Behavior Injection: Preparing Language Models for Reinforcement Learning

Zhepeng Cen[*1], **Yihang Yao**[*1], **William Han**[1], **Zuxin Liu**[2], **Ding Zhao**[1]
[1] Carnegie Mellon University, [2] Salesforce AI Research
[*] Equal contribution, {zcen, yihangya}@andrew.cmu.edu

Abstract

Reinforcement learning (RL) has emerged as a powerful post-training technique to incentivize the reasoning ability of large language models (LLMs). However, LLMs can respond very inconsistently to RL finetuning: some show substantial performance gains, while others plateau or even degrade. To understand this divergence, we analyze the per-step influence of the RL objective and identify two key conditions for effective post-training: (1) RL-informative rollout accuracy, and (2) strong data co-influence, which quantifies how much the training data affects performance on other samples. Guided by these insights, we propose behavior injection, a task-agnostic data augmentation scheme applied prior to RL. Behavior injection enriches the supervised finetuning (SFT) data by seeding exploratory and exploitative behaviors, effectively making the model more RL-ready. We evaluate our method across two reasoning benchmarks with multiple base models. The results demonstrate that our theoretically motivated augmentation can significantly increase the performance gain from RL over the pre-RL model. Website: https://bridge-llm-reasoning.github.io/.

1 Introduction

Large language models (LLMs) have demonstrated remarkable reasoning capabilities and strong performance across a broad range of tasks [1, 2, 3], including mathematical problem solving [4, 5, 6], code generation [7, 8], and embodied decision-making [9, 10]. When guided by chain-of-thought (CoT) prompting [11], LLMs are able to generate intermediate reasoning steps, leading to more structured and interpretable outputs. Despite these advances, LLMs still struggle with complex, multi-step reasoning tasks, such as mathematical competitions [12] and real-world agentic scenarios [13, 14, 15]. A common strategy for improving performance is to scale up training corpora [16, 17, 18, 19]. However, concerns have been raised that high-quality pretraining data may soon be exhausted [20], and continued data scaling introduces significant computational overhead.

An alternative and increasingly prominent direction is to enhance LLM reasoning through reinforcement learning (RL) [21, 22], which enables reward-driven fine-tuning based on verifiable outcomes, ranging from ground-truth correctness [23, 24] to feedback from executable environments [25, 14, 15]. In traditional low-dimensional RL, performance gains are typically attributed to algorithmic improvements [26, 27] and data quality [28, 29, 30], with little attention paid to initialization since models are often trained from scratch. In contrast, the RL process for LLMs begins with a pretrained or fine-tuned model, and we argue that the quality of this initialization, particularly after supervised fine-tuning (SFT), plays a critical role in determining the effectiveness of subsequent RL. In this work, we focus on the *warm-start RL training pipeline* [23], which first fine-tunes an LLM via SFT before applying reinforcement learning. Our goal is to make language models *RL-ready*: rather than modifying RL algorithms themselves, we propose a data-centric strategy to enhance the base model's ability to benefit from RL, thereby improving training efficiency and downstream performance.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

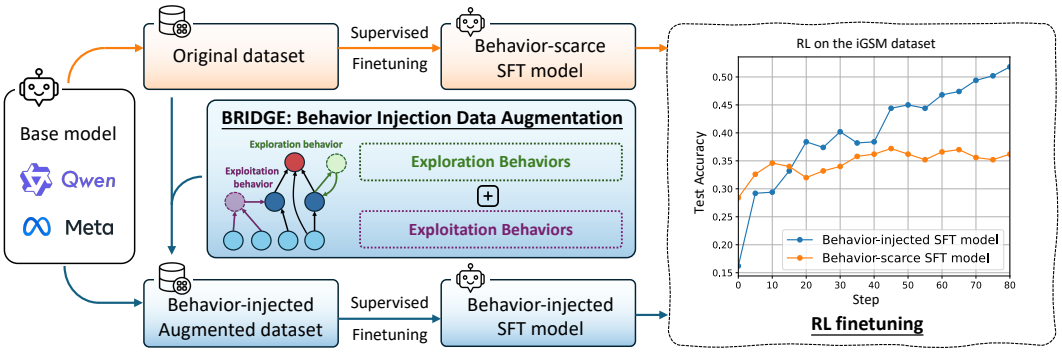

Figure 1: Overview of the BRIDGE pipeline: We augment the SFT data by introducing exploration and exploitation behaviors to prepare LLMs ready for RL finetuning.

Our key insights stem from two perspectives: (1) Analysis of the RL learning objective: we identify two critical factors that influence model improvement during RL tuning: *rollout accuracy* and the *data co-influence coefficient*, which quantifies how strongly RL training data affects generalization to the target domain. (2) Desirable behaviors for RL: while *exploration* and *exploitation* are central in low-dimensional RL [31], they are often underexplored in the context of LLM post-training. Motivated by these findings, we introduce **BRIDGE** (**B**ehavio**R** **I**njection **D**ata au**G**m**E**ntation), a data-centric augmentation strategy applied during the SFT stage. BRIDGE injects desired behaviors into the model before RL, enabling it to generate more informative trajectories during RL rollout and leading to greater final performance improvements. Our contributions are summarized as:

1. In-depth analysis of LLM reinforcement learning. We provide a detailed examination of the RL training process, highlighting two key factors that drive learning efficiency: rollout accuracy distribution and the data co-influence coefficient.

2. Introduction of the BRIDGE augmentation algorithm. We propose BRIDGE, which prepares the model for RL by explicitly injecting exploration and exploitation behaviors during SFT.

3. Comprehensive empirical evaluation. We evaluate BRIDGE across diverse tasks from iGSM and PromptBench. Extensive experiments and ablation studies demonstrate that BRIDGE enhances data co-influence and significantly improves performance in the RL stage.

2 Related Work

RL-based post-training for LLMs. Reinforcement learning (RL) has become a central post-training approach for aligning and extending large language models. Large-scale efforts such as OpenAI-o1 [32] and DeepSeek-R1 [33] illustrated the gains obtainable from reward optimization on general-purpose models. Since then, RL fine-tuning has been pushed into a variety of domain-specialized settings. In mathematics, verifier-guided or programmatically graded rewards help models master challenging problems [34, 35, 24, 36, 37, 38], while logic benchmarks likewise benefit from RL-driven reasoning refinement [39, 40]. Interactive agents leverage RL for textual tool use and multi-step planning [41, 42, 43, 44, 45, 46], mobile-app control [25], device manipulation [47, 48], and web navigation [49]. Additional applications include medical visual QA [50], software-engineering assistance [51], social reasoning [52], and tool-centric instruction following [53].

Analysis of LLM finetuning. To understand how model performance changes during finetuning, researchers study SFT learning dynamics of LLM [54] in terms of data influence [55, 56] or likelihood analysis [57, 58]. In the context of RL finetuning, OpenAI o1 [32] showed that RL significantly improve reasoning by encouraging the generation of longer CoT. Subsequent studies [59, 60, 61, 62] validate this effect and show that RL enables inference-time scaling by favoring more expressive reasoning traces. This finding is aligned with theoretical analyses that characterize the expressivity of CoT [63, 64, 65, 66]. Furthermore, researchers [67, 68, 69] observe that RL fine-tuning often amplifies behaviors already accessible in the base model rather than introducing entirely new ones, which are crucial cognitive operations to performance growth in RL [70]. Distinct from these works, we investigate the tuning dynamics by RL learning objective, and we identify behaviors from the perspective of exploration and exploitation to prepare models for RL tuning.

Data-centric approaches for LLMs. A complementary line of work improves language models not by changing the training algorithm but by enriching the data they see. In supervised fine-tuning (SFT), targeted augmentation has proved especially fruitful: synthetic derivations, curriculum sampling, or structure-preserving rewrites boost mathematical reasoning [16, 71, 72, 73, 74, 4] and extend to code generation domains [75]. Beyond manual augmentation, agentic pipelines automate data acquisition, where LLM-based agents write queries, run tools, and self-filter their outputs, yielding high-quality instruction data at a large scale [76, 19, 77, 78, 79, 80]. For RL settings, several studies curate *seed* datasets whose answers can be programmatically verified, seeding reward-based training with reliable trajectories [81, 14]. Others examine what kinds of pre-RL corpora make a base model more amenable to later policy optimization, highlighting the importance of coverage, diversity, and error profiles [62, 70]. Our work aligns with this data-centric perspective but focuses on *behavior-level* augmentations that explicitly *make models RL-ready*, rather than merely enlarging or filtering the SFT corpus.

3 Method

In this section, we first investigate key factors that affect the model performance growth in the RL stage, and then introduce our method based on the analysis to make the pre-RL model "RL-ready", i.e., being able to boost the performance after the RL stage.

3.1 Preliminaries

Following the previous pipeline [23], we apply SFT to the base models, followed by an RL stage. **Supervised finetuning (SFT)**. SFT is a common practice to initialize LLM finetuning by a demonstration dataset $\mathcal{D}_{\text{SFT}}$, which trains the policy π_θ by minimizing the negative log-likelihood:

$$\min_\theta \mathcal{L}_{\text{SFT}}(\mathcal{D};\theta) = -\mathbb{E}_{(\mathbf{q},\mathbf{a})\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{T}\log \pi_\theta\left(a_t \mid \mathbf{q}, \mathbf{a}^{(<t)}\right)\right], \tag{1}$$

where $\mathbf{q} = [q_1, \ldots, q_L], \mathbf{a} = [a_1, \ldots, a_T]$ are demonstration query and answer from SFT dataset. For simplicity, we use SFT model to refer to the model after SFT training.

RL finetuning. The objective of RL for LLM is to maximize the expectation of reward on a query set Q:

$$\max_\theta \mathcal{J}_{\text{RL}}(Q;\theta) = \mathbb{E}_{\mathbf{q}\sim Q, \mathbf{o}\sim\pi_\theta(\cdot|\mathbf{q})}\left[\sum_{t=1}^{T}\gamma^t r\left(\mathbf{q}, \mathbf{o}^{(\leq t)}\right)\right], \tag{2}$$

where $\mathbf{q} = [q_1, \ldots, q_L], \mathbf{o} = [o_1, \ldots, o_T]$ indicate the query and output respectively, $r(\cdot,\cdot)$ is a reward function, and γ is the discount factor. We mainly consider the rule-based outcome reward, i.e., a binary reward on the last token of output:

$$r(\mathbf{q}, \mathbf{o}) = \mathbf{1}(y(\mathbf{o}) = y_{\text{gold}}),$$

where $y(o)$ is the final answer of the output and y_{gold} is the ground-truth answer of the query. In this paper, we adopt GRPO [23] as the RL algorithm, which samples N outputs $\{\mathbf{o}_i\}_{i=1}^N$ for each query and optimizes a surrogate objective:

$$\begin{aligned}&\mathcal{J}(Q;\theta) = \mathbb{E}_{\mathbf{q}\sim Q, \{\mathbf{o}_i\}\sim\pi_\theta(\cdot|\mathbf{q})}\\ &\frac{1}{N}\sum_{i=1}^{N}\left[\min\left\{\frac{\pi_\theta(\mathbf{o}_i|\mathbf{q})}{\pi_{\text{old}}(\mathbf{o}_i|\mathbf{q})}A_i, \text{clip}\left(\frac{\pi_\theta(\mathbf{o}_i|\mathbf{q})}{\pi_{\text{old}}(\mathbf{o}_i|\mathbf{q})}, 1-\epsilon, 1+\epsilon\right)A_i\right\} - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right],\end{aligned} \tag{3}$$

where ϵ is the clip ratio, β is the KL regularization coefficient, and A_i is the advantage computed by normalizing the reward in each group $A_i \doteq (r_i - \text{mean}(\{r_i\}_{i=1}^N))/\text{std}(\{r_i\}_{i=1}^N)$.

3.2 Training Per-step Influence in RL Finetuning

To understand why LLMs respond to RL divergently, we leverage **per-step influence** [55, 82] to answer the question *how the model performance changes after one RL training step*.

Consider a language model policy π_θ, suppose we update it on one query-output group $(\mathbf{q}, \{\mathbf{o}_i\}_{i=1}^N)$ with learning rate η, then the parameter update is $\Delta\theta = \eta\nabla_\theta\mathcal{J}(\mathbf{q};\theta)$ since the RL objective is to maximize $\mathcal{J}$. According to Taylor expansion, the caused model performance change on other query $\mathbf{q}'$ can be written as

$$\Delta\mathcal{J}(\mathbf{q}';\theta) = \mathcal{J}(\mathbf{q}';\theta+\Delta\theta) - \Delta\mathcal{J}(\mathbf{q}';\theta) = \langle\nabla_\theta\mathcal{J}(\mathbf{q}';\theta), \Delta\theta\rangle + \mathcal{O}(\eta^2).$$

With a sufficiently small learning rate, the per-step influence of $\mathbf{q}$ on the model performance is

$$\Delta\mathcal{J}(Q;\theta) \approx \eta\langle\nabla_\theta\mathcal{J}(Q;\theta), \nabla_\theta\mathcal{J}(\mathbf{q};\theta)\rangle. \tag{4}$$

Then we can derive the per-step influence for the GRPO objective as follows.

Proposition 3.1. *Suppose there are n correct outputs in the sampled group with size N, denote the correct and incorrect outputs as $\{\mathbf{o}_{i+}\}_{i=1}^n$ and $\{\mathbf{o}_{j-}\}_{j=1}^{N-n}$ respectively. When RL training on $\mathbf{q}$ is strictly on policy, with a sufficiently small β, the per-step influence of $\mathbf{q}$ on the model performance is*

$$\Delta\mathcal{J}(Q;\theta) = \eta\mathbb{E}_{\mathbf{q}'\sim Q,\mathbf{o}'\sim\pi_\theta(\cdot|\mathbf{q}')}\sqrt{\alpha(1-\alpha)} \\ A(\mathbf{q}',\mathbf{o}')\cdot\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{K}_\theta[(\mathbf{q}',\mathbf{o}'),(\mathbf{q},\mathbf{o}_{i+})] - \frac{1}{N-n}\sum_{j=1}^{N-n}\mathcal{K}_\theta[(\mathbf{q}',\mathbf{o}'),(\mathbf{q},\mathbf{o}_{j-})]\right], \tag{5}$$

where $\alpha = n/N$ indicates the accuracy rate for the rollout samples, $\mathcal{K}_\theta((\mathbf{q}',\mathbf{o}'),(\mathbf{q},\mathbf{o})) = \langle\nabla_\theta\log\pi_\theta(\mathbf{o}'|\mathbf{q}'), \nabla_\theta\log\pi_\theta(\mathbf{o}|\mathbf{q})\rangle$ indicates the influence between the log-likelihood of query-output samples $(\mathbf{q}',\mathbf{o}')$ and $(\mathbf{q},\mathbf{o})$.

The proof and discussion are in Appendix A.1. The derivation is mainly based on two assumptions, the on-policy training and a small KL coefficient β, both of which hold in practical implementation. Note that this proposition can also be extended to other RL algorithms by replacing the advantage function. More discussion is provided in Appendix A.2.

This proposition shows that the per-step influence is mainly determined by two factors: (1) the accuracy rate of sampled output, α, and (2) the co-influence coefficient $\mathcal{K}_\theta$ between training data $(\mathbf{q},\mathbf{o})$ and target data $(\mathbf{q}',\mathbf{o}')$. For the first factor, 0% or 100% rollout accuracy makes the coefficient $\sqrt{\alpha(1-\alpha)} = 0$, leading to $\Delta\mathcal{J} = 0$, while a medium accuracy can amplify the influence, which aligns with previous theoretical results on reward variance [83] and practical observation [84]. The second factor, expressed through $\mathcal{K}_\theta[\cdot,\cdot]$, quantifies how strongly each training sample affects performance change on the target domain. Take a correct target sample (i.e., $A(\mathbf{q}',\mathbf{o}') > 0$) for example, we expect a large positive $\mathcal{K}_\theta$ when the update draws on correct samples $(\mathbf{q},\mathbf{o}_{i+})$, enabling the language model to learn more from this successful path and gain greater improvement in the RL step. Conversely, when the update uses incorrect samples $(\mathbf{q},\mathbf{o}_{j-})$, we expect $\mathcal{K}_\theta$ for the same correct target sample to be small or negative, allowing the model to learn from the failure path. We defer the detailed explanation of the validation method to examine this factor to section 4.3.

Section 3.2 takeaway

There are two key strategies to enhance performance growth during the RL stage of LLMs:
- Altering the accuracy distribution to increase the information coefficient $\sqrt{\alpha(1-\alpha)}$.
- Shaping the co-influence coefficient of model, governed by $\mathcal{K}_\theta$, to improve knowledge acquisition from both successful and failed rollout experiences.

3.3 BRIDGE: Behavior Injection Data Augmentation

Based on the above analysis, the next question is how to improve performance gains during RL in practical implementations. A straightforward approach is to perform rejection sampling on queries from low-information accuracy regions, i.e., those with excessively high or low α. However, this method has three key drawbacks: (1) Increased computational cost: it requires pre-sampling to identify and discard low-information samples; (2) Distortion of the query distribution: it induces distribution shift issues in optimization; and (3) Limited impact on data co-influence: this approach does little to affect the data co-influence factors involving $\mathcal{K}_\theta$.

Rather than directly reshaping the RL training distribution, we focus on the characteristics of the base model, which fundamentally influence both the accuracy distribution and the co-influence

structure. In particular, we target two essential capabilities of RL agents: *exploration* and *exploitation*. Exploration encourages the model to traverse a broader range of observations and actions, helping it avoid suboptimal local modes. Exploitation enhances the model's ability to leverage its current knowledge for effective decision-making, which can in turn increase the co-influence of training data during RL. By injecting these two behaviors into the base LLMs, we promote more effective learning dynamics and greater performance gains in RL training.

We implement behavior injection through our proposed method, BRIDGE (**B**ehavio**R** **I**njection **D**ata au**G**m**E**ntation), as illustrated in Figure 1. Starting from a vanilla chain-of-thought (CoT) dataset for instruction tuning, we augment the dataset by injecting exploration and exploitation behaviors. We then perform supervised fine-tuning (SFT) on the augmented dataset, followed by reinforcement fine-tuning (RL).

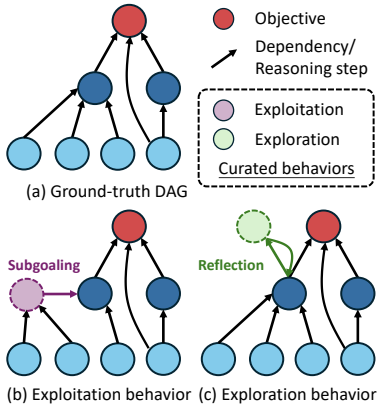

Figure 2: DAG representation of behaviors.

Algorithm 1: BRIDGE

Input: Vanilla QA pairs (q, a), injection prob p
Output: Injected QA pairs (q, a')

Extract DAG $\mathcal{G} = (V, E)$ from (q, a).
Construct exploration behaviors.
Obtain a locked node n_l ahead.
b_1 = "Let's solve [n_l], ..., wait, [n_l] seems to be not solvable yet, let's get back."
Construct exploitation behaviors.
Aggregate all the information `info` to solve an unlocked but not solved node n_u
b_2 = "Let's solve [n_u], ..., since [`info`], [n_u] =[subgoal computation]=[n_u.value]"
$a' \leftarrow a + b_1 + b_2$; # Inject with prob p.
Return: Injected QA pair (q, a')

To further ground our approach in a practical implementation, we adopt a directed acyclic graph (DAG) representation for reasoning tasks [85], as shown in Figure 2. Formally, a reasoning task is represented as $\mathcal{G} = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ denotes the directed edges that capture dependencies and relationships between nodes. Given initial information and inter-node dependencies, the goal is to compute intermediate node results step by step and ultimately derive the final answer at the target node. The ground-truth reasoning chain corresponds to a topological sort of the DAG. Notably, many reasoning and agentic tasks, such as math reasoning [86, 87, 88] and logical reasoning [86, 73], naturally conform to this graph-based representation [77].

Our BRIDGE approach is detailed in Algorithm 1. We first extract the DAG representation from the original query and its corresponding CoT response. Note that such extraction can be achieved by string matching for structured CoT or an oracle LLM for unstructured CoT. More discussion is in the Appendix B.2. After constructing the behaviors, we generate *exploration behaviors* by attempting to solve a locked (i.e., not yet solvable) node ahead of time, followed by reflection. For *exploitation behaviors*, we actively aggregate available information to solve an unlocked node (i.e., solvable but not yet solved) or compute a sub-goal instead of directly reaching the node value, reinforcing the known reasoning path. After generating these behaviors, we inject them into the vanilla CoT corpus, resulting in an augmented dataset.

4 Experiment

4.1 Experiment settings

Tasks. To evaluate the performance growth during RL finetuning, we conduct experiments on two benchmarks: 1) iGSM [74], a grade-school math problem benchmark involving math and common sense reasoning tasks; 2) PromptBench [86], a benchmark involving arithematic and logical reasoning tasks. Note that the difficulty of the tasks in both benchmarks are controllable: iGSM controls the difficulty of the query by the *number of operations* needed to reach the final answer while PromptBench controls it by *reasoning depth* and *number of redundant premises* (i.e., the redundant edges in DAG) in query. We adopt the strict match accuracy as the evaluation metrics for both tasks.

In practice, we apply slight modifications on iGSM to improve the finetuning stability, e.g., removing the modulo operation because we observe that the base models are unable to calculate the modulo with high accuracy and it cannot be significantly enhanced by training on a small SFT dataset [74]. See Appendix B.1 for more details of the tasks.

We select these two benchmarks for analysis because we want to test the problem-solving abilities of LLMs rather than the knowledge storage. While other tasks where domain knowledge can confound the evaluation of reasoning capabilities, the purely synthetic data in iGSM and PromptBench avoid the data contamination issue fundamentally [74, 89]. Although built upon relatively small semantics domains, these tasks require multi-dimensional abilities of LLM such as basic concept understanding, search and planning, and basic arithmetic calculation, which mirrors the key aspects of problem-solving in general reasoning tasks.

Injected behaviors. We adopt two exploitative behaviors, *subgoal computation* and *information analysis*, and one exploratory behavior, *reflection*, respectively. Specifically, *subgoal computation* refers to computing intermediate results for complex equations; *information analysis* involves aggregating relevant information when deriving the equation for a node; *reflection* means actively attempting to solve an unlocked node and then revisiting back. We add the *subgoal computation* to each step of CoT but inject the *analysis* and *reflection* with probability $p = 0.1$. Detailed examples of behaviors in iGSM and PromptBench tasks are illustrated in Appendix B.3.

Baselines. We compare BRIDGE with several data augmentation baselines: 1) `Vanilla`, which uses original SFT data without augmentation; 2) premise permutation augmentation (`PP-Aug`) [90, 73], which augments SFT data by randomly shuffling the premises in the query to improve the reasoning consistency; 3) reasoning chain augmentation (`RC-Aug`) [4], which is implemented by generating new answers with different topological orders for the same query from SFT dataset.

Other experiment settings. We use two families of base models, Qwen-2.5 [91] and Llama-3.2 [92], to validate the effectiveness of our methods. As there is a large domain gap between the pretraining corpus and evaluation tasks, we first train base LLMs on a SFT dataset to expose them to the query set and demonstration answers. Then we use GRPO with the same settings to finetune the SFT models for different augmentation methods. More training details are attached in Appendix B.4.

4.2 Main results

We present the performance comparison on iGSM and PromptBench in Table 1 and 2 respectively.

For the iGSM task, we use data with $15 \sim 20$ operations for finetuning. In SFT stage, the vanilla dataset consists of 2000 data while `PP-Aug` and `RC-Aug` augments dataset size to 8000. BRIDGE also uses 2000 data for SFT but augments them by injecting behaviors. We train each model on corresponding dataset for 5 epochs. In the RL stage, we train on the data with the same difficulty. To avoid overfitting (e.g., the memorization in SFT [93, 39]) and evaluate the generalizability of the finetuned models, we test their performance on two problem sets separately: 1) in-distribution

Table 1: The evaluation results (%) on the iGSM task. We train SFT models for 5 epochs and finetune them with RL for 100 (Qwen 3B) or 200 steps (Qwen 1.5B and Llama 1B). We compare SFT and RL models on both in-distribution and out-of-distribution problem sets. Δ denotes the performance improvement by RL over SFT.

Base		Vanilla		PP-Aug		RC-Aug		BRIDGE (Ours)	
		In-Dist	OOD	In-Dist	OOD	In-Dist	OOD	In-Dist	OOD
Qwen-1.5B	SFT	40.2	29.0	46.6	33.4	33.2	27.4	44.8	34.2
	RL	46.2	36.4	60.4	48.0	46.2	39.6	**91.4**	**83.0**
	Δ	6.0	7.4	13.8	14.6	13.0	12.2	**46.6**	**48.8**
Qwen-3B	SFT	38.0	28.4	38.0	19.8	45.6	32.8	59.2	45.8
	RL	57.2	47.4	62.4	52.0	60.6	45.6	**89.6**	**83.6**
	Δ	19.2	19.0	24.4	32.2	15.0	12.8	**30.4**	**37.8**
Llama-1B	SFT	29.6	19.0	32.4	23.2	31.0	21.4	40.4	26.6
	RL	31.0	24.0	39.0	26.4	33.6	23.4	**64.6**	**44.8**
	Δ	1.4	5.0	6.6	3.2	2.6	2.0	**24.2**	**18.2**

Table 2: The evaluation results (%) on the PromptBench arithmetic reasoning task. We train SFT models for 2 epochs and finetune them with RL for 100 steps. We compare performances on both in-distribution and OOD problem sets. Δ denotes the performance improvement by RL over SFT. **Bold** means the best performance.

Base		Vanilla		PP-Aug		RC-Aug		BRIDGE (Ours)	
		In-Dist	OOD	In-Dist	OOD	In-Dist	OOD	In-Dist	OOD
Qwen-1.5B	SFT	13.6	9.4	16.8	8.8	18.6	13.0	2.2	0.8
	RL	37.8	29.0	41.2	31.2	34.2	21.6	**55.2**	**41.0**
	Δ	24.2	19.6	24.4	22.4	15.6	8.6	**53.0**	**40.2**
Qwen-3B	SFT	31.2	18.0	40.6	29.4	35.2	19.4	44.4	31.0
	RL	50.0	34.6	66.0	50.8	64.0	43.4	**85.8**	**70.0**
	Δ	18.8	16.6	25.4	21.4	28.8	24.0	**41.4**	**39.0**
Llama-1B	SFT	12.6	5.4	11.2	8.8	9.2	4.4	6.0	3.6
	RL	27.8	18.8	28.6	19.0	25.2	17.4	**47.6**	**39.8**
	Δ	15.2	13.4	17.4	10.2	16.0	13.0	**41.6**	**36.2**

(In-Dist) set with operation number $= 20$ and 2) out-of-distribution (OOD) set with operation number $= 25$. Each set consists of 500 problems. We use greedy sampling and compute the accuracy when evaluating on the test sets.

For the PromptBench task, we use data with reasoning depth $= 4$ and 5 for SFT and RL training respectively, both of which have $0 \sim 8$ redundant premises. This simulates the real-world settings that we have labeled CoT answers for easy tasks and only have verifiers for harder tasks where the demonstration annotation is absent. In SFT stage, we train the models on 5000 data (`PP-Aug` and `RC-Aug` augment it to 10000) for 2 epochs. In RL stage, we train the models for 100 RL steps. We also test the performances on 1) in-distribution set with reasoning depth $= 5$ and $0 \sim 8$ redundancy and 2) out-of-distribution set with reasoning depth $= 5$ and $20 \sim 22$ redundancy separately.

In iGSM task, most augmentation methods with different base models increase the SFT accuracy on in-distribution set with models while the improvement on OOD set is less prominent. For RL, BRIDGE achieves both the highest performance growth and best accuracy on in-distribution and OOD sets. In PromptBench task, although BRIDGE has relatively lower pre-RL accuracy, it obtains significantly more remarkable performance growth than baselines during RL and thus achieves the best final score on both in-distribution and OOD sets. The training curves and more training results are in Appendix B.5 where we also present a result of comparison between BRIDGE and accuracy-based rejection sampling.

4.3 Rollout accuracy and Co-influence of Different SFT models

To study how different augmentations lead to divergent RL performances, we test the rollout accuracy α distribution and the finetuning per-step influence $\Delta\mathcal{J}$ of the SFT models on iGSM task. The results are illustrated in Fig 3.

Specifically, we use the SFT model to rollout 8 samples on a iGSM test set consisting of 2000 data with the same difficulty (operation number $= 15 \sim 20$) as the RL training set. We compute the advantages of each answer (by normalizing the reward within the group) and the sampling accuracy distribution of each model based on the rollout results. When measuring the data co-influence $\mathcal{K}_\theta$, the directly computation requires taking the inner product between two gradients with the same size as the model parameters, which is intractable to scale to a large dataset. Instead, we adopt a low-rank approximation [82] that first estimates the LoRA [94] gradients of SFT model and then applies random projection to further reduce the dimensionality. Based on the advantages and the approximated data co-influence $\tilde{\mathcal{K}}$, we can compute the approximated per-step influence of one query-output group on other data $\Delta\tilde{\mathcal{J}}$ by Eq.(5). More implementation details are in Appendix B.7.

We present the sampling accuracy, approximated per-step influence and training curves of different SFT models in Figure 3. In both settings, BRIDGE has the fastest reward growth during the training, which can be explained in terms of both sampling accuracy and data co-influence in finetuning. All of three augmentations increase the accuracy of SFT model, but `RC-Aug` achieves it mainly by more fully correct samples (acc=1). On the contrary, BRIDGE has the largest ratio of samples with

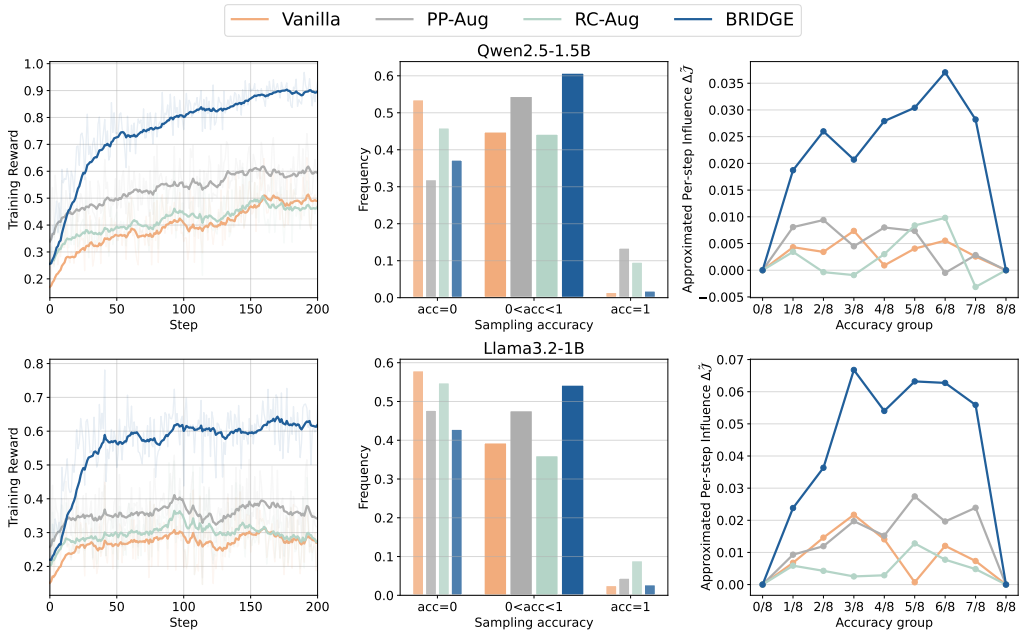

Figure 3: **Left:** Training curve, **Middle:** SFT model rollout accuracy distribution, **Right:**. Per-step influence visualization. The top and bottom plots correspond to the results of Qwen2.5-1B and Llama3.2-1B respectively. We group the approximated per-step influence for samples with different accuracy in right plots, where the influences of samples with all correct or all wrong answers are 0.

medium accuracy ($0 <$acc< 1) that contribute to model improving during RL training, even larger than `PP-Aug` which has better performance at the beginning of RL. Meanwhile, the right plots clearly show the discrepancy among different methods: while `PP-Aug` and `RC-Aug` models have similar per-step influence with vanilla SFT model, our method significantly improves it by proper behavior injection into the SFT data, and thus better prepares the model for RL.

Section 4.2 and 4.3 takeaway

BRIDGE demonstrates superior capability to boost RL performance (Table 1, 2), with:
- Injected behaviors improve the per-step influence $\Delta\mathcal{J}(Q;\theta)$ in Eq.(5). (Figure 3 **Right**).
- Injected behaviors shape the accuracy distribution of rollout samples (Figure 3 **Middle**).

4.4 Ablations of Injected Behaviors

As we add various behaviors to the demonstration dataset to prepare the model for RL, it remains unclear how each behavior contributes to the final improvement. Therefore, we run an ablation by adding only one behavior and present the results in Figure 4.

For Qwen2.5-1.5B models, the *subgoal computation* increases the SFT model accuracy and all three behaviors improve the performance gain in RL, consistent with their per-step influences. For Llama3.2-1B, the models with single behavior still outperform the baseline but the performance gain in RL is marginal. Meanwhile, we observe that there is a large accuracy gap between the model trained by BRIDGE and model with any single behavior after RL finetuning, suggesting the necessity of behavior combination for the Llama model.

We also present ablation studies on the injection probability of behaviors for the iGSM task in Figure 5. Comparing $p = 0$ with $p > 0$, we observe that the injected *reflection* and *analysis* behaviors consistently enhances the final performance. Furthermore, performance differences across various non-zero injection probabilities are nuanced. This suggests that the exact behavior ratio is not critical, and LLMs can adaptively learn to utilize these behaviors during RL as long as the behavior is injected into the model.

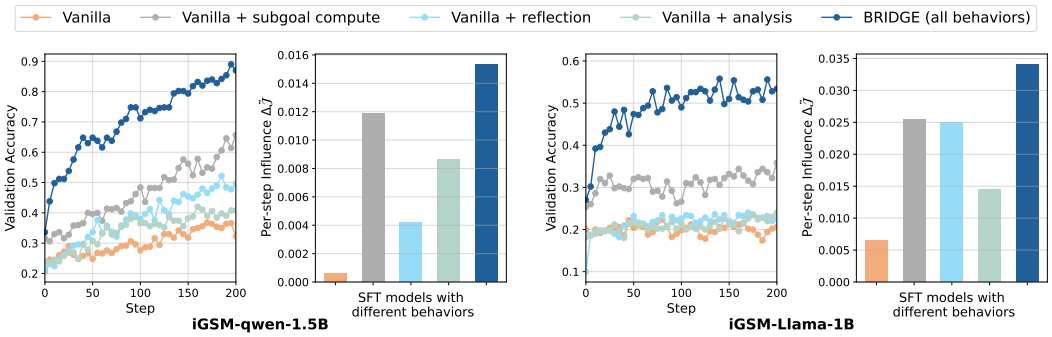

Figure 4: The ablation of models with different behaviors in iGSM task. We present the validation accuracy curves as the RL finetuning performances, where the validation set consists of 500 queries with $21 \sim 25$ operations. We also compare the average per-step influence of the SFT models with different behaviors.

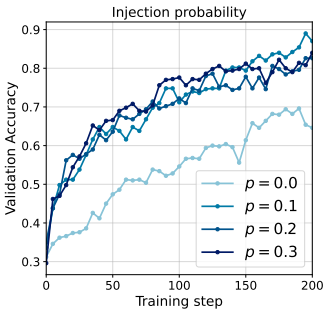

Figure 5: The ablations on behavior injection probability.

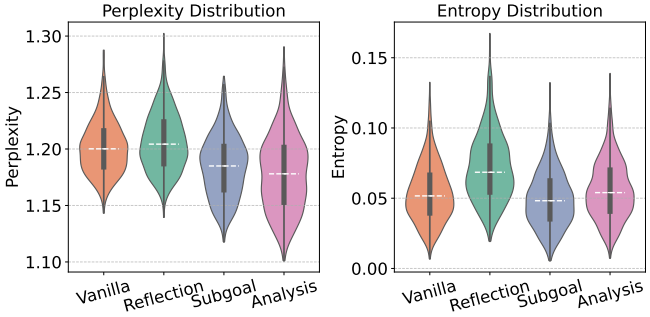

Figure 6: The perplexity and entropy of the SFT models.

4.5 Exploration and Exploitation Effects of Injected Behaviors

In this section, we study the effects of behaviors on exploration and exploitation. We leverage policy entropy and perplexity as the corresponding metrics: we first train the model on datasets with the same size for the same training steps. Then we rollout the SFT models with temperature $= 1$ (the same as rollout in RL finetuning) and compute the policy entropy on model's generation; we also use the log-likelihood on corresponding SFT dataset to compute the perplexity. The results are presented in Figure 6.

While all above injected behaviors contribute to the preparation of RL finetuning, their effects on RL work from different aspects: *subgoal computation* and *analysis* reduce the perplexity on demonstration data and enable the models to generate the ground-truth tokens with higher probability, suggesting these behaviors facilitate the exploitation of LLM. Meanwhile, the *reflection* behavior leads to a larger policy entropy in generation, validating its effectiveness in encouraging LLMs to explore. In BRIDGE, we inject both types of behaviors to LLMs to enhance both exploration and exploitation of LLMs, making them significantly more "RL-ready" compared to the vanilla models.

Section 4.4 and 4.5 takeaway

The injected behaviors can either improve exploration or exploitation capability, specifically,
- Behavior injection is not sensitive to the behavior density, as long as behaviors are successfully injected into the pre-RL model. (Figure 5)
- Both exploration and exploitation behaviors injected through BRIDGE are effective to improving per-step influence $\Delta\mathcal{J}(Q;\theta)$ in Eq.(5), thus benefiting RL. (Figure 4)
- Reflection improves exploration while subgoal computation and analysis improve exploitation, as examined by model entropy and perplexity. (Figure 6)

5 Conclusion

In this paper, we investigate why different language models manifest divergent performances during RL finetuning by analyzing the per-step influence of data for GRPO algorithm. We then propose to inject behaviors, which are RL-favorable in terms of exploration and exploitation, into the SFT dataset to prepare LLMs for RL finetuning. Through various experiments across different benchmarks, we demonstrate that our method effectively makes the models RL-ready and significantly outperforms other augmentation baselines.

One limitation of BRIDGE is that the evaluation is focused on the math and common-sense reasoning tasks. The future work involves extending our method to broader agentic domains with a larger dataset size. Meanwhile, we believe the analysis tools (e.g., per-step influence computation) can inspire future work on data curation and behavior discovery in the community. One potential negative social impact of this paper is the misuse of our method, and unsafe behavior injection can lead to harmful consequences.

References

[1] 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.

[2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[5] 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.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] 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.

[8] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[9] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, et al. Magma: A foundation model for multimodal ai agents. *arXiv preprint arXiv:2502.13130*, 2025.

[10] Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. Tail: Task-specific adapters for imitation learning with large pretrained models. *arXiv preprint arXiv:2310.05905*, 2023.

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

[12] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[13] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[14] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*, 2024.

[15] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

[16] Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774, 2024.

[17] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

[18] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024.

[19] Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482, 2024.

[20] Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of llm scaling based on human-generated data. In *Forty-first International Conference on Machine Learning*, 2024.

[21] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[22] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[24] Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*, 2025.

[25] Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025.

[26] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[27] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[28] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[29] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

[30] Zhang-Wei Hong, Pulkit Agrawal, Rémi Tachet des Combes, and Romain Laroche. Harnessing mixed offline reinforcement learning datasets via trajectory weighting. *arXiv preprint arXiv:2306.13085*, 2023.

[31] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[32] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[33] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[34] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3, 2024.

[35] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[36] Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

[37] Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37:55249–55285, 2024.

[38] Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, et al. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *arXiv preprint arXiv:2411.04282*, 2024.

[39] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*, 2024.

[40] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[41] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

[42] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

[43] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Fan Yang, Zenan Zhou, Weipeng Chen, Haofen Wang, Jeff Z Pan, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

[44] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[45] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[46] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

[47] Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

[48] Qingyuan Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. Vsc-rl: Advancing autonomous vision-language agents with variational subgoal-conditioned reinforcement learning. *arXiv preprint arXiv:2502.07949*, 2025.

[49] Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

[50] Jiazhen Pan, Che Liu, Junde Wu, Fenglin Liu, Jiayuan Zhu, Hongwei Bran Li, Chen Chen, Cheng Ouyang, and Daniel Rueckert. Medvlm-r1: Incentivizing medical reasoning capability of vision-language models (vlms) via reinforcement learning. *arXiv preprint arXiv:2502.19634*, 2025.

[51] Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

[52] Yi-Long Lu, Chunhui Zhang, Jiajun Song, Lifeng Fan, and Wei Wang. Tom-rl: Reinforcement learning unlocks theory of mind in small llms. *arXiv preprint arXiv:2504.01698*, 2025.

[53] Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025.

[54] Yi Ren and Danica J Sutherland. Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*, 2024.

[55] 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.

[56] Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pages 23610–23641. PMLR, 2023.

[57] Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional unalignment: Likelihood displacement in direct preference optimization. *arXiv preprint arXiv:2410.08847*, 2024.

[58] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.

[59] Zhongzhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

[60] Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, nathan lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*, 2025.

[61] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

[62] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

[63] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.

[64] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

[65] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.

[66] Alireza Amiri, Xinting Huang, Mark Rofin, and Michael Hahn. Lower bounds for chain-of-thought reasoning in hard-attention transformers. *arXiv preprint arXiv:2502.02393*, 2025.

[67] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

[68] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

[69] Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

[70] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

[71] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[72] Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew C Yao. Augmenting math word problems via iterative question composing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24605–24613, 2025.

[73] Yihang Yao, Zhepeng Cen, Miao Li, William Han, Yuyou Zhang, Emerson Liu, Zuxin Liu, Chuang Gan, and Ding Zhao. Your language model may think too rigidly: Achieving reasoning consistency with symmetry-enhanced training. *arXiv preprint arXiv:2502.17800*, 2025.

[74] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*, 2024.

[75] John Yang, Kilian Leret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

[76] Elad Levi and Ilan Kadar. Intellagent: A multi-agent framework for evaluating conversational ai systems. *arXiv preprint arXiv:2501.11067*, 2025.

[77] Akshara Prabhakar, Zuxin Liu, Weiran Yao, Jianguo Zhang, Ming Zhu, Shiyu Wang, Zhiwei Liu, Tulika Awalgaonkar, Haolin Chen, Thai Hoang, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*, 2025.

[78] Shuo Tang, Xianghe Pang, Zexi Liu, Bohan Tang, Rui Ye, Tian Jin, Xiaowen Dong, Yanfeng Wang, and Siheng Chen. Synthesizing post-training data for llms through multi-agent simulation. *arXiv preprint arXiv:2410.14251*, 2024.

[79] Hector Vargas Alvarez, Gianluca Fabiani, Nikolaos Kazantzis, Ioannis G Kevrekidis, and Constantinos Siettos. Nonlinear discrete-time observers with physics-informed neural networks. *Chaos, Solitons & Fractals*, 186:115215, 2024.

[80] Saptarshi Sengupta, Harsh Vashistha, Kristal Curtis, Akshay Mallipeddi, Abhinav Mathur, Joseph Ross, and Liang Gou. Mag-v: A multi-agent framework for synthetic data generation and verification. *arXiv preprint arXiv:2412.04494*, 2024.

[81] Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Linhao Zhang, Shulin Xin, Lu Chen, Qi Liu, Xiaojian Zhong, Aoyan Li, et al. Multi-swe-bench: A multilingual benchmark for issue resolving. *arXiv preprint arXiv:2504.02605*, 2025.

[82] 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.

[83] Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. *arXiv preprint arXiv:2503.15477*, 2025.

[84] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[85] Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. Dynamic evaluation of large language models by meta probing agents. In *Forty-first International Conference on Machine Learning*, 2024.

[86] Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *The Twelfth International Conference on Learning Representations*, 2023.

[87] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of Language Models: Part 1, Learning Hierarchical Language Structures. *SSRN Electronic Journal*, May 2023. Full version available at `https://ssrn.com/abstract=5250639`.

[88] Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22, 2024.

[89] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of Language Models: Part 2.2, How to Learn From Mistakes on Grade-School Math Problems. In *Proceedings of the 13th International Conference on Learning Representations*, ICLR '25, April 2025. Full version available at `https://ssrn.com/abstract=5250631`.

[90] Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*, 2024.

[91] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[92] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[93] Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. What do learning dynamics reveal about generalization in llm reasoning? *arXiv preprint arXiv:2411.07681*, 2024.

[94] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[95] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[96] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[97] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[98] Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, 2025.

[99] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[100] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[101] John Schulman. Approximating kl divergence, 2020.

[102] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[103] William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We made the discussion of limitations in section 5. More details about the assumption discussion and computation resources are provided in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The detailed proof of the proposition is provided in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The information needed to reproduce the main experimental results of the paper is introduced in section 4. More details are provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and dataset are provided in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, they are introduced in section 4 and Appendix B.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experiments contain statistical significance tests. We visualize the perplexity and entropy distribution as shown in Figure 6. It indicates that the proposed behaviors are from types of either exploration or exploitation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: They are introduced in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: They are discussed in section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the assets are cited. The licenses are provided in the code. Specifically, we use VeRL with Apache-2.0 License, iGSM with MIT License, and PromptBench with MIT license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets introduced in the paper are well documented and the documentation (README files) is provided alongside the code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Proofs and Discussions of Theoretical Results

A.1 The Proof and Discussion of Proposition 3.1

The proof is as follows:

Proof. We first compute the advantage in one group. For query $\mathbf{q}$ with n correct outputs and $N-n$ wrong outputs ($0 < n < N$), the mean and standard deviation of rewards are n/N and $\sqrt{n(N-n)}/N$ respectively. Accordingly, the advantages of positive and negative samples are

$$\begin{cases} A(\mathbf{q}, \mathbf{o}_+) & = \frac{1-n/N}{\sqrt{n(N-n)}/N} = \sqrt{\frac{N-n}{n}} \\ A(\mathbf{q}, \mathbf{o}_-) & = \frac{0-n/N}{\sqrt{n(N-n)}/N} = -\sqrt{\frac{n}{N-n}} \end{cases}, \tag{6}$$

Note that the advantage will be all 0 if $n = 0$ or $n = N$.

Then we consider the GRPO objective in Eq.(3), when the RL training is on policy, the importance ratio $\pi_\theta(\mathbf{o}|\mathbf{q})/\pi_{\text{old}}(\mathbf{o}|\mathbf{q}) = 1$ and thus we can remove the corresponding clip terms. Meanwhile, we ignore the KL divergence regularization when the coefficient β is small. Consequently, the gradient of simplified GRPO objective on query $\mathbf{q}$ is

$$\nabla_\theta \mathcal{J}(\mathbf{q}; \theta) \tag{7}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\nabla_\theta \pi_\theta(\mathbf{o}_i|\mathbf{q})}{\pi_{\text{old}}(\mathbf{o}_i|\mathbf{q})} A_i \right] \tag{8}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\pi_\theta(\mathbf{o}_i|\mathbf{q})}{\pi_{\text{old}}(\mathbf{o}_i|\mathbf{q})} \nabla_\theta \log \pi_\theta(\mathbf{o}_i|\mathbf{q}) A_i \right] \tag{9}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \frac{1}{N} \left[\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{i+}|\mathbf{q}) A_{i+} + \sum_{j=1}^{N-n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{j-}|\mathbf{q}) A_{j-} \right] \tag{10}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \frac{\sqrt{n(N-n)}}{N} \left[\frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{i+}|\mathbf{q}) - \frac{1}{N-n} \sum_{j=1}^{N-n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{j-}|\mathbf{q}) \right] \tag{11}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \sqrt{\alpha(1-\alpha)} \left[\frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{i+}|\mathbf{q}) - \frac{1}{N-n} \sum_{j=1}^{N-n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{j-}|\mathbf{q}) \right] \tag{12}$$

where $\alpha = n/N$ and we use the policy gradient $\nabla_\theta \pi_\theta(\mathbf{q}|\mathbf{o}) = \pi_\theta(\mathbf{q}|\mathbf{o}) \nabla_\theta \log \pi_\theta(\mathbf{q}|\mathbf{o})$ in derivation.

Similarly, if we do not expand the outputs to correct and incorrect ones in one group, the gradient of GRPO objective on the whole query set is

$$\nabla_\theta \mathcal{J}(Q; \theta) = \mathbb{E}_{\mathbf{q}'\sim Q, \{\mathbf{o}'\}\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(\mathbf{o}'|\mathbf{q}') A_i \right] \tag{13}$$

Therefore, we can obtain the per-step influence by the inner product of $\nabla_\theta \mathcal{J}(\mathbf{q}; \theta)$ and $\nabla_\theta \mathcal{J}(Q; \theta)$.

□

Remark A.1 (Discussions on the assumptions). Our derivation of per-step influence relies on the on-policy training and small KL regularization coefficient assumptions, both of which approximately hold in practical implementation. The on policy training holds when one rollout step corresponds to one optimization step (i.e., one gradient descent step), which is consistent with our implementation (see Appendix B.4), other framework such as open-R1 [95], or DeepSeekMath [23] where each generation is trained only for once. Moreover, the clip term further helps reduce the gap between π_θ and π_{old} even when strictly on-policy training is not guaranteed. For the KL regularization, we observe that most implementations set a very small coefficient β (e.g., 0.01 or smaller), consistent with our assumption. Overall, the per-step influence is mainly dominated by the data co-influence as derived in Proposition 3.1.

Remark A.2 (Relation of data co-influence to neural tangent kernel (NTK)). The data co-influence $\mathcal{K}_\theta$ is also related to NTK [96], a kernel used to describe how neural network evolves during training via gradient descent. Denote $\boldsymbol{\chi} \doteq (\mathbf{q}, \mathbf{o})$ as the query-output pair, If we regard the language model as a neural network with $\boldsymbol{\chi}$ as input and the logit $\mathbf{z}$ on the whole vocabulary (with size $|V|$) of each token as output, we have

$$\nabla_\theta \log \pi_\theta(\mathbf{o}|\mathbf{q}) = \frac{1}{T}\sum_{t=1}^{T}(\pi_\theta^{(t)}(\cdot|\boldsymbol{\chi}) - \mathbf{e}(o_t))^\intercal \nabla_\theta \mathbf{z}_t(\cdot|\boldsymbol{\chi}) \tag{14}$$

where T is the length of output $\mathbf{o}$, o_t denotes the t-th token of the output, $\mathbf{z}_t \in \mathbb{R}^{|V|}$ is the logits of t-th token. Here $\pi_\theta^{(t)}(\cdot|\boldsymbol{\chi}), \mathbf{e}(o_t) \in \mathbb{R}^{|V|}$. $\pi_\theta^{(t)}(\cdot|\boldsymbol{\chi})$ is the output distribution of t-th token and $\mathbf{e}(o_t)$ means the one-hot vector. Note that the result is divided by T because there is nothing. Although we feed the model with the full answer sequence $\mathbf{o}$ in $\boldsymbol{\chi}$, the tokens in later position will not contribute to computing the output distribution of each token because of the existence of causal mask, which is natively integrated in auto-regressive language model.

Then the data co-influence can be decomposed as

$$\mathcal{K}_\theta[\boldsymbol{\chi}, \boldsymbol{\chi}'] = \langle \nabla_\theta \log \pi_\theta(\mathbf{o}|\mathbf{q}), \nabla_\theta \log \pi_\theta(\mathbf{o}'|\mathbf{q}') \rangle \tag{15}$$

$$= \frac{1}{TT'}\sum_{t=1}^{T}\sum_{\tau=1}^{T'}(\pi_\theta^{(t)}(\cdot|\boldsymbol{\chi}) - \mathbf{e}(o_t))^\intercal \langle \nabla_\theta \mathbf{z}_t(\cdot|\boldsymbol{\chi}), \nabla_\theta \mathbf{z}_\tau(\cdot|\boldsymbol{\chi}') \rangle (\pi_\theta^{(\tau)}(\cdot|\boldsymbol{\chi}') - \mathbf{e}(o'_\tau)) \tag{16}$$

where the middle term $\langle \nabla_\theta \mathbf{z}_t(\cdot|\boldsymbol{\chi}), \nabla_\theta \mathbf{z}_\tau(\cdot|\boldsymbol{\chi}') \rangle$ is empirical NTK of the language model [54].

A.2 Extend Proposition 3.1 to Other RL Algorithms

Following previous notation, i.e., given a query with n correct outputs and $N - n$ wrong outputs, the accuracy $\alpha = n/N$ and the advantages of correct and wrong output are A_+, A_- respectively. We consider three RL variants:

Dr.GRPO [97]. The advantages are $A_+ = 1 - \frac{n}{N}, A_- = -\frac{n}{N}$. Plug-in them to Eq.(10), then

$$\nabla_\theta \mathcal{J} = \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \frac{n(N-n)}{N^2} \left[\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{i+}|\mathbf{q}) - \frac{1}{N-n}\sum_{j=1}^{N-n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{j-}|\mathbf{q}) \right] \tag{17}$$

$$= \mathbb{E}_{\{\mathbf{o}_i\}\sim\pi_\theta} \alpha(1-\alpha) \left[\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{i+}|\mathbf{q}) - \frac{1}{N-n}\sum_{j=1}^{N-n} \nabla_\theta \log \pi_\theta(\mathbf{o}_{j-}|\mathbf{q}) \right] \tag{18}$$

Then its per-step influence is to replace the original coefficient $\sqrt{\alpha(1-\alpha)}$ by $\alpha(1-\alpha)$ and other parts remain the same as GRPO.

GPG [98]. It multiples the advantage by a coefficient C (it is α in original paper and we use C instead to avoid ambiguity). The advantages are $A_+ = C \cdot (1 - \frac{n}{N}), A_- = C \cdot (-\frac{n}{N})$. Similar to Dr.GRPO, the coefficient in per-step influence is $C\alpha(1-\alpha)$ and other parts remain the same.

DAPO [99]. The advantage computation in DAPO is the same as the GRPO so the coefficient is still $\sqrt{\alpha(1-\alpha)}$. DAPO introduces other modifications such as query filtering. Therefore, the corresponding per-step influence should replace the original query set Q by a filtered query set Q' which only includes queries with rollout accuracy $\in (0, 1)$.

B More Details of Method and Experiment

B.1 Evaluation Benchmark

We provide a query along with a vanilla demonstration answer for iGSM and PromptBench tasks. Meanwhile, we provide the corresponding DAG representation for each query for easy understanding.

iGSM. In iGSM, we construct a DAG by first generating the nodes, which are with the form of "each X's Y". There are two types of nodes – abstract node and instance node – based on the type of "Y". If "Y" is a class name, the node will be classified as an abstract node; if "Y" is an instance of the class, the node will be classified as an instance node. For example, consider a class "Classroom" with its two instances "Painting room", "Computer room", the "each Oakwood Middle School's Classroom" is an abstract node while "each Oakwood Middle School's Computer room" is an instance node. After generating the nodes, we then generate the dependency of instance nodes randomly as edges to construct a DAG (we do not need to add dependency to abstract nodes because it has implicit dependency, e.g., "each Oakwood Middle School's Classroom" = "each Oakwood Middle School's Painting room" + "each Oakwood Middle School's Computer room"). We can then generate query and answer based on the DAG. In query, we only list the explicit dependencies on instance nodes which may include the conditions for redundant nodes; in answer, the LLM should be able to unlock all instance and abstract nodes to the final target node with a topological order. More implementation details (e.g., how to generate random nodes and random edges, how to set the name of nodes) are systematically introduced in the original paper [74].

To improve training stability, we modified the original iGSM. Here are the main modifications: (1) we remove the modulo operation in all computation steps; (2) we filter out the queries whose ground-truth answers are larger than 1000 or smaller than -1000 to reduce the reliance on complex computation; (3) the original answer template is "Define [node name] as [a random letter], then [the random letter] = [... computation equations]". We rewrite it to increase the answer diversity while keeping the logic of problem-solving (i.e., the order of node selection) and computation details. See below for an example of the new answer template.

One example with operation number op $= 10$ is shown in the following text boxes. The DAG representation of this question is visualized in Figure 7.

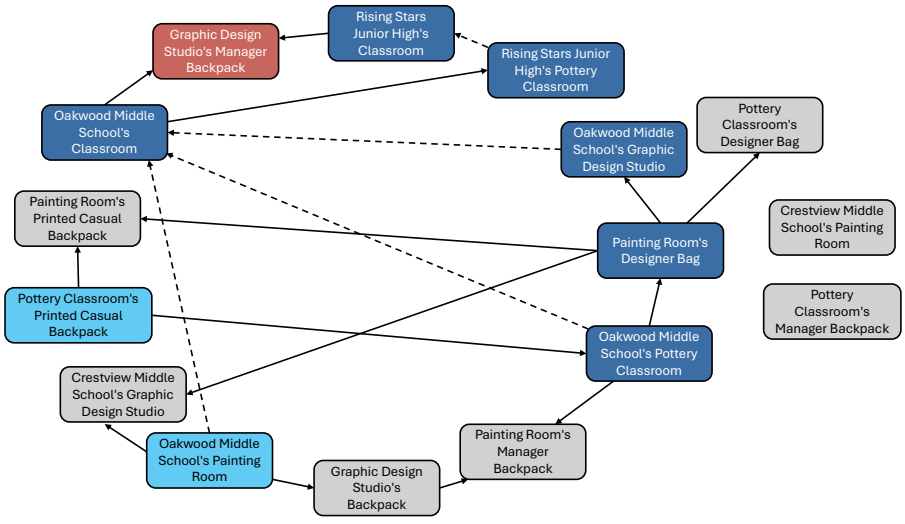

Figure 7: DAG representation of an iGSM task (number of operations $= 10$). The red, light blue, dark blue, and gray rectangles mean the target, leaf, intermediate, and redundant nodes, respectively. The solid arrow indicates the explicit dependency which is explicitly given in query while the dash arrow means the implicit dependency which needs to be inferred by LLM. For example, it is not directly given how to compute the value of each "Rising Stars Junior High's Classroom" in the query. The LLM is required to understand its semantic meaning and then infer that it equals to "Rising Stars Junior High's Pottery Classroom".

Note that the RL training data ($15 \sim 20$ operations) and OOD test set (25 operations) in practical experiments are much harder than the example above.

Query:
The number of each Graphic Design Studio's Manager Backpack equals the difference of each Rising Stars Junior High's Classroom and each Oakwood Middle School's Classroom.
The number of each Painting Room's Designer Bag equals 0 more than each Oakwood Middle School's Pottery Classroom.
The number of each Painting Room's Printed Casual Backpack equals 9 more than the sum of each Painting Room's Designer Bag and each Pottery Classroom's Printed Casual Backpack.
The number of each Crestview Middle School's Graphic Design Studio equals the difference of each Painting Room's Designer Bag and each Oakwood Middle School's Painting Room.
The number of each Pottery Classroom's Designer Bag equals each Painting Room's Designer Bag.
The number of each Pottery Classroom's Manager Backpack equals 0.
The number of each Pottery Classroom's Printed Casual Backpack equals 1.
The number of each Rising Stars Junior High's Pottery Classroom equals 2 times each Oakwood Middle School's Classroom.
The number of each Oakwood Middle School's Painting Room equals 6.
The number of each Crestview Middle School's Painting Room equals 5.
The number of each Painting Room's Manager Backpack equals 0 more than the sum of each Graphic Design Studio's Backpack, each Oakwood Middle School's Pottery Classroom and each Oakwood Middle School's Painting Room.
The number of each Oakwood Middle School's Pottery Classroom equals 5 times each Pottery Classroom's Printed Casual Backpack.
The number of each Oakwood Middle School's Graphic Design Studio equals 8 more than each Painting Room's Designer Bag.
How many Manager Backpack does each Graphic Design Studio have?

Vanilla CoT:
Let's compute the answer step by step.
- According to the information given, the number of each Pottery Classroom's Printed Casual Backpack is 1. Let's denote it as S. So S = 1.
- Next, let Q represent the number of each Oakwood Middle School's Pottery Classroom. Then Q = 5 * S = 5 * 1 = 5.
- Now, we can find the number of each Painting Room's Designer Bag. Let's denote it as U. Then U = 0 + Q = 0 + 5 = 5.
- We can then calculate the number of each Oakwood Middle School's Graphic Design Studio. Let it be m. Then m = 8 + U = 8 + 5 = 13.
- The number of each Oakwood Middle School's Painting Room is 6. Let's denote it as W. So W = 6.
- Then, let's denote the number of each Oakwood Middle School's Classroom as v. Then v = m + Q + W = 13 + 5 + 6 = 24.
- We can then calculate the number of each Rising Stars Junior High's Pottery Classroom. Let it be B. Then B = 2 * v = 2 * 24 = 48.
- Now, we can find the number of each Rising Stars Junior High's Classroom. Let it be p. Then p = B = 48.
- Next, let y represent the number of each Graphic Design Studio's Manager Backpack. Then y = p - v = 48 - 24 = 24.
Thus, the answer is 24.

PromptBench. PromptBench generates data in two stages: (1) DAG construction and (2) Natural language description of the DAG. For DAG construction, we first generate a directed acyclic graph (DAG) with a specified depth and number of redundancies. The DAG is constructed top-down: we begin by generating the root node and then recursively sampling dependencies between each node and its parent node(s). If a node uses a binary operator, it has two parent nodes; if it uses a unary operator, it has one parent node. This process continues recursively until the desired depth is reached. During DAG construction, each node is assigned a unique name generated by a random string generator. Once the DAG structure is complete, we sample values for all leaf nodes from a predefined set and compute the values of internal nodes in a bottom-up manner. For the natural language description stage, we describe the constructed DAG and its associated computation using predefined templates to generate a textual problem description. To improve the training stability, we also filter out the queries whose ground-truth answers are larger than 1000 or smaller than -1000. Full details of the generation process can be found in the original PromptBench paper [86].

One example with depth = 4, number of redundancy = 2 is shown in the following text boxes. The DAG representation of this question is visualized in Figure 8.

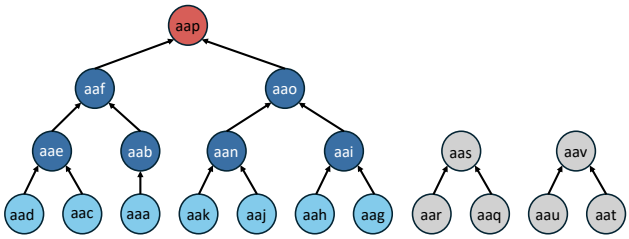

Figure 8: DAG representation of a PromptBench task. The red, light blue, dark blue, and gray circles mean the target, leaf, intermediate, and redundant nodes, respectively.

Query:
The value of aac is 5.
aaf gets its value by adding together the value of aab and aae.
The value of aaj is 5.
The value of aak is 9.
The value of aar is 2.
The value of aaa is 9.
The value of aat is 3.
aao gets its value by adding together the value of aai and aan.
The value of aad is 5.
aas gets its value by multiplying together the value of aaq and aar.
The value of aaq is 5.
aan gets its value by adding together the value of aak and aaj.
The value of aau is 10.
aai gets its value by multiplying together the value of aah and aag.
aap gets its value by subtracting the value of aao from the value of aaf.
The value of aah is 2.
aav gets its value by multiplying together the value of aat and aau.
aab gets its value by squaring the value that aaa has.
The value of aag is 6.
aae gets its value by subtracting the value of aac from the value of aad.
What is the value of aap?

Vanilla CoT: Let's compute the answer step by step.
Let's solve aaa, aaa is 9
Let's solve aab, aab = aaa^2 = 81
Let's solve aac, aac is 5
Let's solve aad, aad is 5
Let's solve aae, aae = aac - aad = 0
Let's solve aaf, aaf = aab + aae = 81
Let's solve aah, aah is 2
Let's solve aaj, aaj is 5
Let's solve aag, aag is 6
Let's solve aai, aai = aag * aah = 12
Let's solve aak, aak is 9
Let's solve aan, aan = aaj + aak = 14
Let's solve aao, aao = aai + aan = 26
Let's solve aap, aap = aao - aaf = 55
Thus, the answer is 55.

Why are iGSM and Promptbench good testbeds for LLM reasoning evaluation? We can observe that both tasks require the problem-solving capabilities of LLM from multiple perspectives, e.g., basic arithmetic calculation, search and planning, which mirrors reasoning tasks in realistic scenarios. Meanwhile, these tasks require few priors (it only needs very basic concept understanding in iGSM task) and focus on the **abilities** instead of **knowledge**. Moreover, all the data are synthetic and the queries are almost infinite (see estimation in [74]). Therefore, it avoiding the data contamination,

which can significantly confound the experiment results, while keeping substantial diversity of the query set.

B.2 The Extraction of DAG representation

For the datasets we used in this paper (iGSM and promptbench), all questions and answers are rule-based and follow a fixed sentence structure, which allows us to extract the DAG representation through simple **string matching**.

Take an iGSM problem for example, each variable mentioned in the question is treated as a node, and we define an edge from one node to another if the value of the former depends on the latter. Consider a premise *The number of each Painting Room's Printed Casual Backpack equals 9 more than the sum of each Painting Room's Designer Bag and each Pottery Classroom's Printed Casual Backpack*, we view *each Painting Room's Printed Casual Backpack* as a node. Since it depends on nodes *each Painting Room's Designer Bag* and *each Pottery Classroom's Printed Casual Backpack* and we draw edges from these two nodes to the former. By iterating all the premises in this way, we construct the node set V and edge set E of the graph. Similarly, we can also apply the same procedure to the answer. However, we will miss the redundant nodes if the answer only includes the minimal topological path to the final node.

For other QA datasets without a fixed format, we can employ an oracle LLM (e.g., GPT4) to parse the node and edge from the QA. Specifically, we can view all intermediate variables / conclusions / corollaries as node and the edge is still their dependency. Here is a prompt to extract (V, E):

Prompt to extract DAG:
You are a helpful data analyst. You will be given a question-answer pair. Your task is to extract the DAG of the question. Here are some instructions for extracting the DAG:
- First separate the answer into multiple steps. If one step includes multiple intermediate variables / conclusions, further separate it until each step includes only one.
- Identify NODE in each step. The node can be intermediate variables / conclusions / corollary.
- Identify the dependency between the nodes.
- Remember that the final graph should be acyclic.
You should return the DAG in the following format:
- the name of the nodes, e.g., node 1: x, node 2: y, node 3: z
- the dependent list of each node, e.g., node 1: [], node 2: [1], node 3: [1, 2]
[few-shot examples]
[QA]

B.3 Examples of Injected Behaviors

In this section, we give an example of injected behaviors on iGSM task.

Injected behaviors in iGSM:
Let's compute the answer step by step.
...
- The number of each Oakwood Middle School's Painting Room is 6. Let's denote it as W. So W = 6.
- Then, let's denote the number of each Oakwood Middle School's Classroom as v. Then v = m + Q + W = 13 + 5 + 6 = 18 + 6 = 24.
...
- Then, let's denote the number of each Graphic Design Studio's Manager Backpack as y. But we haven't calculated the number of each Rising Stars Junior High's Classroom yet, thus the value of y is still unknown.
...
- Now, we can find the number of each Graphic Design Studio's Manager Backpack. Remember that it has been denoted as y. We know that it equals the difference between each Rising Stars Junior High's Classroom and each Oakwood Middle School's Classroom. Then y = p - v = 48 - 24 = 24.
Thus, the answer is 24.

The blue, violet, and cyan parts correspond to the subgoal computation, reflection, and information analysis behaviors. In subgoal computation, we enforce LLM to conduct only one operation (i.e., $+, -$ or $\times$) in every derivation and keep all intermediate computations for complex equations. In reflection, the LLM explores a locked (not solvable yet) node "each Graphic Design Studio's Manager Backpack" and then goes back. In information analysis, the LLM integrates the information gathered from the query for better derivation of computation, which is also an exploitative behavior.

B.4 Training Details

SFT dataset and baseline implementations. For the iGSM task, we use 2000 SFT data for `Vanilla` and BRIDGE. The `PP-Aug` and `RC-Aug` augment each data for additional three times so they have 8000 SFT data in total. For the PromptBench task, we use 5000 SFT data for `Vanilla` and BRIDGE. The `PP-Aug` and `RC-Aug` augment each data for an additional once so they have 10000 SFT data in total.

SFT training. We first train the LLMs by SFT to narrow the domain gap between the pretraining corpus and evaluation tasks and enforce the model to answer the question with the demonstrated template, where the final reward is easy to extract and verify. The other training configurations are attached below, which are shared by all base models on both the iGSM and PromptBench tasks.

Table 3: Configurations of SFT training

Configurations	value
training epoch	5
batch size	128
learning rate	5×10^{-6}
learning rate scheduler	constant

The system prompt along with query and answer templates are shown as follows:

System prompt:
A conversation that the assistant solves the user's problem. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> all the reasoning process here </think> <answer> final answer here </answer>.

SFT data template (take Qwen tokenizer for example)
<|im_start|>system<|im_end|>
{system prompt}
<|im_start|>user<|im_end|>
{query}
<|im_start|>assistant<|im_end|>
<think> {CoT answer} </think>
<answer> The final answer is $\boxed{\{\text{final answer}\}}$ </answer>

RL training. After the SFT stage, we apply RL to further fine-tune the SFT models. Our implementation is based on VeRL [100]. When computing KL divergence, we use the low variance implementation [101], aligning with the GRPO implementation [23]. The shared parameters are listed in Table 4.

In rollout, we set the maximum generation length as 2560 for iGSM and 1536 for PromptBench. We use top $p = 1.0$ for sampling in generation. Besides, we observe that Llama3.2-1B may deviate from answer template so we add a format reward to it, i.e., it will obtain a 0.05 reward if it strictly follows the given answer template (i.e., "<think> ... </think> <answer> ... </answer>") addition to the correctness reward.

B.5 More Experiment Results

We attach the training curves of main experiments in Fig. 9&10.

Table 4: Configurations of RL training

Configurations	value
batch size	256
learning rate	1×10^{-6}
learning rate scheduler	constant
rollout number per query N	32
rollout temperature	1.0
rollout backend	vllm [102]
KL coefficient β	0.001

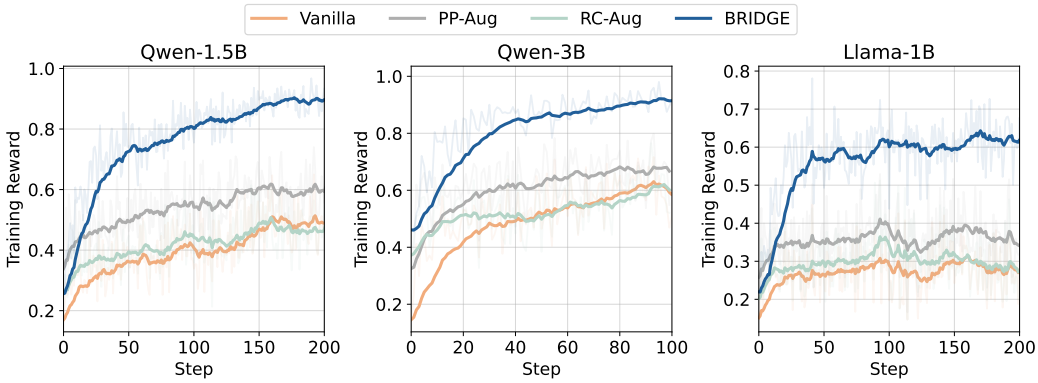

Figure 9: The training curve of experiments in the iGSM task.

B.6 Comparison between BRIDGE and Accuracy-based Rejection Sampling

Based on the analysis of two factors affecting the performance in RL in Proposition 3.1, there is an intuitive idea that filters out queries with excessively high or low rollout accuracy (close to 1 or 0) as discussed in the beginning of sec. 3.3. In this section, we compare the performance of our method with the rejection-sampling-based RL. Specifically, we roll out the SFT models (pre-RL models) for 8 times on a larger dataset that shares the same distribution as the original RL training set. We then retain only the queries with 1 to 7 correct answers, resulting in a filtered RL training dataset with a medium accuracy ratio. Note that the new dataset has the same size as previous RL training dataset. Then we run RL on the rejection sampled dataset starting from the same SFT model. The results are shown in Fig. 11.

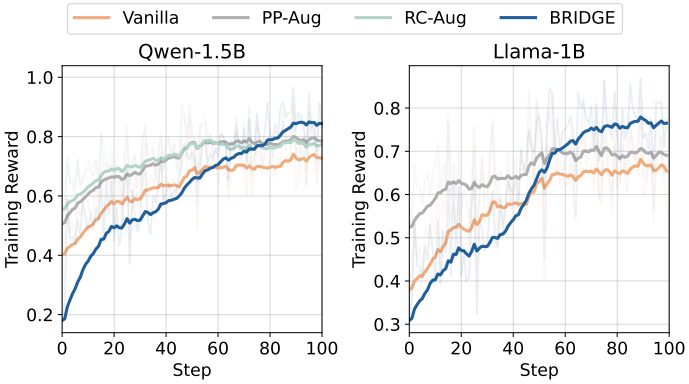

Figure 10: The training curve of experiments in the PromptBench task.

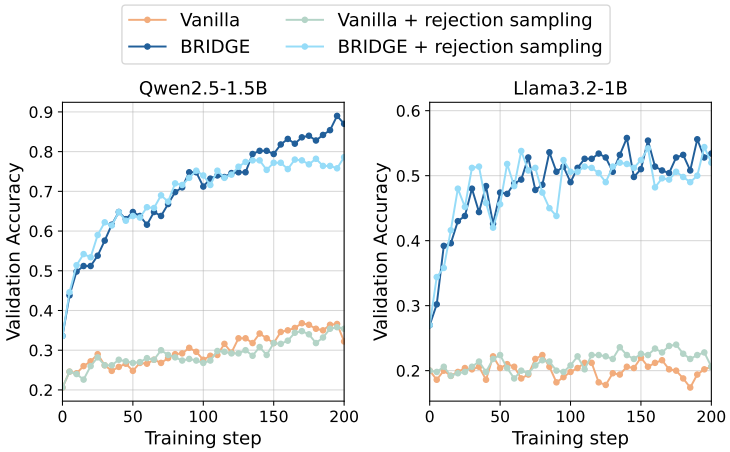

Figure 11: The comparison of BRIDGE with rejection-sampling-based filtering.

We can observe that the rejection sampling does not improve the RL performance over `Vanilla` baseline and is significantly worse than BRIDGE. Meanwhile, when applying rejection sampling to the BRIDGE, the difference is very minor, and it even degrades the final performance on the Qwen-1.5B model. The reason for its poor performance is that rejection sampling can only shape the accuracy distribution of RL rollouts at the initial stage. As training progresses, the distribution may shift and is not guaranteed to remain within the medium-accuracy range. More importantly, this method does not improve the data co-influence. Based on the above results, we can conclude that it is more effective to improve the data co-influence when preparing LLM for RL, which relies on behavior injection, rather than simply adjusting rollout accuracy by distorting the query distribution.

B.7 Details of Data Co-influence and Per-step Influence Estimation

Here is a more detailed description on the computation of per-step influence shown in Fig. 3 right.

1. Begin by 2,000 queries. For each query, we generate 8 rollouts using the model, resulting in a total of 16,000 query–output pairs. Each query is then assigned to an accuracy group based on the accuracy of its rollouts.
2. For each query-output pair, we compute 1) advantage, and 2) the policy gradient $\nabla_\theta \log \pi_\theta(o|q)$.
3. Using Eq.(5), we estimate the per-step influence for each query q. In this formulation, (q', o') ranges over all 16,000 query–output pairs.
4. We then group queries by their accuracy (0/8, 1/8, ..., 8/8), and compute the average per-step influence for each group. These results are visualized in Fig. 3 right. Notably, the groups with 0/8 and 8/8 accuracy have zero per-step influence because the corresponding advantages are zero under the GRPO formulation.

Since computing the inner product $\mathcal{K}_\theta[.,.]$ is computationally intractable, we mainly follow LESS [82] in data co-influence estimation. Specifically, there are two main steps reducing the dimensionality of $\nabla_\theta \log \pi_\theta(\mathbf{o}|\mathbf{q})$: (1) we first compute the LoRA gradient $\nabla_{\theta_{\text{LoRA}}} \log \pi_{\theta_{\text{LoRA}}}(\mathbf{o}|\mathbf{q})$ rather than the full-parameter gradient by backpropagating the likelihood loss to the LoRA modules. (2) Then we apply random projection on the LoRA gradient and get a vector with a smaller dimension, which preserves the inner product of two gradients [103]. For the LoRA module, we specified a rank of 64, an α value of 128, a dropout rate of 0.1, and learned LoRA matrices for all attention matrices. For the random projection, the final dimension of the projection output is 8192. Then we can estimate the data co-influence coefficient $\mathcal{K}_\theta[\cdot,\cdot]$ by the inner product between two query-output samples.

After obtaining the data co-influence coefficient, we can compute the per-step influence $\Delta\mathcal{J}$ by plugging the data co-influence and advantages. **Note that we did not multiple the learning rate η when computing the results in Fig. 3.**

B.8 Computation Overhead

All experiments can be run on a server with $2\times$A100 (80G). Each SFT experiment takes less than 1h. Regarding the RL experiments, it takes $\sim$ 8h for Qwen-1.5B and Llama-1B to complete 200-step RL and $\sim$ 6h for Qwen-3B to run 100-step RL on iGSM while it spends $\sim$ 2h to run RL on PromptBench (100 steps) for Qwen-1.5B and Llama-1B due to shorter rollout length.