The First Few Tokens Are All You Need: An Efficient and Effective *Unsupervised Prefix* Fine-Tuning Method for Reasoning Models

Ke Ji[*,1,2] **, Jiahao Xu**[*,2] **, Tian Liang**[*,2] **, Qiuzhi Liu**[*,2] **, Zhiwei He**[2] **, Xingyu Chen**[2] **,**
Xiaoyuan Liu[2] **, Zhijie Wang**[2] **, Junying Chen**[1] **, Benyou Wang**[†,1] **,**
Zhaopeng Tu[†,2] **, Haitao Mi**[2] **, and Dong Yu**[2]

[1]The Chinese University of Hong Kong, Shenzhen
[2]Tencent

Abstract

Improving the reasoning capabilities of large language models (LLMs) typically requires supervised fine-tuning with labeled data or computationally expensive sampling. We introduce Unsupervised Prefix Fine-Tuning (UPFT), which leverages the observation of Prefix Self-Consistency – the shared initial reasoning steps across diverse solution trajectories – to enhance LLM reasoning efficiency. By training exclusively on the initial prefix substrings (as few as 8 tokens), UPFT removes the need for labeled data or exhaustive sampling. Experiments on reasoning benchmarks show that UPFT matches the performance of supervised methods such as Rejection Sampling Fine-Tuning, while reducing training time by 75% and sampling cost by 99%. Further analysis reveals that errors tend to appear in later stages of the reasoning process and that prefix-based training preserves the model's structural knowledge. This work demonstrates how minimal unsupervised fine-tuning can unlock substantial reasoning gains in LLMs, offering a scalable and resource-efficient alternative to conventional approaches.

1 Introduction

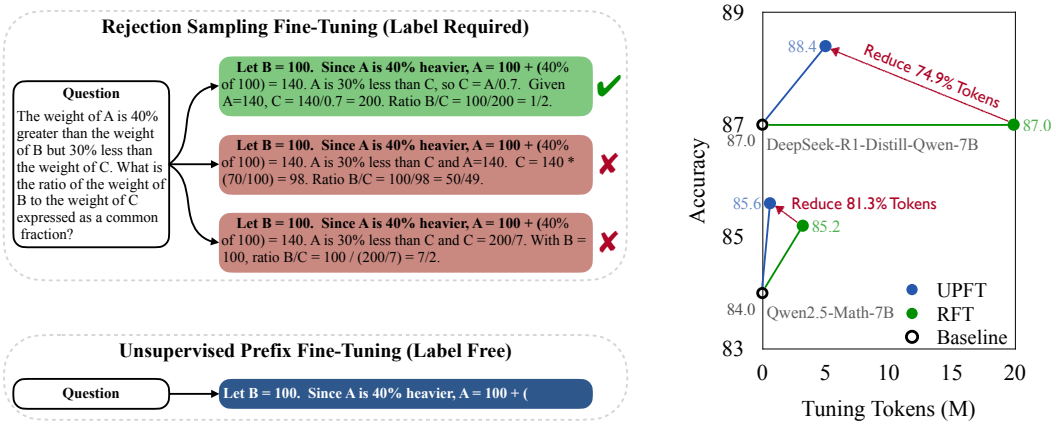
Large language models (LLMs) have demonstrated remarkable performance across a wide range of natural language understanding and generation tasks, primarily due to large-scale pre-training and subsequent instruction fine-tuning on high-quality datasets [16, 27]. Despite these successes, enabling LLMs to exhibit systematic reasoning capabilities remains a challenging endeavor [29, 4, 8, 20]. In multiple domains—from mathematical problem solving to logical and commonsense reasoning—models often rely on large amounts of human-annotated data or extensive sampling-and-filtering pipelines to achieve high accuracy.

Recent inquiry has introduced approaches such as Rejection Sampling Fine-Tuning [33] (RFT) and Self-Taught Reasoner [34] (STaR) to leverage model-generated solutions for iterative **self-improvement**, often requiring multiple candidate responses and subsequent filtering or verification steps. While these methods can yield impressive gains, they are time-consuming, resource-intensive, and assume ready access to correct targets or verification mechanisms — particularly challenging when no reliable ground-truth is available.

[*]Equal Contribution. The work was done when Ke Ji, Zhiwei He, Xingyu Chen, Xiaoyuan Liu, and Zhijie Wang were interning at Tencent.

[†]Correspondence to: Benyou Wang <wangbenyou@cuhk.edu.cn> and Zhaopeng Tu <zptu@tencent.com>.

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



(a) Fine-tuning methods (b) Token-accuracy plot on MATH500

Figure 1: (a): Conventional Rejection Sampling Fine-Tuning (RFT) method (upper panel) involves generating multiple responses to a given question and then applying posterior filtering to discard trajectories that lead to incorrect answers. Finally, the correct trajectory is used for final training. In contrast, the proposed UPFT method (bottom panel) requires only prefix minimal initial tokens of a single generated sample, eliminating the need for labeled data or rejection sampling. (b): Our proposed UPFT matches the performance of supervised RFT, while reduces tuning cost by 75+%.

In this paper, we propose an unsupervised fine-tuning method that requires only a single pass of model-generated responses per question, coupled with prefix-based fine-tuning. Our key insight is that different solution paths often share a common initial reasoning phase, which we call "Prefix Self-Consistency". By fine-tuning on these minimal prefixes (as few as 8 tokens), we effectively guide the model's inherent reasoning structures toward more systematic solution strategies while avoiding the complexity and cost of large-scale or iterative filtering. Moreover, we preserve the model's overall problem-solving format through a small proportion of full-token fine-tuning experiments, ensuring the model does not lose its length generalization and instruction-following abilities.

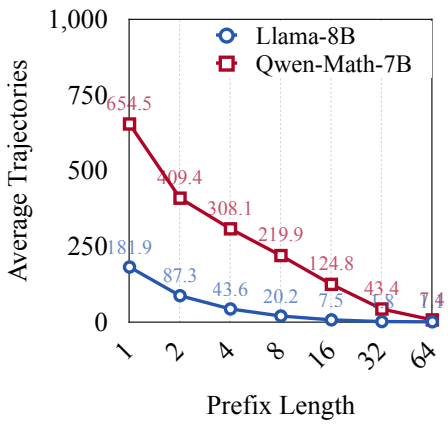
We conduct comprehensive experiments across four training corpora and evaluate our method on four widely used reasoning benchmarks. UPFT demonstrates exceptional data efficiency and flexibility, outperforming conventional full-token fine-tuning in an unsupervised sampling setting. Furthermore, UPFT achieves performance competitive with the widely used RFT method, which relies on labeled verification or large-scale rejection sampling, while requiring only 25% of the training time and 1% of the sampling time. Our approach can be easily adapted to various datasets, tasks, and LLM architectures, highlighting its flexibility and universality for developing robust problem-solving capabilities in large language models.

Our main contributions are as follows:

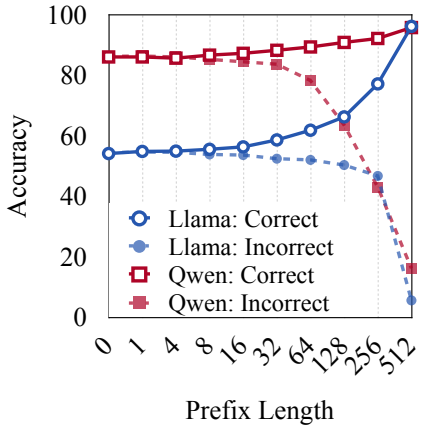
1. We identify Prefix Self-Consistency as a critical phenomenon, showing that early reasoning steps are highly consistent across trajectories, enabling efficient self-improvement learning.
2. We propose an unsupervised fine-tuning method UPFT that leverages only prefix substrings (i.e., minimal initial tokens of model-generated responses), eliminating the need for labeled data or rejection sampling.
3. We conduct comprehensive empirical validation to demonstrate that UPFT exhibits exceptional data efficiency and versatility, outperforming vanilla full-token fine-tuning in unsupervised settings and achieving competitive performance with RFT while drastically reducing sampling and tuning overhead.

2 Prefix Self-Consistency

A central observation of this work is that different solution trajectories for the same question often share a common *initial* reasoning phase, even if their later steps diverge substantially. We term this phenomenon **prefix self-consistency**. In other words, when an LLM generates multiple solutions for a single question, the opening tokens – typically restatements of the problem or the setup of its



(a) Average number of covered trajectories.



(b) Success rate of trajectory prefixes.

Figure 2: An empirical investigation of prefix self-consistency. We investigate (a) the average number of trajectories covered by prefixes at different lengths, and (b) the success rate of 32 rollout samplings from prefixes for both correct and incorrect trajectories.

initial logical steps – tend to be highly consistent across these separate responses. In this section, we identify two key characteristics of reasoning prefixes to demonstrate the existence and plausibility of prefix self-consistency. We report results on 500 questions randomly sampled from the PRM training dataset.

Early reasoning steps are highly consistent across reasoning trajectories. We sampled 1,000 trajectories for each question instance (using a temperature of 0.6) and calculated the average trajectories per prefix of different lengths, as shown in Figure 2(a). The results confirm that reasoning processes exhibit strong prefix self-consistency, with a remarkably high degree of prefix overlap among multiple trajectories for both models. As we increase t (i.e., longer prefixes), the average number of samples per prefix pattern decreases, yet both models maintain consistent patterns well beyond the very first tokens. Notably, the math-specialized Qwen-Math-7B-Instruct preserves shared prefixes more consistently than the general-purpose Llama-3.1-8B-Instruct, as evidenced by the higher values in all cases. This suggests that Qwen-Math-7B's generation process is more tightly anchored in its initial reasoning steps, whereas Llama-3.1-8B introduces more prefix variability. An example is shown in Appendix D.

Errors predominantly occur in later reasoning steps. To empirically validate the plausibility of the reasoning prefix, we conducted 32 rollout samplings for each token in both a correct and an incorrect trajectory for each question. Figure 2(b) illustrates the success rate of these rollout samplings across both correct and incorrect trajectories. Two key observations can be made:

- *Incorrect trajectories diverge significantly from correct ones in later reasoning steps.* The rollout success rate for correct trajectories steadily rises as t grows for both models. For example, with Llama, the success rate starts at 54.2% at $t = 0$ and reaches 96.2% by $t = 512$. This trend is expected: sampling from the later tokens of a correct trajectory leverages more accurate contextual information, increasing the likelihood of staying on a correct path. In contrast, the rollout success rate for incorrect trajectories declines substantially as t increases, indicating that once an incorrect path is taken, recovering through rollout sampling becomes far less likely. This contrast highlights that errors tend to occur in the later stages of generation.
- *Initial reasoning prefixes exhibit self-consistency.* At the earliest token positions ($t \leq 16$), the rollout success rates for both correct and incorrect trajectories are strikingly similar. For Qwen-Math-7B at $t = 4$, the success rates are 85.7% and 86.1% for correct and incorrect trajectories, respectively. This near-identical performance in early steps suggests that the initial tokens generated – whether they lead to a correct or incorrect final answer – are statistically indistinguishable in terms of leading to a correct result when used as a starting point for rollout sampling.

Overall, these results indicate that while mistakes in reasoning are more prone to appear and accumulate in later steps, the initial reasoning prefixes generated by LLMs exhibit a considerable degree of self-consistency. This consistency is reflected in the similar early-stage rollout success rates across both correct and incorrect trajectories. Consequently, enhancing the accuracy and robustness of these early reasoning steps may be a key factor in improving the overall reliability of reasoning tasks.

3 Methodology

Based on the aforementioned observation, we firstly provide a Bayesian explanation in Section 3.1, identify the coverage and accuracy of **Prefix Self-Consistency** in Section 3.2, and propose our UPFT in Section 3.3.

3.1 Understanding Coverage and Accuracy from A Bayesian Perspective

We demonstrate that the SFT process can be naturally interpreted within a Bayesian framework. Consider a dataset $(x, y) \sim \mathcal{D}$, where each x represents an input and y is the corresponding ground-truth answer. For each input x, we conduct the reject sampling process of SFT aims to maximize the following objective:

$$\mathbb{E}_{\substack{(x,y)\sim\mathcal{D},\ r^{(k)}\sim p(\cdot|x;\theta) \\ r\in_R \mathbb{1}(r^{(k)},y)}} \log p(r|x) \tag{1}$$

where $r^{(k)} \sim p(\cdot|x;\theta)$ denotes the generated reasoning trace, and $r \in_R \mathbb{1}(r^{(k)}, y)$ denotes the rejection sampling, which only picks the reasoning trace r whose final answer is y.

From a probabilistic perspective, we can decompose the conditional probability $p(y|x)$ by marginalizing over all possible reasoning traces r that could lead to the answer, and derive the following lower bound:

$$\begin{aligned}
\log p(y|x) &= \log \sum_r p(y, r|x) && (2)\\
&= \log \sum_r p(y|r, x)p(r|x) && (3)\\
&\geq \sum_r p(r|x) \log p(y|r, x) && (4)\\
&= \mathbb{E}_{r\sim p(\cdot|x)} \log p(y|r, x) && (5)
\end{aligned}$$

The correct answer $\log p(y|x)$ is lower-bounded by:

- **Coverage**: The expectation $\mathbb{E}_{r\sim p(\cdot|x)}[\cdot]$ is with respect to the distribution of reasoning trace $p(r|x)$. The term $p(r|x)$ denotes a prior probability of reasoning trace r given input x, which denotes the prefix coverage of the entire reasoning trace space. This corresponds to the average trajectory covered by prefixes in Section 2 indicated by Figure 2(a).
- **Accuracy**: Term $p(y|r, x)$ within the expectation can be viewed as the likelihood of the answer y being correct given a specific reasoning trace r, which is the accuracy. This is also observed by our previous observation in Section 2 in Figure 2(b).

3.2 Prefix Coverage and Accuracy

To incorporate the concept of prefix spans, we utilize the chain rule for conditional probabilities. We can decompose the probability of the full reasoning trace r given input x as:

$$p(r|x) = p(r_{<t}, r_{\geq t}|x) = p(r_{<t}|x)p(r_{\geq t}|r_{<t}, x)$$

where $r_{<t}$ denotes the prefix of r before time step t. Substituting this decomposition of $p(r|x)$ into the original lower bound from Equation (4), we get:

$$
\begin{aligned}
\log p(y|x) &\geq \sum_{r} p(r|x) \log p(y|r,x) \\
&= \sum_{r} p(r_{<t}|x) p(r_{\geq t}|r_{<t},x) \log p(y|r_{<t},r_{\geq t},x) \\
&= \sum_{r_{<t}} \left[p(r_{<t}|x) \sum_{r_{\geq t}} p(r_{\geq t}|r_{<t},x) \log p(y|r_{<t},r_{\geq t},x) \right] \\
&= \sum_{r_{<t}} p(r_{<t}|x) \left[\sum_{r_{\geq t}} p(r_{\geq t}|r_{<t},x) \log p(y|r,x) \right] \\
&= \sum_{r_{<t}} p(r_{<t}|x) L(r_{<t},x) \\
&= \mathbb{E}_{r_{<t} \sim p(\cdot|x)}[L(r_{<t},x)] \qquad (6)
\end{aligned}
$$

where,

$$
\begin{aligned}
L(r_{<t},x) &= \sum_{r_{\geq t}} p(r_{\geq t}|r_{<t},x) \log p(y|r_{<t},r_{\geq t},x) \\
&= \mathbb{E}_{r_{\geq t} \sim p(\cdot|r_{<t},x)}[\log p(y|r_{<t},r_{\geq t},x)] \qquad (7)
\end{aligned}
$$

Equation (6) presents the original lower bound of Equation (5) in terms of prefix spans of the reasoning trace. It also states the importance of coverage and accuracy of the prefix span $r_{<t}$:

- **Prefix Coverage**: The outer expectation in Equation (6) $\mathbb{E}_{r_{<t} \sim p(\cdot|x)}[\cdot]$ is with respect to the distribution of prefixes $p(r_{<t}|x)$. The term $p(r_{<t}|x)$ represents the prior probability of the model generating a prefix reasoning trace $r_{<t}$ given the input x. This denotes the coverage of prefix reasoning traces.
- **Prefix Accuracy**: For each prefix $r_{<t}$, the term $L(r_{<t},x)$ in Equation (7) is the conditional lower bound given the prefix $r_{<t}$. It is the expected log-likelihood of the answer y given that the reasoning process starts with prefix $r_{<t}$, averaging over all possible suffixes $r_{\geq t}$ that can follow $r_{<t}$. This can be viewed as the expected accuracy when reasoning is conditioned to start with the prefix $r_{<t}$.

So far, we have mathematically derived a representation of the lower bound in terms of prefix spans for both prefix coverage and prefix accuracy. To achieve superior performance through prefix fine-tuning, a specific prefix length t is required to trade off both prefix coverage and prefix accuracy terms.

3.3 Our Method: Unsupervised Prefix Fine-Tuning

Motivated by previous theoretical justification, we develop our method Unsupervised Prefix Fine-Tuning (UPFT), which is apt to maximize the coverage of the reasoning trace while maintaining a relatively high accuracy by only learning from the proper prefix of the reasoning trace. This strategy shares several advantages. First, *UPFT exhibits enhanced coverage of all possible correct reasoning traces.* As demonstrated in Section 2 and Section 3.1, the prefix of the reasoning traces represents a set of reasoning traces with the identical prefix, which increases the coverage term of Equation (5)'s lower bound. Second, *by solely decoding a certain length of the prefix, UPFT inherently enjoys improved computational efficiency.* In contrast, conventional Reasoning From Traces (RFT) methods necessitate rejecting sampling over entire reasoning traces. This approach incurs a significant inference burden, particularly in scenarios where generating valid full reasoning traces from the base model proves challenging.

Our strategy consists of the following steps:

- Given a training set $(x, y) \in \mathcal{D}$, we only decode a prefix span $r_{<t}$ of reasoning trace from the base model $r_{<t} \sim p(\cdot|x;\theta)$;
- We conduct the SFT learning on the prefix spans of the reasoning trace $r_{<t}$ with NLL objective;

Structure Tuning To avoid the catastrophic forgetting [19], we jointly conduct the prefix tuning and full trace tuning by a data split ratio $p\%$, which yields a prefix dataset $\mathcal{D}_p$ and full trace dataset $\mathcal{D}_f$. We utilize a specialized task template, as shown in Appendix A, for prefix tuning to enhance the model's ability to learn effectively from prefix-specific patterns. Note that the full trace tuning does not require correctness verification as well. In summary, the overall objective is a combination of UPFT and unsupervised SFT by maximizing the following objective:

$$\mathbb{E}_{\substack{x\sim\mathcal{D}_p \\ r_{<t}\sim p(\cdot|x;\theta)}} [\log p(r_{<t}|x;\theta)] + \mathbb{E}_{\substack{x\sim\mathcal{D}_f \\ r\sim p(\cdot|x;\theta)}} [\log p(r|x;\theta)] \tag{8}$$

Notably, compared to conventional RFT objective, our method does not require any rejection sampling process denoted by $\mathbb{1}(r^{(k)}, y)$ in Equation (1). Consequently, our method is naturally suitable for learning from unproven questions with extreme data efficiency.

4 Experiment

In this experiment, we compare UPFT with traditional supervised methods such as SFT and RFT in both supervised and unsupervised sampling settings. We also evaluate its scalability by varying the self-training corpora and backbone models.

4.1 Experiments Setup

Backbone LLMs For our experiments, we selected three representative open-source backbone models: the general-purpose Llama-3.1-8B-Instruct [4], the math-specialized Qwen2.5-Math-7B-Instruct [29] optimized for mathematical tasks, and the long-reasoning DeepSeek-R1-Distill-Qwen-7B [8], distilled from DeepSeek-R1.

Fine-Tuning We used four datasets to generate self-training data for fine-tuning:

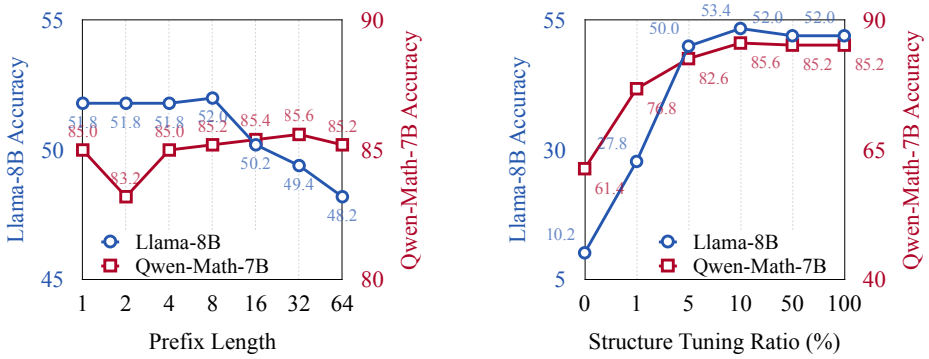
1. **PRM** (12K instances; 15): This dataset includes 4.5K MATH [10] test problems, which are drawn from the PRM800K training set.
2. **OMI2** (600K instances; 26): This is a subset of 600K unique questions extracted from the OpenMathInstruct2 math-instruction tuning dataset.
3. **LIMO** (819 instances; 31): A high-quality training dataset specifically focused on challenging problems.
4. **U-Hard** (100K questions): This dataset, introduced in this work, is designed to explore the potential of our method in an unsupervised setting.

U-Hard was curated through an extensive collection of questions from publicly available online sources. Adhering to the Omni-Math approach [6], we labeled the collected data by difficulty and subsequently filtered out lower-difficulty questions. This process resulted in a dataset composed exclusively of challenging questions, thereby ensuring U-Hard's practical relevance to real-world scenarios. To ensure a fair comparison, UPFT employs the same hyperparameters as conventional SFT when fine-tuning models.Details about hyperparameters is shown in Appendix B. During the inference stage, we adopt a prompted zero-shot setup and use standard greedy decoding, wherein models are directed to answer each question using natural language instructions without any accompanying contextual demonstrations.

Benchmarks We evaluated the performance of our method and the baseline methods on four widely used benchmarks: **GSM8K** [3], **MATH500** [10], **AIME24** [18], and **GPQA Diamond** [24].

Tuning Scenarios We employed two experimental settings:

- **Unsupervised sampling**: we took one sample per question without any posterior filtering.
- **Supervised sampling**: Following RFT [33], we sampled the responses for each question 16 times and used the ground truth answers to select a correct response.



(a) Impact of Prefix Length (b) Impact of Structure Tuning Ratio

Figure 3: Impact of (a) prefix length and (b) structure tuning ratio on reasoning accuracy.

4.2 Ablation Study

In this section, we evaluated the impact of two hyperparameters of UPFT on the PRM-12K dataset using Llama-3.1-8B-Instruct and Qwen2.5-Math-7B-Instruct in the unsupervised setting and report the results on the MATH500 test set.

Impact of Prefix Length Figure 3(a) illustrates how prefix length affects reasoning accuracy. Each model has its own optimal prefix length for peak performance: Llama-3.1-8B-Instruct achieves its highest average accuracy (52.0%) with an 8-token prefix, whereas Qwen2.5-Math-7B-Instruct remains stable over prefix lengths from 8 to 32 tokens. The math-specialized Qwen2.5-Math-7B-Instruct is less sensitive to prefix length, likely because it was trained on the high-quality large-scale math corpus, which reinforces its stability on the MATH500 test set. For subsequent experiments, we set Llama-3.1-8B-Instruct to 8 tokens and Qwen2.5-Math-7B-Instruct to 32 tokens. In the following experiments, we set the prefix length to 8 for Llama-3.1-8B-Instruct and 32 for Qwen2.5-Math-7B-Instruct. Because the responses from the long-reasoning DeepSeek-R1-Distill-Qwen-7B are generally more than four times longer than those of Qwen2.5-Math-7B-Instruct, we use a prefix length of 128 tokens for the former.

Impact of Structure Tuning Ratio Figure 3(b) plots the effect of the structure tuning ratio (p). The performance of both models generally improves with increasing p, peaking at $p = 10\%$. This result supports our hypothesis that even minimal structural supervision effectively guides models towards generating complete responses. However, exceeding this ratio leads to a performance decrease. Consequently, we set p to 0.1 in UPFT for all datasets, except LIMO. For the smaller LIMO dataset, we increased p to 0.3 for structure tuning.

4.3 Unsupervised Fine-Tuning

Table 1 presents the results for the unsupervised sampling setting, where the data is not filtered based on the correctness of the extracted answer. We compare these outcomes to those obtained using conventional SFT [21], which fine-tunes models on training data without any self-improvement or test-time verification.

UPFT demonstrates superior performance compared to SFT in unsupervised fine-tuning. UPFT consistently outperforms conventional SFT across various self-training datasets and backbone models under the unsupervised sampling setting. For instance, using the U-Hard dataset with Qwen2.5-Math-7B-Instruct, UPFT achieves an average accuracy of 54.5% across all benchmarks, exceeding the 51.3% attained by conventional SFT. Likewise, with DeepSeek-R1-Distill-Qwen-7B and U-Hard, UPFT attains an average of 61.6%, whereas SFT achieves 56.4%. These results highlight UPFT's effectiveness in leveraging unsupervised data to enhance reasoning, thereby reducing the reliance on labeled data. They also indicate the versatility and broad applicability of UPFT across

Table 1: Model performance under the **unsupervised sampling setting**, without any filtering based on the correctness of the final answer. "Length" denotes the average length of the tuning samples in each dataset. Models trained with SFT and the proposed UPFT generate responses of similar length.

Fine-Tuning			Testsets				
Method	**Data**	**Avg. Length**	**GSM8K**	**MATH500**	**AIME2024**	**GPQA**	**Ave.**
Llama-3.1-8B-Instruct			82.0	51.0	3.3	8.6	36.2
+ SFT	PRM	175.8	83.8	48.4	3.3	8.6	36.0
+ UPFT	(12K)	15.8	**85.4**	**52.0**	**6.7**	**9.1**	**38.3**
Qwen2.5-Math-7B-Instruct			95.2	84.0	16.7	9.6	51.4
+ SFT	PRM	300.1	95.8	83.4	13.3	9.1	50.4
+ UPFT	(12K)	51.4	95.5	85.6	20.0	9.6	**52.6**
+ SFT	OMI2	533.2	95.4	83.4	13.3	6.6	49.7
+ UPFT	(600K)	67.5	95.4	**86.4**	20.0	9.6	**52.9**
+ SFT	LIMO	491.8	95.8	84.2	20.0	7.6	51.9
+ UPFT	(0.8K)	77.8	95.6	85.8	20.0	8.6	52.5
+ SFT	U-Hard	393.3	95.5	83.4	16.7	9.6	51.3
+ UPFT	(100K)	68.2	**96.0**	85.6	**26.6**	9.6	**54.5**
DeepSeek-R1-Distill-Qwen-7B			88.6	87.0	40.0	13.1	57.2
+ SFT	LIMO	2029.5	89.7	87.0	40.0	12.1	57.2
+ UPFT	(0.8K)	757.7	**92.0**	**89.4**	43.3	**17.7**	60.6
+ SFT	U-Hard	3440.4	89.7	87.0	36.7	12.1	56.4
+ UPFT	(100K)	561.7	91.4	89.2	**50.0**	15.7	**61.6**

different LLM architectures, supporting its claim of being a flexible, universal method for improving reasoning capabilities.

The benefits of UPFT are more pronounced on complex reasoning tasks. The performance gains of UPFT over conventional SFT are especially evident on more challenging benchmarks such as AIME2024 and GPQA. For example, on AIME2024 with the U-Hard dataset, UPFT achieves 26.6% accuracy with Qwen2.5-Math-7B-Instruct, a notable improvement over the 16.7% achieved by conventional SFT. A similar trend is observed with DeepSeek-R1-Distill-Qwen-7B on AIME2024, where UPFT reaches 50.0% compared with SFT's 36.7%, showing UPFT's capability to improve reasoning on difficult problems, aligning with our claim of enhancing reasoning capabilities effectively and efficiently.

The U-Hard dataset maximizes UPFT's potential through difficulty-focused curation. Training with U-Hard yields the strongest performance improvements across models, particularly for complex benchmarks. Qwen2.5-Math-7B achieves 26.6% on AIME2024 with U-Hard - 10 points higher than with PRM data. This demonstrates that challenging questions provide richer signals for prefix-based self-improvement, as they demand more sophisticated initial reasoning setups. Using U-Hard, UPFT achieves the highest average accuracy with both Qwen2.5-Math-7B-Instruct and DeepSeek-R1-Distill-Qwen-7B, surpassing other datasets such as PRM-12K and OMI2-600K. These findings demonstrate that UPFT effectively leverages diverse and challenging questions to refine the model's reasoning skills in an unsupervised manner, showcasing its data efficiency and versatility.

UPFT achieves dramatic efficiency gains through minimal token training. UPFT reduces training sequence length by 82.6-94.7% compared to SFT across datasets, with U-Hard samples averaging 68.2 tokens vs. SFT's 393.3. This directly translates to 6.3-16.7x faster training iterations and reduced memory consumption. Notably, DeepSeek-R1-Distill-Qwen-7B attains better performance with UPFT's 561-token samples than SFT's 3,440-token sequences, proving that strategic prefix selection preserves critical learning signals. These efficiency characteristics validate UPFT's practical value for resource-constrained applications while maintaining or exceeding baseline performance.

Table 2: Model performance under the **supervised sampling setting** on the PRM-12K dataset, with filtering 16 sampled solutions based on the correct extract answer. "#Tokens" denote the number of tokens in each phase. Compared to RFT, V-STaR requires two additional samples to tune the verifier.

Method	#Tokens		Testsets				
	Sampling	Tuning	GSM8K	MATH500	AIME2024	GPQA	Ave.
Llama-3.1-8B-Instruct			82.0	51.0	3.3	8.6	36.2
+ RFT	36.9M	2.3M	86.0	52.0	6.7	9.1	38.5
+ V-STaR		6.8M	85.4	52.6	6.7	8.6	38.3
+ UPFT (*Ours*)	0.2M		85.4	52.0	6.7	9.1	38.3
+ Lable Filter	36.9M	0.2M	85.8	53.4	6.7	9.1	**38.8**
Qwen2.5-Math-7B-Instruct			95.2	84.0	16.7	9.6	51.4
+ RFT	51.7M	3.2M	95.7	85.2	20.0	9.6	52.6
+ V-STaR		9.6M	96.0	85.4	20.0	10.1	**52.9**
+ UPFT (*Ours*)	0.6M		95.5	85.6	20.0	9.6	52.6
+ Lable Filter	51.7M	0.6M	96.0	85.6	20.0	10.1	**52.9**
DeepSeek-R1-Distill-Qwen-7B			88.6	87.0	40.0	13.1	57.2
+ RFT	318.0M	19.9M	90.7	87.0	40.0	11.1	57.2
+ UPFT (*Ours*)	5.0M		91.9	88.4	40.0	14.6	58.7
+ Lable Filter	318.0M	4.5M	92.3	89.2	40.0	13.6	**58.8**

4.4 Supervised Fine-Tuning

In the supervised sampling setting, we compare our approach with two SFT variants in Table 2:

- RFT [33] samples 16 candidate responses with a label filter to identify a correct one.
- V-STaR [11] produces 16 responses for each tuning instance, trains a verifier on them for one iteration, and uses it at test time to select the answer from 4 completions.

UPFT achieves competitive performance with supervised methods while requiring significantly fewer tokens in both sampling and tuning. Across multiple model architectures, UPFT matches or exceeds the performance of supervised baselines such as RFT and V-STaR. On the Llama-3.1-8B-Instruct backbone, UPFT attains an average reasoning benchmark score of 38.3%, matching V-STaR (38.3%) and approaching RFT (38.5%). For Qwen2.5-Math-7B-Instruct, UPFT achieves identical average accuracy to RFT (52.6%), while using only 1.2% of the sampling tokens (0.6M vs. 51.7M). Most notably, UPFT enables DeepSeek-R1 to achieve 58.7% average accuracy, which is 1.5 points higher than RFT, while requiring 16$\times$ fewer tuning tokens than RFT. This underscores how prefix-based fine-tuning effectively captures critical reasoning patterns without incurring the computational overhead of sampling 16 responses per question, as required by supervised baselines. Moreover, UPFT maintains its strong performance even when ground-truth answers are available. By focusing on the shared reasoning prefixes, it does not sacrifice accuracy. These results reinforce our core claim: prefix-based training captures the essential reasoning signals available in full trajectories, validating our Prefix Self-Consistency hypothesis. This hypothesis posits that the essential signals for effective reasoning are largely contained within the initial prefixes of successful solution trajectories, allowing for efficient learning and high performance.

UPFT offers seamless integration with label verification for enhanced accuracy. An additional benefit of UPFT is its inherent flexibility and compatibility with label verification techniques when such information is accessible. As demonstrated in the table's last rows for each model backbone, when UPFT is enhanced with label filtering, it attains 58.8% average accuracy on DeepSeek-7B, surpassing all baselines while maintaining a 4.4$\times$ tuning token advantage over RFT. For Qwen2.5-Math-7B-Instruct, we match V-STaR's 52.9% peak performance without requiring compute-intensive verifier training or best-of-N inference. This dual capability demonstrates UPFT's unique adaptability - it functions as a standalone unsupervised learner while seamlessly integrating supervision when available, making it practical for real-world deployment across the supervision spectrum.

5 Related Work

Unsupervised Learning Unsupervised learning encompasses diverse methodologies, including pseudo-labeling [30], pivot-based approaches [22], and adversarial networks [5]. Self-supervised learning has since emerged as a dominant paradigm, particularly for language models, leading to the success of pre-trained models like BERT [14] and others [9, 23]. More recently, self-improvement techniques, such as self-rewarding [2, 32], further advance unsupervised learning by enabling models to iteratively enhance performance without external supervision. Building upon self-improvement, AL [13] extends this paradigm to unsupervised domain adaptation.

Self-Improvement A family of methods, starting with STaR [34], reinforced self-training [7], and rejection fine-tuning [33], leverages the solutions generated by large language models (LLMs) to iteratively update and improve the models themselves. These techniques involve fine-tuning the model on the generated solutions that produce correct answers. ReST^{EM} [25] views this fine-tuning process as expectation-maximization-based reinforcement learning (RL) for a solution-generating agent. Wang et al. [28] propose using a contrastive loss to enhance the likelihood of correct solutions over incorrect ones. The discovery of successful solutions presents a significant exploration challenge. Luong et al. [17] demonstrated that RL-based fine-tuning of an LLM is particularly difficult unless preceded by several steps of supervised fine-tuning. In An et al. [1], a more powerful LLM was employed to edit the incorrect rationales generated by a smaller model, thereby providing positive data for its fine-tuning. However, Huang et al. [12] argued that LLMs have limited capacity to correct their own reasoning flaws.

In contrast to the above approaches, this paper focuses on discovering self-supervised signals specifically for mathematical reasoning. We introduce UPFT, a simple and effective unsupervised post-training method requiring only questions and the LLM itself.

6 Conclusion

In this work, we presented an unsupervised fine-tuning method that enhances the reasoning capabilities of large language models using only prefix substrings as minimal guidance. Our approach leverages the inherent reasoning structures within pretrained models, exploiting the phenomenon of Prefix Self-Consistency where different reasoning trajectories share common prefixes. Extensive experiments demonstrated that our method outperforms traditional full-token fine-tuning and achieves performance comparable to supervised approaches like RFT, with significantly reduced training and inference times. This work highlights the potential of minimal unsupervised fine-tuning in improving the reasoning abilities of LLMs without relying on external supervision or extensive computational resources. Future work will explore the application of this method to other challenging tasks and investigate the theoretical underpinnings of Prefix Self-Consistency in more depth.

Acknowledgement

This work was supported by Major Frontier Exploration Program (Grant No. C10120250085) from the Shenzhen Medical Academy of Research and Translation (SMART), the Shenzhen Science and Technology Program (JCYJ20220818103001002), NSFC grant 72495131, Shenzhen Doctoral Startup Funding (RCBS20221008093330065), Tianyuan Fund for Mathematics of National Natural Science Foundation of China (NSFC) (12326608), Shenzhen Science and Technology Program (Shenzhen Key Laboratory Grant No. ZDSYS20230626091302006), and Shenzhen Stability Science Program 2023.

References

[1] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

[2] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *ICML*, 2024.

[3] 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.

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

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[6] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

[7] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[8] 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.

[9] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2: 225–250, 2021.

[10] 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.

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

[12] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[13] Ke Ji, Junying Chen, Anningzhe Gao, Wenya Xie, Xiang Wan, and Benyou Wang. Llms could autonomously learn without external supervision. *arXiv preprint arXiv:2406.00606*, 2024.

[14] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[15] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. 2023.

[17] 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*, 2024.

[18] MAA Committees. Aime problems and solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

[19] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[20] OpenAI. Gpt-4 technical report, 2023.

[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] Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pp. 751–760, 2010.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[24] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[25] Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

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

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

[28] Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. Making large language models better reasoners with alignment. *arXiv preprint arXiv:2309.02144*, 2023.

[29] 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.

[30] Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7386–7399, 2020.

[31] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

[32] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

[33] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[34] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. *Neural Information Processing Systems (NeurIPS)*, 2022.

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: Our abstract and introduction clearly describe our contribution, the algorithm, and the experimental results.

   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 mentioned our limitation in Appendix C.

   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: We mentioned our Theory Assumptions in Section 2 and Section 3.

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: We provide a detailed description of our model architecture and present all the training details in Section 4 and Appendix B, including dataset and hyperparameter settings.

   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: We provide our code in the supplemental 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: We provide the full details in Section 4 and Appendix B.

   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: Yes, we mentioned in the Section 4 and Appendix B.

   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: Yes, we mentioned in the Section 4 and Appendix B.

   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: This paper strictly adheres to 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: [NA]

    Justification: This work is foundational research and not tied to particular application.

    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: This 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: Yes, the paper properly credits the creators or original owners of assets (e.g.,code, data, models) and explicitly mentions and respects the relevant licenses and terms of use.

    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: Yes, we mentioned in the Section 4.

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: This 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: This 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: [Yes]

Justification: Our paper investigates the issue of efficient self-training in large language models, conducting research based on multiple open-source large language model backbone.

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 Settings of the Experiments

A.1 Task Template for Prefix Tuning

For the prefix fine-tuning, we simply apply the task template shown in Figure 4.

Task template used for prefix tuning.

```
[question] Please provide the initial step towards resolving the question.
This step may serve as a foundation but might not encompass the entire
solution.
```

Figure 4: The task template used to learn from the prefix of the reasoning traces. [question] represents the question that needs to be answered.

B Hyperparameters

Table 3: The hyperparameters used for our method on all training corpora.

Hyperparam.	**PRM-12K**	**OMI2-60K**	**LIMO**	**U-Hard**
Optimizer	AdamW			
Warmup Ratio	0.03			
Learning Rate	1e-6		2e-6	1e-6
LR Schedule	constant_with_warmup			
Batch Size	1			
Gradient Step	8			
Max Length	4096		16384	
# Epoch	2	1	3	1

To ensure a fair comparison, UPFT employs the same hyperparameters as conventional SFT when fine-tuning models, as listed in Table 3. During the inference stage, we adopt a prompted zero-shot setup and use standard greedy decoding, wherein models are directed to answer each question using natural language instructions without any accompanying contextual demonstrations.

C Limitations

- **Validation with larger-scale models:** In our experiments, the choice of backbone was primarily based on the mathematical capabilities of the model. Our goal is to further enhance the model's reasoning ability at a low cost. In future research within the RL framework, we will consider validating our approach on larger-scale models, such as QwQ-32B-Preview and DeepSeek-R1-Distill-Qwen-32B.
- **Study on sample-independent prefix selection strategy:** Despite achieving satisfactory performance, this work is based on a heuristic prefix selection strategy. In future research, we will continue to design an adaptive sample-independent prefix selection for UPFT.

D Case study of Prefix Data

We provide a case study of prefix data shown in Figure 5. We can see that the different 16 rollout samples have very similar prefixes.

Case Study

Question: Melinda has three empty boxes and 12 textbooks, three of which are mathematics textbooks. One box will hold any three of her textbooks, one will hold any four of her textbooks, and one will hold any five of her textbooks. If Melinda packs her textbooks into these boxes in random order, the probability that all three mathematics textbooks end up in the same box can be written as $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m+n$.

The first 32 words (prefix substring) of all answers:

A1: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A2: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to pack the textbooks

A3: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A4: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A5: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A6: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A7: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A8: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A9: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A10: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A11: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A12: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to distribute the 12

A13: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A14: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to pack the textbooks

A15: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A16: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

Figure 5: With the temperature set to 0.7, we sample 16 times based on Qwen2.5-Math-7B-Instruct for the given question, where A1-A16 represents the corresponding output results.