IMPROVING INFLUENCE-BASED INSTRUCTION TUNING DATA SELECTION FOR BALANCED LEARNING OF DIVERSE CAPABILITIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Selecting appropriate training data is crucial for effective instruction fine-tuning of large language models (LLMs), which aims to (1) elicit strong capabilities, and (2) achieve balanced performance across a diverse range of 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 certain 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, harms performance on these high-influence tasks themselves.

As a remedy, 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 balances data selection by choosing 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 selection frameworks. Surprisingly, training on a 15% subset selected by BIDS can even outperform full-dataset training with a much more balanced performance. Our analysis further highlights the importance of both instance-level normalization and iterative optimization of selected data for balanced learning of diverse capabilities.

1 INTRODUCTION

Supervised instruction finetuning (SFT) plays a crucial role in eliciting strong capabilities from large language models (LLMs). Typically, a pretrained LLM is finetuned on a mixture of different datasets to achieve strong and balanced performance (Ouyang et al., 2022; Touvron et al., 2023; Dubey et al., 2024; Jiang et al., 2023). The importance of SFT data quality (Zhou et al., 2024) has spawned many works on instruction tuning data selection (Cao et al., 2023; Chen et al., 2023; Liu et al., 2023). Influence-based methods estimate each individual training example's influence on the model's prediction on a downstream task (Koh & Liang, 2017; Pruthi et al., 2020). Thanks to recent advances, they have been scaled to LLM-level computations and demonstrated strong potential in facilitating high-quality data selection (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 & 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 certain tasks exhibit larger magnitudes than others, introducing systematic bias in the data selection process when cross-task influence scores are directly compared, as done in many existing works (Yin & Rush, 2024; 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 the LLM to achieve balanced capabilities. Second, perhaps counterintuitively, it may even hurt

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

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 challenges with two key designs. Given a training dataset to select from and a validation dataset representing the diverse target tasks, we formulate the influence-based selection 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 top-ranked examples with the highest influence values, BIDS applies an iterative selection algorithm. At each iteration, this algorithm compares the influence of each candidate training example with the average influence of those already selected ones, and selects the candidate that can provide the largest marginal improvement. If the current selected dataset falls short in influence on certain validation instances, then our algorithm will intuitively favor candidate examples that have high influence on the specific tasks represented by these validation data. In this way, BIDS actually favors training data that contribute most to the underrepresented tasks in the current selected subset, and thus promotes balanced multi-task learning.

In order to show the consistently strong performance of BIDS, we conduct experiments on an extensive suite of training and evaluation data, UltraInteract (Yuan et al., 2024), with base models from two different families—Llama-3-8B (Dubey et al., 2024) and Mistral-7B-v0.3[2]. Across seven tasks spanning five diverse capabilities including coding, math, logical inference, world knowledge and general instruction following, BIDS consistently outperforms both influence- and non-influence-based selection algorithms, not only in terms of macro-average performance across diverse tasks, but also in most individual cases. Surprisingly, a 15% subset selected by BIDS even outperforms full-dataset training in average performance, emphasizing the huge potential of selective training in multi-capability learning of LLMs. Further analysis reveals the positive contributions from both the instance-level normalization and iterative selection. Investigation of the influence distribution of BIDS-selected data also gives valuable insight on how BIDS reduces the influence disparity across tasks, and what might be the properties of a balanced set of influential data.

The contributions of this paper include:

1. We identify the problem of influence-based data selection algorithms in instruction tuning LLMs for learning diverse tasks, and attribute this problem to an inherent bias in cross-task influence through systematic analysis.
2. We propose BIDS, a simple and effective influence-based selection algorithm for balanced learning of diverse capabilities.
3. Through extensive experiments, we confirm the consistent and significant effectiveness 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 individual training examples on model predictions is critical for understanding model behavior and selecting influential training data to improve model performance. Traditional methods, including retraining-based (Ghorbani & Zou, 2019; Ilyas et al., 2022; Park et al., 2023) and gradient-based (Koh & Liang, 2017; Pruthi et al., 2020) approaches, have proven effective but are computationally prohibitive when scaling to LLMs, as they either require 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 & Lowd, 2024; Ko et al., 2024). Several recent advances have sought to address these challenges by extending gradient-based approaches to scale more effectively. Given a large training dataset to select from and a validation set representing some targeted capabilities, 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 highest influence on the validation set. LOGRA (Choe et al., 2024) leverages a low-rank gradient projection algorithm to further improve the efficiency.

[2] `https://huggingface.co/mistralai/Mistral-7B-v0.3`

MATES (Yu et al., 2024) formulates the pointwise data influence between each training point and the whole validation set, and uses a small data influence model to learn this pointwise influence.

Upon closer inspection, these LLM-scale influence-based selection methods share a similar problem formulation. They all need a validation set to represent a targeted data distribution and require the computation of pointwise data influence between each training example and the validation data. In this work, we aim to extend this influence-based data selection paradigm to the setup of multi-task instruction tuning, where the model is expected to simultaneously learn 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 shared among these methods, the results of our work should also provide useful insight 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 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 influence information presented in $\boldsymbol{A}$. Finetuning the LLM on $\mathcal{T}$ is supposed to help the model 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 significantly unbalanced and weak performance in a multi-task learning setup. 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, high-quality 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 and reason over world knowledge, and IFEval (Zhou et al., 2023) for the fine-grained instruction following ability. For more details about the training and evaluation setups, please refer to Appendix A.2.

For the **influence estimation method** throughout this work, we follow the original pipeline introduced by LESS, with an equal number of validation instances sampled uniformly from each of the seven evaluation tasks. In this section, for the **data selection algorithm**, we also start with the **task-wise max** algorithm (Appendix A.3) used by LESS, which, for each training example, first computes its mean influence over validation examples within the same task, followed by selecting training examples with the highest maximum influence across different tasks. We compare this algorithm against a random selection baseline, which represents the average performance of models trained on two sets of randomly selected data.

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: Comparison between LESS and the random baseline. The highest performance for each task and macro-average is **bolded**. LESS only outperforms the random baseline in macro-average under the 5% budget, while lags behind under the other two with imbalanced performance distributions.

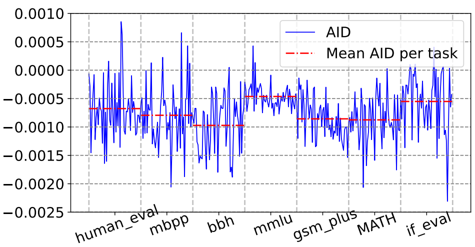


Figure 1: Unnormalized Average Influence Distribution (AID) of the **whole** UltraInteract dataset (Yuan et al., 2024), showing great scale disparities for inter-task and intra-task influence.

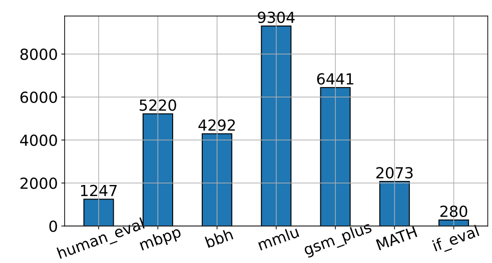


Figure 2: Task frequencies with Highest Influence (THI) under the **10%** budget. MMLU is obviously oversampled in LESS-selected data.

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 a margin over 1.5%, it also consistently and significantly lags behind in BBH by two to three points, and shows no clear trend of 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 performance 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, thus less suitable for general-purpose instruction tuning. But the observations above still raise critical questions, especially given that an equal number of validation instances were used for each task during selection. This suggests a potential inherent bias in the influence values across different tasks, which could skew the selection algorithm towards certain capabilities. If the overall influence on certain task is inherently higher, then the naive task-wise max selection algorithm will naturally prioritize training examples that have high influence on these tasks, 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 have larger space for performance improvement.

What causes the imbalance of LESS? To examine the influence distribution of LESS-selected data, 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.
- The **Task Frequency with Highest Influence (THI)** for a task t is the number of selected training examples that have the highest average influence on t.

Our AID analysis of the whole UltraInteract dataset (Figure 1) reveals both task- and instance-level discrepancies. MMLU receives the highest average influence that is substantially higher than BBH's, despite the fact that MMLU is out-of-distribution for the training data, while BBH is in-distribution.

Moreover, there are also significant influence disparities for validation instances inside the same task. For example, the gap between the highest and lowest instance-wise influence inside IFEval is more than 0.0025, while the globally highest instance-wise influence is less than 0.001. These results answer our question (1) by confirming that the scales of influence values indeed differ significantly across various tasks.

Further, the THI analysis of LESS-selected data (Figure 2) validates that the scale differences indeed make the selection algorithm of LESS disproportionately favor certain tasks over others. Specifically, MMLU has the highest frequency of being the most influential task, which is 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 even frequently underperforms the random baseline on MMLU. For other in-distribution tasks with high THI, such as MBPP, BBH, and GSM-Plus, LESS is either consistently outperformed or shows no clear trend of advantage. As is suggested by these observations, although high-influence tasks tend to have more supporting data in the selected dataset, they do not necessarily have larger room for performance improvement. Besides, such biased sampling may hinder the learning of other necessary capabilities as well. Thus, we answer the 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 the data selection algorithm employed by LESS.

4 BIDS: Selecting Influential Data for Balanced Capability Learning

In this section, we introduce BIDS, a ***B**alanced and **I**nfluential **D**ata **S**election* algorithm to address the issues identified in §3. BIDS has two key design choices: (1) instance-level normalization, and (2) iterative selection favoring underrepresented tasks.

Instance-level normalization. At a higher level, this technique aims to address the scale difference of influence values across different validation instances. This can be achieved by applying a column-wise normalization to the Attribution Matrix. Specifically, for 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 step ensures that the influence scores of different columns are on the same 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 in Appendix A.5) to promote the balance over different capabilities. It begins with an empty set. In each iteration, the algorithm 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 the candidate subset $\mathcal{D} \setminus \mathcal{T}$, 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 example with the highest utility is selected for this iteration. In other words, BIDS actually favors training examples that can bring the largest improvement in influence to the most underrepresented task of the current selected data. This approach essentially differs from LESS, which only scores each training example independently and then selects the top-ranked ones, by considering the interactions of influence distributions among different selected examples and promoting the balance of overall influence distribution of the selected dataset.

5 Experiments

5.1 Experimental Setups

Basic setup. We follow the experimental setup outlined in §3, including the same set of LLMs, datasets, tasks, and influence estimation implementations. To further validate the generalizability of BIDS, we also conduct experiments on base models from different model families, which is detailed in Appendix A.6.

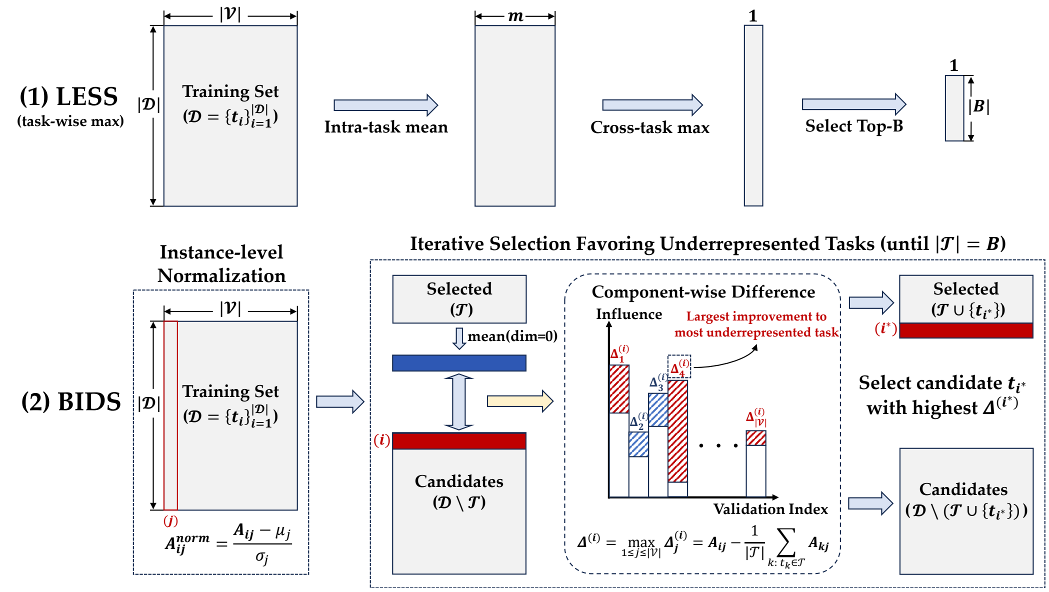


Figure 3: A comparison between BIDS and the task-wise max algorithm used by LESS. For convenience, we represent the training set $\mathcal{D}$ with its Attribution Matrix (AM), in which the i-th row is the $|\mathcal{V}|$-dimensional Influence Distribution of the i-th training example, $\boldsymbol{t}_i$, in $\mathcal{D}$. BIDS differs from LESS in mainly two aspects. First, it applies a column-wise normalization to the AM. Next, instead of directly selecting top-B examples in influence, BIDS applies an iterative algorithm which, at each iteration, obtains the utility $\Delta^{(i)}$ of each candidate example $\boldsymbol{t}_i$ by calculating how much improvement in influence it can bring to the current selected subset $\mathcal{T}$, and selects candidate $\boldsymbol{t}_{i^*}$ with the highest utility $\Delta^{(i^*)}$. Please see §4 for a more detailed walkthrough.

Baselines. Apart from the original **task-wise max** algorithm used by LESS, we compare with two other influence-based intuitive variants that are applicable to the Attribution Matrix. In addition, we also compare with three strong non-influence-based selection methods to further demonstrate the necessity of balanced data selection. These five additional baselines are briefly summarized below. For more details about their mechanisms and implementation, please refer to Appendix A.3.

- **Instance-wise max:** For each training example, it uses the maximum of influence values over all validation instances as the utility score. Training examples with highest scores are selected.
- **Sum** also selects training examples with highest scores, but uses the sum of an example's influence instead of the max.
- **Representation-based Data Selection (RDS; Zhang et al., 2018; Hanawa et al., 2020; Xia et al., 2024)** is a non-influence-based **targeted** data selection method, as it also requires access to validation data that represent the targeted capabilities. It selects the training data that are most similar to the validation instances based on the language model's hidden representations.
- **S2L (Yang et al., 2024)** and **DEITA (Liu et al., 2023)** are two state-of-the-art, non-influence-based **general** data selection methods. They aim to select training data that are generally of high-quality and diverse without knowing the targeted capabilities beforehand. We hypothesize that such a general property may save these two methods from severe bias towards certain tasks or capabilities.

5.2 Results

Comparison under the same budget. As shown in Table 2, across the 5%, 10% and 15% budgets, BIDS consistently outperforms both influence-based and non-influence-based baselines in terms of the macro-average score across all seven benchmarks. Moreover, when compared on specific tasks, BIDS ranks either first or second among the eight selection algorithms on 3/7, 4/7 and 3/7 benchmarks under the three budgets respectively. These results show that BIDS indeed helps achieve strong and balanced performance across multiple different tasks.

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	Random	43.5	48.9	<u>64.8</u>	64.9	41.5	22.5	18.1	43.4
	Task-max (LESS)	43.9	50.7	62.7	**65.1**	<u>42.5</u>	22.6	19.7	43.9
	Sum	**45.6**	51.9	63.6	64.8	42.4	21.3	20.1	<u>44.2</u>
	Instance-max	43.9	<u>52.1</u>	63.2	65.0	**42.6**	22.3	<u>20.6</u>	<u>44.2</u>
	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	<u>22.9</u>	**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	<u>44.6</u>	24.3	19.3	44.4
	Sum	45.6	<u>51.6</u>	61.6	64.6	43.8	23.7	21.0	44.6
	Instance-max	46.5	47.3	64.6	**65.0**	44.1	<u>24.7</u>	<u>22.8</u>	45.0
	RDS	<u>50.0</u>	**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	<u>45.3</u>
	BIDS (ours)	48.2	50.4	<u>65.1</u>	64.9	**45.1**	**25.1**	**23.4**	**46.0**
15%	Random	48.7	<u>51.9</u>	65.2	**65.1**	45.6	25.0	18.8	<u>45.7</u>
	Task-max (LESS)	46.5	51.0	63.2	64.6	44.9	24.9	<u>21.2</u>	45.2
	Sum	48.2	51.0	62.6	64.6	44.8	24.0	19.3	44.9
	Instance-max	47.4	48.1	63.2	<u>65.0</u>	**45.8**	<u>25.1</u>	20.3	45.0
	RDS	<u>50.0</u>	**53.9**	63.7	64.5	41.1	23.5	18.1	45.0
	S2L	**51.8**	50.7	<u>65.4</u>	64.8	44.7	24.2	16.9	45.5
	DEITA	<u>50.0</u>	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 selection algorithms, including both influence-based and non-influence-based ones. The task-specific or macro-average performance is **bolded** if it ranks first under the same budget, and <u>underlined</u> if it ranks second. “BIDS (epochs=4)” is compared with 100% full training. When scaling the training of BIDS to four epochs, it outperforms full-dataset training with both one and four epochs, showing its consistently strong and balanced performance.

Notably, the three non-influence-based selection methods (i.e., RDS, S2L, DEITA) are consistently outperformed by the random baseline across all three budgets. Upon inspection of their task-specific performance, both RDS and DEITA are significantly biased towards the two 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 consolidate the unique 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. Moreover, they also suggest that the imbalance of utility scores (Yin & Rush, 2024) may commonly exist for both influence- and non-influence-based data selection approaches, and deserves careful investigation in future work.

BIDS outperforms full-dataset training. As shown in the last three rows in Table 2, training on a 15% subset selected by BIDS over four epochs consistently outperforms full-dataset training. Further analysis on task-specific performance reveals that BIDS achieves better performance by maintaining balanced and strong performance across six reasoning-related tasks while significantly improving 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. This finding highlights the potential for training on selective subsets to offer more efficient and effective instruction finetuning.

6 Analysis

This section presents ablation studies and analyses of the two key components of BIDS, in terms of their contributions to BIDS’ performance improvements and their effect on the selected data.

Budget	Method	Coding		Logic	Knowledge	Math		Ins-Following	Macro Avg
		HumanEval	MBPP	BBH	MMLU	GSM-Plus	MATH	IFEval	
5%	BIDS	**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%	BIDS	**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%	BIDS	**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 3: Respective contribution of the two components of BIDS. $-$**`Iter`** ablates the iterative selection, and $-$(**`Norm`** + **`Iter`**) further ablates both normalization and iterative selection. The highest performance is **bolded** for each task and macro-average. The performance shows a decreasing trend as more technical components are ablated, which substantiates the positive contributions of both techniques in BIDS.

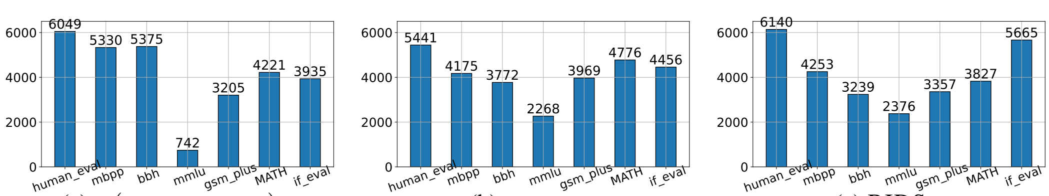
6.1 Ablation

The ablation results are summarized in Table 3. We compare BIDS with the $-$**`Iter`** baseline to ablate iterative selection, and with $-$(**`Norm`** + **`Iter`**) to further ablate both normalization and iterative selection. In other words, $-$(**`Norm`** + **`Iter`**) is the naive **instance-wise max** algorithm applied to the unnormalized Attribution Matrix, and $-$**`Iter`** only additionally applies the instance-level normalization proposed by BIDS to the AM, but without the iterative selection that favors underrepresented tasks. From the table, we observe that normalization alone can already consistently improve the overall performance of selected data under various budgets. And applying the iterative selection not only further elevates the macro-average score, but also improves the balance of cross-task performance. These two observations confirm that both design choices of BIDS contribute positively to the performance gains.

6.2 Changes in Influence Distribution of Selected Data

After confirming the positive contribution from both components of BIDS, we then proceed to explore how they affect the influence distribution of selected data, and whether such effects can provide insight into why BIDS advances balanced learning of diverse capabilities.

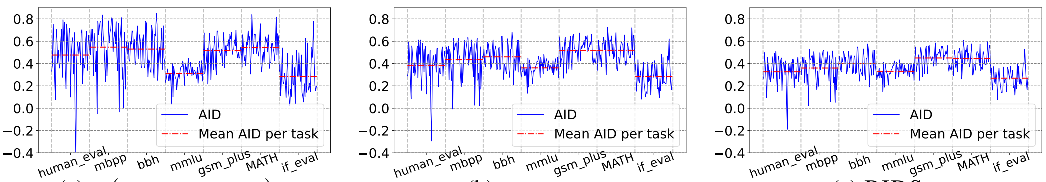
We compare the same models as in §6.1, using a slightly modified version of the two types of data analysis metrics defined in §3. For better AID comparisons, we report influence values after instance-level normalization. We also replace task-wise average influence with instance-wise influence in the THI calculation, since the three algorithms we are comparing are all built upon the instance-wise max approach. Concretely, for each selected training example $\boldsymbol{t}_i$, if its influence on validation instance $\boldsymbol{v}_k$ ($\boldsymbol{v}_k \in \mathcal{V}_j$) is the highest across all $|\mathcal{V}|$ validation instances, then the THI frequency for task j increases by one.



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

Figure 4: Comparative analysis of THI under the 10% budget. Both $-$**`Iter`** and BIDS have more balanced task frequencies compared with $-$(**`Norm`** + **`Iter`**).

Normalization balances THI. Comparing 4a with 4b and 4c, we see that after normalization the task frequency distribution becomes much more balanced. The frequencies for tasks such as MMLU, GSM-Plus, MATH and IFEval all increase by a great extent, while those for BBH and the two coding tasks decrease. This is fairly surprising when compared with the experimental results



(a) $-(\texttt{Norm} + \texttt{Iter})$ (b) $-\texttt{Iter}$ (c) BIDS

Figure 5: Comparative analysis of normalized AID under the 10% budget. From $-(\texttt{Norm} + \texttt{Iter})$ to $-\texttt{Iter}$ to BIDS, the disparity in AID among different tasks and instances gradually diminishes, with both decreasing upper bounds and increasing lower bounds.

in Table 3, where $-\texttt{Iter}$ and BIDS actually show improvements in tasks with both decreased and increased THI frequencies compared with $-(\texttt{Norm} + \texttt{Iter})$. This observation suggests that a balanced selection of influential data may improve data efficiency not only by allocating more budget for capabilities that are underrepresented, but also reducing the redundancy in over-represented capabilities.

Better performance comes with smaller influence discrepancies. The AID results (Figure 5) offer further insights. Moving 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:

- **The maximums of AID decrease.** Despite generally lower influence scores across these evaluation tasks, the performance of BIDS improves consistently compared with the other two ablated baselines. This observation actually reveals a limitation of the first-order linearity assumption by the influence estimation method of LESS: simply selecting high-influence points using a Top-K 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.
- **The minimums of AID increase, especially for validation instances with exceptionally low influence**, such as HumanEval and MBPP. This observation again suggests the effectiveness of one of BIDS' key motivations: improving the model's overall performance by enhancing the capabilities that are most underrepresented in the current selected data.

7 Related Work

Data Selection for Instruction Finetuning. Since the pioneering work LIMA (Zhou et al., 2024) showed that a mere 1000 carefully curated high-quality instruction data can already lead to significant performance improvement, many works have been exploring automatic data selection pipelines guided by different metrics. Quality-guided selection mostly defines the quality for each data point based on natural language indicators (Cao et al., 2023), quality scores from strong evaluators such as GPT-4 (Chen et al., 2023; Parkar et al., 2024), or principled metrics derived from various learning dynamics (Kang et al., 2024; Mekala et al., 2024; Xia et al., 2024; Choe et al., 2024). Diversity-guided methods usually apply clustering algorithms based on certain informative representation of each data point (Yang et al., 2024), and also take inspiration from traditional core-set selection approaches (Das & Khetan, 2023). Both of these dimensions have been proved effective for instruction finetuning of LLMs (Bukharin & Zhao, 2023; Liu et al., 2023), and we remark that our method BIDS considers both quality and diversity metrics by applying an iterative selection algorithm to influence distributions.

Influence Estimation. Influence estimation has long been an important type of data attribution method, which can be classified into gradient-based and retraining-based approaches (Hammoudeh & 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 & Liang, 2017; Pruthi et al., 2020). Retraining-based estimation usually trains a large number of models on different training subsets, and then inspects how their performance changes when a training example is added to these subsets (Ghorbani & Zou, 2019; Ilyas et al., 2022; Park et al., 2023). 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) and instruction tuning (Xia et al., 2024; Liu et al., 2024).

8 Conclusion

In this work, we introduce BIDS, an influence-based instruction tuning data selection algorithm specifically designed for balanced learning of multiple diverse capabilities. Motivated by the observation of an inherent bias in influence across various tasks, BIDS first applies column-wise normalization to the Attribution Matrix that contains pairwise data influence. Together with an iterative selection algorithm favoring underrepresented tasks, BIDS consistently outperforms various selection algorithms as well as full-dataset training with much more balanced performance. Our analysis further provides insight into the properties of an influential dataset with balanced capabilities.

References

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

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A survey on data selection for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=XfHWcNTSHp`. Survey Certification.

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

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

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

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

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

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. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

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

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

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

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

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

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

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

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

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

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

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

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. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into hilbert space. *Contemporary mathematics*, 26:189–206, 1984. URL https://api.semanticscholar.org/CorpusID:117819162.

Feiyang Kang, Hoang Anh Just, Yifan Sun, Himanshu Jahagirdar, Yuanzhi Zhang, Rongxing Du, Anit Kumar Sahu, and Ruoxi Jia. Get more for less: Principled data selection for warming up fine-tuning in llms. *arXiv preprint arXiv:2405.02774*, 2024.

Myeongseob Ko, Feiyang Kang, Weiyan Shi, Ming Jin, Zhou Yu, and Ruoxi Jia. 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*, pp. 26286–26295, 2024.

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

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

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.

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

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. Smaller language models are capable of selecting instruction-tuning training data for larger language models. *arXiv preprint arXiv:2402.10430*, 2024.

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. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

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

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

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

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. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

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

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

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Yu Yang, Siddhartha Mishra, Jeffrey N Chiang, and Baharan Mirzasoleiman. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. *arXiv preprint arXiv:2403.07384*, 2024.

Junjie Oscar Yin and Alexander M Rush. Compute-constrained data selection. *arXiv preprint arXiv:2410.16208*, 2024.

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

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

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

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

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

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

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 & 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.

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 all the three **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 selecting examples with the highest scores, and only differ in the specific definition of s_i.

- **Task-wise Max:** $s_i \triangleq \max_{k=1,\ldots,m}\{\sum_{\boldsymbol{v}_j \in \mathcal{V}_k} \boldsymbol{A}_{ij}\}$.
- **Instance-wise Max:** $s_i \triangleq \max_{j=1,\ldots,|\mathcal{V}|}\{\boldsymbol{A}_{ij}\}$.
- **Sum:** $s_i \triangleq \sum_{j=1}^{|\mathcal{V}|} \boldsymbol{A}_{ij}$.

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[3]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., 2023)** 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[4][5]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

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

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

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

the instance-level normalization, which justifies the use of normal standardization as the normalization method in BIDS.

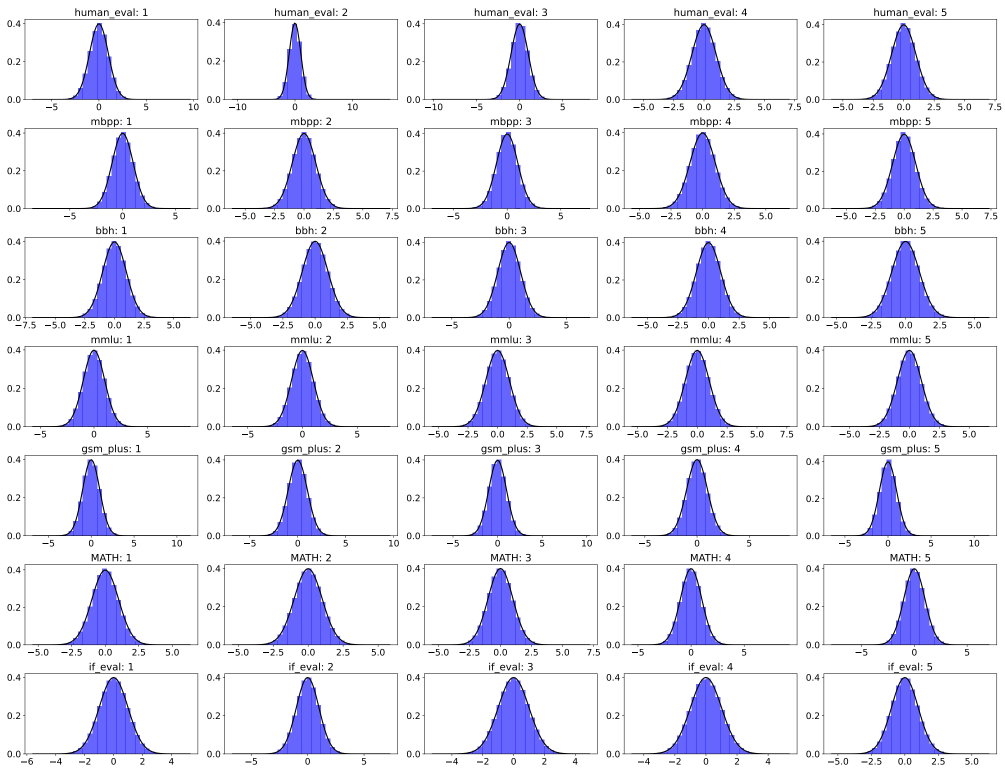


Figure 6: The effect of normal standardization. Five AM columns are sampled for each task. Most of the columns in the AM indeed approximate a standard normal distribution after normal standardization.

A.5 Algorithmic Illustration of the Iterative Selection in BIDS

Algorithm 1 BIDS: Iterative Selection Favoring Underrepresented Tasks

1: **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 $\mathcal{V}$.

2: **Initialization:** $\mathcal{T} = \varnothing$, $\mathcal{D} = \{\boldsymbol{t}_i\}_{i=1}^{|\mathcal{D}|}$

3: **while** $|\mathcal{T}| < B$ **do**

4: $\quad i^* = \underset{i\in\{i|\boldsymbol{t}_i\in\mathcal{D}\setminus\mathcal{T}\}}{\arg\max} \max_{1\leq j\leq|\mathcal{V}|} \{\boldsymbol{A}_{ij} - \frac{1}{|\mathcal{T}|}\sum_{k\in\{k|\boldsymbol{t}_k\in\mathcal{T}\}} \boldsymbol{A}_{kj}\}$

5: $\quad \mathcal{T} = \mathcal{T} \cup \{\boldsymbol{t}_{i^*}\}$

6: **end while**

7: **Return:** $\mathcal{T}$: selected training examples.

Algorithm 1 provides a step-by-step illustration of the iterative selection algorithm in BIDS (§4 and Figure 3). As is shown in line 4, at each iteration, the utility of each candidate example $\boldsymbol{t}_i$ is defined as

$$\Delta^{(i)} \triangleq \max_{1\leq j\leq|\mathcal{V}|} \{\boldsymbol{A}_{ij} - \frac{1}{|\mathcal{T}|}\sum_{k\in\{k|\boldsymbol{t}_k\in\mathcal{T}\}} \boldsymbol{A}_{kj}\}$$

i.e., the largest component of $\boldsymbol{A}_i - \boldsymbol{A}_{\mathcal{T}}$. And the candidate example $\boldsymbol{t}_{i^*}$ with the highest utility $\Delta^{(i^*)}$ is selected for this iteration.

A.6 Results with Different Base Models

In order to further validate the generalizability of BIDS, we compare BIDS with other baseline data selection algorithms using Mistral-7B-v0.3 as the backbone for both selection and training. The results are presented in Table 4. The two algorithms compared here, $-$(**Norm** + **Iter**) and $-$**Iter**, follow the same definition in §6.1. And the random baseline is also the average result of two different random seeds.

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**
	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%	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
	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%	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
	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
	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 4: Additional results when using Mistral-7B-v0.3 as the base model for selection and training. 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. Under all the three budgets, BIDS still maintains the strongest macro-average performance, and ranks either first or second among the five baselines on 5/7, 6/7, 5/7 benchmarks respectively. Also, the performance improvements from $-$(**Norm** + **Iter**) to $-$**Iter** to BIDS are consistent with prior observation with Llama-3-8B in §6.1. Finally, 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.

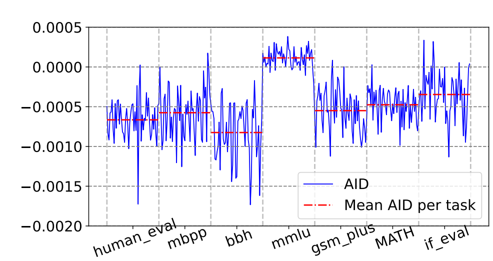


Figure 7: Unnormalized Average Influence Distribution (AID) of the **whole** UltraInteract dataset (Yuan et al., 2024), with the base model being Mistral-7B-v0.3. It still shows great inter-task and intra-task influence scale differences.

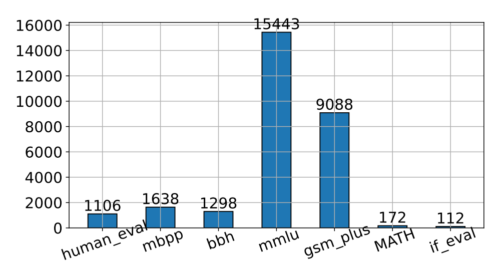
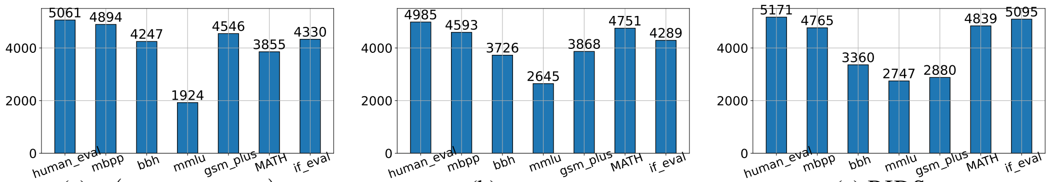


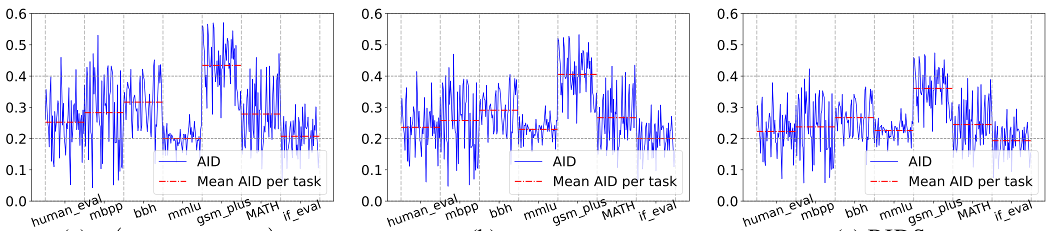
Figure 8: Task frequencies with Highest Influence (THI) of LESS-selected data under the **10%** budget, with the base model being Mistral-7B-v0.3. In this case, MMLU is even more obviously oversampled than prior observation with Llama-3-8B.

Similar to the analysis framework in §3, we also present the AID analysis of the whole UltraInteract dataset (Figure 7) and the THI analysis of LESS-selected data (Figure 8). Then we follow the workflow in §6.2 to present both THI and AID analyses for the three progressive algorithms: $-$(**Norm** + **Iter**), $-$**Iter** and BIDS (Figure 9, 10). The only difference here is that the selection model is Mistral-7B-v0.3 instead of Llama-3-8B.



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

Figure 9: Comparative analysis of 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**$)$.



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

Figure 10: 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.

A.7 Discussion on the Computational Cost of BIDS

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 288 K (Yuan et al., 2024), and the $|V|$ dimension is 350. Therefore, the memory cost for storing the AM using FP 64 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 K (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.8 Qualitative Analysis

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

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 responses.

Concretely, we present three sets of model responses in the task of coding (Table 5), math (Table 6) and general instruction-following (Table 7) 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 5: 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

```python
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

```python
# 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

```python
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

```python
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

```python
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

```python
# 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 6: 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 7: 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?