Improving Influence-based Instruction Tuning Data Selection for Balanced Learning of Diverse Capabilities

Qirun Dai[1,2*], **Dylan Zhang**[1], **Jiaqi W. Ma**[1], **Hao Peng**[1]
[1]University of Illinois Urbana-Champaign [2]The University of Chicago
qirundai@uchicago.edu {shizhuo2,jiaqima,haopeng}@illinois.edu

Abstract

Selecting appropriate training data is crucial for instruction fine-tuning of large language models (LLMs), which aims to (1) elicit strong capabilities, and (2) achieve balanced performance across different tasks. Influence-based methods show promise in achieving (1), by estimating the contribution of each training example to the model's predictions, but often struggle with (2). Our systematic investigation reveals that this underperformance can be attributed to an inherent bias, where some tasks intrinsically have greater influence than others. As a result, data selection is often biased towards these tasks, not only hurting the model's performance on others but also, counterintuitively, harming performance on these high-influence tasks themselves. To address this, we propose BIDS, a ***B****alanced and* ***I****nfluential* ***D****ata* ***S****election* algorithm. BIDS first normalizes influence scores of the training data, and then iteratively chooses the training example with the highest influence on the most underrepresented task. Experiments with both Llama-3 and Mistral-v0.3 on seven benchmarks spanning five diverse capabilities show that BIDS consistently outperforms *both* state-of-the-art influence-based algorithms and other non-influence-based frameworks. Surprisingly, training on a 15% subset selected by BIDS can even outperform full-dataset training with a much more balanced performance. Our analysis highlights the importance of both instance-level normalization and iterative optimization of selected data for balanced learning of diverse capabilities.

1 Introduction

Instruction tuning plays a crucial role in eliciting strong capabilities from LLMs. Typically, a pre-trained LLM is finetuned on a mixture of different instruction datasets, to learn to follow human instructions and achieve strong and balanced performance on a wide range of capabilities (Ouyang et al., 2022; Touvron et al., 2023; Dubey et al., 2024; Jiang et al., 2023). The importance of data quality for instruction tuning (Zhou et al., 2024) has spawned many works on data selection (Cao et al., 2023; Chen et al., 2023; Liu et al., 2024a). Among them, influence-based methods, which estimate each training example's influence on the model's prediction on a downstream task (Koh and Liang, 2017; Pruthi et al., 2020), have shown promise. Thanks to recent advances, they have also been scaled to LLM-level computations and demonstrated strong potential in selecting high-quality instruction tuning data (Xia et al., 2024; Choe et al., 2024; Yu et al., 2024).

However, influence estimation methods are typically designed to measure the data influence for a single task (Koh and Liang, 2017; Pruthi et al., 2020). In this study, we demonstrate that existing influence-based data selection algorithms (Xia et al., 2024) struggle to balance capabilities across diverse tasks, which is crucial in real-world applications.[1] Specifically, our analysis reveals that the influence scores for some tasks can be larger than others' by several magnitudes. This can introduce systematic biases in the data selection process when cross-task influence scores are directly compared, as done in many existing works (Yin and Rush, 2025; Albalak et al., 2024). This leads to a couple of pitfalls. First, biasing towards some tasks hurts the model's performance on others, making it more challenging for LLMs to achieve balanced capabilities. Second, perhaps counterintuitively, it may even hurt the model's performance on the very task that the data is biased towards. These issues call for an influence-based selection algorithm designed for training LLMs to achieve balanced capabilities across diverse tasks.

BIDS, our proposed algorithm, addresses these

*Work done during internship at UIUC.

[1]E.g., it is desirable for a coding agent to faithfully follow user instructions and perform complex reasoning.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7079–7102
November 4-9, 2025

challenges with two key designs. Given a training dataset to select from and a validation dataset representing diverse target tasks, we describe the influence pattern between them with a matrix, where each column consists of the influence scores of different training examples on a specific validation instance. BIDS first applies column-wise normalization to this matrix, thus setting the influence for different validation instances on the same scale. Then, in contrast to prior methods that simply select instances with the highest influence values, BIDS applies an iterative selection algorithm. At each iteration, this algorithm compares the influence of each candidate with the average influence of those already selected, and selects the one that can provide the largest improvement for the current selected subset. If the current dataset falls short in influence on certain validation instances, then our algorithm will intuitively favor training examples that have high influence on the specific tasks represented by these validation data, and thus promotes balanced multi-task learning.

We evaluate BIDS on UltraInteract (Yuan et al., 2024), a strong instruction tuning dataset suite, with Llama-3-8B (Dubey et al., 2024) and Mistral-7B-v0.3 (Jiang et al., 2023). Across seven tasks spanning five capabilities including coding, math, logical inference, world knowledge and instruction following, BIDS consistently outperforms both influence- and non-influence-based baselines. Surprisingly, a 15% subset selected by BIDS even outperforms full-dataset training in average performance. Further analysis highlights the importance of the key design choices of BIDS, and reveals completion length as a potential factor for biased influence. The contributions of this work include: (1) Through systematic analysis, we identify inherent biases in current influence-based instruction tuning data selection algorithms. (2) We propose BIDS to address this issue and promote balanced learning of multiple capabilities. (3) Extensive experiments confirm the strong and consistent performance of BIDS, and provide valuable insights on what makes a balanced set of influential data.

2 Background and Preliminaries

Influence-based instruction tuning data selection. Estimating the influence of training examples on model predictions is critical for understanding model behavior. Traditional methods, including retraining-based (Ghorbani and Zou, 2019; Ilyas et al., 2022; Park et al., 2023) and gradient-based (Koh and Liang, 2017; Pruthi et al., 2020) ones, have proven effective but are computationally prohibitive when scaling to LLMs, as they require either retraining on a large number of subsets, or computing at least a forward and backward pass for each training example in order to obtain its gradient (Hammoudeh and Lowd, 2024; Ko et al., 2024). Several recent advances seek to address these challenges by extending gradient-based approaches to scale more efficiently. Given a large training dataset to select from and a validation set representing certain targeted capability, LESS (Xia et al., 2024) models the influence between each pair of training and validation examples through LoRA-based low-dimensional gradient similarity, and then selects training points with the highest influence on the validation set. LOGRA (Choe et al., 2024) leverages a low-rank gradient projection algorithm to further improve the efficiency of this process. MATES (Yu et al., 2024) formulates the pointwise data influence between each training point and the whole validation set, and uses a small model to learn this pointwise influence.

Upon closer inspection, these LLM-scale influence-based data selection methods share a similar problem formulation. They all need a validation set to represent a targeted data distribution and require pointwise data influence between pair of training and validation instances. We aim to extend this to more general-purpose instruction tuning, where the model simultaneously learns multiple diverse capabilities that may require training data from drastically different distributions. Concretely, since only LESS directly targets instruction tuning among the three LLM-scale approaches, we ground our study on the specific formulation of LESS. But we emphasize that due to the highly similar influence modeling patterns , the results of our work should also provide useful insights for other influence-based selection methods.

Problem Setup and Notations. Assume an instruction tuning dataset $\mathcal{D}$, a validation dataset $\mathcal{V}$, which spans m diverse tasks that we want to optimize the LLM's performance for: $\mathcal{V} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_m$, and an **influence estimation method** that can compute the influence of each training example on each validation instance. We first compute the influence score between each pair of training and validation data, yielding a $|\mathcal{D}| \times |\mathcal{V}|$ matrix $\boldsymbol{A}$. Each row of $\boldsymbol{A}$ corresponds to an individual

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Random	43.5	48.9	**64.8**	64.9	41.5	22.5	18.1	43.4
	LESS	**43.9**	**50.7**	62.7	**65.1**	**42.5**	**22.6**	**19.7**	**43.9**
10%	Random	**47.8**	50.6	**65.0**	**64.9**	43.9	24.0	17.8	**44.9**
	LESS	44.7	**51.3**	62.0	64.7	**44.6**	**24.3**	**19.3**	44.4
15%	Random	**48.7**	**51.9**	**65.2**	**65.1**	**45.6**	**25.0**	18.8	**45.7**
	LESS	46.5	51.0	63.2	64.6	44.9	24.9	**21.2**	45.2

Table 1: Compared with the random baseline, LESS shows weak macro-average and unbalanced task-specific performance, which suggests a potential bias in its selection process.

training example, and each column a validation instance. Element $\boldsymbol{A}_{ij}$ indicates the influence of i-th example from $\mathcal{D}$ on j-th instance from $\mathcal{V}$. We dub $\boldsymbol{A}$ an **Attribution Matrix (AM)** as it reveals the overall attribution pattern from the training set to all target tasks, and each row $\boldsymbol{A}_i$ the **Influence Distribution** of the i-th training example.

Our goal is to design a **data selection algorithm** that can effectively select a subset $\mathcal{T}$ from $\mathcal{D}$ with size under a pre-defined budget, based on the information presented in $\boldsymbol{A}$. Finetuning the LLM on $\mathcal{T}$ is supposed to achieve strong and balanced performance on all targeted tasks. The evaluation tasks are specifically chosen to have minimal overlap in terms of the capabilities they benchmark. The size of validation set for each task is also kept the same to avoid bias in the selection process.

3 Existing Influence-based Selection Fails at Balancing Diverse Tasks

We first show that LESS leads to suboptimal and unbalanced performance in general-purpose instruction tuning. This is quantitatively revealed by our analysis framework, which identifies inherent biases in the scale of influence values across different tasks. Insights drawn in this section pave the way for the design choices of BIDS in §4.

Setting. In this section, we use Llama-3-8B (Dubey et al., 2024) as the base model for both influence estimation and evaluation of selected datasets. For the instruction dataset to select from, we use UltraInteract (Yuan et al., 2024), a state-of-the-art large-scale dataset designed to enhance diverse reasoning capabilities, including mathematical reasoning, coding, and general logical inference. We also follow the evaluation setup of Yuan et al. (2024), with seven datasets spanning five diverse capabilities. We use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for coding, GSM-Plus (Li et al., 2024) and MATH (Hendrycks et al., 2021) for math, and BigBench-Hard (BBH) (Suzgun et al., 2022) for general logical inference. We also use MMLU (Hendrycks et al., 2020) to assess the model's ability to understand world knowledge, and IFEval (Zhou et al., 2023) for fine-grained instruction-following ability. Appendix A.2 contains more training and evaluation details.

We follow LESS for the **influence estimation method** throughout this work, with an equal number of validation instances sampled from each of the seven evaluation tasks to ensure fairness. In this section, for the **data selection algorithm**, we also start with the **task-wise max** algorithm (Appendix A.3) used by LESS. For each training example, it first computes the mean influence over validation instances within the same task, followed by selecting training examples with the highest maximum influence across different tasks. We compare this algorithm against the random selection baseline.

LESS fails to balance different capabilities (Table 1). LESS shows substantial imbalance and variability in task-specific performance across different budgets. Although it consistently outperforms the random baseline in IFEval by more than 1.5%, it underperforms in BBH by 2–3%, and shows no clear advantage in the remaining five tasks. Moreover, with the increase of budget level, LESS is gradually outperformed by the random baseline in more tasks, leading to weaker macro-average under both 10% and 15% budgets.

The underperformance of LESS may stem from the fact that it is not designed for learning multiple diverse capabilities. But given that the number of validation instances was kept equal for each task, it still suggests a potential inherent bias in the influence values, which could skew the selection towards certain capabilities. If the overall influence on certain tasks is inherently higher, then the naive

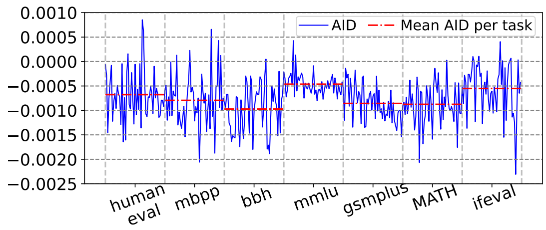


Figure 1: Unnormalized Average Influence Distribution (AID) of the **whole** UltraInteract dataset (Yuan et al., 2024), showing great disparities in scale for inter-task and intra-task influence.

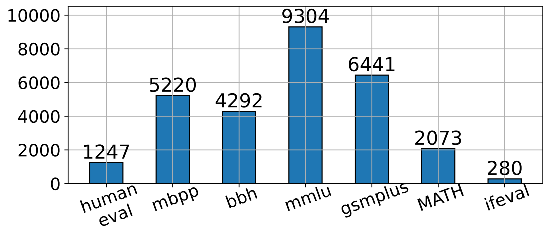


Figure 2: Task frequencies with Highest Influence (THI) under the **10%** budget. MMLU is obviously oversampled in LESS-selected data.

task-wise max algorithm will naturally prioritize them, possibly at the expense of others.

In what follows, we aim to answer the following two questions: (1) whether influence values differ across tasks and to what extent, and (2) whether tasks with higher influence values receive more performance improvements.

What causes the imbalance of LESS? We first define two data analysis metrics:

- **Average Influence Distribution (AID):** $\sum_{i=1}^{N} \boldsymbol{A}_i/N$, is the average of influence distributions of all the training examples. This vector captures the overall influence pattern for all validation data, and is helpful to answering (1).
- **Task Frequency with Highest Influence (THI)** for a task t is the number of selected examples that have the highest average influence on t. Distribution of THI targets (2) in identifying high-influence tasks for the selected subset.

Our UltraInteract AID analysis (Figure 1) reveals discrepancies at both the task and instance levels. Surprisingly, MMLU receives a substantially higher average influence than BBH, despite being more out-of-distribution for the training data. Moreover, there are also significant disparities within the same task. For example, for IFEval, the gap between the highest and lowest instance-wise influence is greater than 0.0025, $2.5\times$ larger than the globally highest instance-wise influence of 0.001. These results answer our question (1) by confirming that the scales of influence values differ significantly across various tasks.

Further, the THI analysis of LESS-selected data (Figure 2) confirms that these scale differences cause LESS to disproportionately favor certain tasks over others. Specifically, MMLU is most frequently identified as the most influential task, consistent with the observations in Figure 1 where MMLU has the highest task-level average influence. However, this does not translate into proportionally better performance—LESS frequently underperforms even the random baseline on MMLU. For other in-distribution tasks with high THI, such as MBPP, BBH, and GSM-Plus, LESS either consistently underperforms or shows no clear advantage. These observations suggest that while high-influence tasks tend to have more examples in the selected training data, they do not necessarily offer greater room for performance improvement. Besides, such biased sampling may hinder the learning of other capabilities. Thus, we answer question (2) by concluding that the inherent difference in the scale of cross-task influence values is indeed a harmful bias, and can severely undermine the performance of LESS-selected data.

4 BIDS: Selecting Influential Data for Balanced Capability Learning

This section introduces BIDS to address the issues identified in §3. BIDS has two key design choices.

Instance-level normalization. This technique addresses the scale differences in influence values across different validation instances, by applying a column-wise normalization to the Attribution Matrix. Specifically, for a validation instance $\boldsymbol{v}_j$, the influence of each training example $\boldsymbol{t}_i$ is normalized by $\boldsymbol{A}_{ij}^{\text{norm}} = (\boldsymbol{A}_{ij} - \mu_j)/\sigma_j$, where μ_j and σ_j are the sample mean and standard deviation of all values in column j of $\boldsymbol{A}$. This normalization ensures the influence scores across different columns are comparable in scale. In other words, if two influence scores of different columns have similar intra-column rankings, then they should also have similar values.

Iterative selection favoring underrepresented tasks. We further propose an iterative greedy selection algorithm (Figure 3 and Algorithm 1)

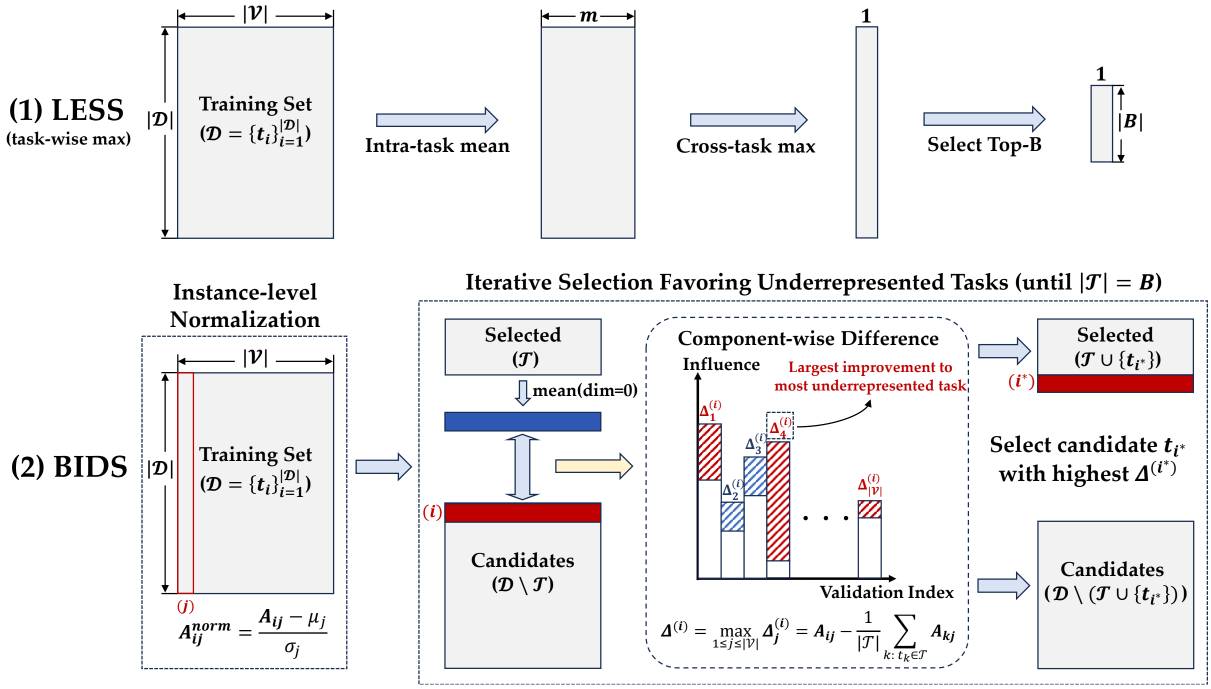


Figure 3: Comparison between BIDS and the task-wise max algorithm used by LESS. For convenience, we represent the training set $\mathcal{D}$ with its Attribution Matrix (AM), in which the i-th row is the $|\mathcal{V}|$-dimensional Influence Distribution of the i-th training example in $\mathcal{D}$. BIDS differs from LESS, which naively selects top-B examples with the highest task-wise max utility scores, in mainly two aspects. §4 contains a detailed walkthrough.

to promote the balance across different capabilities. It begins with an empty set. In each iteration, it first computes the average influence distribution of the current selected subset $\mathcal{T}$, denoted as $\boldsymbol{A}_{\mathcal{T}} \triangleq \frac{1}{|\mathcal{T}|} \sum_{k: \boldsymbol{t}_k \in \mathcal{T}} \boldsymbol{A}_k$. Then it iterates through each training example $\boldsymbol{t}_i$ in $\mathcal{D} \setminus \mathcal{T}$, and calculates a component-wise difference between $\boldsymbol{A}_i$ and $\boldsymbol{A}_{\mathcal{T}}$. The utility $\Delta^{(i)}$ of candidate $\boldsymbol{t}_i$ is then defined as the largest component of $\boldsymbol{A}_i - \boldsymbol{A}_{\mathcal{T}}$, and the candidate with the highest utility is selected.

Algorithm 1 BIDS: Iterative Selection

Input: $\mathcal{D}$: the set of all training examples; $\mathcal{V}$: the set of validation examples; B: the number of examples to be selected; $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{V}|}$: the Attribution Matrix between $\mathcal{D}$ and $\mathcal{V}$.
Initialization: $\mathcal{T} = \varnothing, \mathcal{D} = \{\boldsymbol{t}_i\}_{i=1}^{|\mathcal{D}|}$
while $|\mathcal{T}| < B$ **do**
    $\Delta^{(i)} = \max_{1 \le j \le |\mathcal{V}|} \{\boldsymbol{A}_{ij} - \frac{1}{|\mathcal{T}|} \sum_{k \in \{k | \boldsymbol{t}_k \in \mathcal{T}\}} \boldsymbol{A}_{kj}\}$
    $i^* = \arg\max_{i \in \{i | \boldsymbol{t}_i \in \mathcal{D} \setminus \mathcal{T}\}} \Delta^{(i)}$
    $\mathcal{T} = \mathcal{T} \cup \{\boldsymbol{t}_{i^*}\}$
end while
Return: $\mathcal{T}$: selected training examples.

In other words, BIDS prioritizes training examples that maximize influence improvement for the most underrepresented task in the selected data. It fundamentally differs from LESS, which scores each training example independently and selects the top-ranked ones.

5 Experiments

5.1 Experimental Setups

We follow the setup in §3, including the same LLMs, datasets, and influence estimation implementation (Table 2, 4). To further validate the generalizability of BIDS, we also experiment with base models from other model families (Table 3).

Baselines. In addition to (1) the **random baseline** and (2) the original influence-based **task-wise max** algorithm used by LESS (§3), we compare with three strong non-influence-based methods to further demonstrate the necessity of balanced data selection: (3) **Representation-based Data Selection (RDS; Zhang et al., 2018; Hanawa et al., 2020; Xia et al., 2024)** is a **targeted** selection method that also requires validation data representing the targeted capabilities. It selects training examples that are most similar to validation data based on the model's hidden representations. (4) **S2L (Yang et al., 2024)** and (5) **DEITA (Liu et al., 2024a)** are two state-of-the-art, **general** selection methods. They aim to select high-quality and diverse

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Random	43.5	48.9	64.8	64.9	41.5	22.5	18.1	43.4
	Task-max (LESS)	43.9	50.7	62.7	**65.1**	**42.5**	22.6	19.7	43.9
	RDS	**45.6**	**52.7**	62.2	65.0	34.5	17.2	15.5	41.8
	S2L	40.4	49.0	64.3	65.0	41.8	**23.5**	16.5	42.9
	DEITA	43.9	47.3	**65.0**	**65.1**	41.9	22.3	18.1	43.4
	BIDS (ours)	**45.6**	51.0	64.3	64.9	42.1	22.9	**21.4**	**44.6**
10%	Random	47.8	50.6	65.0	64.9	43.9	24.0	17.8	44.9
	Task-max (LESS)	44.7	51.3	62.0	64.7	44.6	24.3	19.3	44.4
	RDS	50.0	**54.7**	63.2	64.6	39.3	22.4	18.3	44.6
	S2L	46.5	50.7	64.5	**65.0**	43.3	22.7	18.9	44.5
	DEITA	**51.8**	49.9	**65.6**	**65.0**	43.1	24.5	17.5	45.3
	BIDS (ours)	48.2	50.4	65.1	64.9	**45.1**	**25.1**	**23.4**	**46.0**
15%	Random	48.7	51.9	65.2	**65.1**	45.6	25.0	18.8	45.7
	Task-max (LESS)	46.5	51.0	63.2	64.6	44.9	24.9	21.2	45.2
	RDS	50.0	**53.9**	63.7	64.5	41.1	23.5	18.1	45.0
	S2L	**51.8**	50.7	65.4	64.8	44.7	24.2	16.9	45.5
	DEITA	50.0	50.7	**65.8**	64.7	45.0	23.3	16.7	45.2
	BIDS (ours)	49.1	50.7	63.7	64.6	**45.8**	**26.2**	**22.6**	**46.1**
	BIDS (ours; epochs=4)	50.0	53.0	64.4	**64.7**	47.0	26.9	**23.4**	**47.1**
100%	Full (epochs=1)	**52.6**	53.6	**65.5**	64.1	47.2	27.9	17.5	46.9
	Full (epochs=4)	48.2	**54.4**	59.2	63.1	**51.5**	**32.3**	17.9	46.7

Table 2: Comparison between BIDS and various influence-and non-influence-based selection algorithms, with **Llama-3-8B** as the base model. The task-specific or macro-average performance is **bolded** if it ranks first under the same budget, and underlined if it ranks second. “BIDS (epochs=4)” is compared with 100% full training.

	5%	10%	15%	100%
Random	40.2	41.0	41.4	43.0/43.2(4)
Task-max (LESS)	41.0	41.4	41.8	
RDS	38.5	39.6	41.0	
S2L	39.5	41.0	40.7	
DEITA	40.5	40.2	40.8	
BIDS	**41.0**	**42.5**	**42.6/43.3(4)**	

Table 3: Macro-average with **Mistral-7B-v0.3** as the base model. Subscript (4) denotes the result of training for 4 epochs. BIDS shows generalizability to different models. See Appendix A.6 for task-specific results.

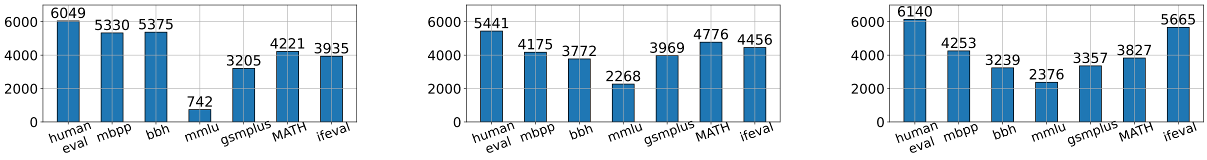
data without knowing the target tasks beforehand. Additionally, we compare to three other influence-based intuitive variants (6) **instance-wise max**, (7) **sum**, and (8) **temperature-based sampling** that are applicable to the Attribution Matrix in Table 4. Appendix A.3 includes more baseline details.

5.2 Results

Comparison under the same budget. As shown in Table 2 and 3, across different budget levels and models, BIDS consistently outperforms all baselines in terms of the macro-average across seven benchmarks. On specific tasks, BIDS is frequently among the strongest, ranking either first or second on 4/7 benchmarks on average. These results demonstrate that BIDS indeed achieves strong and balanced performance across tasks. Finally, results in Table 4 show the uniformly poor performance of influence-based variants other than BIDS, suggesting the necessity of going beyond simple Top-K selection, and taking normalization and cross-sample interactions into consideration.

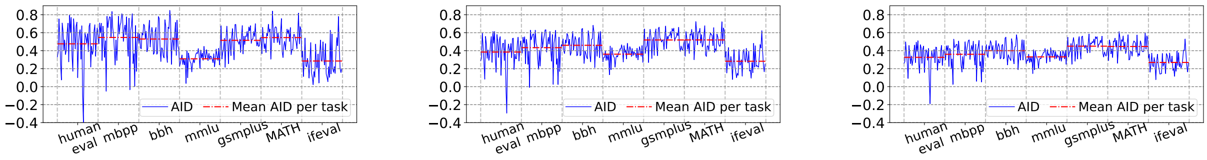
Notably, all non-influence-based methods (RDS, S2L, DEITA) consistently underperform the random baseline across all budgets. Both RDS and DEITA are significantly biased towards the coding tasks under the 10% and 15% budgets, at the cost of serious performance drop on others, especially math and instruction-following. In contrast, S2L shows a shifting trend in its performance bias, from MATH under the 5% to HumanEval under the 15%. These observations confirm the value of BIDS, as existing works on both targeted and general instruction data selection all fail to generalize to the challenging setup of learning multiple diverse capabilities simultaneously.

BIDS outperforms full-dataset training. As shown in the last three rows of Table 2 and the last two columns of Table 3, finetuning on a 15% subset selected by BIDS over four epochs consistently outperforms training on the full dataset. Task-specific results further reveal that BIDS



(a) $-$(Norm + Iter) (b) $-$Iter (c) BIDS

Figure 4: THI under the 10% budget with **Llama-3-8B**. Both $-$**Iter** and BIDS have more balanced task frequencies.



(a) $-$(Norm + Iter) (b) $-$Iter (c) BIDS

Figure 5: Normalized AID under the 10% budget with **Llama-3-8B**. Disparity in AID gradually diminishes, with both decreasing maximums and increasing minimums. Appendix A.6 shows similar trends with Mistral-7B-v0.3.

	Llama-3-8B			**Mistral-7B-v0.3**		
	5%	10%	15%	5%	10%	15%
Task-max	43.9	44.4	45.2	41.0	41.4	41.8
Instance-max	44.2	45.0	45.0	39.7	41.1	42.0
Sum	44.2	44.6	44.9	39.9	40.2	41.2
Temperature	43.6	44.6	45.0	39.6	40.3	41.4
BIDS	**44.6**	**46.0**	**46.1**	**41.0**	**42.5**	**42.6**

Table 4: Macro-average of influence-based algorithms with **Llama-3-8B** and **Mistral-7B-v0.3**. See Appendix A.5 and A.6 for task-specific results.

	Llama-3-8B			**Mistral-7B-v0.3**		
	5%	10%	15%	5%	10%	15%
BIDS	**44.6**	**46.0**	**46.1**	**41.0**	**42.5**	**42.6**
$-$Iter	44.3	45.6	45.7	41.0	41.6	42.2
$-$(Norm + Iter)	44.2	45.0	45.0	39.7	41.1	42.0

Table 5: Respective contributions of the two components of BIDS. Results with both models uniformly show a decreasing trend as more technical components are ablated, substantiating their positive contributions.

achieves better performance by maintaining balanced and strong performance across six reasoning-related tasks while significantly improving general instruction-following. These results demonstrate that BIDS not only excels in selecting influential and balanced data, but also that full-dataset training may not always be optimal for LLMs to learn multiple diverse capabilities, highlighting the efficiency potential of selective instruction finetuning.

6 Ablation

This section presents ablation studies of the two key components of BIDS, and analyzes their effects on the composition of selected data.

Ablation. We compare BIDS with the $-$**Iter** baseline to ablate iterative selection, and with $-$(**Norm** + **Iter**) to further ablate normalization (i.e., the **instance-wise max** algorithm in Appendix A.3). Table 5 shows that both the iterative selection and normalization consistently improve the overall performance across all budgets.

Normalization balances THI. We then proceed to explore how the two components affect the influence distribution of selected data, and whether such effects can provide insights into how BIDS advances balanced learning of diverse capabilities. We replace task-wise average influence with instance-wise influence in THI calculation, since the three algorithms we are comparing are all built upon the instance-wise max approach. Comparing 4a with 4b, after normalization the task frequency distribution becomes more balanced. Frequencies for MMLU, GSM-Plus, MATH and IFEval all increase by a great extent, while those for BBH and coding tasks decrease. 4b to 4c shows a few mild adjustments made by the iterative selection. This is intriguing when compared to the task-specific results in Table 2, where BIDS actually shows superior performance in tasks with both increased (HumanEval, IFEval) and decreased (GSM-Plus, MATH) THI frequencies. This suggests that a balanced selection may improve data efficiency not only by allocating more budget for underrepresented capabilities, but also reducing the

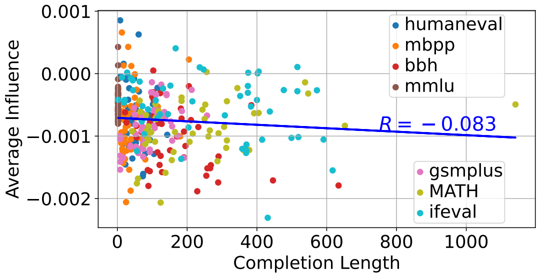


Figure 6: Correlation between **instance-level influence** and completion length is especially intricate, with a global correlation of only -0.083. There is no clear negative correlation **inside** any task.

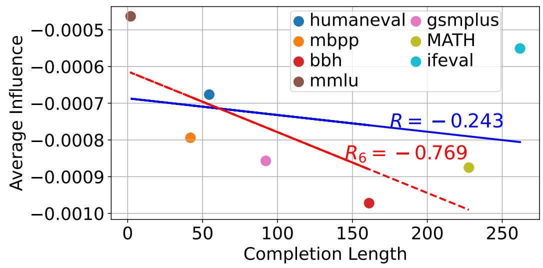


Figure 7: **Task-level influence** has a strong negative correlation ($R = -0.769$) with completion length for 6/7 tasks. However, IFEval contradicts this correlation pattern, suggesting other influencing factors.

redundancy in over-represented ones.

Better performance comes with smaller influence discrepancies. We report influence values after instance-level normalization for better comparisons. From 5a to 5b to 5c, we observe a progressive reduction in the disparity of average influence across tasks, which leads to the following two interesting observations: (1) **The maximums of AID decrease.** Despite generally lower influence scores across the seven tasks, the performance of BIDS improves consistently. This reveals a limitation of the first-order linearity assumption by the influence estimation method: simply selecting high-influence points using a greedy algorithm increases the average influence distribution on almost all tasks, but their effectiveness doesn't linearly add up, thus not necessarily improving task-level or overall performance. (2) **The minimums of AID increase, especially for validation instances with exceptionally low influence**, such as HumanEval and MBPP. This again suggests the effectiveness of BIDS' key motivation: improving the model's overall performance by enhancing the capabilities that are most underrepresented.

7 Completion Length as a Task-specific Factor for Spurious Influence

This analysis aims to explain the counterintuitive phenomenon identified in §3, where the performance of high-influence tasks actually degrades (e.g., MMLU). We focus on the **completion length of target tasks**, which has been shown important for gradient-based influence and data selection (Wang et al., 2024; Liu et al., 2024a; Xia et al., 2024). Ideally, if influence has a salient correlation with a task-specific factor that is **not strongly correlated with downstream performance**, then this can be explained, as such spurious influence no longer translates into better performance.

Figures 6 and 7 show instance- and task-level correlations between influence and completion lengths of validation data. **On the task-level, there is a strong negative correlation** ($R = -0.769$) **for 6/7 tasks.** The shortest completion length of MMLU indeed translates into the highest average influence. Only IFEval, the only task that does not have a single objective answer, contradicts this negative correlation, possibly suggesting other influencing factors. However, **the instance-level correlation pattern becomes more intricate**, with a global correlation of only -0.083. Even for tasks that show strong negative correlations in Figure 7 such as MMLU, HumanEval, MBPP and GSM-Plus, there is no similar trend for instances inside the same task, not to mention IFEval, which even shows a slightly positive correlation.

In summary, completion lengths can partly explain the counterintuitive phenomenon, and should be considered for improving the fidelity of influence. But it is still far from the whole picture, suggesting the potential impact of other task-specific factors such as sample style or complexity (Liu et al., 2024a), which are harder to quantify and deserve more focused investigation in future work.

8 Related Work

Data Selection for Instruction Finetuning. Since Zhou et al. (2024) showed the importance of high quality instruction data curation, many works have been exploring automatic data selection methods. Quality-guided selection defines the quality for each data point using heuristic indicators based on model embeddings or losses (Yang et al., 2023; Ankner et al., 2025; Zhang et al., 2025b), ratings from strong evaluators (Chen et al., 2023; Parkar et al., 2024; Pang et al., 2025), or principled metrics based on training dynamics (Kang et al., 2024; Mekala et al., 2024; Xia et al., 2024; Lin et al., 2025). Diversity-guided methods usually cluster the training data based on informative representations (Yang et al., 2024; Zhang et al., 2025a), and take inspiration from traditional core-set selection approaches (Das and Khetan, 2023). Both have been proved effective for instruction finetuning of LLMs (Bukharin and Zhao, 2023; Liu et al., 2024a; Qin et al., 2025), and BIDS takes both quality and diversity into account by applying an iterative selection algorithm to influence distributions. Additionally, there is also a line of work that surveys how existing data selection techniques perform in compute-constrained scenarios (Yin and Rush, 2025), under different experimental configurations (Diddee and Ippolito, 2025), and for large-scale datasets (Ivison et al., 2025).

Influence Estimation. Influence estimation is generally classified into gradient-based and retraining-based approaches (Hammoudeh and Lowd, 2024; Ko et al., 2024). Gradient-based influence estimation focuses on the gradient trace of each training point, and assesses the gradient alignment between training and validation examples (Koh and Liang, 2017; Pruthi et al., 2020; Pan et al., 2025). Retraining-based estimation trains a large number of models on different subsets, and then inspects how their performance changes when a training example is added to these subsets (Ghorbani and Zou, 2019; Ilyas et al., 2022; Park et al., 2023; Hu et al., 2025). Recently both lines of works have been extended to LLM-scale applications, covering various aspects including pretraining (Engstrom et al., 2024; Yu et al., 2024; Choe et al., 2024; Chang et al., 2025; Yu et al., 2025; Wang et al., 2025) and instruction tuning (Xia et al., 2024; Liu et al., 2024b; Pan et al., 2024). There is also a popular line of work aimed at designing efficient proxies for expensive gradient-based or retraining-based estimation by taking inspiration from in-context probing (Jiao et al., 2025) and uncertainty estimation (Pan et al., 2025).

Gradient-induced Spurious Influence. Although to the best of our knowledge, our work is the first to systematically identify and address the effect of spurious influence on LLM training data selection, a few past works have also offered theoretical or empirical discussions on gradient-induced spurious influence in different machine learning applications. Xia et al. (2024) attribute spurious influence to the representation of example-wise gradients, which is most often made tractable by taking an average over the gradients of all tokens inside each example. This likely leads to the undesirable consequence that shorter training examples (i.e., with fewer tokens) have inherently higher influence values when the influence is computed by gradient dot product, because the norm of averaged token-wise gradients tends to rapidly decay with the length of example. In this work, we follow Xia et al. (2024) by adopting cosine similarity-based (i.e., dot product with length normalization) influence formulation, which should ideally remove the length bias. However, our empirical results in §7 suggest that the correlation between influence values and example length still remains for normalized gradients. One possible reason is that taking the average over token-wise gradients may incur systematic bias in the direction of the resulting gradient as well.

Apart from the application of LLMs in natural language generation tasks, past works have also identified spurious influence for BERT (Devlin et al., 2019) in traditional NLP tasks, and for diffusion models in image generation. Yeh et al. (2022) identify the bias of gradient norms across different layers of a BERT model, and attribute this to a "cancellation effect" in certain weight parameters. Xie et al. (2024) discover the bias of gradient norms over different timesteps in the generation process of diffusion models, where the loss function component from later timesteps is more likely to have a larger gradient norm. In light of these, we suggest in-depth future efforts into the possible factors that may systematically affect both the norm and direction of example gradients.

9 Conclusion

We introduce BIDS, an influence-based instruction tuning data selection algorithm specifically designed for learning multiple capabilities. Motivated by the observation of an inherent bias in influence across different tasks, BIDS first applies column-wise normalization to the Attribution Matrix, and then an iterative selection algorithm favoring underrepresented tasks. BIDS consistently outperforms various baselines as well as full-dataset training with much more balanced performance. Our analysis provides insights into the properties of an influential dataset with balanced capabilities.

Limitations

We discuss the limitations of our work below.

1. **Limitation to influence-based approaches.** While our work focuses on the imbalance issue of influence-based data selection methods, results of non-influence-based methods (i.e., RDS, D2L, DEITA) in Table 2 also show significant bias towards the two coding tasks, at the cost of severely degraded performance on others. These observations suggest the possibility that the imbalance of utility scores (Yin and Rush, 2025) may generally exist for both influence- and non-influence-based data selection approaches. In Appendix A.7, we conduct a preliminary investigation by applying the technical components of BIDS to one of the non-influence-based methods, RDS, and find that the balancing techniques of BIDS work well for RDS, which provides a preliminary verification for our hypothesis above, and also shows the broad applicability of the underlying principles of BIDS. However, the focus of this paper on influence-based approaches limits in-depth investigations into the more general imbalance of utility scores for data selection under a multi-capability learning setup. We hope more analyses and technical solutions can be discussed in future work.

2. **More factors behind the influence bias.** While our work centers on identifying the problem of imbalanced influence and proposing BIDS as a remedy, probing the factors behind the influence bias is also beneficial for a thorough understanding of the counterintuitive phenomenon identified in §3. In §7 we provide a preliminary analysis on the completion length of validation data as a potential factor. Though it can partly account for the phenomenon above, it is still far from a complete explanation, which requires investigation into other task-specific factors that are much harder to quantify. We hope future work can shed light on this.

Acknowledgments

We thank Lifan Yuan and Qingzhi Chen for their insightful discussions on this project. We also sincerely appreciate the anonymous ARR reviewers and chairs for their helpful feedback, which greatly improved this work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. 2025. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *The Thirteenth International Conference on Learning Representations*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Alexander Bukharin and Tuo Zhao. 2023. Data diversity matters for robust instruction tuning. *arXiv preprint arXiv:2311.14736*.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290*.

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

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. 2024. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Devleena Das and Vivek Khetan. 2023. Deft: Data efficient fine-tuning for large language models via unsupervised core-set selection. *arXiv preprint arXiv:2310.16776*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Harshita Diddee and Daphne Ippolito. 2025. Chasing random: Instruction selection strategies fail to generalize. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1943–1957, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Logan Engstrom, Axel Feldmann, and Aleksander Madry. 2024. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

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

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

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. 2020. Evaluation of similarity-based explanations. *arXiv preprint arXiv:2006.04528*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Pingbang Hu, Joseph Melkonian, Weijing Tang, Han Zhao, and Jiaqi W Ma. 2025. Grass: Scalable influence function with sparse gradient compression. *arXiv preprint arXiv:2505.18976*.

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

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. 2025. Large-scale data selection for instruction tuning. *arXiv preprint arXiv:2503.01807*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Cathy Jiao, Weizhen Gao, Aditi Raghunathan, and Chenyan Xiong. 2025. On the feasibility of in-context probing for data attribution. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5140–5155, Albuquerque, New Mexico. Association for Computational Linguistics.

William B. Johnson and Joram Lindenstrauss. 1984. Extensions of lipschitz mappings into hilbert space. *Contemporary mathematics*, 26:189–206.

Feiyang Kang, Hoang Anh Just, Yifan Sun, Himanshu Jahagirdar, Yuanzhi Zhang, Rongxing Du, Anit Kumar Sahu, and Ruoxi Jia. 2024. Get more for less: Principled data selection for warming up fine-tuning in LLMs. In *The Twelfth International Conference on Learning Representations*.

Myeongseob Ko, Feiyang Kang, Weiyan Shi, Ming Jin, Zhou Yu, and Ruoxi Jia. 2024. The mirrored influence hypothesis: Efficient data influence estimation by harnessing forward passes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26286–26295.

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

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.

Xiaotian Lin, Yanlin Qi, Yizhang Zhu, Themis Palpanas, Chengliang Chai, Nan Tang, and Yuyu Luo. 2025. Lead: Iterative data selection for efficient llm instruction tuning. *arXiv preprint arXiv:2505.07437*.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024a. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Zikang Liu, Kun Zhou, Wayne Xin Zhao, Dawei Gao, Yaliang Li, and Ji-Rong Wen. 2024b. Less is more: Data value estimation for visual instruction tuning. *arXiv preprint arXiv:2403.09559*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller language models are capable of selecting instruction-tuning training data for larger language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10456–10470, Bangkok, Thailand. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Xingyuan Pan, Luyang Huang, Liyan Kang, Zhicheng Liu, Yu Lu, and Shanbo Cheng. 2024. G-DIG: Towards gradient-based DIverse and hiGh-quality instruction data selection for machine translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15395–15406, Bangkok, Thailand. Association for Computational Linguistics.

Xingyuan Pan, Chenlu Ye, Joseph Melkonian, Jiaqi W Ma, and Tong Zhang. 2025. Daunce: Data attribution through uncertainty estimation. *arXiv preprint arXiv:2505.23223*.

Jinlong Pang, Jiaheng Wei, Ankit Shah, Zhaowei Zhu, Yaxuan Wang, Chen Qian, Yang Liu, Yujia Bao, and Wei Wei. 2025. Improving data efficiency via curating LLM-driven rating systems. In *The Thirteenth International Conference on Learning Representations*.

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

Ritik Sachin Parkar, Jaehyung Kim, Jong Inn Park, and Dongyeop Kang. 2024. Selectllm: Can llms select important instructions to annotate? *arXiv preprint arXiv:2401.16553*.

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

Yulei Qin, Yuncheng Yang, Pengcheng Guo, Gang Li, Hang Shao, Yuchen Shi, Zihan Xu, Yun Gu, Ke Li, and Xing Sun. 2025. Unleashing the power of data tsunami: A comprehensive survey on data assessment and selection for instruction tuning of language models. *Transactions on Machine Learning Research*. Survey Certification.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jiachen T. Wang, Dawn Song, James Zou, Prateek Mittal, and Ruoxi Jia. 2025. Capturing the temporal dependence of training data influence. In *The Thirteenth International Conference on Learning Representations*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Zhengbo Wang, Jian Liang, Lijun Sheng, Ran He, Zilei Wang, and Tieniu Tan. 2024. A hard-to-beat baseline for training-free CLIP-based adaptation. In *The*

Twelfth International Conference on Learning Representations.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. In *International conference on machine learning*. PMLR.

Tong Xie, Haoyu Li, Andrew Bai, and Cho-Jui Hsieh. 2024. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *Transactions on Machine Learning Research*.

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

Yu Yang, Aaditya K Singh, Mostafa Elhoushi, Anas Mahmoud, Kushal Tirumala, Fabian Gloeckle, Baptiste Rozière, Carole-Jean Wu, Ari S Morcos, and Newsha Ardalani. 2023. Decoding data quality via synthetic corruptions: Embedding-guided pruning of code data. *arXiv preprint arXiv:2312.02418*.

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Kumar Ravikumar. 2022. First is better than last for language data influence. In *Advances in Neural Information Processing Systems*.

Junjie Yin and Alexander M Rush. 2025. Compute-constrained data selection. In *The Thirteenth International Conference on Learning Representations*.

Zichun Yu, Spandan Das, and Chenyan Xiong. 2024. Mates: Model-aware data selection for efficient pretraining with data influence models. *arXiv preprint arXiv:2406.06046*.

Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Wentau Yih, and Chenyan Xiong. 2025. Data-efficient pretraining with group-level data influence modeling. *arXiv preprint arXiv:2502.14709*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, et al. 2024. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Qiu Jiantao, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. 2025a. Harnessing diversity for important data selection in pretraining large language models. In *The Thirteenth International Conference on Learning Representations*.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025b. The best instruction-tuning data are those that fit. *arXiv preprint arXiv:2502.04194*.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

Shimao Zhang, Yu Bao, and Shujian Huang. 2024. Edt: Improving large language models' generation by entropy-based dynamic temperature sampling. *arXiv preprint arXiv:2403.14541*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Appendix

A.1 Influence Estimation Pipeline of LESS

We briefly introduce the influence estimation pipeline of LESS in this section. For more detailed motivation and step-by-step mathematical deduction, we suggest referring to Xia et al. (2024).

Assume a model $\mathcal{M}_s$ which scores and selects data, and another model $\mathcal{M}_t$ which is trained on the selected data. For a training dataset $\mathcal{D}$ and validation dataset $\mathcal{V}$, LESS formulates the pairwise influence between each training example $\boldsymbol{t}_i \in \mathcal{D}$ and validation instance $\boldsymbol{v}_j \in \mathcal{V}$ with the following three steps.

Step 1: Warmup training with LoRA. LESS first trains $\mathcal{M}_s$ on a random subset $\mathcal{D}_{\text{warmup}} \subset \mathcal{D}$ for N epochs using the parameter-efficient finetuning method LoRA (Hu et al., 2021), and checkpoints the model after each epoch to store LoRA parameters $\{\boldsymbol{\theta}_t\}_{t=1}^{N}$.

Step 2: Gradient computation and projection. For each checkpoint $\boldsymbol{\theta}_t$ of LoRA-trained $\mathcal{M}_s$, LESS computes the SGD gradient of validation instance $\boldsymbol{v}_j$, and further uses random projection (Johnson and Lindenstrauss, 1984; Park et al., 2023) to project the gradient to a tractable lower dimension. The resulting projected gradient is denoted as $\nabla\ell(\boldsymbol{v}_j;\boldsymbol{\theta}_t)$. LESS also computes and projects the gradient of training example $\boldsymbol{t}_i$, but uses the Adam gradient defined as follows:

$$\Gamma(\boldsymbol{t}_i, \boldsymbol{\theta}_t) \triangleq \frac{\boldsymbol{m}^{t+1}}{\sqrt{\boldsymbol{v}^{t+1}} + \epsilon}$$

where $\boldsymbol{m}^{t+1}$ and $\boldsymbol{v}^{t+1}$ are the first and second moments in the parameter update rule for Adam optimizer.

Step 3: Gradient matching and influence calculation. Finally, LESS employs the following cosine-similarity-based approach to calculate the alignment between the gradient of each training and validation example, accumulated over all the warmup training epochs:

$$\text{Inf}_{\text{Adam}}(\boldsymbol{t}_i, \boldsymbol{v}_j) \triangleq \sum_{t=1}^{N} \bar{\eta}_t \cos(\nabla\ell(\boldsymbol{v}_j;\boldsymbol{\theta}_t), \Gamma(\boldsymbol{t}_i, \boldsymbol{\theta}_t))$$

where $\bar{\eta}_t$ is the average learning rate in the t-th epoch.

A.2 Details of Training and Evaluation Setups

Based on the LESS pipeline described above, we further introduce the implementation details of the training and evaluation setups in this work. All the experiments are carried out on 2 H100 GPUs with 80 GB memories.

Training Details. We basically follow the same set of hyperparameters as LESS when training both $\mathcal{M}_s$ and $\mathcal{M}_t$. Specifically, a batch size of 128 is used throughout all the training processes in this work, along with a learning rate scheduler with linear warm-up, cosine decay, and a peak learning rate of 2×10^{-5}. For the influence estimation pipeline, we consistently conduct the warmup training of $\mathcal{M}_s$ using four epochs and the full training set. For gradient computation and projection, we uniformly sample 50 validation instances from either the validation or the test split (when there is not a separate validation split) of each of the seven evaluation tasks, leading to a total of 350 validation instances. The projection dimension is set as 8192 for all the training and validation examples. For training $\mathcal{M}_t$ on the selected data, we consistently train for two epochs if not otherwise specified. Notably, we report the average performance of two sets of randomly selected data for all the random baselines in this paper.

Both the warmup training for influence estimation and the training on selected data are carried out with LoRA (Hu et al., 2021). The configurations of LoRA adapters are kept the same throughout the experiments, with a rank of 128, an α value of 512, a dropout rate of 0.1, and LoRA matrices being applied to all the attention modules.

Evaluation Details. We follow the evaluation convention of UltraInteract (Yuan et al., 2024) by using greedy decoding (i.e., temperature $= 0$) for all the evaluation tasks except for IFEval, where we use temperature $= 0.7$ and take the median result of three random seeds due to the high variability of this task.

A.3 Details of Data Selection Baselines

In this section, we first specify the mathematical definitions of the four **influence-based** selection algorithms used in this work. They share the same framework of first assigning an overall utility score s_i to each training example t_i, and then greedily selecting examples with the highest scores, or randomly sampling from the score distribution.

- **Task-wise Max:** $s_i \triangleq \max_{k=1,\ldots,m}\{\sum_{\boldsymbol{v}_j\in\mathcal{V}_k} \boldsymbol{A}_{ij}\}$. It first computes the **average** influence over validation instances within the same task, and then takes the **maximum** influence across different tasks as the utility score.
- **Instance-wise Max:** $s_i \triangleq \max_{j=1,\ldots,|\mathcal{V}|}\{\boldsymbol{A}_{ij}\}$. It directly uses the **maximum** of influence values over all validation instances as the utility score. Notably, instance-wise max is equivalent to $-($**Norm** $+$ **Iter**$)$ in §6.
- **Sum:** $s_i \triangleq \sum_{j=1}^{|\mathcal{V}|} \boldsymbol{A}_{ij}$. It directly uses the **sum** of influence values over all validation instances as the utility score.
- **Temperature-based Sampling:** s_i follows the same definition as **Task-wise Max**. However, instead of greedy selection, this algorithm transforms these utility scores into a probability distribution using temperature-scaled softmax (Ficler and Goldberg, 2017; Zhang et al., 2024), and samples training data from this distribution without replacement. Compared with the deterministic greedy selection, this sampling-based approach increases the proportion of selected data that show preference for low-influence tasks, thus balancing the task distribution.

We then summarize the mechanisms and provide the implementation details of the remaining three **non-influence-based** selection methods.

- **RDS (Zhang et al., 2018; Hanawa et al., 2020; Xia et al., 2024).** We follow the RDS implementation in Xia et al. (2024), which computes the cosine similarity between training and validation examples based on the final layer representation of the last token in each example sequence. Training examples with the highest similarity to any one of the validation examples are selected. In order to ensure fair comparison, we use the same model that computes gradient features in BIDS to extract the final layer representations for RDS.
- **S2L (Yang et al., 2024)** first collects the training loss trajectory for each training example using a small reference model that belongs to the same family as the final model, and then applies K-means clustering to these trajectories. Finally, an equal number of examples are sampled from each cluster to train the final model, ensuring the balance among clusters of different sizes. Specifically, we use Llama-3.2-1B[2]as the reference model, which belongs to the Llama-3 family and is much larger than the Pythia-70M (Biderman et al., 2023) used in Yang et al. (2024), thus improving the fidelity of S2L's performance. The remaining hyperparameters follow the configurations in Yang et al. (2024). We note that although S2L performs selection with a small model, its total computational cost is similar to LESS, as the loss for each example needs to be computed more frequently than the gradient along the training trajectory.
- **DEITA (Liu et al., 2024a)** first uses ChatGPT (Achiam et al., 2023) to curate training data for training a complexity scorer and a quality scorer. Then with the trained scorers, it assigns a complexity score and a quality score to each training example and sorts the training dataset by their product. Finally, it applies an iterative selection algorithm to the sorted dataset based on the embedding of each training example, with the aim of promoting diversity. Specifically, we use the 7B scorer models[3][4]publicly released by the DEITA team, and obtain example-wise embeddings using the final-layer hidden states of a pretrained Llama-3-8B to align with the final evaluation.

A.4 Effect of Normal Standardization on the Attribution Matrix

In §4 we introduce the instance-level normalization technique of BIDS. One potential issue with this normal standardization approach is that it may not work sufficiently well when the distribution of unnormalized influence scores differs much from an approximate normal distribution. In this section we aim to justify the application of normal standardization to the Attribution Matrix (AM). Specifically, we randomly select five validation instances (i.e., five columns in the AM) from each task, and compare their empirical distributions after normalization with a standard normal distribution. The results show that almost all of the sampled columns approximate a standard normal distribution after the instance-level normalization, which justifies the use of normal standardization as the normalization method in BIDS.

[2]https://huggingface.co/meta-llama/Llama-3.2-1B

[3]https://huggingface.co/hkust-nlp/deita-complexity-scorer

[4]https://huggingface.co/hkust-nlp/deita-quality-scorer

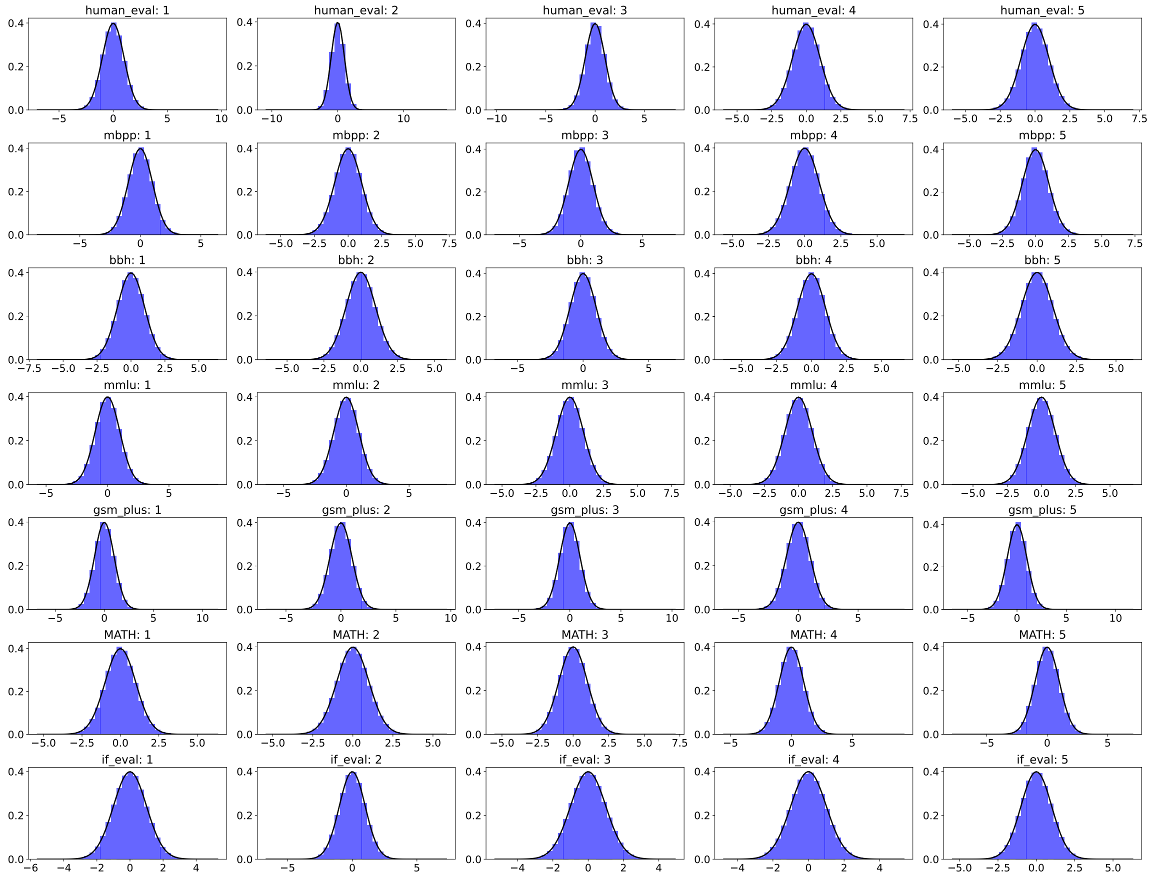


Figure 8: The effect of normal standardization. Five AM columns are sampled for each task. Most of the columns in the AM indeed approximate a standard normal distribution after normal standardization.

A.5 Complete Results with Llama-3-8B

In Table 6, we present the complete task-specific results for Table 4 and 5 in the main text. Across all three budgets, apart from the consistently strongest macro-average, BIDS also ranks either first or second on 4/7, 5/7, 4/7 benchmarks respectively, showing its stronger and more balanced performance than both simple influence-based variants and ablated baselines. This further highlights the necessity of going beyond the naive greedy Top-K selection or temperature-based random sampling, and taking both normalization and interactions among training examples into consideration, in order to tackle spurious or biased cross-task influence.

A.6 Complete Results with Mistral-7B-v0.3

To further validate the generalizability of BIDS, we compare with other baseline data selection algorithms using Mistral-7B-v0.3 as the backbone for both selection and training. The complete task-specific results are presented in Table 7 and 8. The two algorithms compared here, $-($**Norm** $+$ **Iter**$)$ and $-$**Iter**, follow the same definition in §6. And the random baseline is also the average result of two different random seeds.

Similar to the analysis framework in §3, we also present the AID analysis of the whole UltraInteract dataset (Figure 9) and the THI analysis of LESS-selected data (Figure 10). Then we follow the workflow in §6 to present both THI and AID anlyses for the three progressive algorithms: $-($**Norm** $+$ **Iter**$)$, $-$**Iter** and BIDS (Figure 11, 12). The only difference here is that the selection model is Mistral-7B-v0.3 instead of Llama-3-8B.

A.7 Applying BIDS to Other Data Selection Methods than LESS

Theoretically, BIDS is applicable to any data selection method that requires a validation set and computes a score between each pair of training and validation examples (i.e., produces a matrix A where element A_{ij} indicates the utility of i-th training example on j-th validation instance). We empirically apply the two techniques of BIDS, instance-level normalization and iterative selection favoring underrepresented tasks, to RDS, which computes

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Task-max (LESS)	43.9	50.7	62.7	**65.1**	42.5	22.6	19.7	43.9
	Sum	**45.6**	51.9	63.6	64.8	42.4	21.3	20.1	44.2
	Temperature	44.7	47.6	64.2	65.0	42.4	**23.7**	17.7	43.6
	BIDS (ours)	**45.6**	51.0	**64.3**	64.9	42.1	22.9	**21.4**	**44.6**
	−`Iter`	**45.6**	**52.1**	62.5	64.8	42.5	22.5	20.1	44.3
	−(`Norm` + `Iter`)	43.9	**52.1**	63.2	65.0	**42.6**	22.3	20.6	44.2
10%	Task-max (LESS)	44.7	51.3	62.0	64.7	44.6	24.3	19.3	44.4
	Sum	45.6	**51.6**	61.6	64.6	43.8	23.7	21.0	44.6
	Temperature	44.7	50.4	64.4	64.9	44.9	24.2	18.9	44.6
	BIDS (ours)	**48.2**	50.4	**65.1**	64.9	45.1	25.1	**23.4**	**46.0**
	−`Iter`	47.4	48.4	64.6	**65.1**	**45.4**	**25.2**	23.0	45.6
	−(`Norm` + `Iter`)	46.5	47.3	64.6	65.0	44.1	24.7	22.8	45.0
15%	Task-max (LESS)	46.5	51.0	63.2	64.6	44.9	24.9	21.2	45.2
	Sum	48.2	51.0	62.6	64.6	44.8	24.0	19.3	44.9
	Temperature	45.6	**52.1**	**65.1**	64.7	45.5	23.6	18.7	45.0
	BIDS (ours)	**49.1**	50.7	63.7	64.6	**45.8**	**26.2**	**22.6**	**46.1**
	−`Iter`	47.4	50.1	64.9	**65.0**	45.6	26.0	20.8	45.7
	−(`Norm` + `Iter`)	47.4	48.1	63.2	**65.0**	**45.8**	25.1	20.3	45.0

Table 6: Complete results for Table 4 and 5 in the main text, with **Llama-3-8B** as the base model. Notably, −(**`Norm`** + **`Iter`**) (§6; Table 5) is equivalent to the **instance-wise max** algorithm (Appendix A.3; Table 4).

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Random	36.8	44.3	59.5	61.7	37.0	**19.9**	22.2	40.2
	Task-max (LESS)	36.8	**45.6**	**60.1**	**62.3**	**38.6**	19.5	23.8	41.0
	RDS	38.6	45.3	58.2	61.9	33.3	15.4	16.9	38.5
	S2L	36.8	44.4	58.9	61.9	36.5	16.6	21.2	39.5
	DEITA	**43.0**	**45.6**	59.5	62.0	36.1	17.2	19.8	40.5
	BIDS (ours)	37.7	44.4	59.5	61.8	38.0	19.8	**26.1**	**41.0**
10%	Random	37.7	44.8	59.8	61.8	40.0	21.2	22.0	41.0
	Task-max (LESS)	39.5	43.8	60.3	**62.0**	39.7	20.0	24.6	41.4
	RDS	**43.9**	45.0	57.8	61.7	35.6	14.9	18.1	39.6
	S2L	36.8	44.7	59.9	61.7	39.2	**22.0**	22.4	41.0
	DEITA	36.8	45.0	59.1	**62.0**	39.3	18.6	20.4	40.2
	BIDS (ours)	40.4	**46.1**	**60.5**	61.7	**40.5**	21.0	**27.1**	**42.5**
15%	Random	37.7	45.9	**59.6**	**61.9**	41.0	**21.7**	21.8	41.4
	Task-max (LESS)	40.4	44.1	**59.6**	61.3	**41.9**	21.5	24.0	41.8
	RDS	**48.2**	**48.1**	58.4	61.7	36.6	16.4	17.9	41.0
	S2L	34.2	46.7	58.8	61.7	40.3	20.9	22.6	40.7
	DEITA	37.7	46.1	58.5	61.5	40.1	20.3	21.6	40.8
	BIDS (ours)	39.5	47.0	59.5	61.7	41.8	**21.7**	**27.1**	**42.6**
	BIDS (ours; epochs=4)	40.4	47.0	**58.9**	**61.1**	44.1	23.5	**28.1**	**43.3**
100%	Full (epochs=1)	40.4	**49.6**	58.8	60.8	45.2	25.4	20.8	43.0
	Full (epochs=4)	**41.2**	49.3	54.6	59.4	**48.1**	**30.1**	19.6	43.2

Table 7: Complete results for Table 3 in the main text, with **Mistral-7B-v0.3** as the base model. Under all three budgets, BIDS still maintains the strongest macro-average performance, and ranks either first or second among the six candidate selection methods on 5/7 benchmarks on average. The top 15% BIDS-selected subset again outperforms full dataset training in macro average, by steadily improving on coding and math while maintaining its remarkable general instruction-following ability.

pairwise scores using the cosine similarity between hidden-layer representations of training and validation examples. The resulting method is dubbed "RDS-BIDS". This attempt is motivated by the observations in Table 2 that RDS-selected data also demonstrate significant bias towards coding-related tasks and lead to overall poor performance. We maintain the same training and evaluation setup,

Budget	**Method**	**Coding**		**Logic**	**Knowledge**	**Math**		**Ins-Following**	**Macro Avg**
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Task-max (LESS)	36.8	**45.6**	**60.1**	**62.3**	**38.6**	19.5	23.8	**41.0**
	Sum	35.1	43.3	59.7	62.1	38.2	17.6	23.2	39.9
	Temperature	34.2	42.4	59.9	61.8	37.2	19.7	21.8	39.6
	BIDS (ours)	**37.7**	44.4	59.5	61.8	38.0	**19.8**	26.1	**41.0**
	−`Iter`	36.8	44.1	59.1	61.5	38.2	19.6	**27.5**	**41.0**
	−(`Norm` + `Iter`)	33.3	45.0	59.3	61.6	38.0	18.7	22.0	39.7
10%	Task-max (LESS)	39.5	43.8	60.3	**62.0**	39.7	20.0	24.6	41.4
	Sum	36.8	43.0	59.7	61.6	40.1	18.0	22.0	40.2
	Temperature	35.1	45.6	60.0	**62.0**	39.1	18.4	22.2	40.3
	BIDS (ours)	**40.4**	**46.1**	**60.5**	61.7	40.5	**21.0**	**27.1**	**42.5**
	−`Iter`	37.7	45.0	59.7	61.6	40.2	20.2	26.7	41.6
	−(`Norm` + `Iter`)	36.0	43.8	59.7	61.5	**41.6**	20.8	24.6	41.1
15%	Task-max (LESS)	**40.4**	44.1	59.6	61.3	41.9	21.5	24.0	41.8
	Sum	37.7	47.0	59.4	61.3	41.2	18.1	23.8	41.2
	Temperature	37.7	**47.9**	**60.5**	**61.7**	39.7	18.4	24.2	41.4
	BIDS (ours)	39.5	47.0	59.5	**61.7**	41.8	**21.7**	**27.1**	**42.6**
	−`Iter`	38.6	46.4	59.8	61.6	41.8	20.9	26.1	42.2
	−(`Norm` + `Iter`)	39.5	45.0	60.1	61.6	**42.8**	21.5	23.2	42.0

Table 8: Complete results for Table 4 and 5 in the main text, with **Mistral-7B-v0.3** as the base model. Performance improvements from −(**`Norm`** + **`Iter`**) to −**`Iter`** to BIDS are consistent with Llama-3-8B in §6.

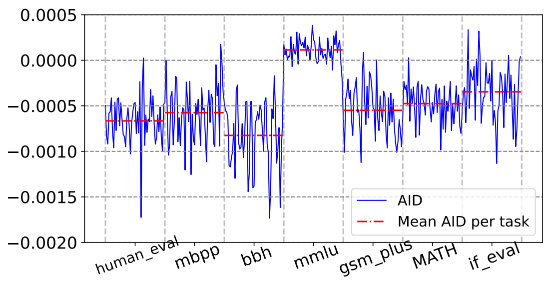


Figure 9: Unnormalized Average Influence Distribution (AID) of the **whole** UltraInteract dataset (Yuan et al., 2024), with Mistral-7B-v0.3 as the base model. It still shows great inter-task and intra-task scale disparities.

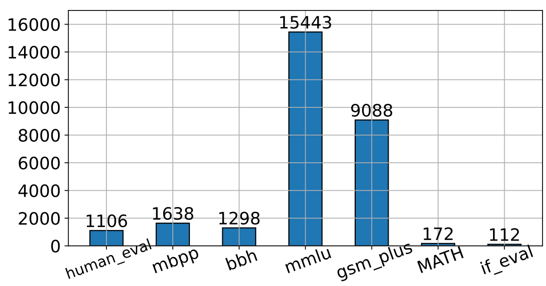
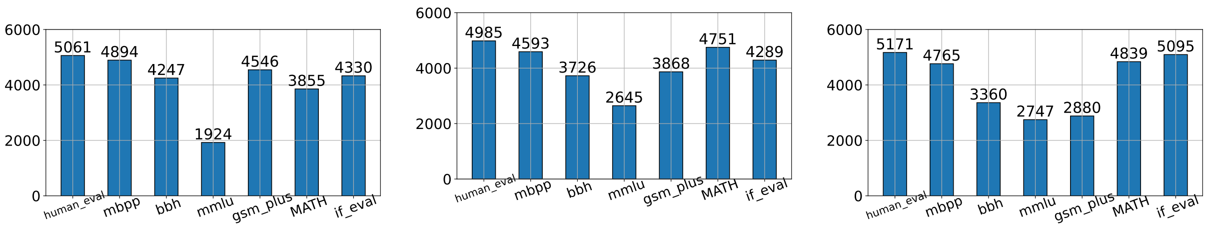


Figure 10: Task frequencies with Highest Influence (THI) of LESS-selected data under the **10%** budget, with Mistral-7B-v0.3 as the base model. In this case, MMLU is even more obviously oversampled than prior observations.

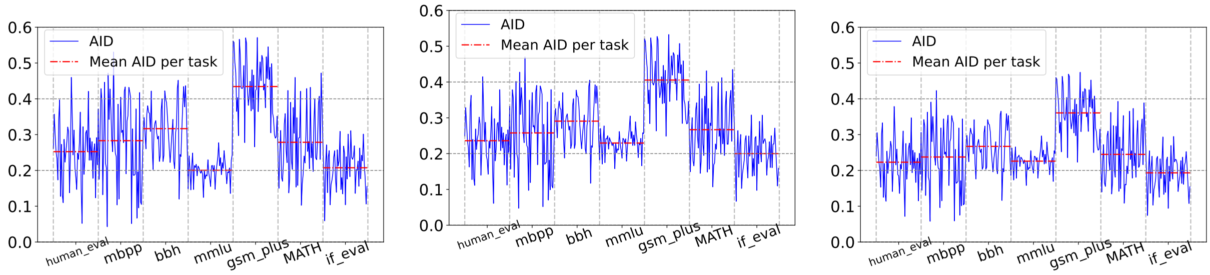


(a) −(`Norm` + `Iter`) (b) −`Iter` (c) BIDS

Figure 11: THI under the 10% budget, with the base model being Mistral-7B-v0.3. Similar to prior observations with Llama-3-8B, both −**`Iter`** and BIDS have more balanced task frequencies than −(**`Norm`** + **`Iter`**).

and show the comparison results across three budgets in Table 9.

We can see that the balancing techniques of BIDS work well for RDS-computed scores. Across all three budgets, RDS-BIDS significantly improves over the original RDS and consistently outperforms the random baseline. In terms of task-specific performance, RDS-BIDS also demonstrates extensive and balanced improvements in tasks other than coding, especially math and instruction-following. This shows the necessity of balancing various types of utility scores in general-



(a) $-$(Norm + Iter) (b) $-$Iter (c) BIDS

Figure 12: Comparative analysis of normalized AID under the 10% budget, with the base model being Mistral-7B-v0.3. Similar to prior observations with Llama-3-8B, from $-$(**Norm** + **Iter**) to $-$**Iter** to BIDS, the disparity among different tasks and instances in AID gradually diminishes, with both decreasing maximums and increasing minimums, although the degree of the original imbalance for Mistral-v0.3 is not as high as Llama-3.

Budget	**Method**	**Coding**		**Logic**	**Knowledge**	**Math**		**Ins-Following**	**Macro Avg**
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Random	43.5	48.9	64.8	64.9	41.5	22.5	18.1	43.4
	RDS	45.6	52.7	62.2	65.0	34.5	17.2	15.5	41.8
	RDS-BIDS	50.9	53.0	64.1	65.0	41.0	19.7	20.0	**44.8**
10%	Random	47.8	50.6	65.0	64.9	43.9	24.0	17.8	44.9
	RDS	50.0	54.7	63.2	64.6	39.3	22.4	18.3	44.6
	RDS-BIDS	47.4	53.6	63.2	65.2	43.6	24.1	21.6	**45.5**
15%	Random	48.7	51.9	65.2	65.1	45.6	25.0	18.8	45.7
	RDS	50.0	53.9	63.7	64.5	41.1	23.5	18.1	45.0
	RDS-BIDS	48.2	53.0	63.8	64.8	45.1	24.2	21.4	**45.8**

Table 9: When the key technical components of BIDS (i.e., (1) instance-level normalization, and (2) iterative selection favoring underrepresented tasks) are applied to the utility matrix of RDS, the resulting method, RDS-BIDS, significantly improves over the original RDS and consistently outperforms the random baseline. These results are obtained with Llama-3-8B as the base model.

purpose instruction tuning, in response to the Limitations (§9) section in the main text, and validates the broad applicability of the underlying principles of BIDS. However, as also stated in §9, the focus of our work on influence-based approaches limits detailed investigation into the more general imbalance of data selection scores under a multi-capability learning setup. We thus leave further exploration in this direction to future work.

A.8 Generalizability of BIDS to Different Evaluation Suites, Training Datasets, and Model Sizes

In this section, we additionally validate the effectiveness of BIDS under three different experimental setups that vary in evaluation suites, training datasets, and size of the base model. Results in this section show that the effectiveness of BIDS is robust and generalizable across different experimental configurations, thus further solidifying the claims in the main text.

Different evaluation suites. First, we keep the same training dataset (i.e., UltraInteract), but drop MMLU and BBH, the two tasks with highest and lowest average influence respectively, from the evaluation suite. In this way we investigate how BIDS performs when the disparity of cross-task influence decreases. This evaluation setup also aligns more with the practical use case where an AI agent should not only have strong math and coding capabilities but also faithfully follow user instructions. From the results in Table 10, we find that despite decreased cross-task influence discrepancies, BIDS is still effective in balancing and improving performance across tasks. Notably, it advances the performance of the original LESS-selected data in all 5 evaluation tasks.

Different training datasets. Next, we investigate how BIDS performs with a different training dataset. For better comparison with prior work, we adopt the original training mixture used in LESS, which is composed of COT (Wei et al., 2022),

Method	Coding		Math		Ins-Following	Macro Avg
	HumanEval	MBPP	GSM-Plus	MATH	IFEval	
Random	**47.8**	**50.6**	43.9	24.0	17.8	36.8
Task-max (LESS)	46.5	49.0	43.0	23.9	19.3	36.3
BIDS	46.5	49.9	**44.5**	**25.3**	**24.6**	**38.2**

Table 10: Results of **Llama-3-8B**, with the training set being UltraInteract and the selection budget being 10%. **MMLU and BBH, the two tasks with highest and lowest average influence respectively, are dropped from the original evaluation suite.** This validates that the effectiveness of BIDS is not a special case under certain task setup, and can generalize to different evaluation suites.

Method	Coding	Logic	Knowledge	Math	Ins-Following	Macro Avg
	HumanEval	BBH	MMLU	GSM-Plus	IFEval	
Random	40.4	40.0	**65.5**	12.2	33.8	38.4
Task-max (LESS)	44.7	24.3	65.0	**16.9**	38.1	37.8
BIDS	**45.6**	**53.5**	64.7	**16.9**	**41.3**	**44.4**

Table 11: Results of **Llama-3-8B**, with the training set being the original mixture used in Xia et al. (2024) (COT (Wei et al., 2022), DOLLY (Conover et al., 2023), FLAN V2 (Longpre et al., 2023), and OPEN ASSISTANT 1 (Köpf et al., 2023)), and the selection budget being 10%. This validates that the effectiveness of BIDS can generalize to different training datasets.

Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
	HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
Random	30.7	39.5	**48.6**	56.6	25.2	**14.1**	12.0	32.4
Task-max (LESS)	28.9	40.7	46.9	56.1	23.6	13.5	14.7	32.1
BIDS	**33.3**	**42.1**	48.2	**56.8**	**25.7**	**14.1**	**17.1**	**33.9**

Table 12: Results of **Llama-3.2-3B**, with the training set being UltraInteract and the selection budget being 10%. This validates that the effectiveness of BIDS can generalize to models of different sizes.

DOLLY (Conover et al., 2023), FLAN V2 (Longpre et al., 2023), and OPEN ASSISTANT 1 (Köpf et al., 2023), and has 270K examples altogether. This training dataset is more general-purposed compared with the reasoning focus of UltraInteract, thus suitable for testing the universal applicability of BIDS. For the evaluation suite, we drop MBPP and MATH, and evaluate on the remaining five tasks to align with the general property of the new training dataset. Table 11 still shows significant improvements of BIDS over other baselines. Notably, we observe the similar counterintuitive phenomenon for LESS to §3, where MMLU has the highest average influence but does not show greater room for performance improvements. We also investigate why LESS shows extremely poor performance in BBH. It turns out that models trained on LESS-selected data tend to stop generation right after generating “Let’s think step by step”, which is likely a consequence of training on imbalanced data mixtures.

Different size of the base model. Moreover, to verify that the effectiveness of BIDS is robust to the size of the base model, we experiment with Llama-3.2-3B using the same training and evaluation setup as the Llama-3-8B and Mistral-7B-v0.3 experiments in the main text. Table 12 again validates that the effectiveness of BIDS can generalize to modern smaller models. Notably, apart from improved average performance, BIDS also outperforms the random baseline and the original LESS in 6/7 and 7/7 tasks respectively, demonstrating the generalizability of its balancing effect.

A.9 Discussion on the Computational Cost

In this section, we aim to discuss and show that BIDS does not incur much memory or latency overhead, and can thus serve as an efficient plug-and-play module. In our training and evaluation setup, the $|D|$ dimension for the Attribution Matrix is about 288K (Yuan et al., 2024), and the $|V|$ dimension is 350. Therefore, the memory cost for storing the AM using FP64 precision is less than 800 MB. The latency cost for running the whole BIDS algorithm is less than 1 minute with CUDA acceleration of a single H100 GPU. More generally, since many popular mixtures of instruction finetuning data are maintained on the scale of hundreds of thousands (Wang et al., 2023; Ivison et al., 2023; Yuan et al., 2024; Yue et al., 2023), the memory and latency cost of BIDS should be light for most practical training setups.

A.10 Qualitative Analysis

In this section, we aim to demonstrate the following two properties of BIDS with some qualitative examples, and better illustrate its effectiveness:

1. Models trained on BIDS-selected data can indeed achieve a stronger balance between mastering task-specific skills (e.g., math reasoning, coding knowledge, etc.) and fully understanding various types of instructions given by the user (e.g., format-following, response style, etc.).
2. Such a stronger balance is indeed helpful to improving the accuracy or human-perceived quality of model response.

Concretely, we present three sets of model responses in the task of coding (Table 13), math (Table 14) and general instruction-following (Table 15) respectively. Each set contains a correct response by a Mistral-7B-v0.3 model trained on top-15% BIDS-selected data, and a false response by the same base model trained on the full (i.e., 100%) UltraInteract, both to exactly the same prompt. We analyze how the BIDS-trained model correctly answers all these prompts due to the greater balance of capabilities it achieves.

Table 13: For the example 1, the model trained on the full dataset fails to handle the corner case of `numbers = []`. For the example 2, the full-trained model also fails at not adding the constraint of `y != x` in its sorting rule. In both cases, BIDS-trained model returns the correct code completion because it better considers and handles corner cases. It reflects that BIDS-trained model balances its capability in correct coding knowledge and comprehensive thinking behavior.

HumanEval: Example 1

Prompt

```
def rolling_max(numbers):
    # Write a Python function ‘rolling_max(numbers: List[int])
     -> List[int]‘ to solve the following problem:
    # From a given list of integers, generate a list of rolling
      maximum element found until given moment
    # in the sequence.
    # >>> rolling_max([1, 2, 3, 2, 3, 4, 2])
    # [1, 2, 3, 3, 3, 4, 4]
```

Wrong Completion: Full-trained model

```
# Initialize the result list with the first element of the
     input list
result = [numbers[0]]
# Initialize the maximum value with the first element of the
     input list
max_value = numbers[0]
# Iterate through the input list starting from the second
     element
for num in numbers[1:]:
    # If the current number is greater than the maximum value,
     update the maximum value and append the current number to
     the result list
    if num > max_value:
        max_value = num
        result.append(num)
    # If the current number is not greater than the maximum
     value, append the maximum value to the result list
    else:
        result.append(max_value)
# Return the result list
return result
```

Correct Completion: BIDS-trained model

```
if not numbers:
    return []
n = len(numbers)
result = [numbers[0]]
for i in range(1, n):
    result.append(max(numbers[i], result[-1]))
return result
```

HumanEval: Example 2

Prompt

```
def find_closest_elements(numbers: List[float]) -> Tuple[float,
      float]
    # From a supplied list of numbers (of length at least two)
     select and return two that are the closest to each other
     and return them in order (smaller number, larger number).
    # >>> find_closest_elements([1.0, 2.0, 3.0, 4.0, 5.0, 2.2])
    # (2.0, 2.2)
    # >>> find_closest_elements([1.0, 2.0, 3.0, 4.0, 5.0, 2.0])
    # (2.0, 2.0)
```

Wrong Completion: Full-trained model

```
closest_pair = min(numbers, key=lambda x: min(abs(x - y) for y
     in numbers))
return closest_pair, min(numbers, key=lambda x: abs(x -
     closest_pair))
```

Correct Completion: BIDS-trained model

```
# Step 1: Sort the list of numbers in ascending order
numbers.sort()
# Step 2: Initialize the closest_numbers variable with the
     first two elements of the sorted list
closest_numbers = (numbers[0], numbers[1])
# Step 3: Iterate through the list to find the two numbers that
      are the closest to each other
for i in range(1, len(numbers) - 1):
    # Step 4: Calculate the difference between the current
     number and the next number
    diff = abs(numbers[i] - numbers[i + 1])
    # Step 5: If the difference is less than the difference
     between the current closest numbers, update the closest
     numbers
    if diff < abs(closest_numbers[0] - closest_numbers[1]):
        closest_numbers = (numbers[i], numbers[i + 1])
# Step 6: Return the two closest numbers in order (smaller
     number, larger number)
return closest_numbers
```

Table 14: For the example 1, BIDS-trained model understands the implicit condition contained in the question that only after a whole number of years can a tree grow oranges for sale, and thus goes on to round the calculated the calculation to integer (Step 7). For the example 2, the model trained on the full dataset fails to understand the meaning of "a person was born 8 years before another" (Step 3) and thus performs the calculation in the wrong way. In both examples, BIDS-trained model correctly answers the question because it balances its capability in math reasoning and natural language understanding.

GSM-Plus: Example 1	**GSM-Plus: Example 2**
Question	**Question**
Solve the following math problem step-by-step. Simplify your answer as much as possible. Present your final answer as \boxed{Your Answer}. Carlos is planting an orange tree. The tree will cost $120 to plant. Each year it will grow 10 oranges, which he can sell for $2 each. It costs $4 a year to water and feed the tree. How many years will it take before he starts earning money on the orange tree?	Solve the following math problem step-by-step. Simplify your answer as much as possible. Present your final answer as \boxed{Your Answer}. Raymond and Samantha are cousins. Raymond was born 8 years before Samantha. Raymond had a son at the age of 25. If Samantha is now 35, how many years ago was Raymond's son born?
Wrong Response: Full-trained model	**Wrong Response: Full-trained model**
Step 1: To determine the number of years it will take for Carlos to start earning money from the orange tree, we need to consider the initial cost of planting the tree and the annual expenses associated with watering and feeding it. Step 2: The tree costs $120 to plant, and it incurs an annual expense of $4 for watering and feeding. Step 3: Each year, the tree produces 10 oranges, which he can sell for $2 each. Step 4: So, the total income from selling the oranges each year is 10 * $2 = $20. Step 5: However, we need to subtract the annual expenses of $4 from the income to find the net profit from the oranges. Step 6: Therefore, the net profit from the oranges each year is $20 - $4 = $16. Step 7: To determine the number of years it will take for Carlos to start earning money, we can divide the initial cost of planting the tree by the net profit from the oranges each year. Step 8: This gives us the number of years it will take for the tree to start generating a positive return on investment. Step 9: So, the number of years it will take before Carlos starts earning money on the orange tree is $120 / $16 = 7.5$ years. Answer: \boxed{7.5}	Step 1: To solve this problem, we need to first understand the given information and then work through the steps to find the age of Raymond's son. Step 2: We're given that Raymond was born 8 years before Samantha, and Samantha is currently 35 years old. Step 3: This means that Raymond was born 35 - 8 = 27 years ago. Step 4: Now, we know that Raymond had a son at the age of 25. So, we need to find out how many years ago Raymond's son was born. Step 5: To do this, we can simply subtract Raymond's age when his son was born (27 years) from the current age of Samantha (35 years). Step 6: This gives us 35 - 27 = 8 years. Step 7: Therefore, Raymond's son was born 8 years ago. Answer: \boxed{8}
Correct Response: BIDS-trained model	**Correct Response: BIDS-trained model**
Step 1: To determine how many years it will take for Carlos to start earning money from his orange tree, we need to consider the costs involved in planting and maintaining the tree, as well as the revenue generated from selling the oranges. Step 2: First, we need to calculate the total cost of planting the tree, which is $120. Step 3: Each year, the tree will grow 10 oranges that can be sold for $2 each, resulting in a total revenue of $20 per year. Step 4: However, it also costs $4 per year to water and feed the tree. Step 5: Therefore, the net revenue from selling the oranges each year is $20 - $4 = $16. Step 6: To determine how many years it will take for Carlos to start earning money, we can divide the total cost of planting the tree by the net revenue each year. This gives us 120 / 16 = 7.5 years. Step 7: Since we can't plant a tree for half a year, Carlos will need to wait 8 years before he starts earning money from the orange tree. Answer: \boxed{8}	Step 1: To solve this problem, we need to understand the relationship between Raymond and Samantha, and then calculate the age of Raymond's son based on the given information. Step 2: Raymond is Samantha's cousin, and he was born 8 years before her. This means that Raymond is 8 years older than Samantha. Step 3: Raymond had a son at the age of 25, so we need to calculate Raymond's age at that time. Since Raymond is 8 years older than Samantha, and Samantha is currently 35, we can calculate Raymond's age as 35 + 8 = 43. Step 4: Now, we know that Raymond had his son at the age of 25, so we can calculate the age of Raymond's son by subtracting 25 from Raymond's age at that time, which is 43 - 25 = 18. Step 5: Therefore, Raymond's son was born 18 years ago. Answer: \boxed{18}

Table 15: For the example 1, the model trained on the full dataset ignores the explicit requirement of forming its output as a single JSON block. It is especially overfit to the response format of `Step X: ...` which is frequently seen in the training data. In contrast, BIDS-trained model handles its CoT in a nifty way by expressing its thinking process in the values of a JSON block. For the example 2, full-trained model not only fails to completely follow the explicitly required format, but also hardly writes a poem, compared with the high-quality one generated by BIDS-trained model. In both examples, BIDS-trained model generates content with significantly higher quality due to the balance between strict format-following and flexible use of parametric knowledge.

IFEval: Example 1	**IFEval: Example 2**
Prompt	**Prompt**
Please provide the names of 5 famous moms in JSON format. Please, use any interesting or weird tone. Your entire output should just contain a JSON block, nothing else.	Write a poem about how I am missing my classes. The poem must have 4 sections marked with SECTION X. Finish the poem with this exact phrase: "Can I get my money back for the classes I missed?"
Wrong Response: Full-trained model	**Wrong Response: Full-trained model**

Step 1: Let's think of 5 famous moms who have made a significant impact in their respective fields.
Step 2: Oprah Winfrey - Talk show host, media mogul, producer, actress, philanthropist
Step 3: Angelina Jolie - Actress, filmmaker, humanitarian
Step 4: Michelle Obama - Former First Lady of the United States
Step 5: Serena Williams - Professional tennis player
Step 6: Create a JSON block with the names of the 5 famous moms.
 	
``javascript
// Step 7: Create a JSON block with the names of the 5 famous moms.
[
"Oprah Winfrey",
"Angelina Jolie",
"Michelle Obama",
"Serena Williams"
]
``
	Section 1:
	I missed my first class today and I'm feeling quite disappointed.
	Section 2:
	I really wanted to learn all the new material, but I had to stay home due to illness.
	Section 3:
	I hope I can make up for the missed class by studying extra hard for the next exam.
	Section 4:
	Can I get my money back for the classes I missed?
	
	Answer:
	"I missed my first class today and I'm feeling quite disappointed.
	I really wanted to learn all the new material, but I had to stay home due to illness.
	I hope I can make up for the missed class by studying extra hard for the next exam.
	Can I get my money back for the classes I missed?"

Correct Response: BIDS-trained model	**Correct Response: BIDS-trained model**

{
"Meryl Streep": "An American actress and filmmaker, known for her versatile and powerful performances in films like Kramer vs. Kramer, Sophie's Choice, The Devil Wears Prada, and many more.",
"Oprah Winfrey": "An American media mogul, talk show host, actress, producer, and philanthropist. She is best known for her talk show The Oprah Winfrey Show, which ran for 25 years.",
"Sarah Jessica Parker": "An American actress, producer, and designer. She is best known for her role as Carrie Bradshaw in the HBO series Sex and the City.",
"Jennifer Garner": "An American actress, producer, and entrepreneur. She is best known for her roles in films like 13 Going on 30, Juno, and Alias.",
"Angelina Jolie": "An American actress, filmmaker, and humanitarian. She is best known for her work as a United Nations Special Envoy and as a Goodwill Ambassador."
}
	SECTION 1:
	I'm sitting here feeling so lonely,
	Missing my classes with all my friends,
	Wishing I could be there with them,
	Instead of sitting at home and mend.
	SECTION 2:
	I'm looking at my empty calendar,
	Wondering what I'm missing out on,
	Wishing I could join in on the fun,
	Instead of staring at the screen and frown.
	SECTION 3:
	I'm listening to the sound of silence,
	While my friends are making memories,
	Wishing I could join in on the laughter,
	Instead of feeling left out and sadder.
	SECTION 4:
	I'm staring at the clock, counting the minutes,
	Until I can see my teachers and classmates,
	Wishing I could just teleport there,
	Instead of waiting for the next class to start.
	
	Finish:
	Can I get my money back for the classes I missed?