GUIDEDSAMPLING: Improving Diversity for Training Large Language Models

Anonymous ACL submission

Abstract

Repeated Sampling (RS) is a simple yet effective inference-time strategy that has been shown to enhance performance on complex tasks. Although its integration into post-training has achieved pass@k improvements, RS often struggles with generating diverse solution candidates (i.e., lack of exploration of solution space). Due to the lack of diversity, multiple samples are often redundant in their generation since they use the same underlying approach to solve a given problem. To address these limitations, we propose a new inference strategy, GUIDEDSAMPLING, which decouples the exploration and generation phases at inference time, increasing diversity during sampling. The exploration phase explores multiple concepts that can be utilized to solve the problem, while the generation phase uses a particular concept to give a final solution. Experimental results show that GUIDEDSAMPLING improves the rate of finding correct solutions by up to $\sim 34.6\%$ over a strong baseline. Furthermore, models trained with trajectories generated via GUIDEDSAMPLING exhibit substantial performance improvements in pass@10, including $17\% \uparrow$ on the MATH, $11.12\% \uparrow$ on GPQA-Diamond, and $5.49\% \uparrow$ on HumanEval, compared to models trained with traditional RS.[1]

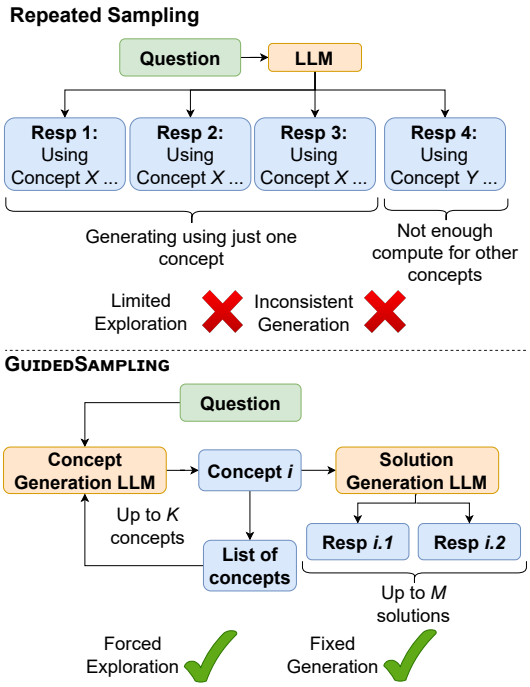

Figure 1: GUIDEDSAMPLING enhances exploration during inference by first generating a set of diverse ideas or theorems to guide subsequent generations of solutions. Unlike repeated sampling (RS), where the model generates the final solution, GUIDEDSAMPLING separates these phases.

1 Introduction

Recent advances in large language models (LLMs) have shown that scaling model size and training data can lead to increasingly capable systems across diverse domains including mathematical reasoning, scientific analysis, and code generation (Kaplan et al., 2020). However, scaling models indefinitely is becoming increasingly infeasible due to the requirement of more data for training ever-larger models (Villalobos et al., 2024). As a result, a growing body of work has shifted focus to alternative ways of boosting performance—not by making models larger, but by making better use of available compute during inference (Hosseini et al., 2024; Kumar et al., 2024; Lightman et al., 2023; Brown et al., 2024). Several studies now suggest that allocating additional compute at inference time can lead to larger performance gains than spending that compute to train bigger models (Snell et al., 2024; Wu et al., 2024).

To this end, various inference-time algorithms have been proposed (Wang et al., 2022; Yao et al., 2023; Zhang et al., 2024). Among them, repeated sampling (RS) (Cobbe et al., 2021) is one of the

[1]The code and data is available at https://anonymous.4open.science/r/sampling_inference-B44E

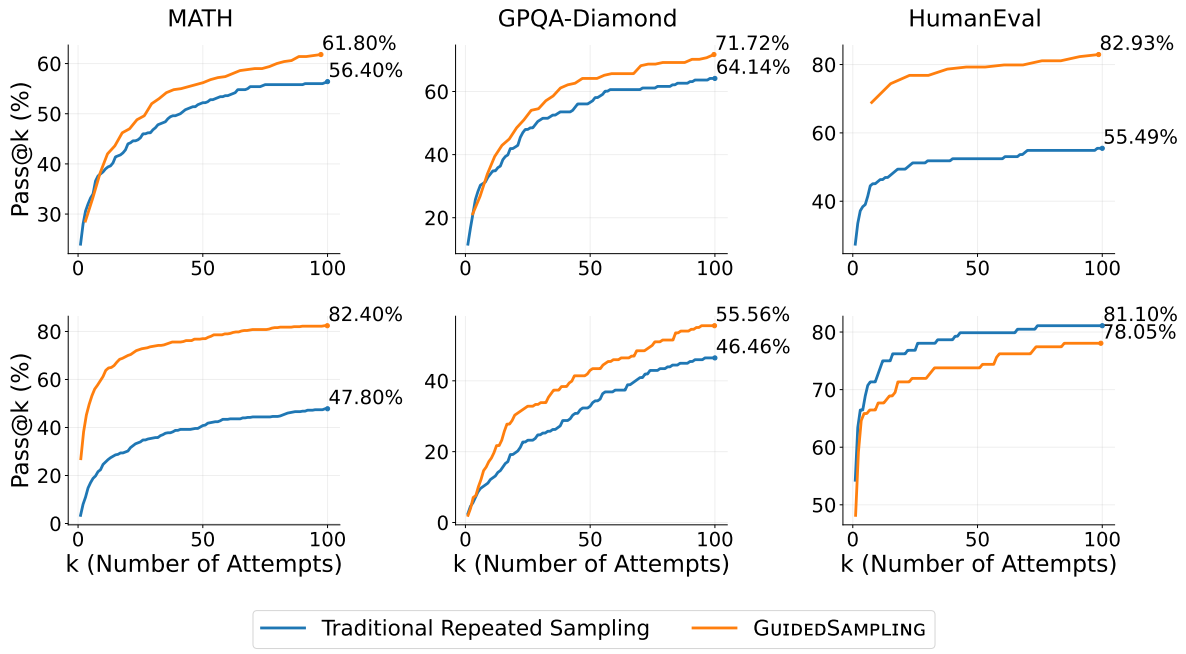

Figure 2: GuidedSampling forces exploration during inference-time, which results in 13.51% average pass@k accuracy compared to traditional repeated sampling. We observe an average improvement of 20% on MATH, 8.34% on GPQA-Diamond, and 12.19% on HumanEval. **First row**: Results for Llama-3.2-3B-Instruct, **Second row**: Results for Qwen2.5-3B-Instruct.

most widely used inference-time algorithms, where multiple outputs are sampled for the same input prompt. Traditional RS implicitly combines two phases: *exploration*, which we define as the diverse theorems or concepts used in solving the given question, and *generation*, where the LLMs use a particular concept and try to generate many solutions for a given problem. However, despite its simplicity, traditional RS suffers from a lack of exploration, leading to the repeated generation of solutions with the same underlying concepts rather than a thorough exploration of the solution space (Brown et al., 2024). To address these limitations, we first propose a novel sampling technique, GuidedSampling, designed to decouple the exploration of diverse concepts from the generation of final solutions. We then demonstrate how training LLMs on trajectories of GuidedSampling shows significant performance gains.

GuidedSampling first explicitly samples diverse concepts or theorems that can be used to solve a given question. Then, in the second phase, these concepts guide the generation of complete candidate solutions. This decoupling is the key reason that GuidedSampling enhances the diversity of solution candidates generated during inference. For instance, consider a problem from MATH (Hendrycks et al., 2021): "Find the maximum value of $\left[\frac{x-y}{x^4+y^4+6}\right]$ over all real numbers x and y.". For this problem, we sample 1000 solutions using traditional RS and GuidedSampling. Our detailed analysis of these candidates shows that $892/1000$ uses only "*AM-GM inequality*" concept to solve the problem, consistently leading to the incorrect solution due to over-utilizing the same theorem. In contrast, only $77/1000$ candidates from GuidedSampling use this theorem, dedicating the remaining compute to exploring other theorems such as "*Cauchy-Schwarz Inequality*", "*Trivial Inequality*", and "*Chebyshev's Inequality*". This improved exploration significantly expands the search space of the model, leading to better accuracy, as is illustrated in Figure 2. More details about GuidedSampling are provided in §3.

Our other core contribution is to use GuidedSampling to improve LLM training. We demonstrate that fine-tuning LLMs on trajectories generated by GuidedSampling outperforms models trained on trajectories from traditional RS. We generate diverse solution trajectories using GuidedSampling on a random subset of 10k instances from OpenMathInstruct-2 (Toshniwal et al., 2024). LLMs fine-tuned on this data exhibited a 17% ↑ in pass@10 accuracy on the MATH benchmark.

These fine-tuned models also demonstrate improved generalization, with pass@10 gains on out-of-domain benchmarks, GPQA-Diamond (Rein et al., 2024) for scientific reasoning ($11.12\% \uparrow$) and HumanEval (Chen et al., 2021) for Python code generation ($5.49\% \uparrow$). Results show that training with an explicit decoupling of exploration and generation leads to better generalizable reasoning capabilities than simply training on correct solutions. Further details are presented in §3.3. In summary, our contributions are as follows:

1. We propose GUIDEDSAMPLING, an inference-time sampling technique that improves the diversity of generated solutions.
2. We show how performance varies when shifting compute between exploration and generation.
3. We demonstrate that fine-tuning LLMs on GUIDEDSAMPLING trajectories significantly improves performance on mathematical reasoning and generalizes the model to other domains.
4. Our proposed approaches show significant improvements over baselines trained with traditional RS on benchmarks including MATH, GPQA-Diamond, and HumanEval.

2 Related Works

Inference Strategies Chain-of-thought (CoT) and its variants (Wei et al., 2022; Kojima et al., 2022) showed that guiding LLMs to produce intermediate reasoning steps during inference boosts the performance on complex tasks such as mathematical and commonsense reasoning. However, as reasoning chains become longer, CoT suffers from error propagation due to complex calculations (Chen et al., 2022). To mitigate this, new methods have been proposed like Self-Consistency (SC), which samples multiple CoT from LLM and then selects the most consistent final answer through majority voting (Wang et al., 2022). Building upon these ideas, better search algorithms such as tree-of-thought (Yao et al., 2023), MCTS (Zhang et al., 2024), and REBASE (Wu et al., 2024) have been proposed, which enable LLMs to perform more deliberate problem solving by exploring multiple reasoning paths in a tree structure. Finally, several agentic systems (Parmar et al., 2025; Estornell and Liu, 2024) have shown that spending more time debating between agents at inference before generating a final solution improves performance. In contrast to prior methods, GUIDEDSAMPLING generates a diverse set of samples with lower inference-time cost than tree search, while achieving greater diversity than both standard prompting and recent agentic approaches. Parallel to our work, Wang et al. (2025) proposed RandIdeaInjection, which first generates a list of distinct ideas and then injects the generated list into the generation process to produce the final response. GUIDEDSAMPLING, on the other hand, works in an iterative loop of generating concepts, adding them individually to generate the final output.

Training LLMs using Synthetic Data Recent works have explored leveraging advanced inference strategies both for generating high-quality synthetic training data and for fine-tuning models to improve their performance. For instance, Self-Taught Reasoner (STaR) (Zelikman et al., 2022) is an iterative method where an LLM is prompted to generate CoT rationales; those rationales that lead to correct answers are then used as high-quality synthetic data to fine-tune the model, while those which lead to incorrect answers are passed back to model for refinement along with the correct final answer, effectively bootstrapping its reasoning abilities from a small initial set. Similarly, ReST^{EM} (Singh et al., 2023), building on principles of reinforced self-training (ReST), employs an iterative Expectation-Maximization-like framework. It uses Best-of-N (BoN) sampling to generate multiple candidate solutions for problems and then refines the model by training on this synthetically generated data. Chow et al. (2024) and Tang et al. (2025) developed reinforcement learning (RL) methods that directly optimize for pass@k metrics and majority voting performance, leading to significant gains in reasoning and code generation. Other methods, such as multi-agent fine-tuning (Subramaniam et al., 2025), train diverse agent models through debate and voting, while Gui et al. (2024) introduced BoNBoN Alignment, distilling the BoN sampling distribution into a single model. While these strategies improve pass@k, they often do not explicitly manage the trade-off between exploration and generation. In contrast, our proposed GUIDEDSAMPLING method introduces a structured exploration phase during training, explicitly balancing diversity and quality. We show that models fine-tuned with GUIDEDSAMPLING outperform those trained using data generated by methods like BoN, STaR, or tree-of-thoughts, and achieve stronger pass@k performance than prior training techniques.

3 GUIDEDSAMPLING

3.1 Background

Traditional RS Repeated Sampling (RS) is a simple strategy to increase the inference-time performance of a model by generating multiple samples from the model's output distribution. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of input queries. For each input $x \in X$, we draw k independent samples from the model-defined conditional distribution $p_\theta(y \mid x)$, i.e.,

$$y_i^{(x)} \sim p_\theta(y \mid x), \quad \text{for } i = 1, \ldots, k$$

This process effectively scales the model's inference-time compute linearly with k. The theoretical appeal of RS lies in its potential to achieve complete coverage of the output space as $k \to \infty$. For any target output y^* such that $p_\theta(y^* \mid x) > 0$, the probability that it is sampled at least once after k draws is:

$$P_k = 1 - (1 - p_\theta(y^* \mid x))^k$$

This quantity monotonically increases with k and asymptotically approaches 1. Thus, under the assumption that all valid outputs are assigned non-zero probability by the model, unlimited sampling ensures that the target output will be generated at least once. This has led to several works adopting RS to generate solutions (Wang et al., 2022; Rozière et al., 2023; Li et al., 2022).

Of course, unlimited sampling is impractical. The value of RS lies in whether increased sampling leads to improved output quality within a feasible compute budget. Several works have pointed out this issue, stating that the lack of diversity in these generated responses is the key limitation of scaling up RS (Brown et al., 2024; Wang et al., 2025).

Diversity Analysis To quantify the lack of diversity in RS, we use Qwen2.5-32B-Instruct (Yang et al., 2024) to extract the core concept or theorem from each solution. We present in prompt for concept extraction in Appendix A.2. We then find the number of distinct concepts which are used to solve a given problem. We find that solutions sampled using RS tend to rely heavily on a few underlying concepts to solve the problem even with an increasing number of compute. For example, while solving problems presented in the HumanEval benchmark, even with 100 responses, Llama-3.2-3B-Instruct gave an average of only 2.75 different concepts

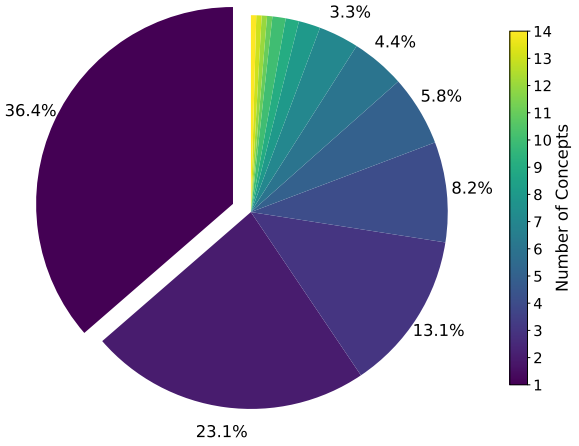

Figure 3: Distribution of the average number of concepts used by Llama-3.2-3B-Instruct for 100 repeated samplings. 37% of the questions are attempted with just 1 concept, while less than 36% of the questions have more than 2 concepts.

that can be used to generate the final answer. Figure 3 represents the distribution of the number of questions for each concept for a compute of 100 responses. We find that in 64% of the questions, less than three concepts were used to solve the questions, with 36.6% just using one concept.

Tree-of-Thoughts (ToT) ToT represents a more sophisticated strategy for enhancing model performance in complex problem-solving tasks by explicitly exploring multiple reasoning paths (Yao et al., 2023). Let P be an initial problem. ToT guides a language model to generate a tree of "thoughts", where each thought t_i is a coherent sequence of text representing an intermediate step towards a solution. The model generates multiple candidate thoughts $T_j = \{t_1^{(j)}, t_2^{(j)}, \ldots, t_m^{(j)}\}$ from a parent thought t_p. Each of these candidate thoughts is then evaluated, often by the LLM itself or a separate verifier, $V(t_i^{(j)} \mid P, t_p)$ to assess its promise. Search algorithms like Breadth-First Search (BFS) or Depth-First Search (DFS) are employed to navigate this tree, allowing the model to look ahead, backtrack if a path seems unpromising, and explore different lines of reasoning (Long, 2023). The theoretical strength of ToT lies in its potential to systematically explore a vast solution space, thereby increasing the likelihood of finding a correct or high-quality solution, especially for tasks where simpler methods like Chain of Thought (CoT) might falter due to their linear, single-path reasoning. This structured exploration aims to address issues like lack

of diversity in generated paths by deliberately generating and considering varied intermediate steps. However, this explicit generation and evaluation of numerous thought branches make tree-of-thought computationally intensive, with costs scaling with the number of candidates explored at each step (m) and the depth of the tree.

While ToT solves the lack of diversity observed in RS, it is significantly more computational as explicit evaluation of each intermediate thought generated at every step of the tree's expansion is required. To mitigate both the lack of diversity in the solutions and less computational cost, we propose GUIDEDSAMPLING, which we elaborate on in the following sections.

3.2 Our Proposed Approach

Our proposed inference strategy, GUIDEDSAMPLING, improves the diversity by separating exploration and generation into two distinct phases. This separation allows for finer control over the diversity of concepts that can be used to solve a problem, an aspect previous approaches like traditional RS fall short of. Moreover, our method explores the concepts just once in the beginning, which leads to better efficiency than the tree-of-thought strategy. Figure 1 highlights the differences between our strategy and RS. We describe these two phases of our strategy in detail below:

Exploration Phase The goal of the Exploration Phase is to discover a diverse set of high-level ideas, concepts, or theorems that could guide the solution of a given question. We start with a dataset or a set of questions denoted by X, from which we sample a specific question $x \in X$ to work on. Given this question x and an LLM parameterized by θ, we aim to identify a set of relevant concepts that could support downstream reasoning or problem-solving, denoted as $\mathcal{C} = \{c_1, c_2, ..., c_K\}$. The process of constructing $\mathcal{C}$ is iterative: the k-th concept is generated by conditioning on the original question x and all previously generated concepts $c_1, \ldots, c_{k-1}$. Formally, this sampling process is expressed as:

$$c_k \sim p_\theta(i \mid x, i_{1:(k-1)})$$

This iterative conditioning mechanism promotes diversity among the concepts, encouraging the model to explore different areas of the solution space rather than repeating similar ideas. The algorithm continues until either K concepts have been generated or the model determines that no more useful ideas can be produced—allowing for early stopping. The prompts used for exploration are presented in Appendix A.1.

Algorithm 1 GUIDEDSAMPLING

Input: Question prompt x, LLM p_θ, maximum number of ideas K, completions per idea M
Output: Set of candidate solutions $\mathcal{S}$

// Exploration Phase
$\mathcal{C} \leftarrow \emptyset$ ▷ Initialize set of concepts
$k \leftarrow 1$
while $k \leq K$ **do**
  $c_k \sim p_\theta(\cdot \mid x, c_1, \ldots, c_{k-1})$ ▷ Sample concept
  if $c_k =$ None **then** ▷ Model indicates no more useful concepts
    break
  end if
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_k\}$
  $k \leftarrow k + 1$
end while

// Generation Phase
$\mathcal{S} \leftarrow \emptyset$ ▷ Initialize set of solutions
for each concept $c_k \in \mathcal{C}$ **do**
  $\mathcal{S}_k \leftarrow \emptyset$ ▷ Initialize solutions for current concept
  for $m = 1$ **to** M **do**
    Sample solution $s_k^{(m)} \sim p_\theta(\cdot \mid x, c_k)$ ▷ Generate solution based on concept
    $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{s_k^{(m)}\}$
  end for
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_k$
end for
return $\mathcal{S}$

Generation Phase Once the set of candidate concepts $\mathcal{C} = c_1, c_2, \ldots, c_K$ has been established during the Exploration Phase, the Generation Phase uses these concepts to produce concrete solutions. For each concept $c_k \in \mathcal{C}$, we generate M potential solutions. These solutions are sampled from the LLM, conditioned on both the original question x and the specific concept c_k:

$$\mathcal{S}_k = \left\{ s_k^{(m)} \sim p_\theta(s \mid x, c_k) \right\}_{m=1}^{M}$$

Each completion $s_k^{(m)}$ represents a full solution that uses the guidance provided by c_k. The full set of candidate solutions is thus $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$.

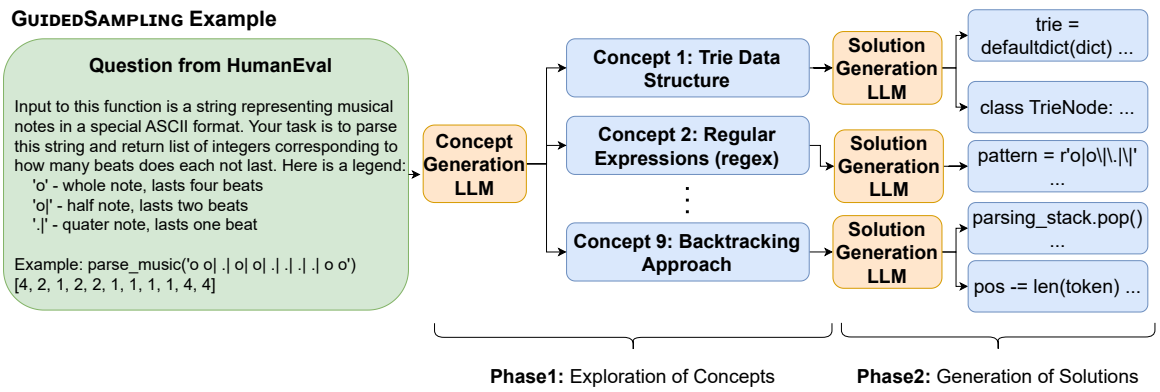

Figure 4: An example illustrating the flow of data in GUIDEDSAMPLING. Phase1 generates up to K different concepts, while Phase2 generates M solutions per concept.

This structured sampling strategy leverages the earlier exploration to guide the solutions more effectively. Instead of relying on unguided or purely random repeated sampling, the model systematically explores multiple reasoning trajectories guided by diverse high-level concepts or theorems. This enhances the diversity of candidate solutions, increasing the likelihood that at least one solution will be correct. We formally define the GUIDEDSAMPLING algorithm in Algorithm 1 and provide an example in Figure 4.

3.3 Training using GUIDEDSAMPLING

Synthetic data has become an increasingly effective tool for enhancing the reasoning capabilities of LLMs (Gupta et al., 2023; Mitra et al., 2024; Chaudhary et al., 2023). In particular, inference-time algorithms are valuable for generating such data when the correctness of the final solution can be programmatically verified (Zelikman et al., 2022; Singh et al., 2023; Shao et al., 2024). We demonstrate that GUIDEDSAMPLING can serve not only as an inference strategy but also as a powerful synthetic data generation mechanism.

Let x denote an input question, and $\mathcal{C} = \{c_1, \ldots, c_K\}$ be the diverse set of concepts generated for x using exploration phase of GUIDEDSAMPLING. For each concept $c_k \in \mathcal{C}$, we sample a solution $s \sim \mathcal{S}$. We define two distinct settings for constructing synthetic training pairs (x, y):

1. **Final-Answer Only (FA):** In this setting, we discard the generated concept and only use the final verified response s as the target output. This encourages the model to learn mappings from problem statements directly to correct answers, i.e. $(x, y) = (x, s)$. The corresponding training objective is the standard fine-tuning loss:

$$\mathcal{L}_{\text{FA}} = -\mathbb{E}_{(x,s)\sim\mathcal{D}_{\text{FA}}} \left[\log P_\theta(s \mid x)\right]$$

where $\mathcal{D}_{\text{FA}}$ is the dataset constructed under the FA regime and P_θ is the model's conditional distribution parameterized by θ.

2. **Idea-Augmented Answer (IAA):** In the IAA setting, we construct an enriched target sequence that includes both the conceptual diversity and the final answer. Specifically, we concatenate the concepts $\mathcal{C}$ with one selected solution s to form the training target:

$$(x, y) = (x, \texttt{concat}(\mathcal{C}, s))$$

This setting encourages the model to internalize multiple reasoning strategies before committing to one concrete solution path. The training objective becomes:

$$\mathcal{L}_{\text{IAA}} = -\mathbb{E}_{(x,\mathcal{C},s)\sim\mathcal{D}_{\text{IAA}}} \left[\log P_\theta(y \mid x)\right]$$

where $\mathcal{D}_{\text{IAA}}$ is the dataset constructed under the IAA regime. The prompt for IAA is provided in Appendix A.3.

To evaluate the effectiveness of data generated from GUIDEDSAMPLING, we first randomly sample 10,000 samples from the training set of OpenMathInstruct-2 (Toshniwal et al., 2024), a mathematical reasoning dataset. We then create reasoning chains using STaR, RS, ToT, and GUIDEDSAMPLING, and select the verified reasoning chains to create corresponding training sets.

We finetune the Llama-3.2-3B-Instruct model using these training sets and evaluate the performance across three benchmarks: MATH, GPQA-Diamond, and HumanEval. We detail the finetuning setup in Appendix B.

4 Results and Discussion

We first evaluate the effectiveness of GUIDEDSAMPLING across three different benchmarks, MATH (mathematical reasoning) (Hendrycks et al., 2021), GPQA-Diamond (scientific reasoning) (Rein et al., 2024), and HumanEval (Python code generation) (Chen et al., 2021). Experiments are conducted under a fixed inference budget of 100 calls using Llama-3.2-3B-Instruct (Grattafiori et al., 2024) and Qwen2.5-3B-Instruct (Yang et al., 2024).

***RQ1*: Does GUIDEDSAMPLING improve solution accuracy compared to Repeated Sampling under a fixed compute budget?** As shown in Figure 2, GUIDEDSAMPLING significantly outperforms RS across all benchmarks. For Llama-3.2-3B-Instruct, we observe improvements in pass@100 of 5.4% on MATH, 7.58% on GPQA-Diamond, and 27.44% on HumanEval. These results highlight that structured exploration enables more effective use of limited compute. Notably, Qwen2.5-3B-Instruct, which has mathematical pretraining, achieves the largest gains on MATH, suggesting that domain-aligned pretraining can further amplify the benefits of guided exploration.

However, the gains from GUIDEDSAMPLING are not uniform across all tasks and models. While Qwen2.5-3B-Instruct achieves strong improvements on MATH, its performance on HumanEval worsens compared to traditional RS. Upon closer analysis, this drop stems from Qwen's limited ability to generate diverse concepts for coding during the exploration phase. On average, Qwen produces only 1.13 distinct concepts per HumanEval problem, indicating that nearly all sampled solutions are guided by the same idea. This lack of diversity not only fails to leverage the core strengths of GUIDEDSAMPLING but can also dilute the model's effectiveness by forcing the model to follow a particular concept. In contrast, Llama-3.2-3B-Instruct generates 7.58 unique concepts on average on HumanEval, enabling richer exploration and stronger performance. These results underscore that successful application of GUIDEDSAMPLING depends critically on the model's ability to generate varied and relevant high-level ideas.

***RQ2*: To what extent does GUIDEDSAMPLING enhance the diversity of generated solutions?** To measure diversity, we use Qwen2.5-32B-Instruct (Yang et al., 2024) to extract the core concept or theorem used in each solution. We then compute the number of distinct concepts generated. On average, RS produces 3.54, 6.72, and 2.66 distinct concepts on MATH, GPQA-Diamond, and HumanEval, respectively. GUIDEDSAMPLING improves this diversity by an average of 17.63%.

We also found the diversity gains from GUIDEDSAMPLING to be model-specific. We find that Llama generates 3.7× more unique ideas on average compared to Qwen, with this gap ranging from 2.82× on GPQA-Diamond to 5.12× on HumanEval. This suggests model architecture and pretraining heavily influence the capacity for generating novel reasoning strategies.

***RQ3*: How does the trade-off between exploration and generation affect overall performance in GUIDEDSAMPLING?** A key design choice in GUIDEDSAMPLING is the allocation of the limited inference compute budget IC between the exploration phase (number of concepts K) and the generation phase (number of samples M per concept, where $M = IC/K$). The number of distinct concepts K directly controls this trade-off: a larger K encourages broader exploration of different approaches, but consequently reduces the compute available for generating solutions using each approach (i.e., smaller M). Conversely, a smaller K allows for more generations using fewer concepts. As demonstrated in Fig. 5, increasing exploration by increasing K initially boosts performance by uncovering more diverse, potentially successful strategies. However, beyond an optimal point, performance may decline as the generation budget M for each concept becomes insufficient to thoroughly develop any single approach.

***RQ4*: Does training language models on GUIDEDSAMPLING-generated data improve downstream task performance compared to training on data from standard sampling methods?** Models fine-tuned on data synthesized via GUIDEDSAMPLING significantly outperform those trained using data from other inference-time algorithms as illustrated in Table 1. Notably, when the models are asked to produce more responses (pass@10), a bigger improvement in performance is observed. On average, the IAA setting yields 9.18% pass@10 improvements compared to the RS,

Method	MATH		GPQA-Diamond		HumanEval	
	pass@1	pass@10	pass@1	pass@10	pass@1	pass@10
Base Model	24.00%	38.40%	11.62%	33.84%	27.44%	45.73%
STaR	37.40%	49.40%	14.14%	48.48%	52.44%	58.54%
Tree-of-Thought (ToT)	**39.40%**	62.40%	15.15%	58.08%	36.59%	53.66%
Repeated Sampling (RS)	37.60%	46.80%	19.70%	49.49%	51.83%	56.71%
GUIDEDSAMPLING (FA)	29.60%	54.60%	**20.20%**	**60.61%**	48.17%	61.59%
GUIDEDSAMPLING (IAA)	38.00%	**63.80%**	15.66%	54.55%	**53.05%**	**62.20%**

Table 1: Performance of Llama-3.2-3B-Instruct trained using different synthetic data creation strategies. FA: Just using the final answer for training the model. IAA: Using both the ideas and the corresponding final solution to create the final data.

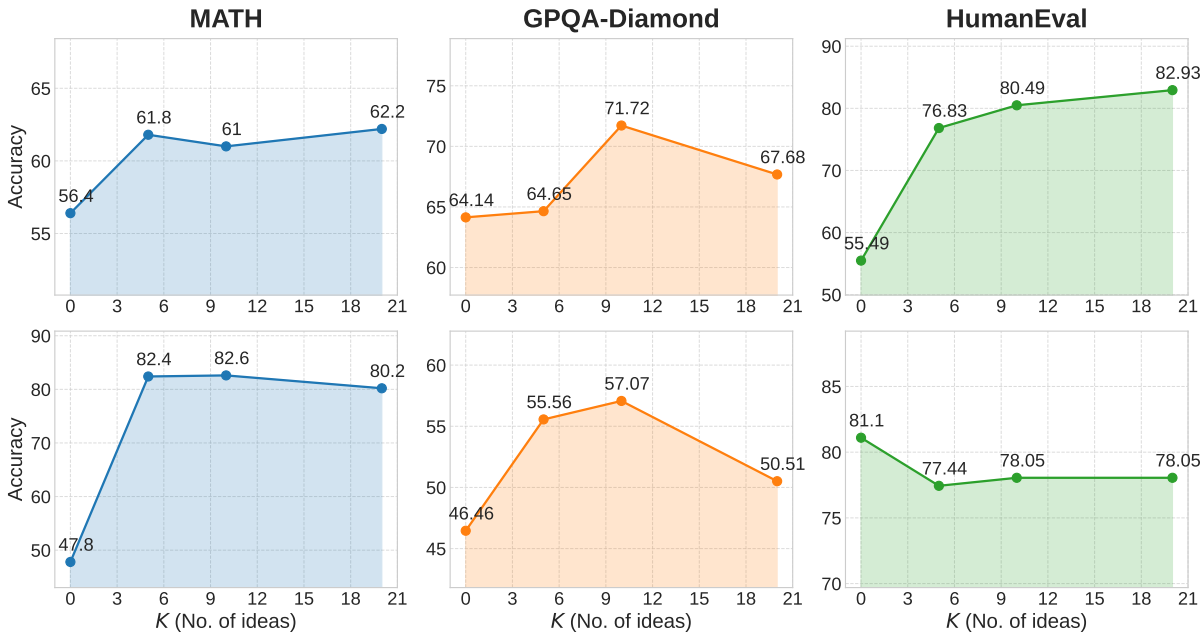

Figure 5: Pass@k performance variation with different exploration (number of ideas K) and exploitation (samples per idea M) compute allocations, given a fixed total compute of 100 calls ($M = 100/K$). Increasing exploration initially helps, but performance declines when the exploitation budget per idea becomes too small. At $K = 0$, GUIDEDSAMPLING becomes traditional RS. The first row shows results for Llama-3.1-3B-Instruct, and the second for Qwen2.5-3B-Instruct.

while FA shows 7.93% pass@10 improvements against RS. Models trained using trajectories from ToT performed better than RS but still lacked behind FA (0.89%) and IAA (2.14%).

***RQ5*: Do models fine-tuned on GUIDEDSAMPLING-generated data learn to produce more diverse reasoning strategies during inference?** To calculate the diversity of the generated samples, we use the same concept extraction method as in Section 4. We observe that diversity increases from 1.67 (RS) to 2.58 (FA) and 3.03 (IAA). Surprisingly, the largest diversity gain occurs on GPQA-Diamond rather than MATH, indicating that diversity learned through mathematical reasoning data can transfer to scientific reasoning tasks. This highlights the generalizability of the GUIDEDSAMPLING framework across domains.

5 Conclusion

We propose a new inference-time strategy, GUIDEDSAMPLING, that improves the diversity of generated solutions. The paper demonstrates how performance varies with shifting compute between the exploration of diverse concepts and the generation of final solutions and shows improvements of up to 34.6%. Furthermore, fine-tuning LLMs on trajectories generated by GUIDEDSAMPLING significantly boosts performance on mathematical reasoning and shows generalizability to other domains like scientific reasoning and code generation.

Limitations & Future Work

While our method is successful in improving the diversity of solutions generated by LLMs, it represents an early step in this area and has some limitations, including but not limited to the following:

Limited model coverage While our evaluation spans two open-source models, applying GUIDEDSAMPLING to proprietary models (e.g., GPT-4o, Gemini-2.5-Pro) remains unexplored due to high inference costs and lack of training access. Extending the method to these models is an important direction for future work.

Exploration cost vs. effectiveness trade-off Although our method improves diversity, the optimal balance between the number of concepts (K) and samples per concept (M) under a fixed compute budget remains task-specific. Developing adaptive strategies for this trade-off is a promising area.

Generality across domains Our work demonstrates promising results in mathematical, scientific, and code generation domains. However, further evaluation is needed to understand how well GUIDEDSAMPLING generalizes to more diverse domains such as legal reasoning, medical, or discovery.

Concept generation quality The success of GUIDEDSAMPLING depends on the quality and diversity of the generated concepts. Investigating techniques to improve/verify the relevance of these concepts (e.g., through external tools or feedback mechanisms) can enhance overall effectiveness.

Ethics Statement

We use AI assistants, specifically Grammarly and ChatGPT, to correct grammatical errors and restructure sentences.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Aditi Chaudhary, Karthik Raman, and Michael Bendersky. 2023. It's all relative!–a synthetic query generation approach for improving zero-shot relevance prediction. *arXiv preprint arXiv:2311.07930*.

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

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. 2024. Inference-aware fine-tuning for best-of-n sampling in large language models. *arXiv preprint arXiv:2412.15287*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

Andrew Estornell and Yang Liu. 2024. Multi-llm debate: Framework, principals, and interventions. *Advances in Neural Information Processing Systems*, 37:28938–28964.

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

Lin Gui, Cristina Gârbacea, and Victor Veitch. 2024. Bonbon alignment for large language models and the sweetness of best-of-n sampling. *arXiv preprint arXiv:2406.00832*.

Himanshu Gupta, Kevin Scaria, Ujjwala Anantheswaran, Shreyas Verma, Mihir Parmar, Saurabh Arjun Sawant, Chitta Baral, and Swaroop Mishra. 2023. Targen: Targeted data generation with large language models. *arXiv preprint arXiv:2310.17876*.

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

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei-ge Chen, Olga Vrousgos, Corby Rosset, et al. 2024. Agentinstruct: Toward generative teaching with agentic flows. *arXiv preprint arXiv:2407.03502*.

Mihir Parmar, Xin Liu, Palash Goyal, Yanfei Chen, Long Le, Swaroop Mishra, Hossein Mobahi, Jindong Gu, Zifeng Wang, Hootan Nakhost, et al. 2025. Plangen: A multi-agent framework for generating planning and reasoning trajectories for complex problem solving. *arXiv preprint arXiv:2502.16111*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code, 2023. *URL https://arxiv. org/abs/2308.12950*.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Rémi Munos. 2025. Optimizing language models for inference time objectives using reinforcement learning. *arXiv preprint arXiv:2503.19595*.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

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

Tianchun Wang, Zichuan Liu, Yuanzhou Chen, Jonathan Light, Haifeng Chen, Xiang Zhang, and Wei Cheng. 2025. Diversified sampling improves scaling llm inference. *arXiv preprint arXiv:2502.11027*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. An empirical analysis of compute-optimal inference for problem-solving with language models.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Appendix

A Prompts

A.1 Exploration Prompts

A.1.1 MATH

The following prompts were used for GUIDED-SAMPLING for the MATH (Hendrycks et al., 2021) benchmark.

MATH Initial Concept Generation

You are an expert mathematician. You will be presented with a mathematical question and your task is to identify and state one single, specific theorem or fundamental concept that is most relevant and useful for solving the problem.

QUESTION:
`{ele['question']}`

Provide only the name of the theorem or concept, or a concise statement of the principle, that is most directly applicable to solving this problem. Do not attempt to solve the original problem. Only provide the theorem or concept.

MATH Subsequent Concept Generation

You are an expert mathematician. You will be presented with a mathematical question and a list of theorems and concepts that have already been proposed as potentially useful for solving the problem. Your task is to provide a *new* and *different* theorem or concept that is most relevant and useful for solving the problem.

QUESTION:
`{ele['question']}`

EXISTING CONCEPTS:
`{ideas_text}`

Provide only the name of the theorem or concept, or a concise statement of the principle, that is most directly applicable to solving this problem. Do not attempt to solve the original problem. Only provide the theorem or concept. If no new, distinct, and useful theorem or concept can be identified, respond with "No additional concepts found."

A.1.2 GPQA-Diamond

The following prompts were used for GUIDED-SAMPLING for the GPQA-Diamond (Rein et al., 2024) benchmark.

GPQA-Diamond Initial Concept Generation

You are an expert scientist and problem solver. You will be presented with a complex, graduate-level science question and your task is to identify and state one single, specific theorem or fundamental concept that is most relevant and useful for solving the problem.

QUESTION:
`{ele['question']}{options}`

Provide only the name of the theorem or concept, or a concise statement of the principle, that is most directly applicable to solving this problem. Do not attempt to solve the original problem. Only provide the theorem or concept.

GPQA-Diamond Subsequent Concept Generation

You are an expert scientist and problem solver. You will be presented with a complex, graduate-level science question and a list of theorems and concepts that have already been proposed as potentially useful for solving the problem. Your task is to provide a *new* and *different* theorem or concept that is most relevant and useful for solving the problem.

QUESTION:
`{ele['question']}{options}`

EXISTING CONCEPTS:
`{ideas_text}`

Provide only the name of the theorem or concept, or a concise statement of the principle, that is most directly applicable to solving this problem. Do not attempt to solve the original problem. Only provide the theorem or concept. If no new, distinct, and useful theorem or concept can be identified, respond with "No additional concepts found."

A.1.3 HumanEval

The following prompts were used for GUIDED-SAMPLING for the HumanEval (Chen et al., 2021) benchmark.

HumanEval Initial Concept Generation

You are an expert python programmer. You will be presented with a programming question and your task is to identify and state one single, specific concept that is most relevant and useful for solving the problem.

QUESTION:
`{ele['question']}`

Provide only the name or short description of the concept, that is most directly applicable to solving this problem. Do not attempt to solve the original question. Only provide the concept.

HumanEval Subsequent Concept Generation

You are an expert python programmer. You will be presented with a programming question and a list of concepts that have already been proposed as potentially useful for solving the question. Your task is to provide a *new* and *different* concept that is most relevant and useful for solving the question.

QUESTION:
`{ele['question']}`

EXISTING CONCEPTS:
`{ideas_text}`

Provide only the name or the short description of the concept, that is most directly applicable to solving this problem. Do not attempt to solve the original question. Only provide the concept. If no new, distinct, and useful concept can be identified, respond with "No additional concepts found."

A.2 Concept Extraction Prompt

IAA Data

You are ConceptTagger, an expert that maps a worked-out solution (chain-of-thought or final answer) to the most specific mathematical or logical concept that makes the solution possible.

Task: For every input consisting of a reasoning explanation (a step-by-step solution, scratch-work, or short justification):
1. Read the explanation.
2. Decide which single mathematical concept, theorem, or canonical formula is essential for the solution.
3. Output that concept's standard name—nothing else.

Choose the narrowest concept that still covers the whole solution.
• Good: "Pythagorean Theorem" (precise).
• Bad: "Geometry" (too broad).
If two or more concepts appear, pick the one without which the problem cannot be solved (typically the first pivotal step).

Here are two examples:

Example 1
Problem: A right triangle has legs of lengths 5 cm and 12 cm. What is the length of the hypotenuse?

Step-by-step solution:
Step 1: Recognize this is a right triangle → apply the Pythagorean Theorem.
Step 2: hypotenuse = $\sqrt{(5^2 + 12^2)}$ = $\sqrt{(25 + 144)} = \sqrt{169} = 13cm$
Concept Used: Pythagorean Theorem

Example 2
Problem: What is the area of a rectangle with a length of 9 meters and width of 4 meters?
Step-by-step solution:
Step 1: Identify the shape as a rectangle.
Step 2: Use the area formula: Area = length × width = 9 × 4 = 36 m²
Concept Used: Area of Rectangle

Formatting Rules:
Output exactly one line with the concept name.
Use Title Case and the singular form (e.g., "Least Common Multiple", not "LCMs").
No extra punctuation, explanation, or line breaks.

A.3 IAA Prompt

IAA Data

I have a few ideas to solve this problem.
a) `{Concept 1}`
⋮
k) `{Concept k}`

To solve the problem I will use the idea i) `{Concept i}`:

`{Step by step solution}`

Final Answer
`{Final Answer}`

B Finetuning Setup

Here we define the hyperparameters that we used for fine-tuning defined in Section 3.3.

All the models were trained on 4 × A100 GPUs, with a learning rate of $5e^{-5}$ and 3 epochs. Batch size and Gradient accumulation steps were 2, and fp16 was used for all experiments. 20% of the data was split for evaluation (random seed as 21), and the checkpoint with the lowest evaluation loss was considered for reporting the results.