	**74.10**	73.70	74.00	73.55	72.35	**74.05**	73.45	**73.54**
LOO-Top	41.95	49.60	53.00	53.75	59.10	54.90	54.05	64.05	54.55
LOO-NADA	**72.40**	**71.30**	**69.00**	**68.70**	**63.85**	**61.20**	**61.10**	**70.00**	**65.89**

Continued on next page

Table 12 (continued): PM-LR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	53.84	58.31	60.26	63.75	67.24	68.81	69.92	69.61	65.46
IF-Top	67.10	67.90	70.20	71.20	71.85	70.95	**72.35**	**74.35**	71.13
IF-NADA	**73.60**	**73.75**	**74.70**	**73.65**	**72.70**	**73.85**	72.30	73.85	**73.42**
TracIn-Top	**73.30**	73.55	**74.60**	**74.55**	**74.15**	**74.00**	**74.15**	73.35	**74.02**
TracIn-NADA	70.05	**74.15**	74.20	73.30	73.90	72.95	73.95	**74.30**	73.58
LOO-Top	50.00	56.80	59.90	55.60	58.15	54.25	**56.95**	**63.95**	57.19
LOO-NADA	**65.65**	**72.40**	**71.85**	**64.45**	**60.00**	**55.40**	55.45	62.50	**62.00**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	50.18	54.47	56.59	61.17	63.32	64.59	66.13	65.51	61.70
IF-Top	73.50	74.00	74.35	73.85	73.95	73.65	**73.85**	**74.10**	73.91
IF-NADA	**73.60**	**75.45**	**75.90**	**74.00**	**75.15**	**74.55**	73.50	73.65	**74.42**
TracIn-Top	**74.25**	**74.50**	**74.10**	73.95	73.45	73.65	**73.75**	**73.70**	**73.87**
TracIn-NADA	71.15	73.40	72.80	**75.20**	**74.85**	**74.90**	73.45	73.25	73.81
LOO-Top	45.05	52.50	57.00	56.00	57.05	53.85	55.60	54.35	54.60
LOO-NADA	**67.55**	**66.70**	**61.30**	**60.70**	**64.15**	**66.60**	**66.55**	**68.60**	**65.36**

Table 13: CP-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	82.80	83.04	84.28	85.24	86.09	86.42	86.95	87.81	85.71
IF-Top	36.15	**60.55**	67.00	76.00	79.35	84.55	87.15	89.40	76.83
IF-NADA	**41.25**	57.65	**68.85**	**76.80**	**82.35**	**85.90**	**89.90**	**89.85**	**78.31**
TracIn-Top	73.20	82.90	84.15	85.45	85.75	87.25	87.65	88.10	85.46
TracIn-NADA	**87.85**	**88.90**	**89.10**	**89.10**	**89.15**	**89.40**	**89.25**	**89.20**	**89.11**
LOO-Top	**70.40**	**79.35**	**81.65**	**86.55**	**89.65**	**89.45**	**89.25**	**89.00**	**86.01**
LOO-NADA	37.50	73.05	78.35	80.05	81.80	83.05	81.65	83.65	78.17
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	79.81	80.15	81.56	82.30	83.55	83.77	85.05	85.11	83.15
IF-Top	76.10	78.90	**82.50**	80.65	80.10	81.05	81.00	83.50	80.81
IF-NADA	**78.55**	**80.30**	81.25	**83.05**	**82.25**	**82.95**	**84.75**	**86.75**	**82.98**
TracIn-Top	81.60	83.15	82.30	82.85	83.40	83.70	84.10	85.50	83.53
TracIn-NADA	**83.00**	**84.10**	**84.75**	**85.15**	**85.55**	**85.70**	**87.10**	**87.55**	**85.71**
LOO-Top	67.70	70.85	74.60	76.60	77.40	80.75	84.55	84.80	78.70
LOO-NADA	**81.80**	**82.95**	**84.15**	**87.70**	**87.70**	**87.35**	**87.20**	**86.10**	**86.10**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	72.60	74.39	77.56	78.54	81.00	80.82	82.53	83.14	79.69
IF-Top	70.00	75.50	77.50	77.30	78.10	79.00	80.05	83.90	78.43
IF-NADA	**78.05**	**82.30**	**84.65**	**84.90**	**85.55**	**85.40**	**85.20**	**86.30**	**84.58**
TracIn-Top	50.00	80.90	81.95	81.90	81.95	81.75	83.10	83.50	80.45
TracIn-NADA	**80.85**	**83.00**	**83.70**	**84.35**	**84.90**	**85.15**	**85.00**	**87.40**	**84.61**
LOO-Top	77.20	43.05	42.15	63.35	69.50	73.40	74.70	76.40	65.84
LOO-NADA	**85.40**	**88.20**	**88.00**	**88.60**	**89.10**	**87.70**	**84.90**	**83.95**	**86.88**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	67.79	69.30	70.16	71.72	73.98	73.69	75.77	76.52	73.07
IF-Top	78.45	81.20	80.60	83.35	83.90	84.35	85.10	85.45	83.38
IF-NADA	**80.25**	**81.80**	**81.55**	**83.90**	**84.90**	**85.80**	**87.30**	**87.35**	**84.77**
TracIn-Top	82.65	84.45	84.25	84.65	84.85	84.65	84.55	84.95	84.51
TracIn-NADA	**84.85**	**85.90**	**85.60**	**86.75**	**86.60**	**87.90**	**88.10**	**87.65**	**87.00**
LOO-Top	76.30	41.75	50.35	53.75	67.80	72.20	76.90	**78.70**	65.84
LOO-NADA	**86.25**	**85.65**	**82.90**	**79.60**	**81.65**	**81.20**	**77.55**	74.25	**80.45**

Table 14: TP-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	73.16	75.31	76.82	78.04	79.49	79.94	80.97	81.75	78.87
IF-Top	39.15	38.65	39.75	43.95	46.35	56.70	73.25	78.25	55.34
IF-NADA	**61.25**	**76.80**	**80.85**	**82.80**	**83.75**	**83.85**	**84.70**	**83.70**	**81.52**
TracIn-Top	65.35	67.40	70.35	72.30	74.70	76.95	80.55	81.45	75.04
TracIn-NADA	**78.20**	**83.15**	**83.80**	**83.20**	**83.20**	**83.15**	**83.10**	**82.75**	**82.90**
LOO-Top	51.15	59.75	64.10	78.20	81.05	**83.25**	**85.45**	**85.15**	76.62
LOO-NADA	**74.40**	**78.90**	**78.35**	**78.85**	**82.80**	82.90	81.90	81.25	**80.61**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	65.58	69.42	72.28	73.99	76.22	77.02	79.28	80.37	75.46
IF-Top	60.00	60.55	64.30	63.15	61.85	68.60	73.70	76.30	67.38
IF-NADA	**68.60**	**74.25**	**81.70**	**82.35**	**83.35**	**84.55**	**83.95**	**84.60**	**81.70**
TracIn-Top	65.70	68.80	71.95	72.45	75.75	78.40	80.85	81.60	75.86
TracIn-NADA	**81.00**	**84.05**	**84.15**	**83.20**	**83.15**	**82.80**	**83.55**	**83.85**	**83.36**
LOO-Top	38.30	48.05	49.30	49.85	48.70	51.95	60.45	66.10	53.33
LOO-NADA	**75.85**	**77.40**	**79.60**	**80.15**	**82.15**	**81.85**	**79.80**	**79.60**	**79.91**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	60.30	61.90	66.36	69.27	73.18	75.16	76.47	77.31	71.54
IF-Top	65.30	68.25	69.45	70.25	72.20	74.30	78.20	80.30	73.44
IF-NADA	**72.85**	**79.20**	**83.90**	**82.95**	**83.10**	**83.20**	**83.80**	**83.45**	**82.38**
TracIn-Top	67.95	71.25	73.65	76.20	77.45	79.25	81.45	83.00	77.48
TracIn-NADA	**82.15**	**83.90**	**81.60**	**82.25**	**82.25**	**83.50**	**82.70**	**83.15**	**82.76**
LOO-Top	59.60	67.40	70.10	65.50	67.05	69.95	72.45	77.25	69.66
LOO-NADA	**79.95**	**81.50**	**83.05**	**78.50**	**78.75**	**77.55**	**76.80**	**79.05**	**78.97**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	55.38	56.81	57.98	62.77	65.92	66.99	71.47	71.46	65.12
IF-Top	71.10	73.60	75.85	78.75	78.60	78.15	80.15	81.50	77.93
IF-NADA	**78.00**	**84.15**	**81.85**	**80.20**	**82.95**	**83.85**	**83.05**	**82.15**	**82.46**
TracIn-Top	70.30	75.75	76.80	79.65	81.85	**82.80**	**83.10**	82.10	80.19
TracIn-NADA	**82.75**	**81.55**	**81.70**	**82.20**	81.85	82.35	82.40	82.10	**82.12**
LOO-Top	64.85	55.90	63.95	64.65	67.05	61.65	66.80	74.65	64.97
LOO-NADA	**80.15**	**72.00**	**70.30**	**75.50**	**79.65**	**78.65**	**76.70**	**78.10**	**76.33**

Table 15: AF-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	82.95	82.99	83.38	84.54	84.78	85.19	85.44	86.43	84.70
IF-Top	74.80	74.20	78.05	80.80	85.45	85.75	87.40	89.15	83.16
IF-NADA	**84.95**	**84.70**	**86.10**	**86.30**	**89.05**	**89.10**	**89.35**	**89.70**	**87.85**
TracIn-Top	16.30	18.40	24.60	26.55	27.50	28.00	62.10	68.35	37.75
TracIn-NADA	**34.90**	**80.00**	**82.70**	**83.65**	**83.85**	**84.10**	**83.45**	**82.95**	**80.41**
LOO-Top	84.40	**87.25**	**88.55**	**90.20**	**89.85**	**89.70**	**90.10**	**90.15**	**89.22**
LOO-NADA	**86.50**	86.55	87.05	87.05	88.05	87.85	88.75	89.75	87.89

Continued on next page

Table 15 (continued): AF-LR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	82.23	80.37	79.62	80.66	82.03	82.17	82.57	83.09	81.70
IF-Top	**84.50**	**84.40**	82.25	83.25	84.25	84.80	85.95	87.00	84.72
IF-NADA	84.05	82.85	**83.70**	**85.75**	**86.70**	**87.60**	**87.55**	**87.80**	**86.15**
TracIn-Top	26.90	22.75	23.95	25.50	26.25	27.45	41.50	63.75	33.31
TracIn-NADA	**79.95**	**83.75**	**83.70**	**84.75**	**84.95**	**86.00**	**86.65**	**86.70**	**85.12**
LOO-Top	23.30	58.60	41.35	46.85	65.65	83.20	84.95	85.75	67.17
LOO-NADA	**85.25**	**85.00**	**84.95**	**85.40**	**85.75**	**85.70**	**88.40**	**87.70**	**86.31**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	78.05	75.44	75.41	76.26	77.33	76.86	77.06	78.85	76.86
IF-Top	75.15	72.60	76.35	75.50	76.40	78.70	80.50	82.40	77.76
IF-NADA	**76.90**	**83.45**	**84.75**	**84.65**	**86.95**	**87.05**	**87.00**	**87.20**	**85.56**
TracIn-Top	35.30	23.35	22.85	24.45	26.60	27.30	41.40	60.70	33.26
TracIn-NADA	**82.75**	**84.30**	**84.25**	**82.10**	**82.55**	**81.95**	**84.45**	**84.20**	**83.40**
LOO-Top	59.35	71.50	73.95	75.15	79.40	84.55	82.70	85.70	78.62
LOO-NADA	**83.15**	**83.60**	**84.25**	**84.70**	**85.50**	**84.75**	**88.85**	**86.95**	**85.71**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	70.67	67.92	67.69	71.39	72.68	72.51	73.70	73.76	71.66
IF-Top	39.10	44.35	52.40	64.60	64.60	66.00	67.20	69.55	60.94
IF-NADA	**66.00**	**76.90**	**81.40**	**81.20**	**83.20**	**83.60**	**85.45**	**84.65**	**81.85**
TracIn-Top	50.00	25.45	27.50	25.95	26.60	26.90	30.85	48.25	31.20
TracIn-NADA	**84.35**	**84.05**	**84.45**	**84.65**	**84.30**	**83.75**	**83.45**	**83.30**	**83.94**
LOO-Top	83.15	84.40	70.80	62.95	64.80	71.60	76.15	79.15	73.71
LOO-NADA	**86.25**	**84.90**	**83.75**	**84.60**	**83.50**	**83.40**	**83.35**	**83.20**	**83.88**

Table 16: VE-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	64.11	68.79	71.00	73.66	74.42	74.35	75.91	76.47	73.31
IF-Top	67.05	70.30	73.40	**76.60**	**76.95**	78.15	**80.20**	80.75	76.57
IF-NADA	**69.85**	**71.70**	**74.55**	76.25	76.80	78.15	79.55	**81.15**	**76.86**
TracIn-Top	50.75	53.60	53.55	55.25	54.35	54.85	55.70	58.15	54.88
TracIn-NADA	**63.35**	**69.95**	**71.60**	**74.95**	**77.00**	**76.60**	**79.50**	**79.30**	**75.38**
LOO-Top	50.00	63.55	64.30	66.20	68.30	73.45	**79.95**	**79.85**	70.81
LOO-NADA	**59.70**	**69.80**	**72.55**	**74.85**	**77.15**	**75.45**	71.40	74.30	**72.72**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	61.28	65.56	66.25	67.41	68.82	69.49	71.06	72.46	68.61
IF-Top	66.95	74.10	**78.65**	**78.65**	77.00	77.50	78.90	**80.90**	77.43
IF-NADA	**71.35**	**75.70**	76.30	77.00	**77.65**	**79.40**	**81.70**	80.15	**78.33**
TracIn-Top	53.90	54.50	54.70	56.90	57.60	57.00	60.05	63.30	57.73
TracIn-NADA	**73.00**	**78.55**	**78.30**	**82.50**	**78.05**	**76.30**	**78.20**	**77.70**	**78.05**
LOO-Top	52.70	67.80	66.60	69.95	72.15	71.80	73.65	**77.60**	70.61
LOO-NADA	**63.35**	**69.25**	**75.75**	**77.15**	**79.00**	**79.50**	**76.55**	73.85	**75.45**

Continued on next page

Table 16 (continued): VE-LR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	57.53	61.81	62.08	64.23	64.57	65.11	66.30	68.02	64.42
IF-Top	40.55	**60.20**	**64.55**	**70.20**	**72.95**	**74.85**	**77.90**	**80.00**	**70.69**
IF-NADA	**48.10**	53.70	57.35	68.90	70.55	71.90	76.90	78.80	68.33
TracIn-Top	66.65	70.65	75.70	77.05	77.40	79.05	80.55	78.90	76.97
TracIn-NADA	**73.95**	**79.90**	**80.70**	**81.70**	**82.40**	**82.55**	**82.60**	**79.60**	**81.14**
LOO-Top	56.65	69.00	68.35	69.95	70.55	**69.50**	**72.45**	**75.05**	70.04
LOO-NADA	**64.50**	**70.60**	**75.40**	**77.70**	**78.20**	68.50	64.25	66.00	**70.16**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	56.47	62.19	60.23	61.02	60.78	61.47	62.49	65.15	61.65
IF-Top	46.15	60.15	57.30	65.45	**73.30**	**73.75**	**74.90**	**75.45**	68.35
IF-NADA	**59.65**	**61.75**	**69.40**	**70.50**	71.20	72.20	73.15	73.45	**70.08**
TracIn-Top	78.50	76.85	78.15	79.65	79.90	79.65	79.25	**79.05**	78.98
TracIn-NADA	**79.00**	**80.50**	**80.20**	**80.65**	**80.95**	**81.40**	**79.85**	78.45	**80.23**
LOO-Top	**74.70**	61.00	64.70	72.00	72.55	**72.70**	**74.80**	**76.70**	71.44
LOO-NADA	70.70	**76.95**	**72.35**	**75.20**	**74.95**	70.55	70.90	65.20	**71.96**

Table 17: CC-LR: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	86.37	86.92	88.52	89.47	89.69	90.22	91.05	91.90	89.68
IF-Top	55.50	**74.30**	**76.25**	80.25	83.10	83.95	85.00	86.05	80.40
IF-NADA	**65.20**	73.60	62.20	**80.55**	**85.30**	**85.45**	**86.85**	**88.10**	**80.47**
TracIn-Top	50.00	49.10	70.10	79.85	85.90	85.45	89.55	90.80	78.67
TracIn-NADA	**82.60**	**91.05**	**92.50**	**93.35**	**93.20**	**93.20**	**92.95**	**93.80**	**92.32**
LOO-Top	28.40	58.40	70.00	75.85	80.90	88.95	**92.10**	**92.95**	78.88
LOO-NADA	**76.60**	**87.00**	**88.50**	**92.05**	**92.25**	**92.30**	91.70	91.50	**90.16**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	79.32	79.83	83.82	84.20	84.70	83.17	84.14	85.43	83.41
IF-Top	86.00	80.10	82.80	88.85	**89.80**	90.50	90.70	92.00	88.19
IF-NADA	86.00	**86.70**	**88.20**	**89.70**	89.75	**90.80**	**92.35**	**92.45**	**90.08**
TracIn-Top	50.00	50.00	83.80	83.75	83.85	84.20	85.55	87.45	79.03
TracIn-NADA	**87.65**	**88.80**	**90.95**	**91.80**	**92.25**	**91.30**	**93.15**	**94.10**	**91.66**
LOO-Top	37.00	47.50	59.25	67.70	68.70	71.45	78.75	87.05	68.15
LOO-NADA	**89.50**	**88.65**	**88.70**	**89.25**	**88.35**	**88.40**	**89.95**	**89.75**	**89.11**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	72.23	75.18	78.40	78.57	76.24	76.13	78.45	79.39	77.22
IF-Top	85.95	86.80	**89.35**	**89.25**	90.05	90.50	**91.65**	**91.35**	**89.87**
IF-NADA	**86.15**	**88.70**	88.65	88.25	**90.25**	**90.55**	91.30	91.20	89.84
TracIn-Top	**83.00**	82.55	80.95	81.55	81.55	81.40	81.55	82.75	81.78
TracIn-NADA	50.00	**90.65**	**91.30**	**92.35**	**88.90**	**88.10**	**81.95**	**83.55**	**85.31**
LOO-Top	**87.20**	82.45	81.05	81.45	83.25	84.10	**88.25**	**89.90**	84.93
LOO-NADA	86.95	**86.50**	**86.95**	**88.25**	**88.15**	**88.15**	86.75	89.00	**87.56**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	67.27	69.26	70.29	71.82	71.21	71.65	72.23	75.18	71.48
IF-Top	82.50	85.15	85.60	86.90	85.40	86.90	89.50	89.10	86.97
IF-NADA	**84.95**	**86.90**	**87.15**	**87.95**	**88.50**	**89.40**	**89.75**	**90.30**	**88.54**
TracIn-Top	84.10	82.30	82.90	83.35	83.15	82.80	**82.30**	**84.05**	82.96
TracIn-NADA	**90.05**	**90.70**	**89.50**	**86.85**	**83.95**	**86.05**	79.25	80.00	**84.84**
LOO-Top	75.75	82.25	82.20	80.15	81.10	**84.55**	**88.50**	**87.05**	83.83
LOO-NADA	**85.25**	**84.80**	**85.05**	**80.65**	**84.20**	84.50	84.60	83.25	**84.06**

Table 18: WD-LR-LOSS: loss (lower is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.4595	0.4524	0.4065	0.4062	0.4024	0.3980	0.3782	0.3710	0.4024
IF-Top	0.5712	0.5799	0.4939	0.4620	**0.3994**	**0.4025**	**0.3863**	**0.3850**	**0.4425**
IF-NADA	**0.4727**	**0.4760**	**0.4670**	**0.4591**	0.4599	0.4552	0.4023	0.3980	0.4426
TracIn-Top	0.4965	0.4879	0.4740	0.4872	0.4907	0.4950	0.4764	0.5033	0.4874
TracIn-NADA	**0.3970**	**0.3405**	**0.3436**	**0.3697**	**0.3650**	**0.3823**	**0.3818**	**0.3672**	**0.3691**
LOO-Top	1.1441	0.8692	0.7368	0.6214	0.6157	0.5383	0.4186	0.4036	0.6071
LOO-NADA	**0.5228**	**0.4774**	**0.4577**	**0.4486**	**0.3939**	**0.3956**	**0.3940**	**0.3335**	**0.4171**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.5164	0.5067	0.4564	0.4512	0.4402	0.4384	0.4298	0.4077	0.4486
IF-Top	0.6482	0.6503	0.5836	0.7498	0.7554	0.7466	0.8471	0.6223	0.7222
IF-NADA	**0.4397**	**0.4468**	**0.4440**	**0.4317**	**0.4307**	**0.4036**	**0.3407**	**0.3398**	**0.4000**
TracIn-Top	0.6931	0.6931	0.5096	0.5144	0.5289	0.5437	0.5842	0.5932	0.5744
TracIn-NADA	**0.4000**	**0.3399**	**0.3474**	**0.3547**	**0.3475**	**0.3669**	**0.3963**	**0.4127**	**0.3717**
LOO-Top	0.7279	0.6525	0.6526	0.6197	0.5901	0.5675	0.6064	0.5474	0.6100
LOO-NADA	**0.4584**	**0.4648**	**0.4942**	**0.4289**	**0.4236**	**0.3691**	**0.3608**	**0.3374**	**0.4059**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.6055	0.5910	0.5519	0.5152	0.5064	0.5040	0.5006	0.4722	0.5218
IF-Top	0.6931	0.4895	0.5466	0.5310	0.5472	0.4635	0.4575	0.4311	0.5002
IF-NADA	**0.4280**	**0.3764**	**0.3970**	**0.4133**	**0.3716**	**0.3643**	**0.3599**	**0.3614**	**0.3778**
TracIn-Top	0.4739	0.4856	0.4961	0.4955	0.5158	0.5267	0.5676	0.5981	0.5282
TracIn-NADA	**0.3636**	**0.3463**	**0.3637**	**0.3703**	**0.3823**	**0.3643**	**0.4131**	**0.4403**	**0.3842**
LOO-Top	0.6931	0.4943	0.4954	0.5062	0.5149	0.5347	0.4664	0.4394	0.5035
LOO-NADA	0.6931	**0.4370**	**0.4422**	**0.4204**	**0.4241**	**0.4117**	**0.3983**	**0.3758**	**0.4289**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.7435	0.6492	0.6130	0.5862	0.5748	0.5637	0.5478	0.5238	0.5844
IF-Top	0.4400	0.4626	0.4819	0.4914	0.4827	0.4683	0.4707	0.4511	0.4704
IF-NADA	**0.4045**	**0.4054**	**0.4096**	**0.3801**	**0.3554**	**0.3562**	**0.3767**	**0.3828**	**0.3804**
TracIn-Top	0.4410	0.4718	0.4780	0.4875	0.4921	0.5068	0.5496	0.6012	0.5123
TracIn-NADA	**0.3879**	**0.3949**	**0.4182**	**0.4667**	**0.4491**	**0.4063**	**0.4930**	**0.4891**	**0.4453**
LOO-Top	0.4400	0.4406	0.4794	0.4996	0.4888	0.5122	0.4115	0.4167	0.4596
LOO-NADA	**0.4115**	**0.3993**	**0.4106**	**0.4043**	**0.4108**	**0.4088**	**0.4038**	**0.3584**	**0.4011**

Table 19: PO-LR-LOSS: loss (lower is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.7528	0.7707	0.7392	0.7325	0.7420	0.7321	0.7086	0.7266	0.7336
IF-Top	**0.9324**	1.1817	**1.1962**	**1.0596**	1.0036	**0.8075**	**0.7293**	0.6135	**0.9101**
IF-NADA	1.0166	**1.1207**	1.3076	1.1732	**0.9964**	0.9491	0.8386	**0.4930**	0.9667
TracIn-Top	0.8211	0.8587	0.8598	0.8653	0.8442	0.8206	0.8080	0.8264	0.8346
TracIn-NADA	**0.6665**	**0.5545**	**0.5513**	**0.4808**	**0.4371**	**0.4092**	**0.3910**	**0.3856**	**0.4598**
LOO-Top	0.9623	1.2641	1.2797	1.1364	1.0748	1.0411	0.8501	0.7873	1.0317
LOO-NADA	**0.8919**	**0.6937**	**0.6709**	**0.5779**	**0.5932**	**0.5685**	**0.4835**	**0.4452**	**0.5830**

Continued on next page

Table 19 (continued): PO-LR-LOSS

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.8181	0.8383	0.8189	0.8007	0.7948	0.7368	0.7252	0.7683	0.7762
IF-Top	1.0561	1.2283	1.2418	**1.2227**	1.1717	**1.1388**	**0.9737**	**0.7469**	1.0884
IF-NADA	**0.9107**	**0.9946**	**1.0542**	1.2247	**1.1707**	1.1482	1.0737	0.8546	**1.0693**
TracIn-Top	0.7656	0.8101	0.9621	0.9774	0.9588	0.9729	0.9754	0.8879	0.9322
TracIn-NADA	**0.6855**	**0.6575**	**0.6374**	**0.6284**	**0.5324**	**0.4449**	**0.4487**	**0.4056**	**0.5299**
LOO-Top	1.0165	1.1511	1.2628	1.2493	1.2771	1.1758	1.1248	0.8348	1.1441
LOO-NADA	**0.8492**	**0.8322**	**0.5587**	**0.4985**	**0.5017**	**0.4942**	**0.4496**	**0.4629**	**0.5466**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.8964	0.8607	0.8931	0.8498	0.8184	0.8347	0.7602	0.7113	0.8171
IF-Top	0.9215	0.9954	1.0970	1.1490	1.2555	1.2281	0.9894	0.7012	1.0533
IF-NADA	**0.8141**	**0.9373**	**0.9859**	**0.9797**	**0.8266**	**0.8150**	**0.7059**	**0.6640**	**0.8261**
TracIn-Top	0.7319	0.6949	0.6447	0.6453	0.7116	0.7347	0.8970	0.8432	0.7557
TracIn-NADA	**0.7030**	**0.6190**	**0.6097**	**0.5954**	**0.5582**	**0.4944**	**0.4828**	**0.4089**	**0.5389**
LOO-Top	0.9704	0.9624	0.9044	1.0562	1.0875	1.0748	0.8741	0.6784	0.9483
LOO-NADA	**0.6948**	**0.5851**	**0.5025**	**0.5078**	**0.4828**	**0.4620**	**0.4974**	**0.4114**	**0.5027**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.9869	1.0181	1.1196	1.0456	1.0904	1.1054	0.9398	0.8707	1.0195
IF-Top	1.0925	1.1656	1.2497	1.1720	1.1730	1.1469	0.9335	0.9163	1.0900
IF-NADA	**0.8541**	**0.7564**	**0.7766**	**0.7338**	**0.7303**	**0.6910**	**0.7284**	**0.7634**	**0.7423**
TracIn-Top	0.6931	0.7384	0.7238	0.7920	0.8747	0.9432	0.9883	0.9143	0.8646
TracIn-NADA	**0.6183**	**0.5738**	**0.5442**	**0.5214**	**0.4978**	**0.4837**	**0.4645**	**0.4772**	**0.5087**
LOO-Top	1.0161	1.1087	0.8466	0.8629	0.8355	0.8528	0.7787	0.6987	0.8552
LOO-NADA	**0.5979**	**0.5517**	**0.5207**	**0.5377**	**0.5415**	**0.5539**	**0.4568**	**0.4167**	**0.5124**

Table 20: CH-RR: MSE (lower is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	0.9116	0.4955	0.4388	0.4250	0.3930	0.3665	0.3524	0.3412	0.4226
TracIn-Top	1.4213	0.9271	0.7849	0.6609	0.6732	0.5776	0.4790	0.4392	0.6689
TracIn-NADA	**0.5606**	**0.4961**	**0.4284**	**0.3949**	**0.3807**	**0.3461**	**0.3277**	**0.3224**	**0.3864**
LOO-Top	1.3392	0.7852	0.7724	0.6408	0.5655	0.5383	0.5068	0.4152	0.6300
LOO-NADA	**1.1518**	**0.6346**	**0.5729**	**0.3860**	**0.3824**	**0.3434**	**0.3219**	**0.3158**	**0.4474**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.1200	0.6814	0.6853	0.5714	0.5075	0.4457	0.4189	0.4221	0.5482
TracIn-Top	4.3379	2.5523	0.8832	0.6789	0.6347	0.6002	0.5117	0.4639	1.0340
TracIn-NADA	**1.5770**	**0.9698**	**0.4103**	**0.3671**	**0.3565**	**0.3385**	**0.3274**	**0.3235**	**0.4865**
LOO-Top	1.6748	1.7461	1.1741	0.7698	0.7142	0.6463	0.4238	0.4109	0.8299
LOO-NADA	**0.9731**	**0.6574**	**0.4800**	**0.4187**	**0.3840**	**0.3724**	**0.3671**	**0.3441**	**0.4515**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.5326	0.8017	0.8714	0.6716	0.5593	0.5121	0.4852	0.5028	0.6568
TracIn-Top	4.5277	1.5862	1.1901	0.8970	0.7279	0.6040	0.4545	0.4176	0.9886
TracIn-NADA	**1.6949**	**0.5456**	**0.3604**	**0.3587**	**0.3552**	**0.3393**	**0.3422**	**0.3375**	**0.4442**
LOO-Top	2.1299	1.4274	0.8810	0.5392	0.4914	0.4305	0.3953	0.3724	0.6903
LOO-NADA	**1.0967**	**0.8051**	**0.5279**	**0.4218**	**0.3919**	**0.3618**	**0.3485**	**0.3460**	**0.4756**

Continued on next page

Table 20 (continued): CH-RR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.8219	1.1092	1.0720	0.9299	0.9205	0.8359	0.7741	0.7806	0.9472
TracIn-Top	2.1094	**0.8908**	**0.6683**	0.5352	0.5361	0.4551	0.4242	0.4252	**0.6268**
TracIn-NADA	**1.9503**	1.4920	0.7772	**0.4617**	**0.4644**	**0.4155**	**0.3808**	**0.3499**	0.6561
LOO-Top	2.0282	1.3904	**0.6144**	**0.5361**	0.4768	**0.4119**	0.3713	**0.3485**	0.6379
LOO-NADA	**1.6094**	**0.9853**	1.1460	0.5516	**0.4679**	0.4301	**0.3584**	0.3495	**0.6297**

Table 21: PP-RR: MSE (lower is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.5662	1.0745	0.9512	0.9330	0.8908	0.8632	0.8341	0.8263	0.9358
TracIn-Top	2.2460	1.5731	1.4540	1.3718	1.3221	1.2966	1.2303	1.1128	1.3736
TracIn-NADA	**1.0768**	**1.0947**	**0.8296**	**0.8077**	**0.8058**	**0.8068**	**0.7904**	**0.7942**	**0.8513**
LOO-Top	1.7447	1.0788	1.1268	1.1558	1.1391	1.0439	1.0272	1.0032	1.1107
LOO-NADA	**1.0057**	**0.8830**	**0.8320**	**0.8223**	**0.8330**	**0.8018**	**0.7837**	**0.7704**	**0.8237**
10% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.6595	1.1408	1.0147	0.9396	0.9016	0.8787	0.8568	0.8469	0.9673
TracIn-Top	1.1882	1.4459	1.3706	1.3610	1.4268	1.3643	1.1828	1.1604	1.3079
TracIn-NADA	**1.1465**	**1.2414**	**1.0217**	**0.9423**	**0.9015**	**0.8387**	**0.8336**	**0.8067**	**0.9347**
LOO-Top	1.4871	1.3615	1.2149	1.0809	1.0673	1.1153	1.0643	1.0087	1.1420
LOO-NADA	**1.4690**	**0.9061**	**0.8539**	**0.8318**	**0.8403**	**0.8304**	**0.7984**	**0.7805**	**0.8655**
20% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.7747	1.2800	1.1301	1.0259	0.9997	0.9579	0.9279	0.9319	1.0608
TracIn-Top	1.3608	1.5244	1.5867	1.4833	1.4067	1.4255	1.2501	1.2251	1.3939
TracIn-NADA	**0.9237**	**1.0156**	**1.0266**	**1.0325**	**1.0138**	**0.9499**	**0.8402**	**0.8267**	**0.9425**
LOO-Top	3.3825	1.0615	1.0252	1.0784	1.0726	1.0350	0.9184	0.8792	1.1330
LOO-NADA	**1.1812**	**0.8785**	**0.8411**	**0.8190**	**0.8253**	**0.8229**	**0.7962**	**0.7983**	**0.8422**
30% corruption; $n = 200$, $n_{\text{eval}} = 2,000$									
Random	1.7690	1.3321	1.2334	1.1559	1.1813	1.1573	1.0961	1.0931	1.2010
TracIn-Top	5.3038	1.5903	1.3703	1.2940	1.3313	1.3225	1.3137	1.2765	1.5695
TracIn-NADA	**1.0511**	**1.0380**	**0.9848**	**0.9658**	**0.9583**	**0.9207**	**0.8840**	**0.8619**	**0.9426**
LOO-Top	1.6816	0.9370	0.8414	0.8459	0.8270	0.8042	0.8195	**0.8168**	0.8838
LOO-NADA	**1.1557**	**0.8079**	**0.7981**	**0.8005**	**0.7986**	**0.7979**	**0.8028**	0.8194	**0.8228**

Table 22: AU-RR: MSE (lower is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
Clean; $n = 200$, $n_{\text{eval}} = 1,843$									
Random	0.9364	0.7687	0.6292	0.5429	0.4983	0.4760	0.4363	0.4160	0.5455
TracIn-Top	2.2919	1.5416	0.9691	**0.6599**	0.7368	0.6732	0.6767	0.5545	0.8857
TracIn-NADA	**0.6598**	**0.6207**	**0.5103**	0.6754	**0.5256**	**0.4638**	**0.4000**	**0.3977**	**0.5061**
LOO-Top	1.4001	1.0662	**0.8299**	1.0785	0.7773	0.6861	0.5675	0.5480	0.7960
LOO-NADA	**0.9390**	**0.6320**	0.9482	**0.8696**	**0.5403**	**0.4829**	**0.4174**	**0.3964**	**0.6017**

Continued on next page

Table 22 (continued): AU-RR

Selector	5%	10%	15%	20%	25%	30%	40%	50%	CA
10% corruption; $n = 200$, $n_{\text{eval}} = 1,843$									
Random	1.5423	1.2803	1.0572	1.0027	0.8066	0.7027	0.5949	0.5719	0.8593
TracIn-Top	7.7563	3.2387	1.7997	1.0897	0.9728	0.7719	0.6521	0.5708	1.5569
TracIn-NADA	**0.7806**	**0.8034**	**0.8776**	**0.6320**	**0.6094**	**0.5117**	**0.5095**	**0.4891**	**0.6210**
LOO-Top	4.6759	3.1478	1.1087	0.8295	0.7694	0.7810	0.7151	0.6238	1.2687
LOO-NADA	**1.5189**	**0.8236**	**1.0801**	**0.6411**	**0.5974**	**0.6226**	**0.4867**	**0.4609**	**0.6966**
20% corruption; $n = 200$, $n_{\text{eval}} = 1,843$									
Random	1.7815	1.5288	1.2386	1.0645	0.8403	0.7823	0.6686	0.6392	0.9681
TracIn-Top	7.1816	3.1503	0.9807	0.8375	0.6691	0.6711	0.5693	0.5072	1.3201
TracIn-NADA	**1.9002**	**0.8837**	**0.6429**	**0.6235**	**0.5689**	**0.5387**	**0.4775**	**0.4418**	**0.6527**
LOO-Top	4.3977	3.3339	1.1253	0.6262	**0.6454**	0.6671	0.5566	0.5213	1.1739
LOO-NADA	**2.3187**	**0.5354**	**0.4899**	**0.6027**	0.6554	**0.5525**	**0.4852**	**0.4544**	**0.6329**
30% corruption; $n = 200$, $n_{\text{eval}} = 1,843$									
Random	1.8611	1.5924	1.3746	1.2806	0.9945	0.9426	0.8037	0.7580	1.1058
TracIn-Top	7.4328	3.6386	1.7283	0.9540	**0.5736**	**0.5396**	0.5111	0.4978	1.4378
TracIn-NADA	**1.8850**	**0.8920**	**1.2992**	**0.8855**	0.7673	0.5683	**0.4889**	**0.4403**	**0.7841**
LOO-Top	7.3722	**1.7592**	0.9185	0.8089	0.6809	**0.5361**	**0.5112**	**0.4893**	1.1299
LOO-NADA	**1.8479**	2.2129	**0.8755**	**0.6154**	**0.5842**	0.5741	0.5401	0.5073	**0.8512**

C.6 Full Primary Fine-Tuning Results

The following tables report all seven budgets for the 23 primary fine-tuning conditions used in the main text. They contain 1,127 accuracy values across seven selectors, together with Random's per-budget standard deviations. Conditions are grouped by task and model to make clean and corrupted training pools directly comparable.

Table 23: AMAZON-BT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	53.54	77.90	83.56	87.70	88.18	88.58	88.74	85.25
Random SD	0.46	0.89	0.37	0.24	0.41	0.32	0.35	—
IF-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
IF-NADA	**52.30**	**81.50**	**85.40**	**87.10**	**88.00**	**88.50**	**89.10**	**85.62**
TracIn-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
TracIn-NADA	**52.10**	**80.50**	**84.70**	**86.80**	**88.00**	**88.50**	**88.50**	**85.30**
LOO-Top	**56.70**	49.80	49.80	49.80	49.80	49.80	49.80	50.09
LOO-NADA	52.30	**81.30**	**84.90**	**87.30**	**88.70**	**88.80**	**89.10**	**85.78**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	53.32	75.16	80.80	85.04	86.64	87.34	87.52	83.35
Random SD	0.27	1.01	0.43	0.39	0.26	0.34	0.42	—
IF-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
IF-NADA	**52.60**	**78.10**	**81.60**	**84.20**	**84.90**	**85.60**	**85.50**	**82.57**
TracIn-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
TracIn-NADA	**52.10**	**78.00**	**81.70**	**83.10**	**85.40**	**85.20**	**86.00**	**82.40**
LOO-Top	**57.30**	49.80	49.80	49.80	49.80	49.80	49.80	50.11
LOO-NADA	52.20	**78.20**	**82.70**	**85.10**	**85.50**	**86.00**	**86.10**	**83.18**

Continued on next page

Table 23 (continued): AMAZON-BT

Selector	2%	6%	10%	20%	30%	40%	50%	CA
20% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	53.08	72.28	76.26	81.20	83.24	84.78	85.80	80.21
Random SD	0.40	1.19	1.10	0.54	0.90	0.76	0.66	—
IF-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
IF-NADA	**52.30**	**72.40**	**76.30**	**79.80**	**81.60**	**82.50**	**82.80**	**78.78**
TracIn-Top	50.20	50.20	50.20	50.20	50.20	50.20	50.20	50.20
TracIn-NADA	**52.00**	**73.10**	**77.20**	**80.30**	**82.10**	**81.20**	**81.80**	**78.79**
LOO-Top	**57.40**	49.80	49.80	49.80	49.80	49.80	49.80	50.12
LOO-NADA	52.10	**74.10**	**77.60**	**80.80**	**82.70**	**83.20**	**83.60**	**79.77**

Table 24: MR-BT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 8,530$, $n_{\text{eval}} = 1,066$								
Random	61.89	72.25	74.77	76.74	77.54	77.97	78.80	76.10
Random SD	2.70	0.53	0.24	0.43	0.32	0.30	0.45	—
IF-Top	51.78	56.00	61.26	65.38	69.51	73.36	75.14	66.97
IF-NADA	**57.97**	**68.95**	**73.17**	**76.27**	**77.77**	**77.67**	**77.86**	**75.22**
TracIn-Top	50.56	50.84	54.22	57.04	62.38	66.42	70.17	60.28
TracIn-NADA	**58.26**	**69.89**	**74.20**	**75.98**	**77.49**	**77.67**	**77.95**	**75.35**
LOO-Top	50.38	50.00	50.00	50.00	50.00	50.00	50.00	50.02
LOO-NADA	**56.85**	**69.32**	**73.17**	**76.64**	**77.58**	**78.42**	**78.42**	**75.45**
10% corruption; $n = 8,530$, $n_{\text{eval}} = 1,066$								
Random	67.30	69.85	71.07	73.40	74.58	75.12	75.57	73.34
Random SD	1.19	0.66	1.12	0.83	0.53	0.59	0.42	—
IF-Top	52.91	58.54	62.29	67.73	72.05	74.86	**76.55**	68.86
IF-NADA	**58.44**	**70.45**	**73.36**	**74.48**	**75.61**	**76.17**	76.36	**74.10**
TracIn-Top	50.56	51.13	56.19	60.88	65.01	70.08	73.26	63.02
TracIn-NADA	**58.26**	**70.17**	**72.70**	**73.26**	**73.83**	**74.11**	**75.05**	**72.78**
LOO-Top	50.19	50.00	50.00	50.00	50.00	50.00	51.69	50.18
LOO-NADA	**58.07**	**68.95**	**73.26**	**75.23**	**75.61**	**75.70**	**75.70**	**73.93**
20% corruption; $n = 8,530$, $n_{\text{eval}} = 1,066$								
Random	68.95	64.09	63.77	64.11	64.52	64.58	64.67	64.50
Random SD	0.47	0.99	1.30	0.84	0.63	1.07	0.48	—
IF-Top	54.41	62.76	66.23	70.92	**74.39**	**74.77**	**75.14**	70.83
IF-NADA	**58.16**	**69.14**	**73.08**	**72.80**	72.80	72.51	73.64	**71.95**
TracIn-Top	50.66	53.75	57.50	64.82	70.26	**75.14**	**76.55**	66.75
TracIn-NADA	**58.26**	**69.61**	**72.05**	**71.58**	**71.20**	71.86	73.17	**71.07**
LOO-Top	50.19	50.00	50.00	50.00	50.00	50.84	54.60	50.66
LOO-NADA	**57.41**	**69.14**	**72.42**	**72.61**	**73.73**	**75.05**	**75.14**	**72.67**

Table 25: YELP-BT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	53.84	75.90	82.40	88.16	89.70	90.64	91.22	86.02
Random SD	0.85	0.75	1.07	0.44	0.68	0.31	0.17	—
IF-Top	**61.80**	51.80	51.80	51.80	51.80	51.80	51.80	52.22
IF-NADA	57.80	**77.70**	**83.10**	**86.60**	**89.90**	**91.20**	**91.70**	**86.33**
TracIn-Top	**61.90**	51.80	51.80	51.80	51.80	51.80	51.80	52.22
TracIn-NADA	56.80	**77.90**	**82.80**	**86.70**	**89.70**	**90.90**	**92.00**	**86.20**
LOO-Top	**62.10**	51.80	51.80	51.80	51.80	51.80	51.80	52.23
LOO-NADA	56.90	**78.10**	**83.20**	**87.60**	**89.10**	**90.30**	**91.20**	**86.14**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	54.32	73.40	78.96	85.68	87.60	88.50	89.30	83.73
Random SD	1.79	1.07	0.95	0.68	0.64	0.28	0.25	—
IF-Top	**61.70**	51.80	51.80	51.80	51.80	51.80	51.80	52.21
IF-NADA	57.40	**74.60**	**78.20**	**80.70**	**83.00**	**88.20**	**89.20**	**81.78**
TracIn-Top	**61.90**	51.80	51.80	51.80	51.80	51.80	51.80	52.22
TracIn-NADA	57.10	**74.80**	**78.30**	**80.70**	**83.50**	**87.40**	**89.00**	**81.72**
LOO-Top	**62.20**	51.80	51.80	51.80	51.80	51.80	51.80	52.23
LOO-NADA	56.90	**74.50**	**81.60**	**84.50**	**83.40**	**84.60**	**87.10**	**82.16**

Table 26: MNLI-LM: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	33.56	37.60	41.16	52.92	58.32	63.72	68.46	54.12
Random SD	0.39	1.13	1.13	1.01	0.50	0.53	0.75	—
IF-Top	31.70	32.10	32.00	32.00	31.90	31.90	33.10	32.07
IF-NADA	**33.10**	**36.80**	**43.00**	**52.60**	**58.80**	**65.00**	**70.80**	**54.84**
TracIn-Top	31.70	32.10	32.00	31.90	31.90	32.00	39.20	32.70
TracIn-NADA	**33.20**	**37.60**	**39.00**	**44.00**	**56.50**	**65.10**	**70.10**	**52.01**
LOO-Top	33.20	34.30	34.30	43.70	51.00	52.50	52.80	45.41
LOO-NADA	**33.40**	**35.80**	**42.90**	**55.90**	**61.30**	**65.00**	**69.40**	**55.82**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	33.42	36.60	40.42	50.32	55.24	59.54	63.58	51.36
Random SD	0.32	1.02	1.22	1.79	0.50	0.65	0.60	—
IF-Top	31.90	32.10	32.00	32.00	31.90	31.90	31.90	31.95
IF-NADA	**32.70**	**37.20**	**40.00**	**51.50**	**56.20**	**61.00**	**65.50**	**52.26**
TracIn-Top	32.30	32.10	32.00	31.90	31.90	31.90	36.70	32.45
TracIn-NADA	**33.80**	**37.40**	**38.40**	**42.30**	**52.30**	**58.20**	**63.80**	**48.60**
LOO-Top	33.10	34.30	34.80	43.00	50.10	51.70	53.40	45.04
LOO-NADA	**33.60**	**36.00**	**40.20**	**53.70**	**57.60**	**62.20**	**64.30**	**53.11**
20% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	33.52	36.46	38.82	46.52	52.66	55.72	59.46	48.56
Random SD	0.26	0.70	0.86	1.37	0.63	0.35	1.63	—
IF-Top	31.80	32.10	32.00	32.00	31.90	31.90	31.80	31.94
IF-NADA	**33.10**	**37.40**	**38.60**	**49.80**	**53.30**	**57.70**	**60.50**	**49.93**
TracIn-Top	31.40	32.10	32.00	31.90	31.80	31.90	31.90	31.89
TracIn-NADA	**32.70**	**37.20**	**38.40**	**41.50**	**49.80**	**54.30**	**58.90**	**46.53**
LOO-Top	32.90	34.30	34.80	43.10	49.80	50.60	52.40	44.66
LOO-NADA	**33.50**	**35.60**	**38.20**	**50.10**	**55.50**	**60.00**	**61.40**	**50.83**

Table 27: MR-LM: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 8,530$, $n_{\text{eval}} = 1,066$								
Random	72.38	76.87	89.31	92.29	92.81	93.25	93.25	90.15
Random SD	0.22	1.53	0.61	0.24	0.44	0.16	0.23	—
IF-Top	69.04	56.47	78.99	89.12	88.65	88.09	89.49	83.81
IF-NADA	**71.67**	**74.58**	**89.21**	**92.03**	**92.59**	**92.96**	**93.15**	**89.74**
TracIn-Top	68.86	57.69	79.46	88.37	88.65	88.56	89.77	83.94
TracIn-NADA	**71.67**	**75.05**	**89.40**	**92.31**	**92.96**	**92.59**	**92.96**	**89.85**
LOO-Top	**72.98**	50.00	50.00	50.00	50.00	50.00	50.00	50.96
LOO-NADA	71.86	**74.86**	**89.59**	**92.21**	**92.87**	**92.78**	**93.25**	**89.90**

Table 28: RT-LM: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 2,490$, $n_{\text{eval}} = 277$								
Random	51.41	51.34	51.41	52.64	54.73	57.33	59.49	54.43
Random SD	0.18	0.14	0.18	0.42	1.95	0.90	1.24	—
IF-Top	51.26	51.26	51.26	51.26	50.54	50.90	52.35	51.15
IF-NADA	51.26	51.26	**51.62**	**53.79**	**54.51**	**59.93**	**60.65**	**55.30**
TracIn-Top	51.26	51.26	51.26	51.26	50.54	50.90	52.35	51.15
TracIn-NADA	**51.62**	**51.62**	**51.62**	**52.71**	**55.96**	**57.40**	**61.01**	**54.93**
LOO-Top	51.26	51.26	51.26	51.26	50.54	51.26	52.35	51.23
LOO-NADA	**51.62**	51.26	**51.62**	**54.51**	**54.87**	**58.48**	**63.18**	**55.51**
10% corruption; $n = 2,490$, $n_{\text{eval}} = 277$								
Random	51.26	51.26	51.34	53.36	55.16	56.82	57.62	54.34
Random SD	0.00	0.00	0.14	0.90	1.64	0.87	0.93	—
IF-Top	51.26	51.26	51.26	51.26	50.54	51.26	51.99	51.19
IF-NADA	51.26	**51.62**	51.26	**52.35**	**54.87**	**58.12**	**58.48**	**54.45**
TracIn-Top	51.26	51.26	51.26	51.26	50.54	50.90	52.35	51.15
TracIn-NADA	51.26	51.26	51.26	**53.43**	**56.32**	**55.23**	**57.76**	**54.27**
LOO-Top	51.26	51.26	51.26	51.26	50.54	51.26	52.35	51.23
LOO-NADA	**51.62**	51.26	**51.62**	**53.79**	**55.23**	**58.12**	**61.73**	**55.21**
20% corruption; $n = 2,490$, $n_{\text{eval}} = 277$								
Random	51.26	51.34	51.55	53.65	55.09	55.31	55.67	53.91
Random SD	0.00	0.14	0.14	0.93	0.78	1.08	1.01	—
IF-Top	51.26	51.26	51.26	51.26	50.54	51.26	52.35	51.23
IF-NADA	51.26	51.26	51.26	**52.35**	**54.51**	**54.87**	**55.60**	**53.37**
TracIn-Top	51.26	51.26	51.26	51.26	50.54	51.62	52.35	51.30
TracIn-NADA	51.26	51.26	**51.62**	**53.07**	**53.79**	**57.40**	**56.32**	**54.02**
LOO-Top	51.26	51.26	51.26	51.26	50.54	51.26	52.35	51.23
LOO-NADA	51.26	**51.62**	**51.62**	**55.23**	**51.99**	**55.23**	**57.40**	**53.79**

Table 29: SST2-LM: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	69.40	85.21	94.50	95.25	95.48	95.57	95.76	93.39
Random SD	0.20	0.60	0.16	0.06	0.12	0.16	0.24	—
IF-Top	63.65	50.92	50.92	50.92	50.92	50.92	50.92	51.45
IF-NADA	**73.17**	**86.12**	**94.50**	**95.41**	**95.76**	**95.76**	**95.87**	**93.77**
TracIn-Top	64.22	50.92	50.92	50.92	50.92	50.92	50.92	51.47
TracIn-NADA	**73.17**	**84.98**	**94.84**	**95.41**	**95.64**	**95.99**	**95.87**	**93.75**
LOO-Top	63.65	50.92	50.92	50.92	50.92	50.92	50.92	51.45
LOO-NADA	**72.82**	**83.83**	**93.58**	**95.53**	**95.76**	**95.76**	**95.76**	**93.44**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	69.27	83.33	93.23	94.68	94.98	95.34	95.39	92.74
Random SD	0.42	1.46	0.51	0.39	0.20	0.22	0.24	—
IF-Top	63.30	50.92	50.92	50.92	50.92	50.92	50.92	51.43
IF-NADA	**72.71**	**80.96**	**93.69**	**94.50**	**95.07**	**94.95**	**95.41**	**92.65**
TracIn-Top	63.30	50.92	50.92	50.92	50.92	50.92	50.92	51.43
TracIn-NADA	**72.82**	**80.73**	**93.58**	**94.50**	**94.50**	**94.72**	**95.41**	**92.46**
LOO-Top	63.76	50.92	50.92	50.92	50.92	50.92	50.92	51.45
LOO-NADA	**72.48**	**81.65**	**93.12**	**94.61**	**95.18**	**95.18**	**95.18**	**92.69**
20% corruption; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	69.82	80.32	90.96	93.49	94.22	94.59	94.66	91.54
Random SD	0.48	0.78	1.47	0.78	0.43	0.46	0.41	—
IF-Top	63.42	50.92	50.92	50.92	50.92	50.92	50.92	51.44
IF-NADA	**72.36**	**80.96**	**91.86**	**93.58**	**94.04**	**94.27**	**94.38**	**91.71**
TracIn-Top	63.07	50.92	50.92	50.92	50.92	50.92	50.92	51.42
TracIn-NADA	**72.71**	**80.62**	**91.51**	**93.58**	**94.04**	**94.72**	**94.95**	**91.81**
LOO-Top	63.19	50.92	50.92	50.92	50.92	50.92	50.92	51.43
LOO-NADA	**72.48**	**81.42**	**91.51**	**92.78**	**94.27**	**94.15**	**95.07**	**91.64**

Table 30: AMAZON-GPT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	94.30	94.84	95.34	95.50	95.72	96.14	96.22	95.63
Random SD	0.17	0.39	0.26	0.09	0.16	0.14	0.04	—
IF-Top	85.40	53.10	54.80	87.90	92.50	94.00	95.20	83.06
IF-NADA	**94.20**	**94.50**	**94.70**	**95.50**	**95.90**	**96.20**	**96.20**	**95.55**
TracIn-Top	**94.50**	93.80	93.70	94.10	94.30	94.60	94.60	94.23
TracIn-NADA	94.00	**94.70**	**94.50**	**95.60**	**96.40**	**96.10**	**96.40**	**95.65**
LOO-Top	84.90	63.70	50.80	50.40	50.20	50.20	65.20	54.46
LOO-NADA	**93.70**	**94.70**	**95.00**	**95.00**	**95.70**	**95.80**	**96.00**	**95.34**

Table 31: ANLI-GPT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	39.06	57.80	62.46	69.14	71.64	73.88	75.16	68.10
Random SD	1.05	0.70	1.28	0.98	1.14	0.92	0.26	—
IF-Top	36.10	33.30	33.30	33.30	33.30	54.20	40.10	38.48
IF-NADA	**40.10**	**57.10**	**62.50**	**69.70**	**72.40**	**74.80**	**75.30**	**68.58**
TracIn-Top	**45.60**	53.00	55.30	56.20	57.90	58.40	58.20	56.38
TracIn-NADA	40.60	**55.40**	**62.20**	**70.60**	**71.00**	**73.80**	**75.20**	**68.09**
LOO-Top	37.20	33.30	33.30	33.30	36.40	53.40	44.50	39.46
LOO-NADA	**41.80**	**58.30**	**62.20**	**69.80**	**72.20**	**73.40**	**75.30**	**68.39**

Table 32: MNLI-GPT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	53.40	71.24	76.78	81.90	85.00	86.34	87.06	81.19
Random SD	0.89	1.12	0.75	1.01	0.62	0.62	0.24	—
IF-Top	36.40	34.00	34.00	34.00	34.00	38.90	62.00	38.04
IF-NADA	**50.20**	**68.60**	**75.80**	**80.40**	**83.60**	**85.90**	**86.80**	**79.97**
TracIn-Top	36.10	34.00	34.00	34.00	34.00	52.00	65.60	41.13
TracIn-NADA	**48.20**	**69.60**	**75.40**	**80.30**	**83.00**	**85.80**	**87.40**	**79.80**
LOO-Top	35.90	34.00	34.00	34.00	34.00	38.30	61.70	37.86
LOO-NADA	**50.70**	**69.70**	**76.50**	**80.90**	**83.70**	**85.80**	**87.80**	**80.39**

Table 33: RT-GPT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 2,490$, $n_{\text{eval}} = 277$								
Random	62.74	63.25	71.26	82.17	85.27	86.43	88.01	80.33
Random SD	0.27	2.11	0.74	0.67	0.80	0.81	0.87	—
IF-Top	60.65	**60.29**	53.07	52.71	52.71	52.71	52.71	53.72
IF-NADA	**62.45**	57.04	**71.84**	**85.56**	**84.84**	**87.73**	**88.81**	**80.86**
TracIn-Top	61.37	**59.57**	53.07	52.71	52.71	52.71	52.71	53.69
TracIn-NADA	**62.82**	55.96	**72.56**	**82.67**	**82.31**	**86.64**	**87.36**	**79.39**
LOO-Top	62.82	**61.37**	53.07	52.71	52.71	52.71	52.71	53.90
LOO-NADA	**64.62**	57.40	**70.40**	**84.48**	**86.28**	**88.09**	**89.17**	**80.96**

Table 34: SST2-GPT: accuracy (%, higher is better). Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	90.23	92.55	93.14	94.17	94.79	95.05	95.07	94.13
Random SD	0.50	0.39	0.37	0.30	0.20	0.17	0.10	—
IF-Top	65.02	51.49	50.92	50.92	50.92	50.92	50.92	51.55
IF-NADA	**90.83**	**92.66**	**93.12**	**93.81**	**94.04**	**95.18**	**95.53**	**94.00**
TracIn-Top	63.19	51.03	51.03	50.92	50.92	50.92	50.92	51.45
TracIn-NADA	**90.48**	**92.43**	**92.43**	**93.81**	**94.04**	**94.72**	**94.84**	**93.70**
LOO-Top	65.25	52.87	50.92	50.92	50.92	50.92	50.92	51.68
LOO-NADA	**90.48**	**92.55**	**93.00**	**93.12**	**94.38**	**95.07**	**95.30**	**93.84**

C.7 Additional Recorded Fine-Tuning Conditions

The detailed result files also contain 11 conditions outside the 23-condition primary summary: SST2-BT and YELP-LM at 0%, 10%, and 20% corruption; AMAZON-GPT at 10% and 20%; ANLI-GPT at 10%; and RT-GPT at 10% and 20%. The following tables report their available primary-family measurements without incorporating them into the main-text aggregate statistics.

For ANLI-GPT at 10% corruption, Random and TracIn-NADA have results only at 2%, 6%, and 10% selection. Their remaining budgets and full-range CAs are therefore left blank with dashes. For RT-GPT at 10% and 20% corruption, only the influence-function pair is available. The other eight additional conditions have complete primary-family results at all seven budgets.

Table 35: SST2-BT: accuracy (%, higher is better). Additional conditions outside the primary summary. Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	50.96	65.11	75.09	80.96	83.46	84.22	84.31	79.08
Random SD	0.06	3.34	2.72	1.11	0.90	0.28	0.34	—
IF-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
IF-NADA	**57.00**	**67.66**	**70.76**	**71.79**	**73.62**	**74.31**	**75.69**	**71.99**
TracIn-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
TracIn-NADA	**57.57**	**70.41**	**73.74**	**71.90**	**71.56**	**71.90**	**75.69**	**71.77**
LOO-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
LOO-NADA	**57.45**	**68.35**	**73.51**	**75.34**	**75.69**	**75.34**	**76.03**	**73.89**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	51.01	62.71	70.85	78.83	82.04	82.98	83.51	77.18
Random SD	0.05	3.53	3.58	1.79	1.14	0.50	0.43	—
IF-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
IF-NADA	**57.68**	**68.00**	**73.74**	**72.36**	**71.33**	**70.99**	**70.99**	**70.94**
TracIn-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
TracIn-NADA	**57.11**	**68.23**	**70.76**	**71.44**	**68.92**	**69.50**	**71.10**	**69.51**
LOO-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
LOO-NADA	**57.11**	**65.14**	**70.18**	**75.23**	**73.51**	**74.89**	**74.08**	**72.35**

Continued on next page

Table 35 (continued): SST2-BT

Selector	2%	6%	10%	20%	30%	40%	50%	CA
20% corruption; $n = 10,000$, $n_{\text{eval}} = 872$								
Random	51.24	61.90	68.12	73.90	78.62	80.32	81.33	74.21
Random SD	0.26	3.65	4.06	1.92	1.15	0.75	0.63	—
IF-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
IF-NADA	**57.45**	**67.78**	**73.39**	**72.25**	**70.41**	**68.00**	**68.46**	**69.77**
TracIn-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
TracIn-NADA	**57.22**	**67.89**	**72.48**	**70.18**	**68.69**	**66.63**	**68.00**	**68.51**
LOO-Top	50.92	50.92	50.92	50.92	50.92	50.92	50.92	50.92
LOO-NADA	**57.00**	**66.51**	**72.36**	**73.39**	**72.36**	**72.59**	**72.36**	**71.50**

Table 36: YELP-LM: accuracy (%, higher is better). Additional conditions outside the primary summary. Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
Clean; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	72.78	77.22	78.12	79.92	84.78	94.64	95.22	84.81
Random SD	0.19	0.39	0.20	0.22	0.47	0.14	0.21	—
IF-Top	70.40	67.70	61.80	51.90	51.80	51.80	70.30	57.31
IF-NADA	**73.10**	**77.60**	**78.10**	**79.10**	**79.60**	**82.90**	**93.50**	**80.98**
TracIn-Top	70.50	67.60	61.90	51.80	51.80	51.80	70.30	57.30
TracIn-NADA	**73.20**	**77.60**	**78.10**	**79.00**	**79.60**	**83.00**	**93.40**	**80.97**
LOO-Top	70.40	67.60	61.90	51.80	51.80	51.80	51.80	55.36
LOO-NADA	**72.90**	**77.60**	**77.90**	**79.00**	**79.10**	**83.90**	**94.00**	**81.07**
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	72.64	76.62	77.86	80.00	83.00	91.40	94.16	83.57
Random SD	0.27	0.41	0.37	0.17	0.34	0.83	0.30	—
IF-Top	70.40	67.70	62.00	51.80	51.80	51.80	56.30	55.86
IF-NADA	**73.00**	**76.70**	**77.30**	**78.30**	**78.40**	**81.00**	**86.60**	**79.25**
TracIn-Top	70.20	67.30	62.00	51.80	51.80	51.80	56.70	55.86
TracIn-NADA	**72.80**	**77.10**	**77.70**	**78.50**	**78.80**	**81.20**	**86.90**	**79.53**
LOO-Top	70.20	67.70	61.90	51.90	51.80	51.80	51.80	55.39
LOO-NADA	**72.90**	**76.60**	**77.60**	**78.10**	**78.20**	**81.20**	**86.70**	**79.25**
20% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	72.78	75.50	76.96	79.16	80.48	85.98	90.70	81.17
Random SD	0.37	1.23	0.73	0.15	0.30	0.78	0.66	—
IF-Top	70.20	67.60	61.90	51.80	51.80	52.50	55.90	55.93
IF-NADA	**72.90**	**75.00**	**76.60**	**78.00**	**78.10**	**79.30**	**81.50**	**77.99**
TracIn-Top	70.50	67.50	61.90	51.80	51.80	52.70	56.20	56.01
TracIn-NADA	**73.10**	**75.30**	**77.00**	**78.30**	**78.40**	**79.70**	**81.10**	**78.25**
LOO-Top	70.30	67.90	61.80	51.90	51.80	51.80	51.80	55.39
LOO-NADA	**72.80**	**75.90**	**77.10**	**77.70**	**78.20**	**79.40**	**81.40**	**78.10**

Table 37: AMAZON-GPT: accuracy (%, higher is better). Additional conditions outside the primary summary. Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	93.42	93.84	93.98	94.54	95.06	95.44	95.58	94.76
Random SD	0.26	0.33	0.22	0.34	0.39	0.31	0.12	—
IF-Top	82.80	50.30	50.20	59.70	75.90	83.90	86.60	69.71
IF-NADA	**93.70**	**93.20**	**93.40**	**93.50**	**94.10**	**94.50**	**94.80**	**93.94**
TracIn-Top	94.20	**93.40**	92.30	91.30	90.90	91.00	92.00	91.67
TracIn-NADA	**94.50**	93.20	**93.00**	**94.50**	**94.70**	**95.40**	**95.70**	**94.53**
LOO-Top	85.20	50.40	50.20	50.20	50.20	50.20	50.20	51.68
LOO-NADA	**93.90**	**93.50**	**93.60**	**93.80**	**94.60**	**94.60**	**94.70**	**94.18**
20% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	92.16	91.72	92.10	92.68	93.20	93.72	94.42	93.00
Random SD	1.09	0.62	0.32	0.47	0.60	0.61	0.77	—
IF-Top	78.10	50.20	50.20	50.20	62.50	76.20	81.60	62.61
IF-NADA	**93.50**	**88.90**	**90.90**	**90.30**	**92.10**	**92.60**	**93.00**	**91.54**
TracIn-Top	**94.20**	**91.70**	**90.30**	89.00	87.40	86.90	88.00	88.76
TracIn-NADA	93.50	91.20	89.20	**91.50**	**92.00**	**92.30**	**93.40**	**91.69**
LOO-Top	83.60	50.20	50.20	50.20	50.20	50.20	50.20	51.59
LOO-NADA	**93.80**	**91.50**	**91.00**	**91.70**	**92.60**	**93.10**	**93.80**	**92.37**

Table 38: ANLI-GPT: accuracy (%, higher is better). Additional conditions outside the primary summary. Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
10% corruption; $n = 10,000$, $n_{\text{eval}} = 1,000$								
Random	39.46	54.54	59.38	—	—	—	—	—
Random SD	1.24	1.32	0.96	—	—	—	—	—
IF-Top	35.80	33.30	33.30	33.30	33.30	44.80	33.70	35.84
IF-NADA	**39.10**	**54.10**	**59.70**	**67.50**	**70.80**	**71.20**	**73.80**	**66.18**
TracIn-Top	38.20	51.00	54.60	55.80	56.70	56.60	56.60	54.93
TracIn-NADA	**38.80**	**54.90**	**57.30**	—	—	—	—	—
LOO-Top	36.80	33.30	33.30	33.30	33.30	50.60	33.70	37.09
LOO-NADA	**43.30**	**55.60**	**58.90**	**66.40**	**69.10**	**70.90**	**69.10**	**65.22**

Table 39: RT-GPT: accuracy (%, higher is better). Additional conditions outside the primary summary. Columns give nominal selection percentages; CA is the normalized curve average.

Selector	2%	6%	10%	20%	30%	40%	50%	CA
10% corruption; $n = 2,490$, $n_{\text{eval}} = 277$								
Random	—	—	—	—	—	—	—	—
IF-Top	60.65	59.21	53.07	52.71	52.71	52.71	52.71	53.63
IF-NADA	**62.09**	59.21	**76.17**	**80.14**	**85.20**	**85.20**	**85.56**	**79.74**
TracIn-Top	—	—	—	—	—	—	—	—
TracIn-NADA	—	—	—	—	—	—	—	—
LOO-Top	—	—	—	—	—	—	—	—
LOO-NADA	—	—	—	—	—	—	—	—

Continued on next page

Table 39 (continued): RT-GPT

Selector	2%	6%	10%	20%	30%	40%	50%	CA
20% corruption; $n = 2,490$*,* $n_{\text{eval}} = 277$								
Random	—	—	—	—	—	—	—	—
IF-Top	62.82	**58.84**	52.71	52.71	52.71	52.71	52.71	53.64
IF-NADA	62.82	58.12	**69.68**	**74.73**	**80.87**	**82.67**	**83.39**	**75.95**
TracIn-Top	—	—	—	—	—	—	—	—
TracIn-NADA	—	—	—	—	—	—	—	—
LOO-Top	—	—	—	—	—	—	—	—
LOO-NADA	—	—	—	—	—	—	—	—

C.8 Attribution and Selection Configurations

Tabular selection settings. The logistic-regression and ridge-regression tasks uses exponential aggregation, selection strength 2.5, global max-absolute normalization, threshold 0, `nada_quality_alpha`=0, and `component_weights=none`. We denote the stored selection-strength parameter by λ. The aggregation choice corresponds to $F_\lambda(x) = 1 - \exp(-\lambda x)$ with $\lambda = 2.5$. The quality coefficient is zero, so no additional scalar-quality term is configured.

Precomputed fine-tuning orders. The fine-tuning reports record selection strength 2.5 and load method-specific selections from condition-specific order archives, with paths of the form `results/precomputed/<condition>/<benchmark>.npz`. The condition directories are `clean`, `flip10`, and `flip20`. Within each reported condition, the primary selectors share the same recorded order-source path and fine-tuning settings. The saved maximum prefix size is the 50% selection budget: 5,000, 4,265, or 1,245, depending on the task.

D Instantiation details

This appendix gives the formal component matrices used by the NADA instantiations in Section 4. All matrices follow the utility-oriented sign convention introduced in Section 3: positive entries represent helpful effects on the corresponding evaluation component.

D.1 Influence-Function Components

Let

$$R_N(\theta) = \frac{1}{n} \sum_{j \in N} \ell(z_j, \theta), \qquad \widehat{\theta} \in \arg\min_\theta R_N(\theta),$$

and define

$$g_i = \nabla_\theta \ell(z_i, \widehat{\theta}), \qquad H_{\widehat{\theta}} = \nabla^2_\theta R_N(\widehat{\theta}).$$

For an evaluation component with differentiable loss $\ell_c(\theta)$, let

$$g_c = \nabla_\theta \ell_c(\widehat{\theta}).$$

To define the influence function, infinitesimally upweight training example i:

$$\widehat{\theta}_{\epsilon,i} \in \arg\min_\theta \{R_N(\theta) + \epsilon \ell(z_i, \theta)\}.$$

Under the standard differentiability and invertibility conditions, its parameter influence is

$$\mathcal{I}_{\text{param}}(i) := \left. \frac{d\widehat{\theta}_{\epsilon,i}}{d\epsilon} \right|_{\epsilon=0} = -H^{-1}_{\widehat{\theta}} g_i.$$

The corresponding effect on component loss is

$$I_{\text{loss}}(i, c) = g_c^\top \mathcal{I}_{\text{param}}(i) = -g_c^\top H^{-1}_{\widehat{\theta}} g_i.$$

Because NADA uses utility-oriented scores, we reverse the loss-influence sign and define

$$A^{\mathrm{IF}}_{i,c} = -I_{\mathrm{loss}}(i,c) = g_c^\top H_{\widehat{\theta}}^{-1} g_i. \tag{14}$$

In practice, we use the damped approximation

$$A^{\mathrm{IF}}_{i,c} \approx g_c^\top \left(H_{\widehat{\theta}} + \rho I\right)^{-1} g_i, \qquad \rho \geq 0. \tag{15}$$

Thus, $A^{\mathrm{IF}}_{i,c} > 0$ means that upweighting example i is locally estimated to reduce the loss on component c.

Influence-NADA applies the generic NADA objective to A^{IF}. Its accumulated component proxy is

$$\widehat{u}^{\mathrm{IF}}_c(\mathcal{M}) = \sum_{i \in \mathcal{M}} A^{\mathrm{IF}}_{i,c}.$$

D.2 TracIn Components

Let

$$\theta_1, \ldots, \theta_T$$

be saved checkpoints from the model-training trajectory, and let η_t denote the learning rate associated with checkpoint t. Define

$$g_i^{(t)} = \nabla_\theta \ell(z_i, \theta_t), \qquad g_c^{(t)} = \nabla_\theta \ell_c(\theta_t).$$

A first-order expansion of the component loss after a gradient step on example i gives

$$\ell_c\left(\theta_t - \eta_t g_i^{(t)}\right) - \ell_c(\theta_t) \approx -\eta_t \left(g_c^{(t)}\right)^\top g_i^{(t)}.$$

The utility-oriented TracIn component score is therefore

$$A^{\mathrm{TracIn}}_{i,c} = \sum_{t=1}^{T} \eta_t \left(g_i^{(t)}\right)^\top g_c^{(t)}. \tag{16}$$

A positive value indicates that updates on example i tend to reduce the component loss over the observed training trajectory. The accumulated TracIn proxy is

$$\widehat{u}^{\mathrm{TracIn}}_c(\mathcal{M}) = \sum_{i \in \mathcal{M}} A^{\mathrm{TracIn}}_{i,c}.$$

D.3 Leave-One-Out Components

Leave-one-out directly compares the full feasible pool with the pool obtained after removing one training example. For component c, we define

$$A^{\mathrm{LOO}}_{i,c} = u_c(N) - u_c(N \setminus \{i\}). \tag{17}$$

A positive value means that removing example i decreases the utility of component c, indicating that i is helpful for that component.

The corresponding accumulated proxy is

$$\widehat{u}^{\mathrm{LOO}}_c(\mathcal{M}) = \sum_{i \in \mathcal{M}} A^{\mathrm{LOO}}_{i,c}.$$

Because each LOO score is measured relative to the nearly complete feasible pool, this accumulated quantity is not assumed to equal the result of retraining directly on $\mathcal{M}$. LOO-NADA instead evaluates whether these full-coalition removal effects remain informative when combined over candidate subsets.

E LIMITATIONS AND FUTURE WORK

Dependence on the component attribution matrix. NADA operates on existing component-level attribution scores, and its effectiveness depends on their quality and alignment with the downstream task. Our results support improvements over matched scalar top-m selection, while performance relative to Random remains setting-dependent.

Theoretical and computational scope. Our theoretical results apply under their stated assumptions: the separation result concerns binary component utilities, and the greedy approximation guarantee requires nonnegative scores. Extending the analysis to broader settings and developing more efficient selection procedures are natural directions for future work.

Evaluation scope and future directions. Our experiments cover benchmark tasks and controlled corruption settings. Further evaluation on naturally noisy data and larger training collections would help characterize broader applicability. Adaptive component construction, weighting, and aggregation also offer opportunities to extend the framework across tasks and selection budgets.