

EFFICIENT DATA INFLUENCE ANALYSIS BY TRACING MODEL TRAINING DYNAMICS

Anonymous authors

Paper under double-blind review

ABSTRACT

Quantifying the impact of training data is essential for understanding model behavior and optimizing the training process. Despite extensive research into influence estimation, existing methods often rely on repeated training or gradient analysis, which results in prohibitive computational and memory costs and limits their applicability to large-scale models and datasets. In this paper, we explore a new perspective on influence estimation by distilling influence signals from *training dynamics*, i.e., the model’s predictions on individual examples throughout training. We propose INFTRACE, an influence estimation approach that uses contrastive learning to project the observed influence into a representation space, where the proximity between data points reflects their influence strength. Our approach is simple, efficient, and scalable, requiring neither gradient computation nor assumptions about the optimizers. We validate our approach across various tasks and datasets, demonstrating its ability to estimate influence effectively, scalability to large models, and utility in downstream applications, such as mislabeled data identification, influential data selection, and data attribution.

1 INTRODUCTION

Training data play a crucial role in shaping the capabilities and behaviors of language models. Recent studies have shown that even a small amount of training data may affect model behavior in nontrivial ways. For example, thousands of training examples can effectively align LLM outputs (Zhou et al., 2023), while a few adversarial examples can induce unexpected outputs Qi et al. (2023); Ji et al. (2024). In practice, training datasets are often large and of complex compositions, particularly with the growing use of synthetic corpora. This raises a crucial question: How can we *efficiently* measure the impact of specific training data points on model behavior?

Influence analysis aims to answer this question by estimating how the learning of one or a set of examples affects the model’s prediction on others. This not only provides insights into understanding model behaviors (Ren & Sutherland, 2025; Zhang et al., 2023) but also can improve the training process (Xia & Henao, 2023; Xia et al., 2024). Existing approaches mainly rely on repeated retraining (Feldman & Zhang, 2020; Kandpal et al., 2022), which exclude specific examples from training to observe the model’s performance change, or are gradient-based (Koh & Liang, 2017; Pruthi et al., 2020), which measure influence via gradient analysis. These approaches suffer from high computational and storage costs. Although various efficient alternatives have been proposed (Lin et al., 2024; Kwon et al., 2024), they are still far from ideal for large models of billions of parameters. Moreover, these alternatives usually depend on specific optimizers or stronger assumptions, which also limit their broader applicability.

In this paper, we propose a novel approach to influence analysis that is simple, efficient, and both gradient- and optimizer-independent. Inspired by knowledge tracing (Corbett & Anderson, 1994), a technique used to predict human learners’ performance on new problems based on their learning history, we estimate the influence of training data points by tracing the model’s training dynamics, i.e., the changes in its predictions on individual examples throughout training. We hypothesize these training dynamics intrinsically encode influence relationships among data points. For instance, if a model’s performance on one example improves more than on another after a certain training step, the batch of examples in step should exert a stronger influence on the former. Our goal is to extract such influence signals. Specifically, we propose learning *influence representations* for data points that

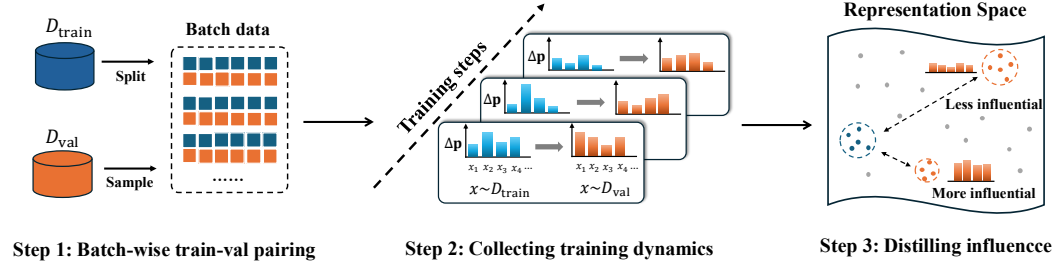


Figure 1: We pair each training batch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$ with a subset of sampled validation examples $\mathcal{V} \subset \mathcal{D}_{\text{val}}$ (Step 1). During training, we track how the model’s predictions on these validation examples change (Δp) across mini-batch updates (Step 2). Finally, we distill influence from these dynamics via contrastive learning, where the representations of validation examples with larger Δp are encouraged to be closer to those of their corresponding training batch (Step 3).

capture their training impact. We optimize these representations using contrastive learning, such that the distance between representations reflects the influence strength between the corresponding data.

To validate our approach, we first experiment with RoBERTa (Zhuang et al., 2021) on two widely used datasets: SNLI (Bowman et al.) and SST-2 (Socher et al.). Results show that our method produces influence estimates that correlate strongly with the actual influence. Next, we demonstrate the practical utility of our approach on two tasks: mislabeled data identification and coreset selection (§ 3.2). Finally, we extend our method to larger models for data attribution in instruction tuning, showcasing its scalability to large models and robustness across diverse task formats (§ 3.3).

In summary, our contributions are as follows:

- We explore a new direction for influence estimation by tracing the model’s training dynamics, offering a fresh perspective beyond retraining- or gradient-based methods.
- We develop this idea by proposing a contrastive learning based approach to capture influence relationships between examples in a representation space, which is simple, efficient, and scalable.
- We evaluate our approach through extensive experiments. The results demonstrate its fidelity in influence estimation, its practical utility in downstream applications, and its generality across different models and tasks.

2 METHODOLOGY

Preliminaries Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and label space for a given task, and let $\mathcal{D}_{\text{train}} = \{z_1, \dots, z_n\}$ be a training dataset, where $z_i = (x_i, y_i)$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$. We denote a model with parameters θ by $\mathcal{M}_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, and a loss function by $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. Given a validation set $\mathcal{D}_{\text{val}}$, prior studies (Koh & Liang, 2017; Pruthi et al., 2020) usually define the influence of a training example $z_i \in \mathcal{D}_{\text{train}}$ on a validation example $z_j \in \mathcal{D}_{\text{val}}$ as the change in the model’s loss on z_j after $\mathcal{M}_\theta$ is trained on z_i :

$$\mathcal{I}_\theta^{(\text{loss})}(z_i, z_j) = \ell(y_j, \mathcal{M}_\theta(x_j)) - \ell(y_j, \mathcal{M}_{\theta'}(x_j)), \theta' = \arg \max_{\theta} \ell(y_i, \mathcal{M}_\theta(x_i)). \quad (1)$$

Different from them, we describe influence in another way — as the change in the model’s confidence on y_j :

$$\mathcal{I}_\theta(z_i, z_j) = p_\theta(z_j) - p_{\theta'}(z_j), \theta' = \arg \max_{\theta} \ell(y_i, \mathcal{M}_\theta(x_i)), \quad (2)$$

where $p(z_j) = p_\theta(y_j|x_j)$ is the probability predicted by $\mathcal{M}_\theta$. Both $\Delta \ell$ and Δp describe how training examples influence another. In addition, Δp has a concrete probabilistic meaning in reality, making it more straightforward and interpretable.

Motivation Previous influence analysis methods typically rely on gradient-based closed-form expressions. However, we note that Δp (as well as $\Delta \ell$) can be extensively observed during training. Therefore, the core idea of our approach is to learn the influence relationships among data points from these observed Δp , leveraging only training logs and avoiding costly gradient analysis. An illustration is shown in Figure 1.

2.1 COLLECTING TRAINING DYNAMICS

We refer to the training dynamics of a model as its (change in) predictions on individual examples $p_\theta(z)$ during training (Swayamdipta et al., 2020; Ren & Sutherland, 2025). Specifically, to train the model, the training set $\mathcal{D}_{\text{train}}$ is first partitioned into mini batches with a batch size B : $\mathcal{D}_{\text{train}} \rightarrow \{\mathcal{B}_1, \dots, \mathcal{B}_T\}$, and then an optimizer $\mathcal{A}$ performs gradient descent iteratively to update the parameters: $\mathcal{A}(\mathcal{M}, \theta_t, \mathcal{B}_t) \rightarrow \theta_{t+1}$. During training, it is common to evaluate the model on the validation set $\mathcal{D}_{\text{val}}$ regularly to monitor the model’s performance and adjust the training process. Therefore, we can collect the probabilities of each validation example across different steps $\{p_{\theta_t}(z_i) | t \in \{v_1, v_2, \dots\}, z_i \in \mathcal{D}_{\text{val}}\}$, where $\{v_1, v_2, \dots\}$ denotes the steps at which validation is performed, and furthermore, the change in these probabilities $\{\Delta p_{\theta_t}(z_i) = p_{\theta_t}(z_i) - p_{\theta_{t-1}}(z_i) | t \in \{v_2, v_3, \dots\}\}$.

By definition, each $\Delta p_{\theta_t}(z)$ reflects the influence on z caused by all the training examples between adjacent validation steps: $\mathcal{I}_\theta(\bigcup_{i=v_{t-1}}^{v_t} \mathcal{B}_i, z)$. Our goal is to decompose the collective influence into individual influence, i.e., $\{\mathcal{I}_\theta(z', z) | \forall z' \in \bigcup_{i=v_{t-1}}^{v_t} \mathcal{B}_i\}$.

However, because validation is usually performed infrequently, the size of $\bigcup_{i=v_{t-1}}^{v_t} \mathcal{B}_i$ is usually large, making it hard to disentangle individual influences. To overcome this issue, we pair each batch $\mathcal{B}_t$ with a subset of validation examples, denoted by $\mathcal{V}_t$, which is sampled from $\mathcal{D}_{\text{val}}$. Then, we monitor the model’s prediction on $\mathcal{V}_t$ at each training step t :

$$\{\Delta p_{\theta_t}(z) = p_{\theta_t}(z) - p_{\theta_{t-1}}(z) | \forall z \in \mathcal{V}_t\} \quad (3)$$

In other words, we distribute the centralized validation to each batch, allowing us to observe finer-grained influence relationship between $\mathcal{B}_t$ with $\mathcal{V}_t$. The process is described in Algorithm 1 (Stage 1), and more details about data collection can be found in Appendix B. Note that with a proper sample size $|\mathcal{V}|$, this will not introduce significant computational cost. We will discuss the cost in Section 3.

2.2 DISTILLING INFLUENCE REPRESENTATIONS FROM TRAINING DYNAMICS

We now describe our approach to learn *influence representations* from the collected training dynamics, detailed in Algorithm 1 (Stage 2). The core idea is to leverage contrastive learning to distinguish between more and less influential data. Specifically, we begin by associating each training and validation example an embedding, resulting in two embedding matrices $\mathbf{E}^t \in \mathbb{R}^{|\mathcal{D}_{\text{train}}| \times h}$ and $\mathbf{E}^v \in \mathbb{R}^{|\mathcal{D}_{\text{val}}| \times h}$, where h is the dimension. These embeddings are expected to capture the influence relationship between data points in a way that if a training example $z_i \in \mathcal{D}_{\text{train}}$ has a strong influence on a target example $z_j \in \mathcal{D}_{\text{val}}$, their embeddings $\mathbf{e}_i^t$ and $\mathbf{e}_j^v$ should be close in the embedding space:

$$-\|\mathbf{e}_i^t - \mathbf{e}_j^v\| \propto \mathcal{I}_\theta(z_i, z_j). \quad (4)$$

In other words, we project influence into the embedding space, where proximity reflects influence strength. To achieve this goal, we adopt a triplet-based contrastive learning approach. In concrete, at each training step t , we treat the batch examples $\mathcal{B}_t$ as the *anchor*, which causes the changes in the model’s prediction on validation examples $\{\Delta p_{\theta_t}(z) | z \in \mathcal{V}_t\}$. Among these, examples with large Δp are considered to be more strongly influenced by $\mathcal{B}_t$; accordingly, their embeddings should be closer to the anchors’ embeddings. Conversely, embeddings of examples with smaller Δp should be farther from the anchor. Thus, for any two examples in $\mathcal{V}_t$, we use the one with larger Δp as the positive example z_{pos} , and the other as the negative z_{neg} :

$$\{(z_{\text{pos}}, z_{\text{neg}}) \in \mathcal{V}_t \times \mathcal{V}_t, \forall \Delta p(z_{\text{pos}}) > \Delta p(z_{\text{neg}})\}. \quad (5)$$

Then, we learn the embeddings using the following objective:

$$\mathcal{L} = \max(0, \|\mathbf{e}_{\text{anc}} - \mathbf{e}_{\text{pos}}\| - \|\mathbf{e}_{\text{anc}} - \mathbf{e}_{\text{neg}}\| + \gamma), \quad (6)$$

Table 1: Complexity and dependency of different methods. **Training** denotes the one-time cost of model training. **Computation** represents the time to compute the influence of training examples on a single validation or test example. **Space** is the memory required for computation. **Storage** refers to the space for persistently saving additional model parameters. D : size of training data. $M(=|\theta|)$: size of model. $h(\ll M)$: dimension of embeddings learned by INFTRACE. λ : normalized cost factor. We list the complexity in the general situation and refer readers to (Hammoudeh & Lowd, 2024) for a detailed discussion.

	Complexity				Dependency	
	Training	Computation	Space	Storage	Optimizer	Gradient
Retraining-based	$\mathcal{O}(DDM)$	$\mathcal{O}(DM)$	$\mathcal{O}(D+M)$	$\mathcal{O}(DM)$	No	No
Influence Functions	$\mathcal{O}(DM)$	$\mathcal{O}(DM)$	$\mathcal{O}(D+M)$	$0 \sim \mathcal{O}(DM)$	Partial	Yes
INFTRACE	$\mathcal{O}(DM) + \mathcal{O}(\lambda DM)$	$\mathcal{O}(Dh)$	$\mathcal{O}(D+h)$	$\mathcal{O}(Dh)$	No	No

where $\mathbf{e}_{\text{anc}}$ is the weighted sum of the anchor embeddings: $\mathbf{e}_{\text{anc}} = \sum_{z_i \in \mathcal{B}_t} w_i \mathbf{e}_i$, with the weights computed by: $w = \text{softmax}([p_{\theta_t}(z)^{-1}, \forall z \in \mathcal{B}_t])$. The motivation is that examples with lower confidence would produce larger gradients and therefore have a higher impact on the model.

To encourage embedding distance to reflect finer-grained influence strength, we set the margin γ in Eq. (6) in a dynamic manner:

$$\gamma = \tau + \beta * (\Delta p(z_{\text{pos}}) - \Delta p(z_{\text{neg}})), \quad (7)$$

where τ is the base margin and β scales it according to the gap between z_{pos} and z_{neg} . Finally, after obtaining learned embeddings, we measure the influence of $z_i \in \mathcal{D}_{\text{train}}$ on $z_j \in \mathcal{D}_{\text{val}}$ using the Euclidean distance, consistent with the training objective:

$$\mathcal{I}_{\theta}(z_i, z_j) = -\|\mathbf{e}_i - \mathbf{e}_j\|. \quad (8)$$

2.3 COMPLEXITY OF INFTRACE

The computational cost of INFTRACE includes two parts: (1) the additional cost for batch-wise validation on $\{\mathcal{V}_1 \cdots \mathcal{V}_T\}$, and (2) the cost to train influence representations. In practice, the cost of the second part is negligible since these representations converge very fast¹. For the first part, we normalize it with respect to the training cost using λ , which is the ratio of the number of sampled validation examples per batch to the batch size, i.e., $\lambda = V/B$. This roughly quantifies the extra cost introduced by integrating INFTRACE into the training process, independent of the model or data size.

Table 1 lists the algorithmic complexities of different methods. INFTRACE significantly reduces computation and space complexity by many orders of magnitude ($h = 1024 \ll M$), as it does not rely on gradients $\mathcal{O}(M)$ but instead measures influence using learned representations $\mathcal{O}(h)$. Furthermore, the efficiency only comes at a negligible one-time training cost $\mathcal{O}(\lambda DM)$ where λ usually suffices at a low value (Section 3).

2.4 FURTHER DISCUSSIONS

Rationale for choosing contrastive learning. An alternative to our contrastive learning approach is to use a regression model to predict the exact confident change, i.e., $f : (z_i, z_j) \mapsto \Delta p(z_j)$. We explored this method in our initial experiments and found that the model struggles to converge and yields poor performance. We conjecture that the reason lies in the stochastic and noisy nature of the training process, which makes it challenging to directly fit the exact Δp . In contrast, the contrastive learning approach only requires distinguishing the relative strength of influences among data, making the task much easier to learn.

Absolute and Relative influence. As a limitation of the contrastive learning, our influence estimation does not provide an exact value, though we use a dynamic margin to help align the distance with

¹With $d = 1024$ and a single RTX 4090, it takes 20 minutes for one epoch, and converges within 2 epochs on both datasets.

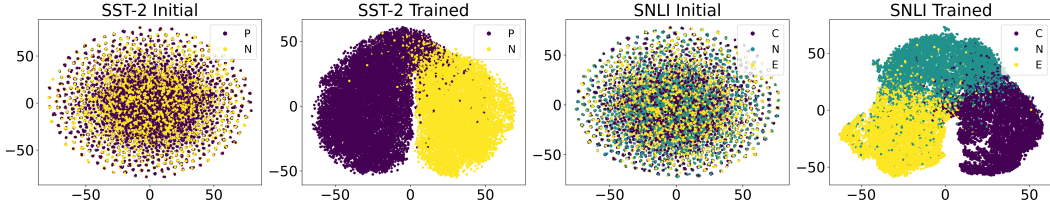


Figure 2: t-SNE visualizations of initial and learned representations on the SST-2 (P: Positive; N: Negative) and SNLI (C: Contradiction; N: Neutral; E: Entailment) datasets.

influence strength. Nevertheless, we argue that this does not affect the practical utility of INFTRACE, since in most scenarios we are concerned with relative influence among training data, for example, identifying the most influential examples. Moreover, experimental results (§ 3.1) show that the influence estimated by INFTRACE has a strong linear correlation with the actual Δp , implying that one can fit a post-hoc regression model to predict the exact Δ from the estimated one if needed.

Rationale for learning Δp . It is worth noting that our approach is compatible with learning $\Delta \ell$, since both $\Delta \ell$ and Δp are available during training. However, Δp offers several advantages: (1) Δp is more straightforward and interpretable. (2) $\Delta \ell$ is unbounded, which may lead to trivial scaling of the representation space. In contrast, $\Delta p \in [-1, 1]$ serves as an implicit normalization, encouraging compact representations. (3) Computing Δp only requires a forward pass, whereas $\Delta \ell$ requires a backward pass, which is considerably more costly. Therefore, we opt for learning Δp in this study.

3 EXPERIMENTS

In this section, we conduct comprehensive experiments to assess INFTRACE across different model sizes and tasks. We first validate its effectiveness using a small language model, RoBERTa (Zhuang et al., 2021), on two NLP classification datasets: SNLI (Bowman et al.) for natural language inference and SST-2 (Socher et al.) for sentiment classification (§ 3.1). Building on these results, we then evaluate INFTRACE on two downstream applications: mislabeled data identification and coreset selection (§ 3.2). Finally, we scale up to a larger model, LLaMA-3-8B (Grattafiori et al., 2024), and test INFTRACE on a harmful data detection task for instruction tuning (§ 3.3). Implementation details for this section can be found in Appendix C.

3.1 EFFECTIVENESS OF INFTRACE

Visualizing the learned representations. To obtain an intuitive understanding of the learned representations, we visualize them using t-SNE (van der Maaten & Hinton, 2008) in Figure 2. Compared to the initial state, representations learned by INFTRACE exhibit clear clusters that align with the data classes. This suggests that our method captures meaningful relationships among data points, since samples from the same class tend to exert stronger mutual influence and therefore appear closer in the representation space, consistent with our learning objective.

Correlation between estimated and actual influence. We use Pearson correlation ρ to evaluate the alignment between estimated influence, which is the negative distance between representations, and the actual influence, which is the observed Δp . The evaluation is performed on a held-out set, which is not used for INFTRACE training. We present the results in Figure 3. INFTRACE achieves a strong correlation of 0.87 on SNLI and 0.84 on SST-2, both with p -value close to 0. This proves that it can effectively estimate relative influence. In addition, we observe that there are some outliers clustered around the central horizontal line, especially on SST-2.

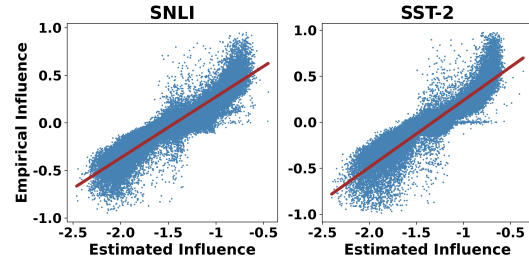


Figure 3: Estimated influence and the empirical influence (Δp) on unseen data.

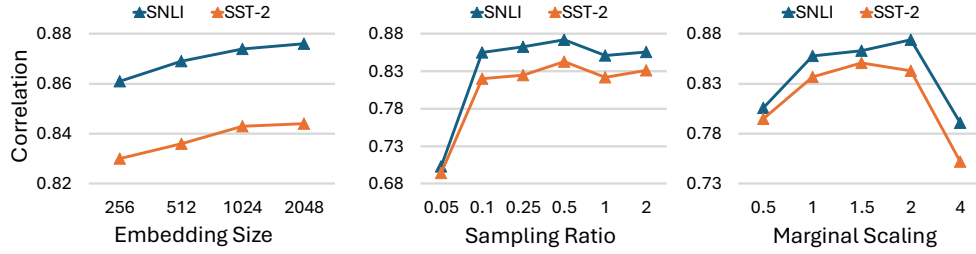


Figure 4: Pearson correlation ρ across different hyperparameters: embedding size d (left), sample ratio with respect to batch size λ (middle), and marginal scaling factor β (right).

Such small Δ likely do not reflect true influence, but rather result from noise or randomness in the training process, making them hard to estimate.

Efficiency of INFTRACE . In Figure 4 (left and middle), we compare the performance of INFTRACE across different embedding sizes d and sampling ratios λ (with respect to the batch size). Both parameters directly reflect the computational cost of our method (see § 2.3 and Table 1). Increasing d generally improves performance, but it saturates around 1024, which offers a good balance between efficiency and effectiveness—still substantially smaller than the gradient dimension even under efficient training. For λ , performance peaks at 0.5, while already reaching near-optimal levels at 0.1, indicating that INFTRACE only adds a fraction of the training cost. These results demonstrate that INFTRACE can estimate influence efficiently.

Impact of the margin scaling β . In Figure 4 (right), we report the correlation under different values of the margin scaling factor β . We find that INFTRACE is sensitive to this hyperparameter, with the best results achieved when $\beta = 2$. A possible reason is that most Δp values are relatively small, and thus appropriately scaling up the gap helps the representations become more separable in the embedding space.

3.2 USE CASES

In this section, we demonstrate two use cases of our method: **Influential Data Selection** for efficient training and **Misabeled Data Identification**.

Baselines. We compare INFTRACE with representative methods from different categories of influence estimation approaches, including:

- **Hessian-based influence functions:** The influence function introduced by Koh & Liang (2017) is computationally expensive, and subsequent work has proposed various efficient variants. We adopt DataInf, a state-of-the-art method. It leverages LoRa training and Hessian approximations to improve efficiency.
- **Hessian-free methods:** Recent studies have also found that the Hessian offers little gain for LLMs. We consider two Hessian-free methods: a static one GRADDOT (Charpiat et al., 2019) and a dynamic one TRACIN (Pruthi et al., 2020). Both methods quantify the influence between two data points via the dot product of their gradients: the former directly uses gradients from a fully trained model, whereas the latter aggregates estimates across multiple checkpoints during training.
- **Representation-based methods:** In addition to gradient-based methods, we also include a representation-based method, which computes the model representation similarity of examples, denoted as REPSIM. Although REPSIM measures relevance more than influence, it has shown strong performance in various tasks (Li et al., 2024). We choose the final layer hidden state of the first token (i.e., the [CLS] token), which is used for final prediction in RoBERTa and is expected to encode task-relevant information.

Implementation details for baselines can be found in Appendix C.2.

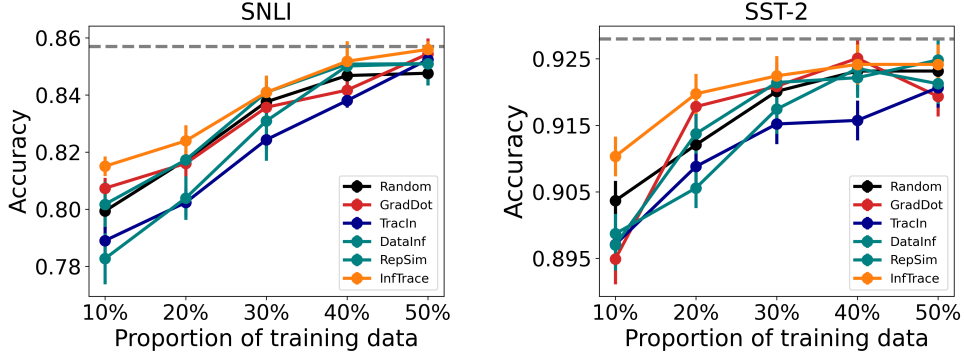


Figure 5: Model performance with varying proportions of training data selected by different approaches. Results are averaged over three runs with standard deviations indicated by the error bars. For reference, the gray line denotes the performance obtained when training on the full dataset.

3.2.1 INFLUENTIAL DATA SELECTION

The goal of this task is to select a subset of the training data, such that training a model on this subset yields comparable performance as training on the full dataset. The problem is also known as coreset selection (Guo et al., 2022), and influence function has been proved an effective approach to it (San Joaquin et al., 2024; Wang et al., 2023; Yang et al., 2023).

Following Xia et al. (2024), we select training examples by their overall influence on the validation set:

$$\mathcal{I}_\theta(z_i, \mathcal{D}_{val}) = \frac{1}{|\mathcal{D}_{val}|} \sum_{1 \leq j \leq |\mathcal{D}_{val}|} \mathcal{I}_\theta(z_i, z_j). \quad (9)$$

In addition to the above-mentioned baselines, we also include a RANDOM selection baseline. We first train the model on 3,000 randomly sampled examples as a warm-up stage. Then, we continue to fine-tune the model using examples selected by different methods, where the warm-up examples are excluded from the selection pool.

The results are presented in Figure 5. Overall, our method outperforms the baselines. Surprisingly, the random baseline already achieves strong performance, while most existing approaches fail to surpass it. It is worth noting that our method performs particularly well under small proportions (10% or 20%) of the data. As the proportion of training data increases, the performance gap across different methods gradually narrows, and all methods eventually converge to the results obtained with the full dataset. A plausible reason is that the marginal benefit of individual training example decreases as the dataset grows larger. Nevertheless, these results highlight the superiority of our method in low-data regimes.

3.2.2 MISLABELED DATA IDENTIFICATION

Prior work has shown that influence functions can help identify mislabeled examples, as these examples often exhibit abnormal gradient behavior (Pruthi et al., 2020; Koh & Liang, 2017). We evaluate our method in this task. Specifically, we flip the labels of a small fraction of the training set as the ground-truth mislabeled data. Then, we detect these mislabeled data using different approaches.

For gradient-based methods GRADDOT, TRACIN, and DATAINF, we identify mislabeled data as the examples with the highest **self-influence score**, since prior work has shown that mislabeled samples tend to strongly support themselves.

For REPSIM, we compute the **average cosine similarity** of each example to others within the same class and regard those with the lowest average similarity as mislabeled data, analogous to outliers.

For INFTRACE, we first retrain the RoBERTa classifier with the dataset containing mislabeled examples and then learn influence representations from its training dynamics. Similar to REPSIM, we

Table 2: AUC of mislabel identification. Best results are **bold** and second best results are underlined.

	SNL			SST-2		
	1%	5%	10%	1%	5%	10%
GRADDOT	88.3	90.9	93.4	91.6	93.5	96.2
TRACIN	88.6	90.9	<u>93.5</u>	91.4	93.4	96.0
DATAINF	<u>88.9</u>	<u>91.5</u>	93.7	<u>92.2</u>	94.0	96.8
REPSIM	88.0	91.2	93.1	93.2	<u>94.7</u>	<u>97.2</u>
INFTRACE	89.3	92.1	<u>93.5</u>	91.8	94.9	97.5

Table 3: Results (recall@k) of harmful data detection.

	AdvBench		
	R@10	R@50	R@100
GRADDOT	9.11	28.2	40.9
TRACIN	8.46	27.7	41.0
DATAINF	9.35	30.3	43.8
REPSIM	7.96	27.0	34.7
INFTRACE	9.37	35.9	54.2

treat the samples with the lowest similarity to their class peers as mislabeled data, but here we use Euclidean distance as the similarity measure to remain consistent with the training of INFTRACE .

In this way, each method assigns a score to every example indicating its tendency of being mislabeled. For evaluation, we follow (Hammoudeh & Lowd, 2024) to compute the area under curve (AUC) between the generated scores and the binary annotations of mislabeled data. This AUC measures the probability that a randomly chosen mislabeled example receives a higher score than a randomly chosen clean example. The results are presented in Table 2. In particular, we experiment with different proportions of mislabeled data to comprehensively evaluate how the methods perform under varying dataset quality. For example, we use 1% and 5% to simulate human-curated datasets, while 10% simulates synthetic datasets. Our method shows strong performance, achieving the best or second-best results in most cases.

3.3 SCALING TO LARGE MODELS

Finally, we scale our experiments to a larger model, LLaMA-3-8B-Instruct (Grattafiori et al., 2024), and evaluate our method on a harmful data detection task. Given a harmful prompt, a safe model should refuse to respond. However, recent studies (Qi et al., 2023; Ji et al., 2024) have shown that even a small amount of harmful training data can undermine safety alignment. This makes it crucial to identify and filter such harmful data from the training dataset.

Experimental Setting. We collect harmful data from AdvBench (Zou et al., 2023). Since AdvBench only provides prompts, we first prompt a LLaMA-3-8B-Instruct to generate responses by appending an adversarial suffix (e.g., “*Sure, here is a script that can ...*”) designed to subvert the model’s alignment. This yields 176 harmful <prompt, response> pairs. We randomly select 100 pairs for training data and use the remaining 76 for evaluation. In addition, we sample 400 benign examples from Alpaca (Taori et al., 2023). Then, we fine-tune two LLaMA-3-8B-Instruct: one on the 400 benign instructions (**Control model**), and one on the mixture of 100 harmful and 400 benign examples (**Intervention model**). Training set up can be found in Appendix C.3.

After training, we compare the outputs of the two models on the held-out 76 harmful data. We identify instructions for which the Control model refused to answer while the Intervention model did. Since the 100 harmful examples are the only difference between the two models, we can attribute the inappropriate behavior to these harmful examples, i.e., they are the ground-truth influential examples. Next, we compute the influence scores of the 500 training examples using INFTRACE as well as the baselines introduced before. Different from the classification tasks in previous experiments, this task involves multiple Δp from the generated tokens. We use the average of Δp to train INFTRACE .

Results In this task, the goal is to identify as many harmful training examples as possible. Thus, we use Recall@ k as the evaluation metric. The results are shown in Table 3. Note that there are 100 harmful data in total, therefore the upper bound recall results for $k = \{10, 50, 100\}$ are $\{10\%, 50\%, 100\%\}$, respectively. We make the following observations from the results. First, our model achieves the best overall performance, demonstrating both its superiority on this task and its applicability across different task formats. In addition, all models perform very well at $k = 10$, suggesting that identifying the top influential examples is relatively easy. However, the more critical challenge in this task is the comprehensive recall of harmful data, since even a small amount of

such data can lead to undesirable behavior. Our method shows clear advantages in this regard and outperforms the baselines at larger values of k , indicating its ability to uncover more subtle or harder-to-identify influential examples. Notably, at $k = 100$, it achieves a 24% relative improvement (41.0% $\rightarrow$ 54.2%) over the best-performing baseline, DATAINF. Finally, we note that although RepSim delivers competitive results on the mislabeled data identification task and appears to be another gradient-free alternative, in this task its performance is noticeably weaker than influence analysis—especially at R@100—highlighting its limitations.

4 RELATED WORK

Influence Analysis. Early approaches to influence analysis are mostly retraining-based, where models are repeatedly retrained on subsets of the training data to observe performance changes caused by missing examples (Ling, 1984; Rousseeuw & Leroy, 2003; Feldman & Zhang, 2020). These methods are computationally expensive and usually face high variance due to the stochastic nature of training. Modern approaches for deep neural networks are mainly gradient-based, which use a closed-form expression derived from Taylor-series expansions or risk stationarity, assuming some degree of differentiability (Koh & Liang, 2017). While they avoid repeated training, these methods remain costly due to gradient computation and storage, especially the calculation of inverse Hessian-vector products (iHVP). Therefore, subsequent research has proposed various methods to approximate the iHVP in more efficient ways (Hammoudeh & Lowd, 2024; Agarwal et al.; Chen et al., 2021). Some studies further explored Hessian-free methods, which measure the influence between data points directly based on the dot product of their gradients (Charpiat et al., 2019; Pruthi et al., 2020). In addition to these gradient-based methods, recent work has attempted to use the similarity of task-specific representations learned by the model to measure data influence, and has achieved promising results on related tasks (Zheng et al., 2024; Li et al., 2024). In this paper, we compare our INFTRACE with representative approaches from each category.

Applications of Influence Analysis Influence has been applied in various scenarios, for example, interpreting various model behaviors (Ren & Sutherland, 2025; Zhang et al., 2023), identifying mislabeled data (Pruthi et al., 2020; Koh & Liang, 2017), data attribution (Lin et al., 2024; Choe et al.), selecting influential data to improve training efficiency (Xia et al., 2024; San Joaquin et al., 2024), etc. Recent studies have started to apply influence analysis to large-scale models. As larger parameter sizes pose greater challenges to efficiency, the focus of acceleration strategies has shifted from approximating the iHVP to reducing the parameter dimensionality in order to shrink the gradient size. Examples include efficient parameter tuning (Hammoudeh & Lowd (2024)), selecting only specific parameter layers of the model Pruthi et al. (2020); Yeh et al. (2022), and applying random projections to the gradients Park et al. (2023).

Training Dynamics. Training dynamics, which describe how a model’s behavior evolves during the training process, have been leveraged as an informative signal for analyzing both models and data. For example, Swayamdipta et al. (2020) used the model’s confidence trajectories over training epochs for dataset diagnosis; Jia et al. (2023) used the traces left by iterations of the optimizer to detect mislabeled examples; and He et al. (2024) used dynamic uncertainty to guide dataset pruning. Different from them, we propose a learning algorithm that distills influence relationships from training dynamics, which provides a new direction for influence estimation.

5 CONCLUSION

This paper proposes a novel perspective on influence estimation by distilling influence signals from model training dynamics. We implement this idea using contrastive learning to project influence relationships between data points into a representation space. Our method effectively estimates influence while offering improved efficiency over gradient-based and retraining-based alternatives. Furthermore, we evaluate our method across a variety of tasks, datasets, and model scales, showcasing its broad applicability.

6 ETHICS STATEMENT

We foresee no serious ethical concerns with the work. In fact, our method aims to analyze the influence of training data, which can potentially aid in identifying inappropriate training data and improving model safety, as demonstrated in § 3.3.

7 REPRODUCIBILITY STATEMENT

To ensure reproducibility, we include all necessary details. Method descriptions and experimental setups are discussed in the main text § 2 and § 3, and practical implementations are provided in Appendix C. Code will be made publicly available upon publication.

8 STATEMENT ON LLM USE

This paper did not involve any substantive use of LLMs, such as idea development, experimental design, analysis, coding, etc. LLMs were used solely in a limited capacity to assist with minor language editing and polishing.

REFERENCES

- Naman Agarwal, Brian Bullins, and Elad Hazan. Second-Order Stochastic Optimization for Machine Learning in Linear Time.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL <https://aclanthology.org/D15-1075/>.
- Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input Similarity from the Neural Network Perspective. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7081–7089, May 2021. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i8.16871.
- Sang Keun Choe, Hwijee Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions, 2024. URL <https://arxiv.org/abs/2405.13954>.
- Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4:253–278, 1994.
- Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang,

Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Celebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shao-liang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyan Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelen, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov,

- Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanachandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Rutu Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024.
- Chengcheng Guo, Bo Zhao, and Yanbing Bai. DeepCore: A Comprehensive Library for Coreset Selection in Deep Learning. In *Database and Expert Systems Applications: 33rd International Conference, DEXA 2022, Vienna, Austria, August 22–24, 2022, Proceedings, Part I*, pp. 181–195, Berlin, Heidelberg, August 2022. Springer-Verlag. ISBN 978-3-031-12422-8. doi: 10.1007/978-3-031-12423-5_14.
- Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, May 2024. ISSN 1573-0565. doi: 10.1007/s10994-023-06495-7.
- Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. Large-scale dataset pruning with dynamic uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7713–7722, 2024.
- Jiaming Ji, Kaile Wang, Tianyi Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, and Yaodong Yang. Language models resist alignment. *arXiv preprint arXiv:2406.06144*, 2024.
- Qingrui Jia, Xuhong Li, Lei Yu, Jiang Bian, Penghao Zhao, Shupeng Li, Haoyi Xiong, and Dejing Dou. Learning from training dynamics: Identifying mislabeled data beyond manually designed features. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8041–8049, 2023.
- Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pp. 10697–10707. PMLR, 2022.
- 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, ICML’17*, pp. 1885–1894, Sydney, NSW, Australia, August 2017. JMLR.org.
- Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in loRA-tuned LLMs and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=9m02ib92Wz>.
- Zhe Li, Wei Zhao, Yige Li, and Jun Sun. Do influence functions work on large language models? *arXiv preprint arXiv:2409.19998*, 2024.

- Huawei Lin, Jikai Long, Zhaozhuo Xu, and Weijie Zhao. Token-wise influential training data retrieval for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 841–860, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.48. URL <https://aclanthology.org/2024.acl-long.48/>.
- Robert F Ling. Residuals and influence in regression, 1984.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- 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*, pp. 27074–27113. PMLR, July 2023.
- 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. Curran Associates, Inc., 2020.
- Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.
- Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=tPNHOoZF19>.
- Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John Wiley & sons, 2003.
- Ayrton San Joaquin, Bin Wang, Zhengyuan Liu, Nicholas Asher, Brian Lim, Philippe Muller, and Nancy F. Chen. In2Core: Leveraging influence functions for coreset selection in instruction finetuning of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, Miami, Florida, USA, November 2024.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics. URL <https://aclanthology.org/D13-1170/>.
- Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928.
- Xiao Wang, Weikang Zhou, Qi Zhang, Jie Zhou, SongYang Gao, Junzhe Wang, Menghan Zhang, Xiang Gao, Yun Wen Chen, and Tao Gui. Farewell to aimless large-scale pretraining: Influential subset selection for language model. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 555–568, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.35. URL <https://aclanthology.org/2023.findings-acl.35/>.

- Meng Xia and Ricardo Henao. Reliable active learning via influence functions. *Transactions on Machine Learning Research*, 2023.
- Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *ICML*, 2024. URL <https://openreview.net/forum?id=PG5fV50maR>.
- Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=4wZiAXD29TQ>.
- Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. First is better than last for language data influence. *Advances in Neural Information Processing Systems*, 35:32285–32298, 2022.
- Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362, 2023.
- Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. On prompt-driven safeguarding for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML’24*, pp. 61593–61613, Vienna, Austria, July 2024. JMLR.org.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.
- Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized BERT pre-training approach with post-training. In Sheng Li, Maosong Sun, Yang Liu, Hua Wu, Kang Liu, Wanxiang Che, Shizhu He, and Gaoqi Rao (eds.), *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pp. 1218–1227, Huhhot, China, August 2021. Chinese Information Processing Society of China. URL <https://aclanthology.org/2021.ccl-1.108/>.
- Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A ALGORITHM

Algorithm 1 INFTRACE Learning Algorithm

Input: Model $\mathcal{M}$ parameterized by θ , training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{val}}$, optimizer $\mathcal{A}$.

```

1: Stage 1: Collecting training dynamics  $\mathcal{C}$ 
2:  $\mathcal{C} \leftarrow \emptyset$ 
3: Split  $\mathcal{D}_{\text{train}}$  to mini batch:  $\{\mathcal{B}_1, \dots, \mathcal{B}_T\}$ 
4: Sample subsets from  $\mathcal{D}_{\text{val}}$ :  $\{\mathcal{V}_1, \dots, \mathcal{V}_T\}$ 
5: for  $1 \leq t \leq T$  do
6:    $P_{\text{val}}^{(t)} = \{p_{\theta_t}(z), \forall z \in \mathcal{V}_t\}$ 
7:    $\theta_{t+1} \leftarrow \mathcal{A}(\mathcal{M}, \theta_t, \mathcal{B}_t)$ ,  $P_{\text{train}} = \{p_{\theta_t}(z), z \in \mathcal{B}_t\}$ 
8:    $P_{\text{val}}^{(t+1)} = \{p_{\theta_{t+1}}(z), z \in \mathcal{V}_t\}$ 
9:    $\Delta P_{\text{val}}^{(t)} = \{p_{\theta_{t+1}}(z) - p_{\theta_t}(z), z \in \mathcal{V}_t\}$ 
10:   $\mathcal{C} \leftarrow \mathcal{C} \cup (\Delta P_{\text{val}}^{(t)}, P_{\text{train}}^{(t)})$ 
11: end for
12: Stage 2: Distilling influence from  $\mathcal{C}$ 
13: Initialize:  $\mathbf{E}^t = \{\mathbf{e}_i^t, \forall z_i \in \mathcal{D}_{\text{train}}\}$ ,  $\mathbf{E}^v = \{\mathbf{e}_j^t, \forall z_j \in \mathcal{D}_{\text{val}}\}$ 
14: for  $1 \leq t \leq T$  do
15:   for  $(z_p, z_n) \in \mathcal{V}_t \times \mathcal{V}_t$ ,  $\Delta p_{\theta_t}(z_p) > \Delta p_{\theta_t}(z_n)$  do
16:      $\mathbf{e}_{\text{anc}} = \sum_{e_i \in \mathcal{B}} w_i \mathbf{e}_i$ , where  $w_i \propto \frac{1}{p_{\theta_t}(z_i)}$ 
17:     Update  $\mathbf{E}^t, \mathbf{E}^v$  w.r.t:  $\text{Max}(0, \|\mathbf{e}_{\text{anc}} - \mathbf{e}_p\| - \|\mathbf{e}_{\text{anc}} - \mathbf{e}_n\| + \gamma)$ 
18:   end for
19: end for
20: return  $\mathbf{E}^t, \mathbf{E}^v$ 

```

B MORE DETAILS OF THE METHOD

Sampling strategy. For INFTRACE to effectively learn influence relationships among data points, it is essential that their training dynamics are adequately observed throughout training. Therefore, at each timestep t , we sample $\mathcal{V}_t$ according to its frequency sampled in previous steps:

$$w(z_i)^{(t)} = \frac{a_i^{(t)}}{\sum_{j=1}^{|\mathcal{D}_{\text{val}}|} a_j^{(t)}}, \quad (10)$$

$$a_i^{(t)} = \frac{1}{1 + \sum_{j=1}^{t-1} \mathbf{1}\{z_i \in \mathcal{B}_j\}}. \quad (11)$$

where $w(z)^{(t)}$ is the sampling weight for z at a certain step t , which is inversely proportional to how many times it has been sampled before.

Data Filtering. In Eq. 5, we collect positive and negative examples according to the gap between their prediction change Δp . In our experiments, we discard pairs with a small gap between Δp , as such data are more likely to reflect training noise rather than true influence. Including these pairs often results in poorer performance. Specifically, we rank the collected pairs by the gap in Δp and retain only the top 20% with the largest values.

C IMPLEMENTATION DETAILS

C.1 IMPLEMENTATION OF INFTRACE

To learn INFTRACE, we first fine-tune RoBERTa on SNLI and SST-2 for 2 epochs using AdamW (Loshchilov & Hutter, 2019) with the same hyperparameters, where the learning rate is $1e-5$ and batch size is 32. While our method is scalable to larger datasets, baseline methods are computationally

expensive. Therefore, we reduce the data size by randomly sampling 32,000 training examples from each dataset so that we can compare our method with baselines on the same data in an efficient way. Both datasets are licensed under a Creative Commons Attribution-ShareAlike 4.0 International License and are used for their intended purpose. Using $|\mathcal{V}| = 16$, we collect $16 \times 15 \times \frac{1}{2} \times 2000 = 240,000$ triplets from 2,000 training steps. All the experiments with RoBERTa are done on a single RTX 4090 with 24GB of Memory.

Then, we split the collected triplets into 80% for training, 10% for validation, and 10% for testing. We train INFTRACE on the training set using AdamW with a learning rate of $5e-5$ and a batch size of 32. The embedding dimension h is set to 1024, and the margin scaling factor β is set to 2 (Eq. 7). Training is conducted for up to 10 epochs. During training, we evaluate the model on the validation set every 500 steps using Pearson correlation. We apply early stopping based on validation performance. INFTRACE typically converges quickly, often reaching the best performance within 2 or 3 epochs.

C.2 IMPLEMENTATION OF BASELINES

During the training of RoBERTa, we stored the model every 500 steps, resulting in four checkpoints. For the static influence function GRADDOT, we obtain gradients from the final checkpoint and use the gradient dot product to compute influence. While for the dynamic influence function, we follow ? to aggregate the influence from all the checkpoints:

$$\mathcal{I}_\theta(z, z') = \sum_{i=1}^k \eta_i \nabla \ell(\theta_i; z) \cdot \nabla \ell(\theta_i, z'), \quad (12)$$

where η_i is the learning rate for the corresponding checkpoint. Since gradient vectors are typically high-dimensional, even for models like RoBERTa, we extract gradients only from the last five layers, including four encoder layers and the final classifier. The gradients are concatenated and then compressed to a 1024-dimensional vector using sparse random projection, matching the dimensionality of our learned influence embeddings. This dimensionality reduction strategy is widely adopted in prior work on influence functions to improve efficiency (Xia et al., 2024; Kwon et al., 2024; Lin et al., 2024).

C.3 LLAMA FINETUNING

In § 3.3, we fine-tune LLaMA-3-8B-Instruct using LoRA with the rank of $r = 16$, resulting in $\approx 0.52\%$ trainable parameters. We set the learning rate to $2e-4$, use a batch size of 4, and train it for 3 epochs.