Generalized Group Data Attribution

Dan Ley[*] **Suraj Srinivas** **Shichang Zhang** **Himabindu Lakkaraju**

Harvard University

Abstract

Data Attribution (DA) methods quantify the influence of individual training data points on model outputs and have broad applications such as explainability, data selection, and noisy label identification. However, existing DA methods are often computationally intensive, limiting their applicability to large-scale machine learning (ML) models. To address this challenge, we introduce the Generalized Group Data Attribution (GGDA) framework, which computationally simplifies DA by attributing to groups of training points instead of individual ones. GGDA is a general framework that subsumes existing attribution methods and can be applied to new DA techniques as they emerge. It allows users to optimize the trade-off between efficiency and fidelity based on their needs. Our empirical results demonstrate that GGDA applied to popular DA methods such as Influence Functions, TracIn, and TRAK results in up to **10x-50x** speedups over standard DA methods while gracefully trading off attribution fidelity. For downstream applications such as dataset pruning and noisy label identification, we demonstrate that GGDA significantly improves computational efficiency and maintains effectiveness, enabling practical applications in large-scale ML scenarios that were previously infeasible.

1 Introduction

In the era of data-driven ML, the impact of training data on model performance has become increasingly evident. Data Attribution (DA) methods have risen as powerful tools designed to address this, identifying the influence of each training data point for specific predictions, offering insights into the relationships between training data and model behavior [1, 2, 3, 4, 5, 6, 7, 8, 9]. DA applications are far-reaching, encompassing model debugging, strategic data selection, and mislabelled data detection. Existing techniques, however, incur prohibitive computational costs, e.g., by requiring retrainings in the order of the training data size (intractable for large-scale models [2, 7]), and often struggle to faithfully represent the true impact of training data on model outcomes [10]. Efficiency and fidelity issues combined have severely restricted widespread adoption in large-scale applications.

To address these, we propose Generalized Group Data Attribution (GGDA), a framework that shifts the paradigm from attributing influence to individual data points to considering meaningful groups of data points collectively, enabling GGDA to substantially reduce computational overhead and making it particularly well-suited for large-scale applications where traditional DA methods falter. To validate the effectiveness of GGDA, we conduct extensive experimental evaluations across various datasets, model architectures, and DA methods. Our results show that: (1) GGDA significantly improves the computational efficiency of DA methods, resulting in upto **10x-50x** speedups depending on the setting, (2) GGDA favourably trades off fidelity with efficiency compared to standard DA approaches, and (3) For downstream applications such as dataset pruning and noisy label identification, GGDA outperforms standard DA methods, while maintaining significant speedups.

[*]Corresponding author: dley@g.harvard.edu

2 Related Work

Data Attribution quantifies the influence of training samples on model predictions. One type is retraining-based, systematically retraining models with varying subsets of training data to measure sample influence. This includes Leave-One-Out (LOO) influence [11], and its extensions to subsets in Data Shapley [2, 12] and DataModels [7], which can capture complex data interactions, but are computationally expensive due to reliance on retraining. In contrast, gradient-based methods offer more scalable solutions, providing closed-form attribution scores. Influence functions [1] approximate the effect of upweighting a training sample on the loss function. TracIn [3] traces loss changes during the training process, and TRAK [8] employs random projection to assess influence. Gradient-based methods significantly reduce computation, but may suffer from performance degradation on non-convex neural networks due to reliance on convexity assumptions and Taylor approximations.

Grouping in Data Attribution. Prior research has explored attributing effects to groups. [13] analyzed influence functions for large groups, finding strong correlations between predicted and actual group effects on linear models (we focus on non-linear models). [14] developed a second-order influence function for better fidelity but at increased computational costs, whereas our focus is enhancing efficiency. [2] also touched on grouping data points as a strategy to handle large datasets, but our research systematically extends attribution methods to groups and experimentally validates that it is possible to generalize both re-training-based and gradient-based attribution methods.

Data Attribution Acceleration. Various approaches have been proposed to accelerate data attribution methods, including subsampling [15] and better gradient/Hessian approximations [4]. These techniques focus on optimizing existing attribution algorithms to improve computational efficiency. While valuable, these methods are orthogonal to our GGDA framework, which fundamentally alters the attribution paradigm by considering groups of data points. As such, GGDA can be combined with these techniques to achieve greater efficiency gains, highlighting its complementary nature.

3 Generalized Group Data Attribution Framework

This section introduces formal DA definitions. We examine limitations of conventional methods which attribute to individual points, showing that they often result in prohibitive computation. We then present the rationale behind Generalized Group Data Attribution (GGDA) as a solution.

Notation. Given a dataset of n training data points $\mathcal{D} = \{(\mathbf{x}_0, \mathbf{y}_0), (\mathbf{x}_1, \mathbf{y}_1), ...(\mathbf{x}_n \mathbf{y}_n)\}$ in the supervised learning setting,[2] and a learning algorithm $\mathcal{A}(\mathcal{D}) : 2^{\mathcal{D}} \to \mathbb{R}^p$ that outputs weights $\theta \in \mathbb{R}^p$, where $f(\mathbf{x}_{\text{test}}; \theta) \in \mathbb{R}$ is the model prediction function for a given test input $\mathbf{x}_{\text{test}}$, we define a counterfactual dataset $\mathcal{D}_S = \mathcal{D} \setminus \{(\mathbf{x}_i, \mathbf{y}_i; i \in S)\}$ as the training dataset with points in S removed. We use $\theta_S \sim \mathcal{A}(\mathcal{D}_S)$ to denote the model weights trained on $\mathcal{D}_S$.

Definition 1 *(Data Attribution) Given a learning algorithm $\mathcal{A}$, dataset $\mathcal{D}$ and test point $\mathbf{x}_{test}$, data attribution for the i^{th} training point $(\mathbf{x}_i, \mathbf{y}_i)$ is a scalar estimate $\tau_i(\mathcal{A}, \mathcal{D}, \mathbf{x}_{test}) \in \mathbb{R}$ representing the influence of $(\mathbf{x}_i, \mathbf{y}_i)$ on $\mathbf{x}_{test}$.*

To ground our discussion with a concrete example, we first introduce the conceptually most straightforward DA method, the Leave-One-Out (LOO) estimator. This computes the average prediction difference between a model trained on the original dataset vs one with a single training point excluded.

Definition 2 *(Leave-One-Out estimator) The LOO estimator of a learning algorithm $\mathcal{A}$, dataset $\mathcal{D}$ and test point $\mathbf{x}_{test}$, w.r.t. to training point $(\mathbf{x}_i, \mathbf{y}_i)$ is given by $\tau_i(\mathcal{A}, \mathcal{D}, \mathbf{x}_{test}) = \mathbb{E}_{\theta \sim \mathcal{A}(\mathcal{D})} f(\mathbf{x}_{test}; \theta) - \mathbb{E}_{\theta_i \sim \mathcal{A}(\mathcal{D} \setminus \{(\mathbf{x}_i, \mathbf{y}_i)\})} f(\mathbf{x}_{test}; \theta_i)$*

The LOO estimator forms the conceptual basis for DA methods, but has several critical drawbacks.

Drawback: Computation Scales Intractably. Computationally, methods like the leave-one-out (LOO) estimator, which trains $n+1$ models, become impractical with large n. Additionally, gradient-based approaches like influence functions, still require (1) computation of per-training-sample gradients w.r.t. parameters (n forward and backward passes) and (2) computation of the Hessian inverse of models parameters (dimensions of $p \times p$ for a model $\theta \in \mathbb{R}^p$).

[2]This can be easily extended to unsupervised learning.

Drawback: Pointwise Attribution may be Ill-Defined. Recall that the LOO estimator involves repeatedly training models on *nearly identical* training sets. However, as data scales, a single missing example is unlikely to lead to statistically significant change in model outputs, an intuition captured in literature via properties such as (1) stability of learning algorithms [16], or (2) differential privacy [17]. The conceptual basis for many methods, the LOO estimator, may be ill-defined under such conditions.

Drawback: Focus on Individual Predictions Limits Scope. We may wish to identify training points that cause models to be inaccurate, non-robust, or unfair, all of which are "bulk" model properties. In these scenarios, DA has been adapted to analyze individually the test points that are inaccurate or non-robust, which is undesirable given (1) many model properties, such as fairness and f-measure, are non-decomposable [18], i.e., cannot be written in terms of contributions of individual test points, and (2) it is computationally wasteful to aggregate test point results over directly computing the metric.

To resolve these, we propose two modifications to the DA framework, of (1) partitioning the training dataset into groups of training points and directly attributing model behaviour to these groups (similar data points may affect models in similar ways), and (2) expanding the scope of DA to focus beyond individual predictions to arbitrary "property functions" of the model parameters, such as accuracy, robustness, and fairness. Collapsing groups to singletons or setting property functions to invididual loss subsumes standard DA. We now formally define these two new components:

Group $\mathcal{Z}$: Given a dataset $\mathcal{D}$, we define a partition of the dataset into k groups such that $\mathcal{Z} = \{\mathbf{z}_0, \mathbf{z}_1, ...\mathbf{z}_k\}$, where each $\mathbf{z}_j$ is a set of inputs $\mathbf{x}_i$ such that $\mathbf{z}_j \cap \mathbf{z}_i = \emptyset$ for $i \neq j$, and $\bigcup_i \mathbf{z}_i = \mathcal{D}$. In general, the cardinality of each group $|\mathbf{z}_i|$ may vary. A group size of 1 subsumes standard DA.

Property function g: Given a model with weights $\theta \in \mathbb{R}^p$, we define a property function $g(\cdot) : \mathbb{R}^p \to \mathbb{R}$. Property functions generalize (1) pointwise functions like log probability of a given test point, (2) aggregate functions like accuracy on a test set. Given these, we are ready to define GGDA.

Definition 3 *(Generalized Group Data Attribution) Given a learning algorithm $\mathcal{A}$, dataset $\mathcal{D}$ with groups $\mathcal{Z}$ and property function g, GGDA wrt to the j^{th} group $\mathbf{z}_j$ estimates $\tau_j(\mathcal{A}, \mathcal{D}, \mathcal{Z}, g) \in \mathbb{R}$ representing the influence of group $\mathbf{z}_j$ on model property g*

To ground this with a concrete example, we present the GGDA variant of the LOO estimator below. The idea is very similar to the usual LOO estimator, except here that we leave the j^{th} group out, and measure the resulting change in the property function g.

Definition 4 *The (GGDA LOO) estimator of a learning algorithm $\mathcal{A}$, dataset $\mathcal{D}$, groups $\mathcal{Z}$ and property function g, wrt to the j^{th} group $\mathbf{z}_j$ is given by $\tau_j(\mathcal{A}, \mathcal{D}, \mathcal{Z}, g) = \mathbb{E}_{\theta \sim \mathcal{A}(\mathcal{D})} g(\theta) - \mathbb{E}_{\theta_j \sim \mathcal{A}(\mathcal{D} \setminus Z_j)} g(\theta_j)$*

Comparing this to the usual LOO attributor, we observe that the GGDA variant requires training of $k+1$ models as opposed to $n+1$ models, which is advantageous when $k << n$. In summary, the main advantage of GGDA methods is that they <u>scale as the number of groups k</u>, whereas DA methods <u>scale with the number of data points n</u>. <u>While extending LOO was fairly simple, it is, in general, non-trivial to extend arbitrary DA methods</u> to GGDA methods, particularly in a way that makes the computation scale with the number of groups k, reflecting the advantage that group attribution offers.

4 Generalizing Existing Data Attribution Methods to Groups

We first show how to generalize Influence Functions to the GGDA setting, following two principles:

- the GGDA variants must scale with k, the number of groups, and not n, the number of datapoints;
- the GGDA variants must subsume the ordinary DA variants when groups are defined as singleton datapoints, and the property function as the loss on a single test point.

Generalized Group Influence Functions. Given a loss function $\ell \in \mathbb{R}^+$, Influence GGDA is:

$$\tau_{\text{inf}}(\mathcal{A}, \mathcal{D}, \mathbf{x}_{\text{test}}) = \underbrace{\nabla_\theta \ell(\mathbf{x}_{\text{test}}; \theta)^\top}_{\mathbb{R}^{1\times p}} \underbrace{H_\theta^{-1}}_{\mathbb{R}^{p\times p}} \underbrace{\left[\nabla_\theta \ell(\mathbf{x}_0; \theta); \nabla_\theta \ell(\mathbf{x}_1; \theta); ...\nabla_\theta \ell(\mathbf{x}_n; \theta)\right]}_{\mathbb{R}^{p\times n}} \quad \theta \sim \mathcal{A}(\mathcal{D})$$

The primary computational bottlenecks for computing influence functions involve (1) computing the Hessian inverse, and (2) computing the n gradients terms, which involve independent n forward and backward passes. In Appendix A.1, we show that due to linearity of influence functions, group influence is given by the sum of individual influences. The corresponding GGDA generalization is:

$$\tau_{\text{inf}}(\mathcal{A}, \mathcal{D}, \mathcal{Z}, g) = \underbrace{\nabla_\theta g(\theta)^\top}_{\mathbb{R}^{1\times p}} \underbrace{H_\theta^{-1}}_{\mathbb{R}^{p\times p}} \underbrace{\left[\nabla_\theta \ell(\mathbf{z}_0;\theta); \nabla_\theta \ell(\mathbf{z}_1;\theta); ...\nabla_\theta \ell(\mathbf{z}_k;\theta)\right]}_{\mathbb{R}^{p\times k}} \qquad \theta \sim \mathcal{A}(\mathcal{D}) \qquad (1)$$

where $\ell(\mathbf{z}_i) := \sum_{\mathbf{x}\in\mathbf{z}_i} \ell(\mathbf{x})$. From equation 1, we observe that the corresponding group influence terms involve $\underline{k \text{ batched gradient computations}}$ instead of $\underline{n \text{ per-sample gradient computations}}$, where all $\sim n/k$ points belonging to each group are part of the same batch. Typically, $k << n$, leading to considerable overall runtime benefits for GGDA influence functions. We use this key insight for two more methods: TracIn and TRAK (presented in Appendix A).

5 Experiments

This section presents highlights from a comprehensive evaluation of our proposed GGDA framework. We conduct experiments on multiple datasets using various models to demonstrate the effectiveness and efficiency of our approach compared to existing data attribution methods. We evaluate across four diverse datasets encompassing tabular (HELOC), image (MNIST and CIFAR-10), and text data (QNLI). These datasets represent a range of task complexities, data types, and domains, allowing us to assess the generalizability of our approach across different scenarios. The models we fit on these datasets range from simple logistic regression (LR) to medium-sized artificial neural networks (ANNs, e.g., MLPs and ResNet) and pre-trained language models (BERT). Full configuration details for datasets, models, attribution methods, grouping methods, and evaluations are listed in Appendix B, while the remainder of results across our grid of experiments is in Appendix C.

5.1 Grouping Methods

We employ several grouping methods to evaluate the effectiveness of our GGDA framework. For each grouping method, we also experiment with various group sizes to analyze the trade-off between computational efficiency and attribution effectiveness. The choice of grouping method and group size can significantly impact the performance of GGDA, as we subsequently demonstrate.

Groupings comprise of: **Random**, assigning data points to groups of a specified size uniformally randomly (serving as a baseline); **K-Means**, grouping data points based on raw features; **Representation K-Means (Repr-K-Means)**, grouping data points on hidden representations from the model we attribute on, capturing higher-level semantic similarities; and **Gradient K-Means (Grad-K-Means)**, grouping on the gradient of the loss wrt activations (NOT parameters) of the pre-classification layer of the model, grouping instances with similar effects on the model's learning process.

5.2 Evaluation Metrics

To illustrate the effectiveness of GGDA at identifying important datapoints, in comparison to DA, it is critical to have metrics that allow us to compare both on equal terms. To this end, we utilize:

Retraining Score (RS) This metric quantifies the impact of removing a proportion of training data points on model performance. The process involves (1) identifying the most influential groups using GGDA, (2) removing a given percentile of high-importance points, (3) retraining the model, and (4) comparing the performance of this model to the original model's performance on the test set. A significant decrease in performance (i.e., an increase in loss or decrease in accuracy) indicates that the removed points were indeed important for the model's learning process. This metric provides a tangible measure of the effectiveness of GGDA in identifying highly useful training instances.

Noisy Label Detection (AUC) This metric quantifies the effectiveness with which GGDA methods can identify mislabelled training data points. This involves randomly flipping a proportion of the training set labels, and training a model on the corrupted dataset. We then examine the labels of the training data points that belong to groups with the lowest scores, and plot the change of the fraction of detected mislabeled data with the fraction of the checked training data, following [2, 5]. The final metric is computed as the area under the curve (AUC).

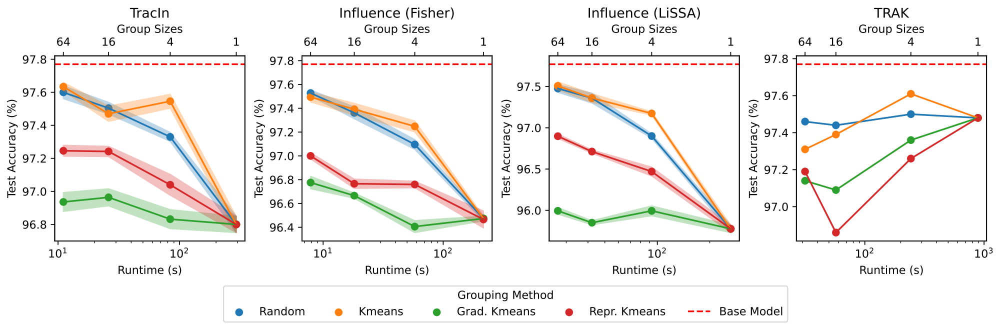

Figure 1: **Ablation over grouping methods.** Test accuracy (%) retraining score on MNIST after removing the top 20% most important training points (lower is better). Grad-K-Means grouping is superior, achieving comparable performance to individual attributions at orders of magnitudes faster runtime. Error bars represent standard error computed on 10 differently seeded model retrainings.

5.3 Results

In the following subsections, we present results from across our range of datasets and models, discussing GGDA performance in terms of retraining score, computational efficiency, and effectiveness in downstream tasks such as dataset pruning and noisy label detection. Our pipeline trains a model on the full training set and determines groups as described in Section 5.1. We ablate extensively over a range of group sizes, beginning at 1 (standard DA) and increasing to 1024, in ascending powers of 2.

5.3.1 Retraining Score

To evaluate the retraining score metric, we compute *Test Accuracy* upon removal of 1%, 5%, 10%, and 20% of the most important points in the train set. For groups, this translates to sequentially removing the most important groups up until the desired number of data points (points are randomly selected from the final group in the case of overlap). Lower test accuracies are thus more favorable.

Which Grouping Method to Use? We plot a subset of group sizes that decreases from left to right as runtimes simultaneously increase, until GGDA subsumes DA at size 1. We find that across the board, Grad-K-Means most consistently demonstrates superior performance. Figure 1 illustrates this for MNIST (similarly conclusive plots for remaining datasets and models are in Appendix C.1).

Results across Removal Percentages. Based on our results above, we employ Grad-K-Means as the primary grouping method, and trial incremental increases on the removal percentage for the retraining metric (using values of 1%, 5%, 10%, 20%). Our results in Figure 2 show that GGDA, in general, has a favorable accuracy efficiency tradeoff. In particular, we find that the performance levels are roughly equivalent across group sizes $\{1, 4, 16, 64\}$, but with orders-of-magnitude speedup.

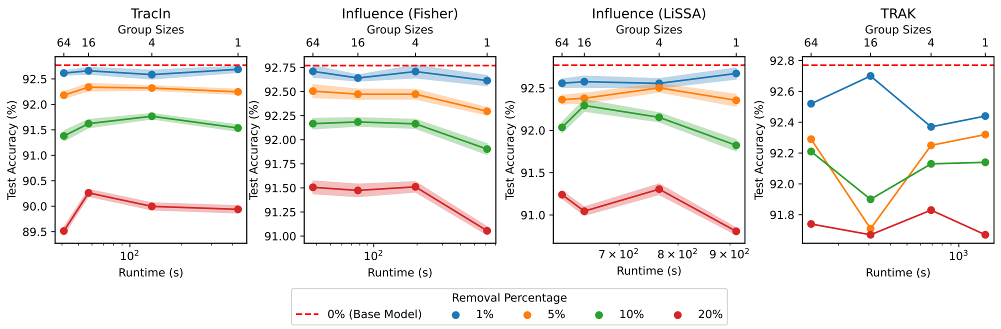

Figure 2: **GGDA attribution fidelity across removal percentages**. Test accuracy (%) retraining score using Grad-K-Means grouping on CIFAR-10 after removing 1%, 5%, 10%, and 20% of the most important points (lower is better). GGDA methods save orders of magnitude of runtime while gracefully trading off attribution fidelity. See Appendix C.1 for similar findings across all datasets and models.

5.3.2 Dataset Pruning

To evaluate the effectiveness of GGDA in dataset pruning, we compute *test accuracy* after removing varying percentages (25%, 50%, 75%) of the least important points from the training set (retraining score). Higher test accuracies after pruning are more favorable. Our results demonstrate that GGDA significantly outperforms individual DA methods in this task, achieving higher test accuracies while requiring substantially less compute. Table 1 presents example results for a range of datasets using TracIn, illustrating the superior performance and efficiency of GGDA compared to individual DA.

Table 1: Test accuracies (%) after pruning the MNIST, CIFAR-10, and QNLI datasets with TracIn DA/GGDA methods and varying removal percentages (using size 1024 groupings).

	MNIST		**CIFAR-10**		**QNLI**	
Removal %	DA	GGDA	DA	GGDA	DA	GGDA
25	95.8 ± 0.1	97.5 ± 0.0	89.9 ± 0.1	92.1 ± 0.1	74.9 ± 4.4	85.9 ± 1.2
50	93.6 ± 0.2	97.1 ± 0.1	84.4 ± 0.2	90.7 ± 0.1	64.8 ± 4.3	84.2 ± 1.6
75	85.8 ± 3.3	96.1 ± 0.1	74.4 ± 0.3	87.5 ± 0.1	46.4 ± 5.3	83.3 ± 3.5
Runtime (s)	311 ± 5.8	6.59 ± 0.07	492 ± 73.2	35.2 ± 0.07	1043 ± 14.3	100 ± 0.1

5.3.3 Noisy Label Identification

Noisy label detection is a critical task in ML, addressing inaccurate labelling in real-world datasets. We analyze the trade-off between detection accuracy (AUC) and computational efficiency (runtime). We present our findings for CIFAR-10 and QNLI in Table 2, demonstrating superior performance for GGDA, with the remainder in Appendix C.3. For BERT models, the Fisher approximation variant of influence functions shows improved performance when combined with grouping, while also reducing runtime by an order of magnitude. TracIn demonstrates superior performance on the CIFAR-10 dataset, particularly when grouping datapoints with similar penultimate-layer gradients.

Table 2: Mean noisy label detection AUC and runtimes for increasingly large Grad-K-Means groups, measured across all datasets and attribution methods. For HELOC, we display results from the ANN-S model.

Dataset	Attributor	Noisy Label Detection AUC / Runtime (s) per GGDA Group Size				
		1 (DA)	4	16	64	256
HELOC	TracIn	0.580 / 10.6	0.572 / 2.95	0.568 / 0.90	0.566 / 0.40	0.566 / 0.28
	Inf. (Fisher)	0.604 / 9.25	0.582 / 2.51	0.552 / 0.76	0.495 / 0.34	0.510 / 0.23
	Inf. (LiSSA)	0.650 / 18.5	0.631 / 11.9	0.627 / 10.2	0.621 / 9.77	0.620 / 9.65
	TRAK	0.504 / 46.4	0.549 / 12.0	0.528 / 3.92	0.512 / 1.22	0.503 / 0.77
MNIST	TracIn	0.637 / 208	0.646 / 55.9	0.650 / 18.1	0.642 / 8.18	0.529 / 5.69
	Inf. (Fisher)	0.509 / 287	0.551 / 90.7	0.586 / 27.8	0.554 / 9.49	0.502 / 4.58
	Inf. (LiSSA)	0.640 / 220	0.647 / 71.7	0.647 / 35.2	0.639 / 25.5	0.534 / 23.2
	TRAK	0.495 / 868	0.501 / 240	0.502 / 87.7	0.504 / 23.9	0.501 / 33.0
CIFAR-10	TracIn	0.700 / 558	0.702 / 172	0.704 / 59.0	0.705 / 31.6	0.707 / 24.1
	Inf. (Fisher)	0.630 / 497	0.614 / 172	0.602 / 69.8	0.573 / 40.5	0.569 / 35.0
	Inf. (LiSSA)	0.656 / 835	0.657 / 544	0.661 / 468	0.678 / 452	0.685 / 447
	TRAK	0.501 / 1814	0.490 / 674	0.486 / 319	0.482 / 210	0.497 / 188
QNLI	TracIn	0.543 / 1743	0.539 / 427	0.537 / 172	0.536 / 136	0.536 / 119
	Inf. (Fisher)	0.667 / 2207	0.695 / 493	0.684 / 187	0.605 / 145	0.508 / 135
	Inf. (LiSSA)	0.501 / 2043	0.504 / 686	0.489 / 395	0.490 / 403	0.512 / 392
	TRAK	0.500 / 4947	0.498 / 1698	0.497 / 841	0.507 / 739	0.502 / 788

6 Conclusion

In this work, we have proposed a new framework, Generalized Group Data Attribution (GGDA), that offers computational advantages over data attribution while maintaining similar fidelity. While our work focuses on influence-based approaches, any data attribution approach can, in principle, be extended to the group setting to obtain these computational benefits. The key insight here is that many downstream applications of attribution involve manipulating training datasets at scale, and a fine-grained approach of individual point influence may be unnecessary in such cases.

There are several exciting avenues for future work. First, it is desirable to conduct a formal analysis of the fidelity tradeoffs in the group setting for various methods of interest. Second, while our experiments focus on small-to-medium-scale settings to systematically demonstrate the advantages of group attribution, evaluation on a genuinely large-scale, billion-data-point learning setting can help uncover any remaining computational bottlenecks limiting the practical adoption of data attribution.

References

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

[2] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

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

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

[5] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8239–8247, 2021.

[6] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8780–8802. PMLR, 2022.

[7] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

[8] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

[9] Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023.

[10] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[11] John Tukey. Bias and confidence in not quite large samples. *Ann. Math. Statist.*, 29:614, 1958.

[12] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

[13] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.

[14] Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pages 715–724. PMLR, 2020.

[15] Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[16] Andre Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6(3):55–79, 2005.

[17] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[18] Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *International Conference on Machine Learning*, pages 199–208. PMLR, 2015.

[19] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR, 2020.

[20] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

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

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Qian Wang and Christopher D Manning. Stanford question answering dataset. *arXiv preprint arXiv:1806.02847*, 2018.

[25] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[26] T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Appendix

This appendix is formatted as follows:

- We provide full mathematical derivations in Appendix A.
- We provide experimental setup details in Appendix B.
  - Appendix B.1 contains dataset and model details.
  - Appendix B.2 contains clustering details.
  - Appendix B.3 contains attributor method parameters.
- Extended experimental results in full are located in Appendix C.

A Derivations

A.1 Generalized Group Influence functions

In this section, we present a derivation for group influence functions, mirroring the influence function derivation adopted by [1]. The argument involves two steps. First, we analyse the change in model parameters upon upweighting the loss of a group of training points. Second, we compute how the change in model parameters affects the downstream property function. To begin, we consider what happens if we weight a group of training samples, Z. The modified loss and minimizer are:

$$L_\epsilon(\theta) = \frac{1}{n}\left[\sum_{i=1}^{n} \ell(z_i, \theta) + \epsilon \sum_{z \in Z} \ell(z, \theta)\right]$$

$$L_\epsilon(\theta) = L(\theta) + \frac{\epsilon}{n}\sum_{z \in Z} \ell(z, \theta) \text{ and } \hat{\theta}_\epsilon = \arg\min_\theta \left\{L(\theta) + \frac{\epsilon}{n}\sum_{z \in Z} \ell(z, \theta)\right\}$$

$$\text{With gradient } \nabla L_\epsilon(\theta) = \nabla L(\theta) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla \ell(z, \theta)$$

Let $\partial\theta = (\hat{\theta}_\epsilon - \hat{\theta})$. For a single step:

$$f(\hat{\theta}_\epsilon) \approx f(\hat{\theta}) + \nabla f(\hat{\theta})\partial\theta$$

We now make the following simplifying assumptions, following [1]

Assumption 1: For small ϵ, the modified loss L_ϵ is locally linear between $\hat{\theta}$ and $\hat{\theta}_\epsilon$.

We can approximate the gradient $\nabla L_\epsilon(\hat{\theta}_\epsilon)$ by starting from $\hat{\theta}$ and taking a single step towards $\hat{\theta}_\epsilon$ (for $f = \nabla L_\epsilon$):

$$\nabla L_\epsilon(\hat{\theta}_\epsilon) \approx \nabla L_\epsilon(\hat{\theta}) + \nabla^2 L_\epsilon(\hat{\theta})\partial\theta$$

Assumption 2: $\hat{\theta}_\epsilon$ minimizes $L_\epsilon \implies \nabla L_\epsilon(\hat{\theta}_\epsilon) = 0$

$$\nabla L_\epsilon(\hat{\theta}) + \nabla^2 L_\epsilon(\hat{\theta})\partial\theta = 0$$
$$\implies [\nabla L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla \ell(z, \hat{\theta})] + [\nabla^2 L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla^2 \ell(z, \hat{\theta})]\partial\theta = 0$$

Rearranging, we have

$$\partial\theta = -[\nabla^2 L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla^2 \ell(z, \hat{\theta})]^{-1}[\nabla L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla \ell(z, \hat{\theta})]$$

Using $(A + \epsilon B)^{-1} \approx A^{-1} - \epsilon A^{-1}BA^{-1}$, we set $A = \nabla^2 L(\hat{\theta})$ and $B = \frac{1}{n}\sum_{z \in Z} \nabla^2 \ell(z, \hat{\theta})$, yielding:

$$\partial\theta = -\underbrace{\left[[\nabla^2 L(\hat{\theta})]^{-1} - \frac{\epsilon}{n}\sum_{z \in Z}[\nabla^2 L(\hat{\theta})]^{-1}\nabla^2 \ell(z, \hat{\theta})[\nabla^2 L(\hat{\theta})]^{-1}\right]}_{(A+\epsilon B)^{-1} = [\nabla^2 L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla^2 \ell(z, \hat{\theta})]^{-1}}[\nabla L(\hat{\theta}) + \frac{\epsilon}{n}\sum_{z \in Z} \nabla \ell(z, \hat{\theta})]$$

Assumption 3: $\hat{\theta}$ minimizes $L \implies \nabla L(\hat{\theta}) = 0$

$$\partial\theta = -\left[[\nabla^2 L(\hat{\theta})]^{-1} - \frac{\epsilon}{n}\sum_{z\in Z}[\nabla^2 L(\hat{\theta})]^{-1}\nabla^2\ell(z,\hat{\theta})[\nabla^2 L(\hat{\theta})]^{-1}\right]\frac{\epsilon}{n}\sum_{z\in Z}\nabla\ell(z,\hat{\theta})$$

As $\epsilon \to 0$, we have $\epsilon = \partial\epsilon$ and drop $o(\epsilon)$ terms:

$$\partial\theta = -\nabla^2 L(\hat{\theta})^{-1}\frac{\epsilon}{n}\sum_{z\in Z}\nabla\ell(z,\hat{\theta})$$
$$\implies \frac{\partial\theta}{\partial\epsilon} = -\nabla^2 L(\hat{\theta})^{-1}\frac{1}{n}\sum_{z\in Z}\nabla\ell(z,\hat{\theta})$$

Assumption 4: Hessian at $\theta = \hat{\theta}$ exists and is positive definite.

$$\implies \frac{\partial\theta}{\partial\epsilon} = -\frac{1}{n}H_{\hat{\theta}}^{-1}\sum_{z\in Z}\nabla\ell(z,\hat{\theta})$$

This is the rate of change of parameters θ as we uniformally upweight a group of training samples Z by ϵ in the loss function (note that upweighting occurs before normalizing by $1/n$). This concludes the first part of the analysis, i.e., computing the change in model parameters due to the up-weighting.

The second is fairly trivial, as the gradient of the property function gives the change in property function with the change in model parameters. Combining both, the overall attribution of a group to some test property, $g(\theta)$ is:

$$\frac{\partial g}{\partial\epsilon} = \frac{\partial g}{\partial\theta}\times\frac{\partial\theta}{\partial\epsilon} = -\frac{1}{n}\nabla g H_{\hat{\theta}}^{-1}\sum_{z\in Z}\nabla\ell(z,\hat{\theta})$$

A.2 TracIn as a Simplified Influence Approximation

[3] propose an influence definition based on aggregating the influence of model checkpoints in gradient descent. Formally, tracein can be described as follows:

$$\tau_{\text{identity-inf}}(\theta,\mathcal{D},\mathbf{x}_{\text{test}}) = \nabla_\theta\ell(\mathbf{x}_{\text{test}};\theta)^\top\left[\nabla_\theta\ell(\mathbf{x}_0;\theta);\nabla_\theta\ell(\mathbf{x}_1;\theta);...\nabla_\theta\ell(\mathbf{x}_n;\theta)\right] \tag{2}$$
$$\tau_{\text{tracein}}(\mathcal{A},\mathcal{D},\mathbf{x}_{\text{test}}) = \sum_{i=1}^{T}\tau_{\text{identity-inf}}(\theta_i,\mathcal{D},\mathbf{x}_{\text{test}}) \quad \text{where } \{\theta_i | i\in[1,T]\} \text{ are checkpoints} \tag{3}$$

The core DA method used by tracein is the standard influence functions, with the Hessian being set to identity, which drastically reduces computational cost. The GGDA extension of tracein thus amounts to replacing the `identity-inf` component (equation 2) with equation 1, with the Hessian set to identity.

A.3 Background on Fisher information matrix

One of the popular techniques for Hessian approximation in machine learning is via the Fisher Information Matrix. When the quantity whose Hessian is considered is a log-likelihood term, it turns out that the Hessian has a simple form:

$$\begin{aligned} H_\theta &= -\sum_{\mathbf{x}}\mathbb{E}_{y'\sim p(y|x)}\nabla^2_\theta\log p(y' \mid \mathbf{x},\theta) \\ &= \sum_{\mathbf{x}}\mathbb{E}_{y'\sim p(y|x)}\nabla_\theta\log p(y' \mid \mathbf{x},\theta)\nabla_\theta\log p(y' \mid \mathbf{x},\theta)^\top = F_\theta \end{aligned}$$

Here, the term F_θ is the Fisher Information Matrix. Thus, when the loss term is a log-likelihood term, $\ell(\mathbf{x}, \theta) = \mathbb{E}_{y' \sim p(y|x;\theta)} \log p(y' \mid \mathbf{x}; \theta)$, the Hessian can be written in this manner [19]. In practical applications involving the Fisher, it is common to use the so-called Empirical Fisher matrix, which is given by:

$$\hat{F}_\theta = \sum_{\mathbf{x}} \nabla_\theta \log p(\mathbf{y} \mid \mathbf{x}, \theta) \nabla_\theta \log p(\mathbf{y} \mid \mathbf{x}, \theta)^\top \quad \text{(where } \mathbf{y} \text{ is the ground truth label for } \mathbf{x}) \tag{4}$$

While the limitations of using the empirical Fisher approximation have been well-documented [20], primarily owing to issues with the bias of the estimate, it is employed practically as a convenient approximation. Computationally, while the Hessian of log-likelihood requires both a double-backpropagation and a Monte Carlo expectation, the Fisher only requires a single backpropagation for gradient computation with the Monte Carlo estimate. On the other hand, the empirical Fisher approximation forgoes the Monte Carlo estimate.

A.4 TRAK as Ensembled Fisher Influence

TRAK [8] is an influence method derived for binary classification setting. Specifically, given a logistic classifier $\ell(\theta) = \log(1 + \exp(-y_{gt} f(\mathbf{x})))$, TRAK without projection + ensembling (Equation 13 in the TRAK paper) is equivalent to the following (note distinction between ℓ vs f):

$$\tau_{\text{trak}} = \underbrace{\nabla_\theta f(\mathbf{x}_{test}; \theta)}_{\mathbb{R}^{1\times p}} \underbrace{\hat{H}_\theta^{-1}}_{\mathbb{R}^{p\times p}} \underbrace{\left[\nabla_\theta \ell(\mathbf{x}_0; \theta); \nabla_\theta \ell(\mathbf{x}_1; \theta); ... \nabla_\theta \ell(\mathbf{x}_n; \theta)\right]}_{\mathbb{R}^{p\times n}}$$

$$\hat{H}_\theta = \sum_{i=1}^{n} \nabla_\theta f(\mathbf{x}_i; \theta) \nabla_\theta f(\mathbf{x}_i; \theta)^\top$$

Comparing the Hessian approximation term $\hat{H}$ with the formula for the empirical Fisher approximation in the main paper, we observe that TRAK computes outer products of model outputs, whereas the empirical Fisher involves outer products of the loss function. We show below that these two quantities are related:

$$\begin{aligned}
\ell(\mathbf{x}, \theta) &= \log(1 + \exp(-y_{gt} f(\mathbf{x}, \theta)/T)) \\
\nabla_\theta \ell(\mathbf{x}, \theta) &= \frac{\exp(-y_{gt} f(\mathbf{x}, \theta)/T)}{1 + \exp(-y_{gt} f(\mathbf{x}, \theta)/T)} (-y_{gt}/T) \nabla_\theta f(\mathbf{x}, \theta) \\
\nabla_\theta \ell(\mathbf{x}_i; \theta) \nabla_\theta \ell(\mathbf{x}_i; \theta)^\top &= \underbrace{\frac{\exp(-2 y_{gt} f(\mathbf{x}, \theta)/T)}{(1 + \exp(-y_{gt} f(\mathbf{x}, \theta)/T))^2 T^2}}_{C(T)} \nabla_\theta f(\mathbf{x}_i; \theta) \nabla_\theta f(\mathbf{x}_i; \theta)^\top \quad (y_{gt} \in \{+1, -1\})
\end{aligned}$$

Where T is a temperature parameter. If we set $T \to \infty$, then $C(T) \to \frac{1}{4T^2}$. Thus in this case,

$$\nabla_\theta \ell(\mathbf{x}_i; \theta) \nabla_\theta \ell(\mathbf{x}_i; \theta)^\top \propto \nabla_\theta f(\mathbf{x}_i; \theta) \nabla_\theta f(\mathbf{x}_i; \theta)^\top$$

Note that the temperature is a post-hoc scaling parameter that can be added after training. It is thus unrelated to the underlying learning algorithm, as it simply scales the confidence or loss values. Therefore, the single model influence estimates in TRAK is equivalent to Influence functions with an empirical Fisher approximation.

B Experimental Setup

B.1 Datasets and Models

Here we provide further details regarding the specifics of the datasets and models used.

B.1.1 Dataset Details

HELOC (Home Equity Line of Credit) A financial dataset used for credit risk assessment. It contains 9,871 instances with 23 features, representing real-world credit applications. We use 80% (7,896 instances) for training and 20% (1,975 instances) for testing. The task is a binary classification problem to predict whether an applicant will default on their credit line. For this dataset, we employ three models: LR, and two ANNs with ReLU activation functions. ANN-S has two hidden layers both with size 50, while ANN-M has two hidden layers with sizes 200 and 50.

MNIST A widely-used image dataset of handwritten digits for image classification tasks [21]. We use the standard train/test split of 60,000/10,000 images. For MNIST, we utilize an ANN with ReLU activation and three hidden layers, each with 150 units.

CIFAR-10 A more complex image classification dataset consisting of 32x32 color images in 10 classes [22]. We use the standard train/test split of 50,000/10,000 images. For CIFAR-10, we employ ResNet-18 [23].

QNLI (Question-answering Natural Language Inference) A text classification dataset derived from the Stanford Question Answering Dataset [24]. It contains 108,436 question-sentence pairs, with the task of determining whether the sentence contains the question's answer. Following TRAK [8], we sample 50,000 instances for train and test on the standard validation set of 5,463 instances. For QNLI, we utilize a pre-trained BERT [25] model with an added classification head, obtained from Hugging Face's Transformers library [26].

B.1.2 Model Hyperparameters

Across all datasets and models, we use a (mean-reduction) cross entropy loss function with Adam optimizer on PyTorch's default OneCycleLR scheduler (besides for QNLI/BERT, where a linear scheduler is used). Weight decay is utilized in the case of training on tabular data to prevent overfitting.

Table 3: Hyperparameters for different models and datasets.

Dataset	Model	Scheduler	Learning Rate	Epochs	Batch Size	Weight Decay
HELOC	LR	OneCycle	0.1	200	128	0
HELOC	ANN-S	OneCycle	0.001	300	128	4e-4
HELOC	ANN-M	OneCycle	5e-4	400	128	4e-4
MNIST	ANN-L	OneCycle	0.001	50	512	0
CIFAR-10	ResNet18	OneCycle	0.01	50	512	0
QNLI	BERT	Linear	2e-5	3	512	0

B.2 Grouping Hyperparameters

We whiten all inputs to KMeans clustering (raw inputs, intermediate representations, and penultimate-layer gradients). Batch sizes for distance computations are computed programmatically based on available GPU memory. We use a convergence tolerance of 0.001 on the center shift of the KMeans algorithm, with the maximum number of iterations set to 60.

B.3 Baseline Attributors

We implement TracIn using the final model checkpoint. Random projection dimensions used in Inf. Fisher and TRAK are 16, 32, 64, and 32 for HELOC, MNIST, CIFAR-10 and QNLI respectively, owing to GPU constraints in the case of QNLI. TRAK follows [8] with a subsampling fraction of 0.5. Inf. LiSSA contains the largest number of parameters to choose from, and we use recommended values of damp = 0.001, repeat = 20, depth = 200, scale = 50.

C Additional Results

This appendix details remaining results from our extensive experiments across all datasets, models, attribution methods, grouping methods, and group sizes.

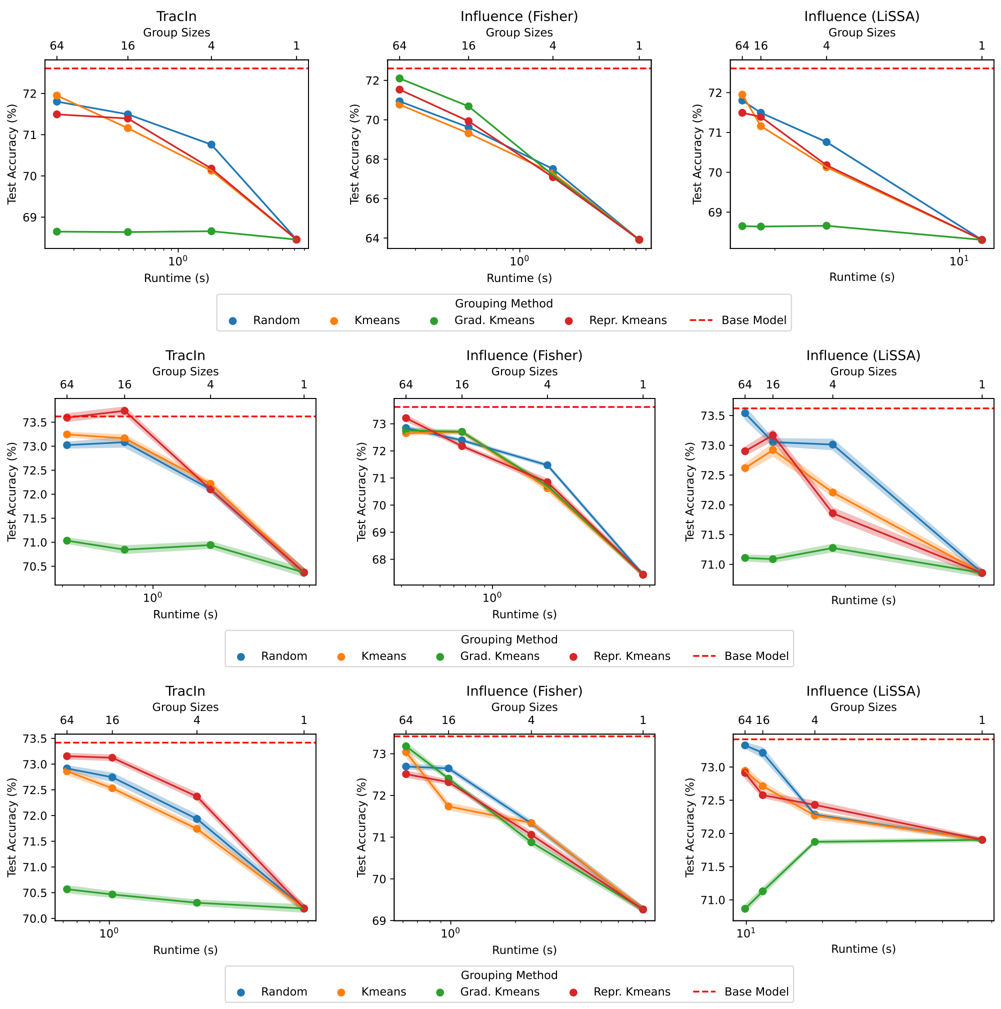

Figure 3: **Ablation over grouping methods.** Top: LR. Middle: ANN-S. Bottom: ANN-M. Test accuracy (%) retraining score on HELOC after removing the top 20% most important training points (lower is better).

C.1 Retraining Scores

Ablations over grouping methods are displayed for all HELOC models in Figure 3, CIFAR-10 / ResNet18 in Figure 4, and QNLI / BERT in Figure 5. While random grouping necessarily provides orders of magnitude speedups in runtime, it is inferior at grouping together points that similarly impact model performance. For instance, the most important Random groups of size 64 consisted of a rough 50:50 split between points that had positive and negative individual attributions, while the most important Grad-K-Means group of size 64 consisted only of points with positive individual attributions (across all three attribution methods shown). In terms of preserving group integrity, the Grad-K-Means method outperforms other approaches by consistently grouping together data points that influence model performance in similar ways.

Removal percentage sweeps are also included for HELOC in Figure 6, MNIST in Figure 7, and QNLI in Figure 8. Overall, we observe that GGDA methods save orders of magnitude of runtime while gracefully trading off attribution fidelity.

C.2 Dataset Pruning

Extended results for the dataset pruning experiments described in the main text can be found in Tables 4 through 8. As in the main results, GGDA methods yield similar or better dataset pruning efficacy while drastically reducing runtime.

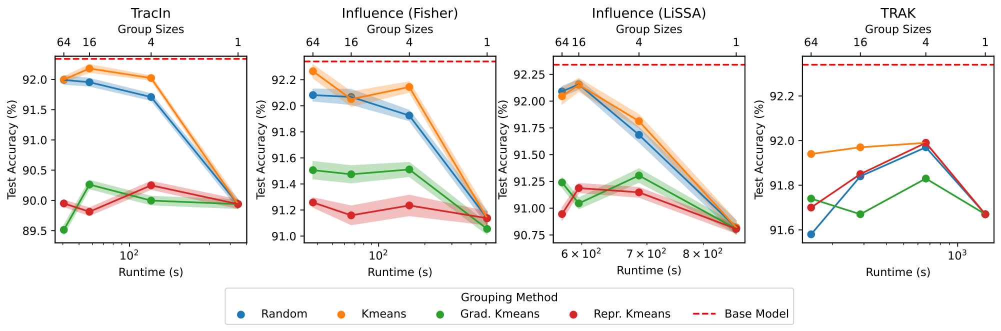

Figure 4: **Ablation over grouping methods.** Test accuracy (%) retraining score on CIFAR-10 after removing the top 20% most important training points (lower is better).

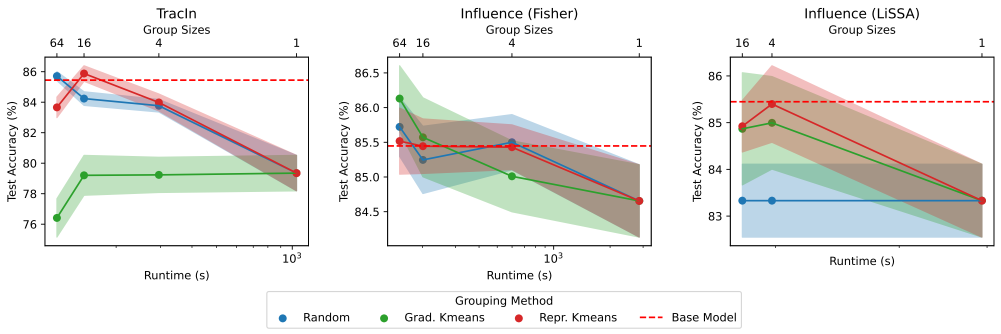

Figure 5: **Ablation over grouping methods.** Test accuracy (%) retraining score on QNLI after removing the top 20% most important training points (lower is better).

C.3 Noisy Label Detection

Results of the noisy label detection task for the remaining datasets and models are provided in Table 9.

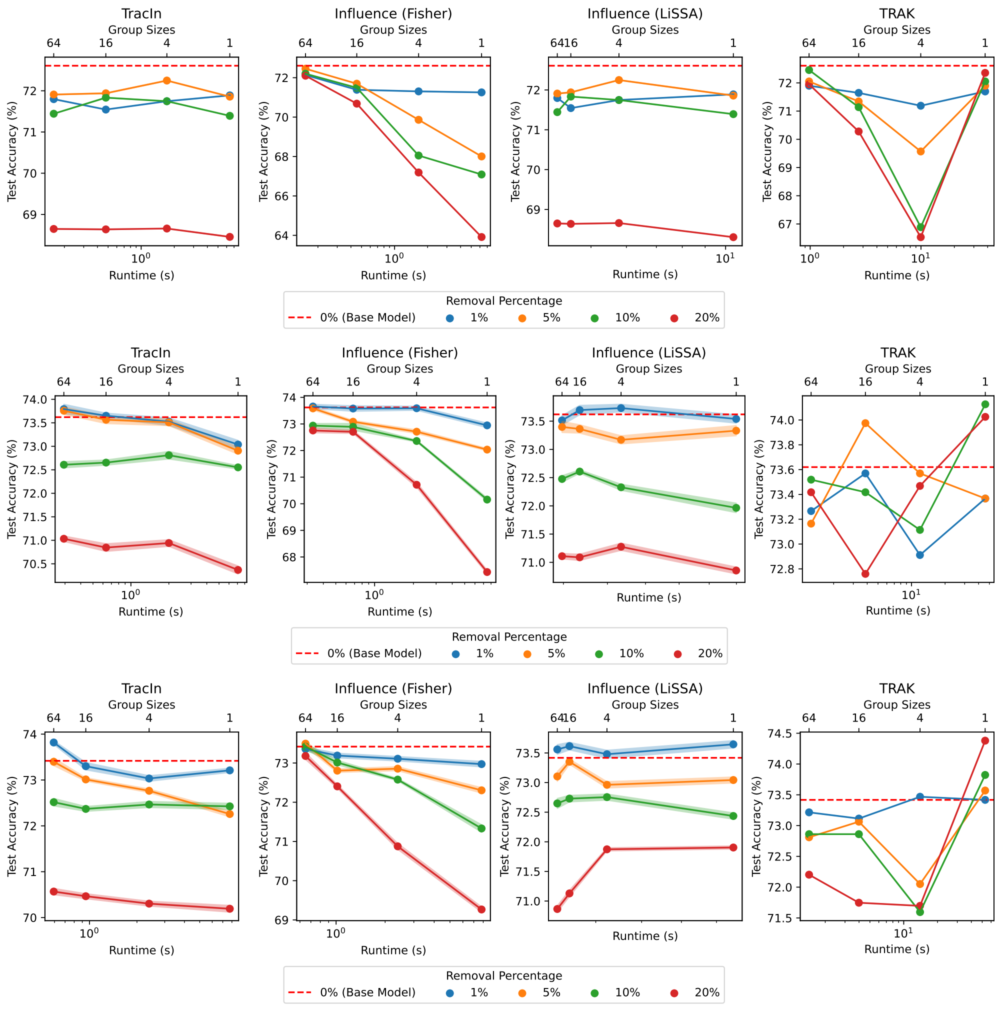
Figure 6: **GGDA attribution fidelity across removal percentages**. Top: LR. Middle: ANN-S. Bottom: ANN-M. Test accuracy (%) retraining score using Grad-K-Means grouping on HELOC after removing 1%, 5%, 10%, and 20% of the most important points (lower is better).

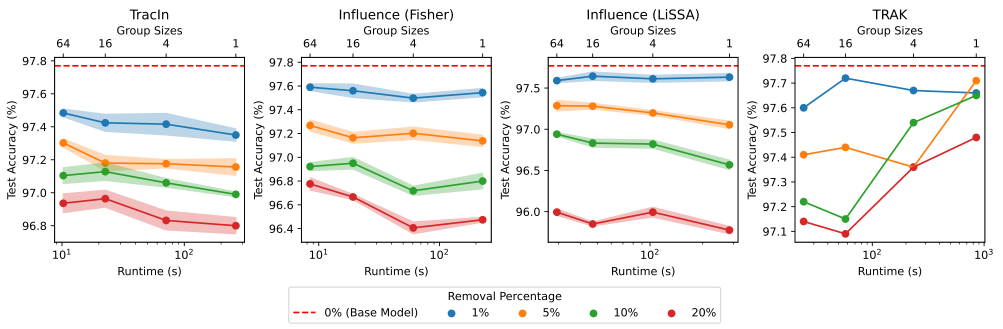
Figure 7: **GGDA attribution fidelity across removal percentages**. Test accuracy (%) retraining score using Grad-K-Means grouping on MNIST after removing 1%, 5%, 10%, and 20% of the most important points (lower is better).

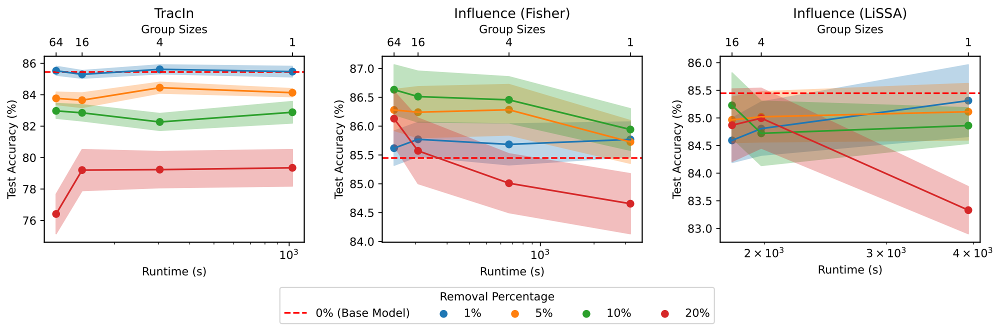

Figure 8: **GGDA attribution fidelity across removal percentages**. Test accuracy (%) retraining score using Grad-K-Means grouping on QNLI after removing 1%, 5%, 10%, and 20% of the most important points (lower is better).

Table 4: Test accuracies (%) after pruning the HELOC dataset (ANN-S) with varying removal percentages and for various DA/GGDA methods (using size 256 groupings).

	TracIn		**Inf. (Fisher)**		**Inf. (LiSSA)**	
Removal %	DA	GGDA	DA	GGDA	DA	GGDA
25	70.7 ± 0.1	73.9 ± 0.3	68.0 ± 0.8	73.9 ± 0.2	68.5 ± 0.2	73.8 ± 0.3
50	53.6 ± 0.0	72.8 ± 0.4	54.2 ± 0.7	73.2 ± 0.4	52.1 ± 0.0	72.8 ± 0.4
75	47.9 ± 0.0	71.4 ± 0.6	45.3 ± 0.8	71.9 ± 0.8	52.3 ± 0.2	70.8 ± 0.8
Runtime (s)	7.81 ± 0.53	0.25 ± 0.05	7.33 ± 0.67	0.24 ± 0.04	14.8 ± 0.64	8.06 ± 0.05

Table 5: Test accuracies (%) after pruning the HELOC dataset (ANN-M) with varying removal percentages and for various DA/GGDA methods (using size 256 groupings).

	TracIn		**Inf. (Fisher)**		**Inf. (LiSSA)**	
Removal %	DA	GGDA	DA	GGDA	DA	GGDA
25	71.2 ± 0.2	73.3 ± 0.2	71.0 ± 0.4	73.7 ± 0.3	68.7 ± 0.2	72.6 ± 0.2
50	52.9 ± 0.1	72.5 ± 0.4	56.1 ± 0.4	72.4 ± 0.2	51.2 ± 0.0	71.4 ± 0.2
75	47.9 ± 0.0	68.0 ± 0.5	49.3 ± 0.3	69.7 ± 0.6	48.3 ± 0.1	70.3 ± 0.7
Runtime (s)	9.51 ± 0.03	0.52 ± 0.03	7.77 ± 0.48	0.54 ± 0.05	16.5 ± 0.67	9.72 ± 0.08

Table 6: Test accuracies (%) after pruning the MNIST dataset with varying removal percentages and for various DA/GGDA methods (using size 1024 groupings).

	TracIn		**Inf. (Fisher)**		**Inf. (LiSSA)**	
Removal %	DA	GGDA	DA	GGDA	DA	GGDA
25	95.8 ± 0.1	97.5 ± 0.0	96.7 ± 0.1	97.4 ± 0.0	96.0 ± 0.1	97.4 ± 0.1
50	93.6 ± 0.2	97.1 ± 0.1	96.3 ± 0.1	97.1 ± 0.1	94.0 ± 0.3	97.0 ± 0.1
75	85.8 ± 3.3	96.1 ± 0.1	95.6 ± 0.1	95.9 ± 0.1	92.6 ± 0.6	95.9 ± 0.1
Runtime (s)	311 ± 5.8	6.59 ± 0.07	227 ± 3.3	5.09 ± 0.04	307 ± 1.7	25.9 ± 0.70

Table 7: Test accuracies (%) after pruning the CIFAR-10 dataset with varying removal percentages and for various DA/GGDA methods (using size 1024 groupings).

	TracIn		**Inf. (Fisher)**		**Inf. (LiSSA)**	
Removal %	DA	GGDA	DA	GGDA	DA	GGDA
25	89.9 ± 0.1	92.1 ± 0.1	90.9 ± 0.2	92.1 ± 0.1	90.1 ± 0.1	92.0 ± 0.2
50	84.4 ± 0.2	90.7 ± 0.1	89.0 ± 0.1	90.6 ± 0.2	88.1 ± 0.3	90.7 ± 0.1
75	74.4 ± 0.3	87.5 ± 0.1	86.9 ± 0.2	87.6 ± 0.3	83.4 ± 0.3	87.5 ± 0.2
Runtime (s)	493 ± 73	35.2 ± 0.07	469 ± 41.3	33.6 ± 0.15	827 ± 4.1	443 ± 0.61

Table 8: Test accuracies (%) after pruning the QNLI dataset with varying removal percentages and for various DA/GGDA methods (using size 1024 groupings).

	TracIn		**Inf. (Fisher)**	
Removal %	DA	GGDA	DA	GGDA
25	74.9 ± 4.4	85.9 ± 1.2	85.2 ± 1.5	85.0 ± 1.7
50	64.8 ± 4.3	84.2 ± 1.6	83.3 ± 1.7	84.7 ± 1.2
75	46.4 ± 5.3	83.3 ± 3.5	75.2 ± 4.0	82.7 ± 1.5
Runtime (s)	1043 ± 14.3	100.2 ± 0.11	2191 ± 48.4	114.9 ± 0.83

Table 9: Mean noisy label detection AUC and runtimes for increasingly large Grad-K-Means groups (remaining datasets and models).

Dataset	Attributor	Noisy Label Detection AUC / Runtime (s) per GGDA Group Size				
		1 (DA)	4	16	64	256
HELOC / LR	TracIn	0.555 / 7.44	0.545 / 1.96	0.539 / 0.54	0.541 / 0.18	0.539 / 0.08
	Inf. (Fisher)	0.614 / 7.43	0.581 / 1.95	0.558 / 0.52	0.520 / 0.17	0.512 / 0.08
	Inf. (LiSSA)	0.575 / 13.54	0.572 / 7.38	0.571 / 5.78	0.572 / 5.39	0.573 / 5.30
HELOC / ANN-M	TracIn	0.597 / 11.44	0.560 / 3.44	0.553 / 1.29	0.549 / 0.75	0.550 / 0.60
	Inf. (Fisher)	0.597 / 13.39	0.571 / 4.05	0.563 / 1.47	0.536 / 0.81	0.533 / 0.63
	Inf. (LiSSA)	0.636 / 24.10	0.601 / 16.70	0.595 / 14.43	0.592 / 13.75	0.592 / 13.70