LIGHTREASONER: CAN SMALL LANGUAGE MODELS TEACH LARGE LANGUAGE MODELS REASONING?

Anonymous authors
Paper under double-blind review

ABSTRACT

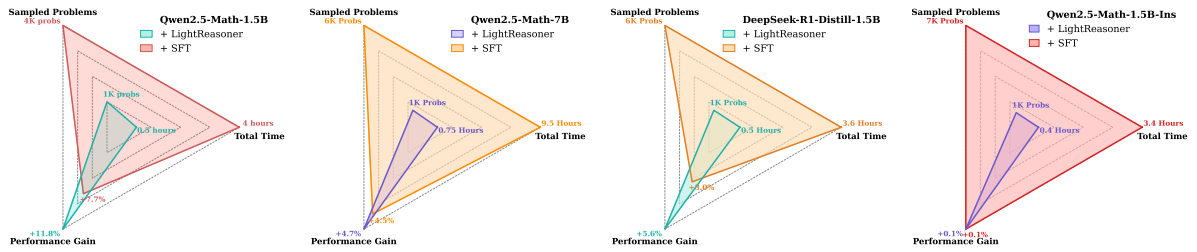
Large language models (LLMs) have demonstrated remarkable progress in reasoning, often through supervised fine-tuning (SFT). However, SFT is resource-intensive, relying on large curated datasets, rejection-sampled demonstrations, and uniform optimization across all tokens—even though only a fraction carry meaningful learning value. In this work, we explore a counterintuitive idea: can smaller language models (SLMs) teach larger language models (LLMs) by revealing high-value reasoning moments that reflect the latter's unique strength? We propose **LightReasoner**, a novel framework that leverages the behavioral divergence between a stronger *expert* model (LLM) and a weaker *amateur* model (SLM). LightReasoner operates in two stages: (1) a *sampling stage* that pinpoints critical reasoning moments and constructs supervision examples capturing the expert's advantage through expert–amateur contrast, and (2) a fine-tuning stage that aligns the expert model with these distilled examples, amplifying its reasoning strengths. Across seven mathematical benchmarks, LightReasoner improves accuracy by up to 28.1%, while reducing time consumption by 90%, sampled problems by 80%, and tuned token usage by 99%, all without relying on ground-truth labels. By turning weaker SLMs into effective teaching signals, LightReasoner offers a scalable and resource-efficient approach for advancing LLM reasoning. For reproducibility, the implementation is available at: https://anonymous.4open.science/r/LightReasoner.

1 INTRODUCTION

Driven by large-scale pre-training and various post-training strategies, large language models (LLMs) have achieved remarkable progress in language understanding and generation (Kaplan et al., 2020; Touvron et al., 2023). However, they continue to struggle with systematic reasoning tasks that demand step-by-step logical precision, particularly in mathematics. A prevalent strategy to address this limitation is to align models with demonstrations of correct problem-solving trajectories. To this end, recent approaches (Yang et al., 2024; Guo et al., 2025) often integrate rejection sampling (Yuan et al., 2023) into supervised fine-tuning (SFT), reducing the direct reliance on fully human-curated data. While effective, this strategy is resource-intensive: it requires generating multiple candidate solutions, filtering them against ground-truth answers to retain only correct trajectories, and then fine-tuning on every token of those trajectories, treating trivial and crucial reasoning steps as equally valuable.

These limitations have motivated researchers to explore more targeted strategies for strengthening LLM reasoning. As early as the Chain-of-Thought paper (Wei et al., 2022), it was shown that LLMs acquire latent reasoning abilities during pre-training, which can be elicited through appropriate prompting. Complementary to this, recent works (Zhao et al., 2025; Prabhudesai et al., 2025) show that LLMs can improve by exploiting internal feedback signals, such as self-certainty, instead of relying on external rewards or labeled data. Another line of research (Lin et al., 2024a) finds that during pre-training, learning progress is disproportionately concentrated on a small subset of tokens. Instead of uniformly training on entire trajectories, selectively optimizing these high-impact tokens offers significant gains in both efficiency and performance. Together, these insights highlight the promise of harnessing natural learning dynamics to improve LLM reasoning.

Motivated by these insights, we pose a counterintuitive possibility: can smaller, weaker models help guide the training of their larger, stronger counterparts? Rather than viewing the former as merely inferior, we treat it as a valuable teaching signal—one that precisely reveals where reasoning complexity begins to challenge weaker model capabilities. Specifically, we ask whether the divergent behaviors between a strong *expert* model and a weaker *amateur* model can pinpoint the high-impact reasoning moments where learning should focus, and whether these moments can be transformed into effective training signals to enhance model reasoning. In §2.2, we analyze the Expert-Amateur behavioral dynamics through the Kullback–Leibler (KL) divergence between their next-token predictions at each generation step. The intuition is straightforward: when models of different capabilities disagree strongly, those moments often mark critical reasoning steps that distinguish successful from failed problem-solving. Our analysis shows that such steps, where the Expert confidently predicts the correct token while the Amateur diverges or shows uncertainty, are signaled by pronounced KL values. Exploiting these moments yields targeted supervision signals that reinforce the Expert's reasoning strengths.



(a) Qwen2.5-Math-1.5B **(b) Qwen2.5-Math-7B** **(c) DeepSeek-R1-1.5B** **(d) Qwen2.5-Math-1.5B-Ins**

Figure 1: Efficiency and performance comparison between **SFT** and **LightReasoner** across different base models. LightReasoner achieves competitive or superior accuracy while substantially reducing resource consumption.

Building on these analyses, we introduce **LightReasoner**, a novel framework that leverages expert-amateur divergence to enhance LLM reasoning. As illustrated in Figure 4, LightReasoner operates in two stages: **(1) Sampling stage.** For each reasoning trajectory, both Expert and Amateur models generate next-token predictions under identical prefixes. Steps where the KL divergence between their distributions exceeds a threshold are retained (§2.3.1). From these selected steps, we construct supervision examples that encode Expert's *relative* advantage by contrasting the two models' prediction distributions (§2.3.2). **(2) Fine-tuning stage.** The *same* Expert model is then trained to align with these contrastive signals, increasing probability of tokens where its advantage over Amateur is most pronounced, thereby reinforcing its strengths and avoiding amateur-like tendencies (§2.3.3).

Our experimental evaluation highlights four key advantages of LightReasoner: • **Strong Performance Gains (§3.2).** LightReasoner consistently achieves comparable or superior performance to supervised fine-tuning (SFT) across 5 models and 7 mathematical reasoning benchmarks. • **Order-of-Magnitude Efficiency (§3.3).** LightReasoner delivers substantial computational savings: reducing total time cost by 90%, requiring 80% fewer sampled problems, and tuning 99% fewer tokens compared to SFT with rejection sampling, while entirely eliminating the need for ground-truth labels. • **Domain Expertise Drives Effective Contrasts (§3.4).** Our analysis shows that the most effective expert-amateur contrasts arise not from model size gaps, but from domain-specific expertise differences. This reinforces our insight that weaker models, when used in a novel way, can provide effective teaching signals for stronger models. • **Synergistic Design (§3.5).** Ablation studies confirm the essential roles of divergence-based step selection and contrastive supervision, along with their mutually-reinforcing effect, in enabling LightReasoner to strengthen the expert model's reasoning efficiently and reliably.

2 Methodology

2.1 Preliminaries

Autoregressive Language Model Generation. Given vocabulary $\mathcal{A}$ and input prompt a_0, a language model generates a response $a_{1:T} = [a_1, \ldots, a_T]$ autoregressively. At each step t, the model predicts the next token based on the prefix $s_t = [a_0, \ldots, a_{t-1}]$ and outputs distribution $\pi_{\text{LM}}(\cdot \mid s_t)$ over $\mathcal{A}$. The joint likelihood factorizes as:

$$P(a_{1:T} \mid a_0) = \prod_{t=1}^{T} \pi_{\text{LM}}(a_t \mid s_t). \tag{1}$$

In this autoregressive framework, the quality of reasoning emerges from the accumulation of individual token-level decisions. Thus, improving a model's reasoning ability fundamentally amounts to refining its policy π_{LM}. A central challenge, then, is to determine *which* token-level decisions truly matter for reasoning quality, and how to provide learning signals that target them effectively.

Learning from Behavioral Divergence. Existing approaches often depend on costly human annotations or external verification mechanisms, which suffer from scalability and resource limitations, hindering their practicality for continuous improvement. We observe that models of different capabilities exhibit systematic differences in their token-level decision patterns. This insight motivates our approach: leveraging *behavioral divergence* between models to automatically identify critical decision points and extract effective learning signals.

Critical Decision Points for Reasoning Enhancement. The foundation of our approach lies in the observation that reasoning ability is shaped *not* by uniform performance across all tokens, but by a handful of critical decision points. Cognitive science shows that expert problem-solving hinges on key moments where the correct logical choice determines success Chi et al. (1981). These are precisely where expert and amateur models diverge most. Certain reasoning steps exert disproportionate influence on final outcomes, creating natural bottlenecks in reasoning chains. Recent studies further confirm that language models benefit most from training on such critical

token subsets (Lin et al., 2024a;b). We therefore propose that targeting these high-stakes decision points creates a leverage effect: small improvements at bottlenecks can yield large overall gains. This mirrors human expertise development, where experts master the pivotal transitions that challenge beginners Ericsson & Smith (1991). Our method operationalizes this intuition by exploiting expert–amateur differences to identify reasoning bottlenecks and focus enhancement where it matters most.

To this end, we leverage two models with distinct reasoning capabilities: (1) an **Expert** model π_E, which we aim to improve, and (2) an **Amateur** model π_A, serving as a weaker baseline. As π_E generates a response $a_{1:T}$ through a sequence of prefixes $s_{1:T}$, we evaluate both models at each step t on the same prefix s_t. This produces paired distributions $(\pi_E(\cdot \mid s_t), \pi_A(\cdot \mid s_t))$, whose divergences reveal exactly where the Expert departs from amateur-level reasoning, providing targeted supervisory signals that concentrate on the most consequential decisions.

2.2 Token Informativeness via Expert-Amateur Divergence

To pinpoint critical decision points, we quantify the disagreement between Expert and Amateur models at each generation step t using **Kullback–Leibler (KL) divergence**:

$$D_{\mathrm{KL}}(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)) = \sum_{a \in \mathcal{A}} \pi_E(a \mid s_t) \log \frac{\pi_E(a \mid s_t)}{\pi_A(a \mid s_t)}. \tag{2}$$

Large KL values signals reasoning bottlenecks where the Expert's knowledge departs sharply from amateur-level choices. In mathematical problem solving, these pivotal moments are especially consequential: solution trajectories often hinge on recurring decision points where expert guidance proves decisive (Ji et al., 2025; Prabhudesai et al., 2025). Our analysis highlights three consistent patterns that validate this perspective:

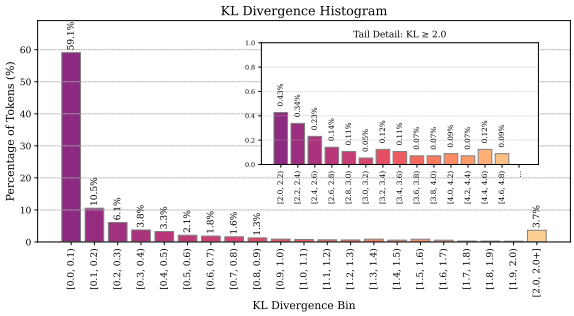


Figure 2: Most tokens show minimal KL divergence, with only few exhibiting elevated values.

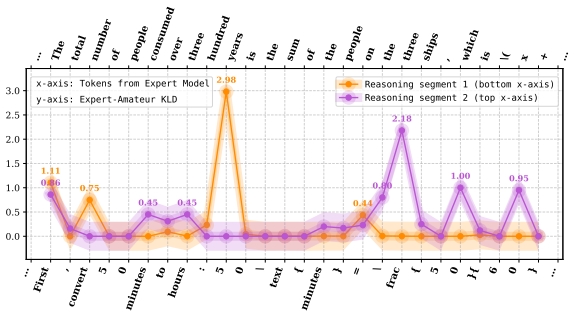


Figure 3: Predictable tokens yield near-zero KL divergence, while critical steps trigger notable spikes.

• **Divergence Concentrates on Critical Tokens (Figure 2):** 60% of tokens exhibit negligible divergence (KLD $\in [0.0, 0.1)$), while only 20% exceed 0.4, confirming that reasoning bottlenecks arise at specific moments.

• **Divergence Tracks Reasoning Complexity (Figure 3):** KL divergence stays near zero for routine tokens but spikes at demanding steps such as arithmetic operations and logical transitions, tracking underlying complexity.

• **Divergence Reveals Hidden Disagreements:** When the Expert and Amateur disagree on top-1 tokens, the average KL jumps to 1.99, versus 0.166 under agreement. Even among agreements, 10% of tokens still show KLD > 0.4, exposing subtle hidden misalignments beneath apparent consensus.

2.3 LightReasoner: Self-Supervised Enhancement Through Model Contrast

We present **LightReasoner** (Figure 4), a self-supervised framework for enhancing LLM reasoning without reliance on human annotations or external verifiers. It is built on three core components: (1) an informative step-selection mechanism that pinpoints critical reasoning steps via divergence-based metrics (§2.3.1); (2) a method for constructing contrastive soft labels that capture the Expert's reasoning advantage (§2.3.2); (3) a training objective that distills this contrastive signal back into the Expert (§2.3.3). The full procedure is summarized in Algorithm 1.

2.3.1 Informative Step Selection

Reasoning trajectories consist of tokens with uneven learning value. Routine steps often yield close Expert–Amateur agreement, while valuable supervision arises at critical decision points where expert reasoning is decisive. To capture these moments, we introduce *informative step selection*, which uses KL divergence to quantify reasoning criticality: large divergences mark bottlenecks where expert knowledge separates success from

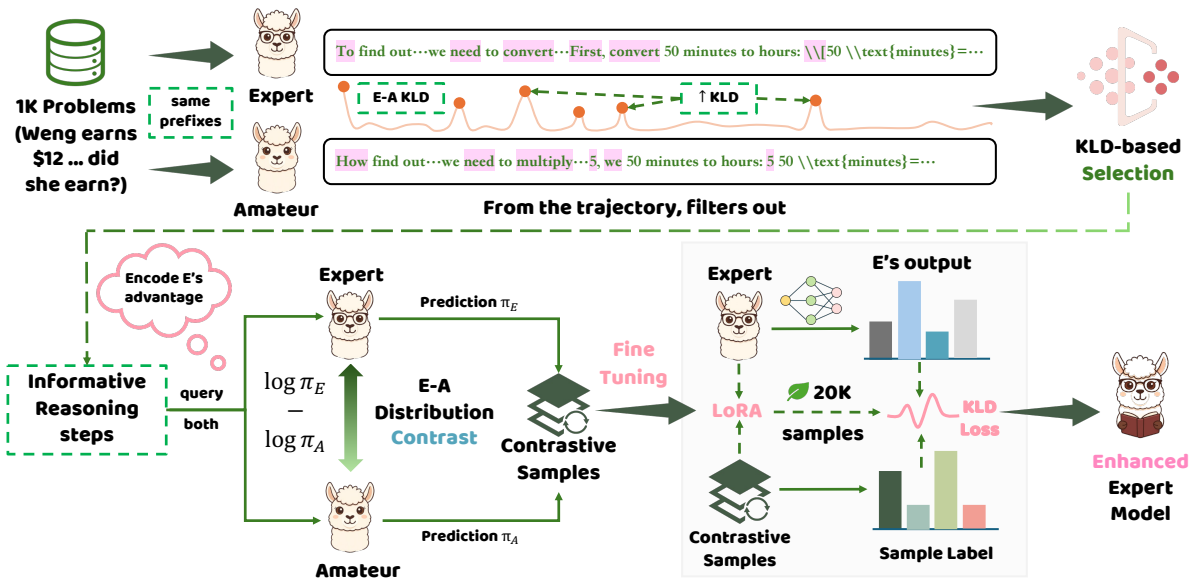


Figure 4: Overview of the LightReasoner framework. **Sampling Stage**: Expert and Amateur models generate distributions π_E and π_A. Informative step selection retains steps with $D_{\mathrm{KL}}(\pi_E \parallel \pi_A) > \beta$, and contrastive supervision constructs soft labels v_C capturing the Expert's advantage through Expert-Amateur contrast. **Fine-tuning Stage**: The Expert model is enhanced by minimizing the KLD between its output and v_C.

Algorithm 1 LightReasoner: A self-supervised framework for LLM reasoning enhancement

Input: Expert model π_E, Amateur model π_A, dataset $\mathcal{Q}$
Output: Enhanced Expert model π'_E

```
// Phase 1: Contrastive Sampling
for all problem q ∈ Q do
    Generate a CoT trajectory using π_E: {(s_t, π_E(· | s_t))}_{t=1}^T
    for t = 1 to T do
        Compute π_A(· | s_t)
        Compute D_KL(π_E(· | s_t) ‖ π_A(· | s_t))
        if D_KL(π_E(· | s_t) ‖ π_A(· | s_t)) > β then
            for all a_i ∈ A do
                if π_E(a_i | s_t) ≥ α · max_a π_E(a | s_t) then
                    Add a_i to A_mask
                    Compute v'_C(a_i | s_t) ← log( π_E(a_i|s_t) / π_A(a_i|s_t) )
            Normalize: ṽ_C(· | s_t) ← softmax(v'_C(· | s_t)) over A_mask
            for all a_i ∈ A do
                if a_i ∈ A_mask then
                    v_C(a_i | s_t) ← ṽ_C(a_i | s_t)
                else
                    v_C(a_i | s_t) ← 0
            Store (s_t, v_C(· | s_t)) for training
// Phase 2: Contrastive Fine-tuning
for all stored (s_t, v_C(· | s_t)) do
    Compute current output π_E(· | s_t)
    Compute loss L(s_t) ← D_KL(v_C(· | s_t) ‖ π_E(· | s_t))
    Update π_E using gradient ∇L(s_t) to obtain the fine-tuned model π'_E
```

failure. We implement this via β-*filtering*, retaining only steps where the Expert's advantage is pronounced. For a given prefix s_t, a step is selected if

$$D_{\mathrm{KL}}(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)) > \beta. \tag{3}$$

This filtering concentrates fine-tuning on the reasoning steps that matter most, preventing the learning signal from being diluted by trivial token predictions.

2.3.2 CONTRASTIVE DISTRIBUTIONAL SUPERVISION

After identifying informative steps, we convert Expert–Amateur disagreement into effective training signals. Instead of one-hot targets, which discard distributional information, we construct *contrastive soft targets* that encode the Expert's *relative* advantage. For a step s_t passing the β-filter, we first define the *masked support set*:

$$\mathcal{A}_{\text{mask}} = \left\{ a \in \mathcal{A} : \pi_E(a \mid s_t) \geq \alpha \cdot \max_{b \in \mathcal{A}} \pi_E(b \mid s_t) \right\}, \tag{4}$$

where $\alpha \in (0, 1]$(Li et al. (2022)) removes low-confidence tokens, preventing noisy tail probabilities from distorting the supervision signal. For each $a \in \mathcal{A}_{\text{mask}}$, we compute the unnormalized *contrast score* (Li et al., 2022):

$$v'_C(a \mid s_t) = \log\big(\pi_E(a \mid s_t)\big) - \log\big(\pi_A(a \mid s_t)\big), \tag{5}$$

which quantifies the Expert's advantage margin. We normalize these scores via *softmax* to to form a valid distribution $\tilde{v}_C$ and then extend $\tilde{v}_C$ to the full vocabulary:

$$v_C(a \mid s_t) = \tilde{v}_C(a \mid s_t) \cdot \mathbb{1}[a \in \mathcal{A}_{\text{mask}}], \quad a \in \mathcal{A}. \tag{6}$$

The resulting $v_C(\cdot \mid s_t) \in \mathbb{R}^{|\mathcal{A}|}$ encodes the expert superiority as structured probabilistic supervision, enabling targeted refinement that reinforces expert-like decisions.

2.3.3 SELF-DISTILLATION TRAINING OBJECTIVE

The framework is completed with a self-distillation objective that transfers the signal encoded in v_C back into the Expert model. Specifically, the Expert distribution π_E is trained to match v_C by minimizing the KL divergence:

$$\mathcal{L}(s_t) = D_{\text{KL}}\big(v_C(\cdot \mid s_t) \,\big\|\, \pi_E(\cdot \mid s_t)\big) = \sum_{a \in \mathcal{A}} v_C(a \mid s_t) \left[\log v_C(a \mid s_t) - \log \pi_E(a \mid s_t)\right] \tag{7}$$

Since the first term is constant with respect to π_E, equation 7 is equivalent to cross-entropy minimization:

$$\tilde{\mathcal{L}}(s_t) = -\sum_{a \in \mathcal{A}} v_C(a \mid s_t) \, \log \pi_E(a \mid s_t), \tag{8}$$

which reinforces the Expert's probability mass on tokens where its advantage over the Amateur is most pronounced. Detailed derivations and further discussions are provided in Appendix C and Appendix D.

3 EXPERIMENT

To assess the effectiveness of LightReasoner, we structure our experiments around four research questions: • **RQ1:** How does LightReasoner improve performance across different baseline models? • **RQ2:** To what extent does LightReasoner reduce training costs while preserving performance? • **RQ3:** Which factors drive the success of Expert–Amateur collaboration? • **RQ4:** How do specific design choices influence LightReasoner's effectiveness?

3.1 EXPERIMENTAL SETUP

3.1.1 MODELS AND DATA

LightReasoner leverages Expert–Amateur model pairing to generate supervision signals. The Expert models include Qwen2.5-Math-1.5B and 7B, their Instruct counterparts, and the DeepSeek-R1-Distill variant (Yang et al., 2024; Guo et al., 2025). The Amateur is fixed as Qwen2.5-0.5B (Yang et al., 2024), a base model with general linguistic reasoning ability but without specialized mathematical pretraining.

For generating supervision samples, we use the GSM8K training set (Cobbe et al., 2021), exclusively selected for its emphasis on step-by-step, broadly applicable logical reasoning rather than domain-specific notation. This ensures that the Amateur, despite lacking math-specific training, can still produce interpretable outputs suitable for contrastive supervision. Chain-of-thought prompting (Wei et al., 2022) is employed to elicit reasoning trajectories. Comprehensive model specifications and dataset descriptions are provided in Appendix G.1 and Appendix G.2.

3.1.2 TRAINING CONFIGURATION

We fine-tune the same Expert model used during sampling on the curated LightReasoner dataset using LoRA (Hu et al., 2022) for parameter-efficient training, reducing computational cost while preserving general knowledge. Key hyperparameters are as follows: a masking threshold of $\alpha = 0.2$ to filter out low-probability Expert tokens (Li et al., 2022); a KL divergence threshold of $\beta = 0.4$ to isolate informative reasoning steps; and reasoning trajectories capped at 128 tokens, as early steps are typically less error-prone (Ji et al., 2025). Models are fine-tuned for 1000 steps, with each step comprising 16 contrastive samples, which we find sufficient for convergence. Full configuration details are provided in Appendix G.3.

Model	GSM8K	MATH	SVAMP	ASDiv	Minerva Math	Olympiad Bench	MMLU STEM	AVG.
Qwen2.5-Math-1.5B								
Baseline	42.5	34.2	68.8	68.1	9.9	23.7	49.8	42.4
+ SFT	69.2	57.1	64.1	70.2	**15.1**	**27.6**	47.7	50.1
+ LightR	**70.6**	**59.3**	**76.0**	**79.8**	11.4	27.1	**54.9**	**54.2**
Qwen2.5-Math-1.5B-Instruct								
Baseline	84.8	75.8	94.2	94.7	29.4	37.5	57.4	67.7
+ SFT	85.4	75.8	93.5	94.7	31.6	37.5	56.2	67.8
+ LightR	**86.7**	75.5	93.0	94.1	**32.0**	**37.8**	55.2	**67.8**
DeepSeek-R1-Distill-Qwen-1.5B								
Baseline	75.2	54.2	79.9	84.9	16.2	19.1	22.3	50.3
+ SFT	78.2	**60.3**	81.5	87.4	**18.4**	21.2	26.2	53.3
+ LightR	**79.5**	60.2	**83.5**	**87.5**	18.0	**36.5**	**26.2**	**55.9**
Qwen2.5-Math-7B								
Baseline	57.5	51.8	67.9	72.7	14.0	16.0	69.8	50.0
+ SFT	64.4	**63.3**	76.2	76.6	12.1	**20.5**	68.5	54.5
+ LightR	**67.9**	57.8	**77.2**	**80.6**	12.1	16.9	**70.5**	**54.7**
Qwen2.5-Math-7B-Instruct								
Baseline	95.2	83.2	93.9	95.3	33.8	41.5	69.3	73.2
+ SFT	95.4	83.1	**94.1**	95.2	**38.2**	40.7	68.2	**73.6**
+ LightR	**95.8**	**83.6**	93.1	95.2	34.2	39.0	67.8	72.7

Table 1: **Zero-shot pass@1**[2] accuracy (%) across evaluation benchmarks. LightReasoner consistently achieves comparable or superior performance over SFT across 5 baseline models and 7 mathematical benchmarks.

3.1.3 Evaluation

For a strong competitor, we implement SFT on the baseline models (Appendix G.4), fine-tuning with LoRA on demonstrations of correct reasoning trajectories collected via rejection sampling (Yuan et al., 2023). We evaluate the baseline models, the SFT-trained variants, and the LightReasoner fine-tuned counterparts on a diverse suite of benchmarks ranging from basic arithmetic to expert-level mathematical reasoning (Appendix G.1). All evaluations are conducted with the official Qwen2.5-Math toolkit[1], and we report **zero-shot pass@1**[2]as the primary metric.

3.2 Performance Improvements (RQ1)

Table 1 presents the **zero-shot pass@1**[2] accuracy across a comprehensive suite of mathematical reasoning benchmarks. Across 5 models and 7 datasets, LightReasoner delivers consistent performance improvements and demonstrates fundamental advances in mathematical reasoning capability. These results highlight LightReasoner's strong generalization, which we analyze through three perspectives below.

- **Cross-Dataset Generalization for Reasoning Enhancement.** Despite being trained exclusively on GSM8K, our model achieves consistent gains across diverse benchmarks including MATH, SVAMP, and ASDiv. This cross-domain success shows that LightReasoner cultivates fundamental reasoning patterns rather than memorizing dataset-specific heuristics. The power of contrast-based selection is particularly evident here: by focusing on reasoning bottlenecks where expert knowledge diverges from amateur patterns, the method captures transferable logical structures that extend beyond the training domain.
- **Adaptive Enhancement across Model Architectures.** Our approach delivers consistent improvements across models of different capacities, though the mechanisms vary with model sophistication. For non-instruct models (e.g., Qwen2.5-Math-1.5B), we observe dramatic gains (+28.1% on GSM8K and +25.1% on MATH), showing that contrastive supervision can activate latent reasoning circuits previously dormant. For heavily optimized instruct models (e.g., Qwen2.5-Math-1.5B-Instruct), the improvements are modest but steady (+1.9% on GSM8K and +2.6% on Minerva Math), suggesting our method refines existing reasoning pathways. This differential effectiveness highlights how contrastive distributional supervision adapts flexibly to varying model capabilities.

[1] `https://github.com/QwenLM/Qwen2.5-Math`

[2] Tested on unseen datasets without task-specific demonstrations; performance measured as the percentage of problems correctly solved in a single greedy attempt; MMLU STEM is evaluated using 5-shot; all others are zero-shot.

Method	Time	Probs.	Tokens	Gain
Qwen2.5-Math-1.5B				
+ SFT	4.0h	3952	1.77M	+7.7%
+ LightR	**0.5h**	**1000**	**0.02M**	**+11.8%**
Qwen2.5-Math-7B				
+ SFT	9.5h	6029	2.20M	+4.5%
+ LightR	**0.75h**	**1000**	**0.02M**	**+4.7%**
DeepSeek-R1-Distill-Qwen-1.5B				
+ SFT	3.6h	6023	5.95M	+3.0%
+ LightR	**0.5h**	**1000**	**0.02M**	**+5.6%**
Qwen2.5-Math-1.5B-Instruct				
+ SFT	3.4h	7153	2.08M	+0.1%
+ LightR	**0.4h**	**1000**	**0.02M**	+0.1%

Table 2: **Efficiency comparison between SFT and LightReasoner.** We report total time, sampled problems, tuned tokens, and average performance improvement across seven evaluation benchmarks.

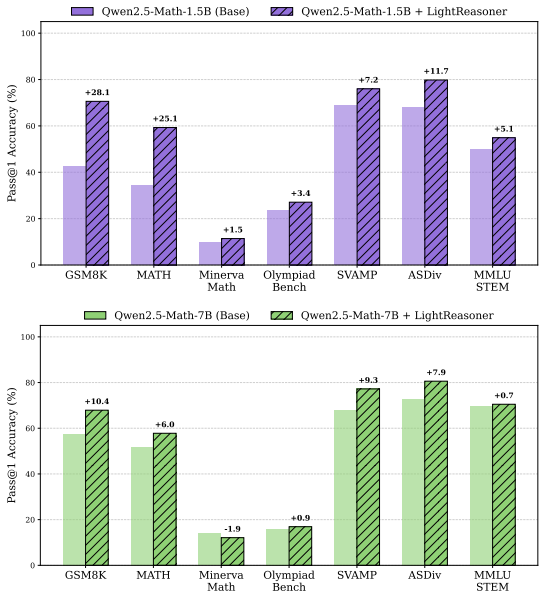


Figure 5: **LightReasoner** consistently improves zero-shot pass@1 accuracy across 7 mathematical evaluation benchmarks for baseline models.

- **Superior Efficiency over SFT.** Direct comparisons show that LightReasoner matches or surpasses SFT performance while consuming substantially fewer resources. On Qwen2.5-Math-1.5B, it delivers an average performance gain of +4.1% over SFT with 90% less training time and 99% fewer tuned tokens (§3.3). These efficiency gains stem from two synergistic components: informative step selection (§2.3.1), which concentrates learning on the $\sim$20% of tokens that drive reasoning (§2.2), and contrastive supervision (§2.3.2), which transforms these steps into precise training signals that encode the expert's advantage over amateurish tendencies.

3.3 Efficiency Study (RQ2)

The LightReasoner framework achieves remarkable efficiency through targeted reasoning enhancement. We evaluate the efficiency across three important dimensions: (1) **Time budget**: the total sampling and fine-tuning time measured on a single NVIDIA H200 GPU without inference accelerators (e.g., vLLM); (2) **Training instances**: the number of distinct training set problems required to generate the supervision dataset; (3) **Tuned tokens**: the computational overhead at the token level, where LightReasoner trains on selective next-token predictions while SFT optimizes over full reasoning trajectories. As shown in Table 2 and Figure 1, LightReasoner consistently outperforms SFT while using only a fraction of the resources. These efficiency improvements stem from three core design principles of LightReasoner that directly address SFT's computational bottlenecks:

Table 3: **Efficiency contrasts at a glance.** ↑ and ↓ indicate whether each aspect helps or hurts efficiency.

Attribute	Efficiency	SFT	LightR
Full trajectories	↓	✓	✗
All-token tuning	↓	✓	✗
GT verification	↓	✓	✗
Prefix termination	↑	✗	✓
Selective tokens	↑	✗	✓
Verification-free	↑	✗	✓

- **Sampling efficiency through prefix termination.** SFT must generate complete reasoning trajectories under rejection sampling. By contrast, LightReasoner halts generation at 128 tokens, sharply reducing sampling cost while preserving supervision quality. This approach leverages the observation that early reasoning steps provide more reliable signals with fewer cascading errors (Ji et al., 2025), enabling LightReasoner to capture informative patterns without expending computational resources on error-prone later steps.
- **Training efficiency via selective token filtering.** LightReasoner focuses on steps where expert–amateur divergence peaks, concentrating learning on reasoning bottlenecks rather than redundant tokens. This selectivity explains the large gap in token usage between SFT and LightReasoner. The benefit is further amplified by contrastive supervision, which converts the Expert's advantage into direct training signals. By avoiding SFT's indiscriminate tuning over full trajectories, LightReasoner achieves rapid reinforcement of reasoning ability.
- **Data efficiency through verification-free supervision.** Traditional SFT relies on costly "generate-and-verify" loops that require explicit checking against ground-truth answers, imposing a major limitation. LightReasoner avoids this by using the Amateur as a contrastive baseline, turning relative performance gaps into supervision

Amateur Model	Δ Perf.	GSM8K	MATH	SVAMP	ASDiv	MMLU STEM	AVG.
Expert: Qwen2.5-Math-1.5B							
Qwen2.5-0.5B	38.2	70.6	59.3	76.0	79.8	54.9	**68.1**
Qwen2.5-1.5B	35.1	63.4	57.1	69.7	75.7	54.8	64.1
Qwen2.5-Math-1.5B-Ins	-42.3	41.4	35.5	67.5	66.4	55.0	53.2
Expert Only (Baseline)	/	42.5	34.2	68.8	68.1	49.8	52.7
Expert: Qwen2.5-Math-7B							
Qwen2.5-0.5B	53.2	67.9	57.8	77.2	80.6	70.5	**70.8**
Qwen2.5-1.5B	50.1	69.0	56.0	77.6	78.9	69.5	70.2
Qwen2.5-Math-1.5B	15.0	56.9	50.2	63.5	63.4	70.7	60.9
Qwen2.5-Math-1.5B-Ins	-27.3	59.4	49.0	68.3	69.6	70.3	63.3
Expert Only (Baseline)	/	57.5	51.8	67.9	72.7	69.8	63.9

Table 5: **Impact of expertise-driven contrast.** We fix the Expert model and vary the Amateur to examine how the expertise gap affects LightReasoner performance. The column ΔPerf. denotes the Expert–Amateur accuracy difference on GSM8K. For extra clarity, each group concludes with the Expert's original performance.

Ablation Setting	GSM8K	MATH	SVAMP	ASDiv	Minerva Math	Olympiad Bench	AVG.
Qwen2.5-Math-1.5B	42.5	34.2	68.8	68.1	9.9	23.7	41.2
+ SFT	69.2	57.1	64.1	70.2	15.1	27.6	50.6
+ GT Supervision	39.4	34.8	70.4	69.7	6.2	13.8	39.1
+ Full LightReasoner	**70.6**	**59.3**	**76.0**	**79.8**	**11.4**	**27.1**	**54.0**
✗ **Select** ✓ **Contrast**	67.6	58.8	78.7	80.5	11.0	26.4	53.8
✓ **Select** ✗ **Contrast**	62.0	53.1	56.6	61.0	10.7	25.5	44.8
✗ **Select** ✗ **Contrast**	55.5	50.2	50.0	65.4	10.4	24.0	42.6

Table 6: **Ablation study** on the LightReasoner framework. We progressively remove key components, **step selection** and **contrastive supervision**, from the LightReasoner framework to isolate their contributions.

signals without correctness validation. This design removes the dependency on ground-truth verification and naturally extends to domains where definitive solutions may be unavailable. By decoupling learning from outcome validation, LightReasoner focuses on strengthening reasoning processes rather than merely outcomes.

3.4 Beyond Scale: Domain Expertise Drives Contrast (RQ3)

Prior contrastive methods rely on rigid parameter-size disparities to create Expert–Amateur contrast (e.g., OPT-13B vs. OPT-125M in *Contrastive Decoding (CD)* (Li et al., 2022); 65B vs. 1.5B in its follow-up study (O'Brien & Lewis, 2023)), restricting applicability to only scenarios with very large Expert models and introducing substantial computational demands. To overcome this limitation, we hypothesize that domain-specific expertise provides a more flexible and broadly applicable axis of contrast than the raw model scale. To validate this hypothesis, we fixed the Expert model and systematically varied the Amateur, progressively narrowing their expertise gap. As shown in Table 5 and Figure 6, this design demonstrates how Expert–Amateur expertise relationships determine LightReasoner's effectiveness and offers valuable insights into optimal contrastive pairings.

Table 4: **Key differences** between traditional Contrastive Decoding (CD) methods and LightReasoner. ↑ and ↓ denote whether each attribute helps or hurts the practicality of the method.

Attribute	Practicality	CD	LightR
Contrast usage	/	Inference	Training
Size-based contrast	↓	✓	✗
Expertise contrast	↑	✗	✓
Persistent benefits	↑	✗	✓
Independent inference	↑	✗	✓

- **Domain expertise as a superior axis of contrast.** Our results demonstrate that domain-specific knowledge, rather than parameter count, is the primary driver of effective contrastive supervision. The clearest evidence comes from pairing the Qwen2.5-Math-1.5B expert with the generalist Qwen2.5-1.5B amateur, which yields striking performance gains (+12.1% average gain) despite identical model sizes. This finding frees LightReasoner from rigid scale requirements and extends contrastive supervision to a broader range of models.
- **Effectiveness depends on the expertise gap.** As illustrated in Figure 6, performance gains are closely correlated with the size of the expertise gap. When Amateur capabilities approach those of the Expert, contrastive

signals weaken and benefits diminish. In the extreme case of pairing the Math-1.5B or Math-7B Expert with the stronger Math-1.5B-Instruct model—where the expertise gap is effectively negative—fine-tuning yields negligible gains or even degradation, confirming that expertise differentials are essential for contrastive learning.

3.5 Ablation Study (RQ4)

To assess the contribution of each core component in the LightReasoner framework, we conducted a systematic ablation study. By progressively removing individual mechanisms and measuring the resulting impacts, we isolate how each design contributes to overall performance. The results, presented in Table 6, highlight the essential role of every component and provide key insights into the design of LightReasoner. We summarize the findings below:

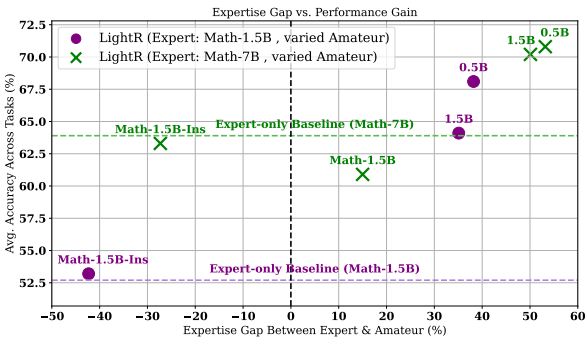


Figure 6: **Expert-Amateur Pairing Effects.** Each point represents an amateur model paired with a fixed expert model. The performance gains achieved by LightReasoner decrease as the expertise gap closes.

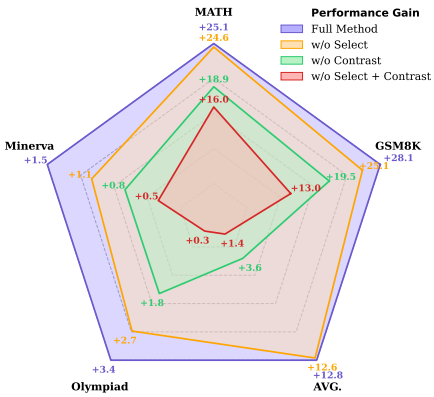


Figure 7: **Impact of Ablation.** Removing key components from LightReasoner consistently degrades performance, emphasizing the critical role of both step selection and contrastive supervision.

- **Impact of Informative Step Selection.** To assess the role of step selection, we removed this mechanism, leaving LightReasoner as a full-sampling pipeline without KL-based pruning. This ablation led to a clear performance decline (e.g., –3.0% on GSM8K), indicating that many steps contribute noise rather than meaningful learning. The step-selection filter addresses this by discarding trivial cases, enabling more targeted training.
- **Impact of Contrastive Distributional Supervision.** Removing contrastive supervision reduces LightReasoner to fine-tuning the Expert on its own paths (filtered by step selection). Without the Amateur's contrast to capture the Expert's relative margin, average performance drops by 9.2%. This demonstrates the central role of contrastive supervision in amplifying the Expert's strengths while steering it away from amateur-like tendencies.
- **Synergy Between Contrastive Supervision and Step Selection.** When both mechanisms are removed, performance drops by 12.4%—greater than the sum of the individual ablations (–9.2% and –0.2%). This superadditive decline reveals their mutual dependence: without step selection, the contrastive signal is diluted by trivial steps; without contrastive supervision, selected steps cannot be transformed into effective learning signals. Together, these mechanisms form a tightly coupled system, explaining the amplified benefits when used jointly.
- **Insights from Competitive Approaches.** To further validate our method, we compared LightReasoner against alternative strategies (Table 6). Fine-tuning on human-curated GSM8K solutions (*GT Supervision*) yielded weak results, while conventional SFT on correct self-generated trajectories provided clear gains but still lagged behind the ablation variant with contrastive supervision alone. These findings highlight a key principle: pretrained models learn most effectively from signals grounded in their own behavior. By selectively amplifying the Expert's advantage on the most informative reasoning steps, LightReasoner achieves faster and more robust improvements than either human-curated supervision or conventional SFT.

4 Conclusion

In this work, we introduced LightReasoner, a novel framework for advancing reasoning in large language models (LLMs) by exploiting the behavioral divergence between expert and amateur models. Grounded in the insight that critical reasoning steps carry disproportionate learning value, LightReasoner integrates targeted step selection with contrastive supervision to amplify the expert's strengths while minimizing resource demands and training complexity. Experiments across diverse benchmarks demonstrate that LightReasoner not only outperforms traditional supervised fine-tuning (SFT), but also delivers order-of-magnitude efficiency gains by reducing computation, data usage, and reliance on labeled ground truth. Taken together, these results highlight LightReasoner as a scalable and practical solution for building more capable reasoning models.

522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579

REPRODUCIBILITY STATEMENT

We have taken extensive measures to ensure the reproducibility of our work. Sampling, training, and evaluation protocols for LightReasoner and competitive baselines are detailed in Sections §2 and §3. Additional supporting materials are provided in the appendix: theoretical aspects, including assumptions and proofs, are given in Appendix C, D, E, and F; empirical details, such as dataset descriptions and backbone model specifications, are presented in Appendix G.1 and G.2, with further experimental setup information reported in Appendix G.3 and G.4. The complete LightReasoner framework code, along with complementary analysis scripts, is available in an anonymous repository at https://anonymous.4open.science/r/LightReasoner.

REFERENCES

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, et al. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Haw-Shiuan Chang, Nanyun Peng, Mohit Bansal, Anil Ramakrishna, and Tagyoung Chung. Explaining and improving contrastive decoding by extrapolating the probabilities of a huge and hypothetical lm. *arXiv preprint arXiv:2411.01610*, 2024.

Michelene TH Chi, Paul J Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive science*, 5(2):121–152, 1981.

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.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

K Anders Ericsson and Jacqui Smith. *Toward a general theory of expertise: Prospects and limits*. Cambridge University Press, 1991.

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.

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

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

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

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

Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xingyu Chen, Xiaoyuan Liu, Zhijie Wang, Junying Chen, Benyou Wang, et al. The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. *arXiv preprint arXiv:2503.02875*, 2025.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*, 2024a.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *arXiv preprint arXiv:2411.19943*, 2024b.

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. In *International Conference on Machine Learning*, pp. 22631–22648. PMLR, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023.

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.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

Phuc Phan, Hieu Tran, and Long Phan. Distillation contrastive decoding: Improving llms reasoning with contrastive decoding and distillation. *arXiv preprint arXiv:2402.14874*, 2024.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, 2023.

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

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions, 2025. URL `https://arxiv.org/abs/2304.12244`.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

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, 2023. URL `https://arxiv.org/abs/2308.01825`.

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

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

Appendix

CONTENTS

A Related Work **13**
A.1 Contrastive Decoding . . . 13
A.2 Post-Training . . . 13

B The Use of Large Language Models **13**

C From KL Divergence to Contrast Score **14**

D Connection between Selection, Contrast, and Training **14**

E Relation to Reinforcement Learning **15**

F Entropy Dynamics **15**

G Supplementary Details **16**
G.1 Datasets . . . 16
G.2 Baseline Models . . . 16
G.3 LightReasoner . . . 16
G.3.1 α-masking . . . 17
G.3.2 β-filtering . . . 17
G.3.3 Prompting . . . 17
G.3.4 LoRA Configuration . . . 18
G.4 Supervised Fine-tuning (SFT) . . . 18
G.4.1 Rejection Sampling . . . 18
G.4.2 Model Fine-tuning . . . 18

H Case Study **18**

A Related Work

A.1 Contrastive Decoding

Contrastive decoding (CD) has emerged as a novel technique for improving inference-time decoding in language models by contrasting output distributions from a strong expert and a weaker amateur model (Li et al., 2022). Follow-up work shows that CD can enhance reasoning by promoting confident, expert-aligned predictions (O'Brien & Lewis, 2023). More recently, Chang et al. (2024) present a theoretical framework that interprets contrastive scores as approximations to an idealized model and propose training procedures to align the amateur model accordingly. In parallel, Phan et al. (2024) apply distillation techniques such as dropout and quantization to the expert model to create a weaker variant, which replaces the amateur model as the contrast source during inference, thereby reducing memory cost.

Despite their promise, existing contrastive approaches are restricted to the *inference* stage. By contrast, our method is the **first** to leverage the model-contrast paradigm during *training*, distilling expert–amateur divergence into persistent supervision without requiring concurrent model execution at inference, thereby avoiding memory and latency overheads. This novel direction, combined with our informative step-selection mechanism and the use of expertise gap in place of size disparity, leads the contrastive paradigm into a new domain with broader utility and flexibility.

A.2 Post-Training

Post-training techniques refine pre-trained large language models (LLMs) by adapting their general knowledge for downstream applications and aligning them more closely with human intent. We organize the discussion into three main categories.

Supervised Fine-Tuning (SFT). SFT adapts pre-trained models by fine-tuning them to capture desired behaviors. Early approaches relied on human-annotated instruction datasets, with instruction tuning (Ouyang et al., 2022; Longpre et al., 2023) becoming widely adopted by training on instruction–response pairs to bridge the gap between next-token prediction and user intent. To reduce reliance on costly manual annotation, automated methods such as Self-Instruct (Wang et al., 2023) and Evol-Instruct (Xu et al., 2025) use large LLMs to generate synthetic instruction–response pairs at scale. In reasoning domains, SFT is often combined with rejection sampling (Yang et al., 2024; Guo et al., 2025), where model-generated trajectories are filtered for correctness before being used for supervision.

Reinforcement Learning (RL). RL extends beyond SFT by optimizing models against preference signals rather than fixed demonstrations, enabling alignment with subjective human judgments. Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) follows a three-stage pipeline: initialize with SFT, train a reward model on human preference comparisons, and optimize the policy with Proximal Policy Optimization (PPO). To improve scalability, Reinforcement Learning with AI Feedback (RLAIF) (Bai et al., 2022) replaces human labels with judgments from stronger models. More recent approaches simplify this paradigm by directly optimizing from preference comparisons, as in Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Generalized RPO (GRPO) (Guo et al., 2025). These methods reduce human effort but still rely on preference-based optimization to guide alignment.

Unsupervised Post-Training. A related line of work explores unsupervised post-training methods that leverage internal model signals to improve performance without external supervision. For example, UPFT (Ji et al., 2025) enhances reasoning by reinforcing self-generated intermediate prefixes, while methods such as Self-Refine (Madaan et al., 2023) and Self-Rewarding Correction (Xiong et al., 2025) adapt model behavior using self-evaluated feedback. These approaches reduce reliance on human supervision but often require additional scaffolding or careful calibration of internal signals.

In contrast, LightReasoner provides a practical alternative to conventional post-training. By capturing token-level divergences between an Expert model and a weaker Amateur, it isolates informative reasoning steps and uses their behavioral contrast to reinforce the Expert's strengths. This enables targeted fine-tuning without labels or external verification, making LightReasoner an efficient and effective approach to LLM reasoning enhancement.

B The Use of Large Language Models

We used OpenAI's ChatGPT solely as a writing assistant to polish grammar, phrasing, and readability of the paper. All research content is entirely the work of the authors.

C FROM KL DIVERGENCE TO CONTRAST SCORE

In §2.2, we analyzed Expert–Amateur disagreement via KL divergence and observed that reasoning steps where the Expert holds a clear advantage correspond to higher KL divergence values. A natural idea, therefore, is to train the Expert model to maximize this divergence, which is equivalent to minimizing the loss:

$$\mathcal{L}_{\text{KL}}(s_t) = -D_{\text{KL}}(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)) \,. \tag{9}$$

Treating π_A as fixed, the gradient with respect to the Expert parameters θ_E is:

$$\nabla_{\theta_E} \mathcal{L}_{\text{KL}}(s_t) = \nabla_{\theta_E} \Big[-D_{\text{KL}}\big(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)\big) \Big] \tag{10}$$

$$= -\nabla_{\theta_E} \sum_{a \in \mathcal{A}} \pi_E(a \mid s_t) \, \log \frac{\pi_E(a \mid s_t)}{\pi_A(a \mid s_t)} \tag{11}$$

$$= -\sum_{a \in \mathcal{A}} \nabla_{\theta_E} \Big[\pi_E(a \mid s_t) \, \log \tfrac{\pi_E(a|s_t)}{\pi_A(a|s_t)} \Big] \tag{12}$$

By *product rule*:

$$\nabla_{\theta_E} \mathcal{L}_{\text{KL}}(s_t) = -\sum_{a \in \mathcal{A}} \Big[\nabla_{\theta_E} \pi_E(a \mid s_t) \, \log \tfrac{\pi_E(a|s_t)}{\pi_A(a|s_t)} + \pi_E(a \mid s_t) \, \nabla_{\theta_E} \log \tfrac{\pi_E(a|s_t)}{\pi_A(a|s_t)} \Big] \tag{13}$$

By *chain rule*:

$$\nabla_{\theta_E} \mathcal{L}_{\text{KL}}(s_t) = -\sum_{a \in \mathcal{A}} \nabla_{\theta_E} \pi_E(a \mid s_t) \, \Big(\log \tfrac{\pi_E(a|s_t)}{\pi_A(a|s_t)} + 1 \Big). \tag{14}$$

At this point, the log-ratio naturally emerges:

$$w(a \mid s_t) \;=\; \log \frac{\pi_E(a \mid s_t)}{\pi_A(a \mid s_t)} \;=\; \log \pi_E(a \mid s_t) - \log \pi_A(a \mid s_t), \tag{15}$$

which coincides exactly with the *contrast score* proposed in Li et al. (2022). This observation motivates our choice: rather than directly maximizing KL divergence, we adopt the contrast score as the central mechanism for constructing the supervision signal to guide the Expert model.

D CONNECTION BETWEEN SELECTION, CONTRAST, AND TRAINING

As shown in §C, a central quantity in our framework is the *contrast score* (Li et al., 2022):

$$w(a \mid s_t) \;=\; \log \pi_E(a \mid s_t) - \log \pi_A(a \mid s_t), \tag{16}$$

This log-ratio underlies step selection, contrastive supervision, and the training objective.

Step Selection. Expert–Amateur disagreement is quantified by the KL divergence:

$$D_{\text{KL}}(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)) = \sum_{a \in \mathcal{A}} \pi_E(a \mid s_t) \, w(a \mid s_t). \tag{17}$$

Contrastive Supervision. Restricting to a masked vocabulary $\mathcal{A}_{\text{mask}}$, we normalize $w(a \mid s_t)$ via:

$$\tilde{v}_C(a \mid s_t) = \frac{\exp(w(a \mid s_t))}{\sum_{b \in \mathcal{A}_{\text{mask}}} \exp(w(b \mid s_t))}, \quad a \in \mathcal{A}_{\text{mask}}, \tag{18}$$

and extend to the full vocabulary by:

$$v_C(a \mid s_t) = \tilde{v}_C(a \mid s_t) \cdot \mathbb{1}[a \in \mathcal{A}_{\text{mask}}] \,, \quad a \in \mathcal{A}, \tag{19}$$

Training Objective. The Expert is distilled toward $v_C(a \mid s_t)$ by minimizing forward KL:

$$\mathcal{L}(s_t) = D_{\text{KL}}(v_C(\cdot \mid s_t) \,\|\, \pi_E(\cdot \mid s_t)) = \sum_{a \in \mathcal{A}} v_C(a \mid s_t) \, [\log v_C(a \mid s_t) - \log \pi_E(a \mid s_t)]. \tag{20}$$

Unification. All three components are governed by the same log-ratio $w(a \mid s_t)$:

1. **Step selection** quantifies divergence through $w(a \mid s_t)$ and retains informative reasoning steps.
2. **Contrastive supervision** transforms $w(a \mid s_t)$ into a learning signal, capturing the Expert model's difference margin from the Amateur.
3. **Training** propagates gradients implicitly weighted by $w(a \mid s_t)$, guiding toward high-contrast tokens.

In this way, selection, supervision, and optimization are mathematically unified and mutually reinforcing, which explains the *coherence* and effectiveness of our approach from a new angle.

E Relation to Reinforcement Learning

Policy gradient methods. In reinforcement learning, the objective of policy gradient methods is to maximize the expected return

$$J(\theta) \;=\; \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t\, r_t\right], \tag{21}$$

where r_t is the reward at step t and $\gamma \in [0,1]$ is a discount factor. Under the actor–critic framework, the policy gradient theorem states that

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \ln \pi_\theta(a_t \mid s_t)\, A^\theta(a_t, s_t)\right], \tag{22}$$

where $A^\theta(a_t, s_t)$ denotes the advantage of action a_t at state s_t.

Parallel with our framework. In our case, the training objective is the KL loss

$$\mathcal{L}(s_t) = D_{\mathrm{KL}}\big(v_C(\cdot \mid s_t) \,\|\, \pi_E(\cdot \mid s_t)\big) = \sum_{a\in\mathcal{A}} v_C(a \mid s_t)\,\big[\ln v_C(a \mid s_t) - \ln \pi_E(a \mid s_t)\big]. \tag{23}$$

The first term in equation 23 is constant with respect to π_E, and this reduces to the cross-entropy:

$$\tilde{\mathcal{L}}(s_t) = -\sum_{a\in\mathcal{A}} v_C(a \mid s_t)\, \ln \pi_E(a \mid s_t), \tag{24}$$

Its gradient with respect to the Expert parameters is:

$$\nabla_{\theta_E} \mathcal{L}(s_t) = -\sum_{a\in\mathcal{A}} \nabla_{\theta_E} \ln \pi_E(a \mid s_t)\, v_C(a \mid s_t). \tag{25}$$

Comparing the two forms equation 22 and equation 25, we see that our update shares the same structure as a policy gradient step: a log-policy term $\nabla_{\theta_E} \ln \pi_E(a \mid s_t)$ weighted by a signal. In standard RL, this weight is the advantage $A^{\pi_\theta}(s_t, a_t)$; in our framework, the contrastive target $v_C(a \mid s_t)$ plays the role of the advantage signal. The only distinction is the optimization direction: policy gradient methods perform gradient ascent to maximize $J(\theta)$, while we perform gradient descent to minimize $\mathcal{L}(s_t)$.

Inner connection. This analogy suggests that our framework can be viewed as a variant of policy gradient training, where the advantage signal arises from Expert–Amateur divergence rather than environment rewards or human feedback. Unlike RL pipelines that depend on explicit reward models or preference annotations, our approach is closer in spirit to Direct Preference Optimization (DPO) (Rafailov et al., 2023), since it bypasses reward modeling and directly optimizes a divergence-based objective. Exploring how LightReasoner could be extended into a full RL framework with explicit policy updates and reward signals remains an interesting direction for future work.

F Entropy Dynamics

In reinforcement learning (RL), policy updates instantiate the exploration–exploitation trade-off. In LLM post-training, recent work shows that this trade-off manifests as changes in *policy entropy* (Cui et al., 2025): without explicit control, entropy often collapses early, leading to over-confident policies and performance saturation. Empirically, pretrained models with higher initial entropy tend to achieve better downstream performance after RL, while the RL process itself largely converts entropy into reward (Yue et al., 2025). This makes entropy preservation a crucial aspect of LLM post-training.

Policy entropy. Formally, the entropy of a policy π relative to a dataset $\mathcal{D}$ is:

$$H(\pi, \mathcal{D}) = -\,\mathbb{E}_{\mathbf{y}_{<t}\sim\mathcal{D}}\left[\sum_{y_t} \pi(y_t \mid \mathbf{y}_{<t})\, \log \pi(y_t \mid \mathbf{y}_{<t})\right], \tag{26}$$

Entropy change under policy gradient. For a single state s, the intrinsic policy entropy is:

$$H(\pi \mid s) := -\sum_{a} \pi(a \mid s)\, \log \pi(a \mid s). \tag{27}$$

For a tabular softmax policy updated by one step of vanilla policy gradient with step size η, the stepwise entropy change satisfies the first–order approximation:

$$H(\pi^{k+1} \mid s) - H(\pi^k \mid s) \;\approx\; -\eta\; \mathrm{Cov}_{a\sim\pi^k(\cdot|s)}\big(\log \pi^k(a \mid s),\; \pi^k(a \mid s)\, A^k(s,a)\big), \tag{28}$$

where $A^k(s,a)$ is the advantage, satisfying $\mathbb{E}_{a\sim\pi^k}[A^k(s,a)] = 0$. Thus, when high-probability actions tend to carry positive advantage (positive covariance), entropy decreases; whereas if advantage is concentrated on low-probability actions (negative covariance), entropy can increase (Cui et al., 2025).

Entropy change with contrast score. As established in §E, our framework is equivalent to policy gradient when the contrast score is used as the advantage. Consequently, the first–order change in policy entropy can be written as:

$$H(\pi_E^{k+1} \mid s) - H(\pi_E^k \mid s) \;\approx\; -\eta \,\mathrm{Cov}_{a\sim\pi_E^k(\cdot|s)}\Big(\,\log \pi_E^k(a \mid s),\; \pi_E^k(a \mid s)\, v_C^k(s,a)\Big). \tag{29}$$

Let $\mathcal{A}_{\text{mask}}(s) \subseteq \mathcal{A}$ denote the masked action set and let $m_\alpha(s,a) \in \{0,1\}$ be its indicator:

$$m_\alpha(s,a) = \begin{cases} 1, & a \in \mathcal{A}_{\text{mask}}(s), \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Define the masked, temperature–scaled contrast:

$$v_C^k(s,a) := \frac{1}{\tau}\, m_\alpha(s,a)\, v_C'^k(s,a), \qquad v_C'^k(s,a) := \log \frac{\pi_E^k(a \mid s)}{\pi_A^k(a \mid s)}, \tag{31}$$

with $\tau > 0$ a temperature (scaling) parameter. Substituting equation 31 into equation 29 yields:

$$H(\pi_E^{k+1} \mid s) - H(\pi_E^k \mid s) \;\approx\; -\frac{\eta}{\tau}\, \mathrm{Cov}_{a\sim\pi_E^k(\cdot|s)}\Big(\,\log \pi_E^k(a \mid s),\; \pi_E^k(a \mid s)\, m_\alpha(s,a)\, \log \tfrac{\pi_E^k(a|s)}{\pi_A^k(a|s)}\Big). \tag{32}$$

Intuitively, equation 32 highlights that the covariance is dominated by terms of the form:

$$\pi_E(a \mid s)\, m_\alpha(s,a)\, \log \tfrac{\pi_E(a|s)}{\pi_A(a|s)} \tag{33}$$

The α-mask restricts updates to actions with non-negligible probability under the Expert. If Expert and Amateur agree on an action, the log-ratio is close to zero, so the contribution is negligible even when π_E is large. In contrast, when the Expert assigns mass to actions where it substantially outperforms the Amateur (large positive log-ratio), the covariance is typically positive and entropy decreases. Thus, the contrastive advantage expends entropy selectively on high-contrast, non-negligible tokens, rather than on aligned or low-value regions, thereby improving the efficiency of the update process.

G Supplementary Details

We provide additional details to complement the descriptions in §3.1.1 and §3.1.2, covering both our proposed method, LightReasoner, and the competitive supervised fine-tuning (SFT) baseline.

G.1 Datasets

We exclusively use the **GSM8K** (Cobbe et al., 2021) training set, a collection of grade-school math problems emphasizing step-by-step reasoning, to generate contrastive samples. To evaluate the transferability of the learned skills, we assess our models on a diverse suite of benchmarks: **MATH** (Hendrycks et al., 2021), a collection of high school competition problems; **SVAMP** (Patel et al., 2021) and **ASDiv** (Miao et al., 2021), testing numerical reasoning through linguistically varied arithmetic problems; **Minerva Math** (Lewkowycz et al., 2022), quantitative problems from advanced STEM courses; **OlympiadBench** (He et al., 2024), challenging problems from international math olympiads; and **MMLU-STEM** (Hendrycks et al., 2020), which evaluates broad knowledge across science and math. This range spans from foundational arithmetic to expert-level reasoning, enabling a thorough assessment of both generalization and specialization.

G.2 Baseline Models

Our method leverages pairing between *Expert* and *Amateur* models to generate training signals:

- ***Expert Models*** span varying capabilities and sizes to ensure robust evaluation: **(1) Qwen2.5-Math-1.5B** and **(2) Qwen2.5-Math-7B** are derived from Qwen2.5 base models via pretraining on a 1T-token math corpus (Yang et al., 2024). **(3) Qwen2.5-Math-1.5B-Instruct** and **(4) Qwen2.5-Math-7B-Instruct** receive additional multi-stage post-training, including supervised fine-tuning and GRPO-based reinforcement learning (Yang et al., 2024). **(5) DeepSeek-R1-Distill-Qwen-1.5B** is fine-tuned based on the corresponding Qwen2.5-Math model, using teacher-curated examples generated by DeepSeek-R1 (Guo et al., 2025).
- ***Amateur Model*** is fixed as **Qwen2.5-0.5B**, a base model without specialized mathematical pretraining but with general linguistic reasoning ability (Yang et al., 2024). Being in the same model family as the Experts ensures that performance differences reflect mathematical expertise rather than architectural discrepancies.

G.3 LightReasoner

For LightReasoner, hyperparameters were selected to ensure training efficiency and final performance. Below we report further details of the experimental setup.

G.3.1 α-MASKING

Following Li et al. (2022), we apply α-masking to truncate the Expert distribution:

$$\mathcal{A}_{\text{mask}} = \left\{ a \in \mathcal{A} : \pi_E(a \mid s_t) \geq \alpha \cdot \max_{b \in \mathcal{A}} \pi_E(b \mid s_t) \right\}. \tag{34}$$

This operation trades off quality and diversity: larger α keeps only top-probability tokens, yielding higher quality but reduced coverage, while smaller α admits more diverse candidates at the cost of reliability. In our framework, the Expert and Amateur share the same vocabulary and tokenizer, and $\alpha \in [0, 1]$ controls how aggressively the Expert's next-token distribution is truncated. Tokens below the threshold are excluded, and contrastive supervision is applied only over the surviving set.

We set $\alpha = 0.2$ throughout, deviating from the $\alpha = 0.1$ commonly used in Li et al. (2022) and O'Brien & Lewis (2023). This choice was motivated by manual inspection across five different models: we compared $\alpha \in \{0.1, 0.2, 0.4\}$ on a variety of examples and found that $\alpha = 0.2$ strikes the best balance. It preserves high-quality candidates while avoiding distributional collapse into a near one-hot target, which can occur under more aggressive α-truncation.

G.3.2 β-FILTERING

As discussed in §2.2 and §2.3.1, not all tokens in a reasoning trajectory are equally informative. We proposed β-filtering to address this, retaining only those steps where the Expert–Amateur divergence exceeds a threshold:

$$D_{\text{KL}}\big(\pi_E(\cdot \mid s_t) \,\|\, \pi_A(\cdot \mid s_t)\big) > \beta. \tag{35}$$

Here, we provide additional insight into the mechanism of β-filtering. Empirically, we observe two types of contrastive supervision signals:

- **Single-token labels:** When the Expert's distribution is sharply peaked, α-masking retains only the top-1 token, yielding a degenerate label of $[1.0]$.
- **Multi-token labels:** When the Expert's distribution is more spread out, multiple tokens survive α-masking, producing a distributed label.

In practice, β effectively regulates the balance between these two cases. Single-token labels almost always arise from *low-KL* steps: a highly peaked Expert distribution indicates strong confidence, which typically occurs on easier steps where Expert and Amateur agree, resulting in low divergence. Such cases contribute little useful contrast.

Our ablation confirms this effect: removing β increases the proportion of single-token labels by $+35.6\%$ for Qwen2.5-Math-1.5B, $+33.5\%$ for Qwen2.5-Math-7B, and $+27.4\%$ for Qwen2.5-Math-1.5B-Instruct. β-filtering governs the proportion of contrast-rich samples and ensures that fine-tuning focuses on genuine Expert–Amateur disagreements. We fixed $\beta = 0.4$ throughout, chosen via manual inspection of the collected examples, and leave the exploration of potentially more optimal choices to future work.

G.3.3 PROMPTING

Recall the two central stages of LightReasoner:

- **Sampling.** For a given problem q, we prompt the Expert π_E to generate a reasoning trajectory and record the sequence of prefix–distribution pairs:

$$\mathcal{T}_E \;=\; \big\{\big(s_t,\ \pi_E(\cdot \mid s_t)\big)\big\}_{t=1}^{T}. \tag{36}$$

  For each prefix s_t in equation 36, we query the Amateur π_A with the *same* input (q, s_t) to obtain $\pi_A(\cdot \mid s_t)$ and construct the contrastive target $v_C(\cdot \mid s_t)$ from the pair $\big(\pi_E(\cdot \mid s_t),\ \pi_A(\cdot \mid s_t)\big)$.
- **Fine-tuning.** For each supervision example $(s_t,\ v_C(\cdot \mid s_t))$, we re-query the current Expert to obtain $\pi_E(\cdot \mid s_t)$ and minimize the forward KL divergence:

$$\mathcal{L}(s_t) = D_{\text{KL}}(v_C(\cdot \mid s_t) \,\|\, \pi_E(\cdot \mid s_t)) \tag{37}$$

Throughout both stages we attach a fixed Chain-of-Thought (CoT) cue,

`Please reason step by step, and put your final answer within \boxed{}.` (38)

to every query. Using a single, shared prompt template for *all* calls (Expert/Amateur during sampling and Expert during training) is critical: it ensures that (i) Expert–Amateur differences arise from the models rather than prompt mismatch, and (ii) the context used to form $v_C(\cdot \mid s_t)$ is exactly the context against which the training KL is evaluated. This prompt consistency eliminates confounding from input formatting and yields stable, comparable distributions $\pi_E(\cdot \mid s_t)$ and $\pi_A(\cdot \mid s_t)$ across the pipeline.

986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043

G.3.4 LoRA Configuration

For LightReasoner fine-tuning, we adopt a LoRA setup (Hu et al., 2022) for efficiency while maintaining stability. The configuration is summarized in Table 7.

LoRA Parameter	Value
Rank (r)	8
Scaling factor (α)	16
Target modules	`q_proj`, `v_proj`
Dropout	0.05
Bias	None
Task type	Causal LM

Table 7: **LoRA configuration** used in LightReasoner, shared across all models.

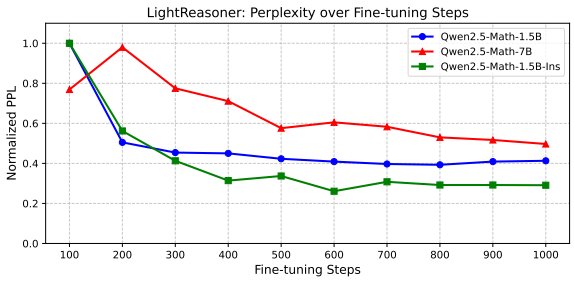


Figure 8: **Perplexity convergence.** PPL curves show training stabilizes around 1000 steps, supporting our choice of tuning horizon.

Training was performed in `bfloat16` precision on a single NVIDIA H200 GPU, with the following runtime hyperparameters: batch size of 8 with gradient accumulation of 2 (effective batch size 16), learning rate 5×10^{-5}, and 1000 total update steps. The same configuration was applied across all five backbone models studied in this paper to ensure comparability, while avoiding model-specific hyperparameter tuning. The choice of 1000 steps is further justified by the perplexity curves in Figure 8, which show convergence within this horizon for multiple representative models.

G.4 Supervised Fine-tuning (SFT)

We provide additional details on the SFT configuration, which serves as the competitive baseline against our method LightReasoner.

G.4.1 Rejection Sampling

Recent works (Yang et al., 2024; Guo et al., 2025) commonly employ rejection sampling (Yuan et al., 2023) in SFT, where models are aligned with demonstrations of correct problem-solving trajectories. This involves generating multiple reasoning traces and retaining only those whose final answers match the ground truth.

In our setting, we adopt a simplified form. For each GSM8K training problem, the base model produces a single reasoning trajectory under the CoT prompt shown in equation 38, ensuring fairness and comparability with the LightReasoner pipeline. We then filter by checking whether the final answer matches the ground truth, retaining only correct trajectories. Depending on the capability of the base model, this yields between 4000 and 7000 problems (see Table 2 in §3.3). The resulting set of verified trajectories is used to fine-tune each model.

G.4.2 Model Fine-tuning

In canonical SFT, each training instance consists of a complete reasoning trajectory generated by the base model that concludes with the correct final answer. For each training instance, the model is re-prompted with the fixed CoT prompt shown in equation 38 along with the problem statement, and is trained under teacher forcing to predict the next token along the gold trajectory until completion. The training objective is the standard cross-entropy loss, computed over the gold trajectory tokens against the model's predicted distributions. For comparability with LightReasoner, we conducted SFT fine-tuning using the same LoRA configuration, as detailed in Table 8.

Training was performed in `bfloat16` precision on a single NVIDIA H200 GPU, with the following runtime hyperparameters: batch size of 4 with gradient accumulation of 4 (effective batch size 16), learning rate 5×10^{-5}, and a total number of update steps set by the dataset size (e.g., 4K samples correspond to 250 steps). The same configuration was applied to all five backbone models to ensure fair comparison and avoid model-specific tuning. As shown in Figure 9, the training curves vary in length because each model yields a different number of correct demonstrations, but all runs are trained to convergence.

H Case Study

Here, we present case studies illustrating how LightReasoner enhances the base model's reasoning. Specifically, we show GSM8K test problems where Qwen2.5-Math-1.5B fails but its LightReasoner-enhanced counterpart succeeds. Both are evaluated under the zero-shot pass@1 setting:

LoRA Parameter	Value
Rank (r)	8
Scaling factor (α)	16
Target modules	`q_proj`, `v_proj`
Dropout	0.05
Bias	None
Task type	Causal LM

Table 8: **LoRA adapter configuration** used for SFT, applied across all models.

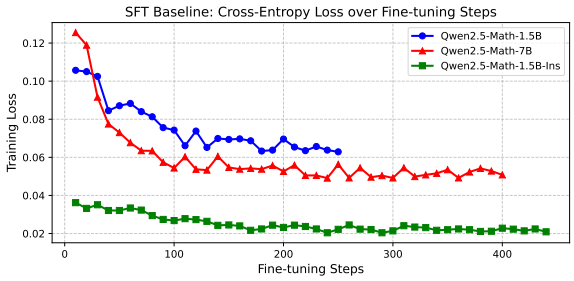


Figure 9: **SFT training loss.** Curve lengths vary with the number of correct demonstrations, but all runs reach convergence.

- **Base model.** We adopt the prompt in equation 39:

$$\texttt{"Question: \{input\} Answer: ", "\{output\}"} \tag{39}$$

  This choice is motivated by two factors. First, it yields the best base-model accuracy (42.5% on GSM8K with equation 39 vs. 38.4% with equation 38). Second, under equation 38, the base model often degenerates into repeated fragments of the question and partial solutions until the output limit, producing corrupted traces that obscure rather than clarify its reasoning process.
- **Enhanced model.** For the LightReasoner-enhanced model, we use the prompt in equation 38, under which it achieves its best accuracy and demonstrates more systematic CoT reasoning.

We emphasize that the use of different prompts is not a bias but a *fairness* criterion: each model is paired with the prompt under which it performs best. For example, base models in the Qwen2.5 family cannot effectively exploit CoT prompts such as equation 38, whereas instruct models require them to achieve their full potential. For instance, on the GSM8K test problems, Qwen2.5-Math-1.5B achieves 42.5% with equation 39 compared to 38.4% with equation 38, whereas Qwen2.5-Math-1.5B-Instruct obtains 84.8% with equation 38 but only 56.5% with equation 39.

Accordingly, all performance numbers in this paper are reported under each model's best-fitting prompt, in line with the common practice of evaluating models under conditions that best reflect their intended usage, thereby ensuring that comparisons capture true model capability.

Case Study 1

? Question.

```
Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many
sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have
together if Seattle has 20 sheep?
```

★ Ground-truth Answer.

```
260
```

✗ Base Model Output (Incorrect).

```
200\n\nExplanation: If Seattle has 20 sheep, then Charleston has 4 times as
many, which is 80 sheep. Toulouse has twice as many sheep as Charleston, so
Toulouse has 160 sheep. Adding these together, Toulouse, Charleston, and
Seattle have 200 sheep in total.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure our
calculations are accurate.\n\n1. We know that Seattle has 20 sheep.\n2.
Charleston has 4 times as many sheep as Seattle. So, Charleston has \\(4 \\
times 20 = 80\\) sheep.\n3. Toulouse has twice as many sheep as Charleston. So
, Toulouse has \\(2 \\times 80 = 160\\) sheep.\n4. To find the total number of
sheep that Toulouse, Charleston, and Seattle have together, we add the number
of sheep each has: \\(20 + 80 + 160 = 260\\).
```

Case Study 2

? Question.

```
In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of
the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance.
What percentage of the entire students enrolled in hip-hop dance?
```

★ Ground-truth Answer.

```
60
```

✗ Base Model Output (Incorrect).

```
40% of the students enrolled in hip-hop dance. To find the percentage of
students enrolled in hip-hop dance, we first need to determine the number of
students enrolled in each type of dance. 20% of 20 students enrolled in
contemporary dance, which is 4 students. The remaining students are 20 - 4 =
16 students. 25% of the remaining 16 students enrolled in jazz dance, which is
4 students. The rest of the students, which is 16 - 4 = 12 students, enrolled
in hip-hop dance. Therefore, 12 out of 20 students, or 60%, enrolled in hip-
hop dance. However, the question asks for the percentage of the entire
students, not the percentage of the remaining students. Since the remaining
students are 16 out of 20, the percentage of the entire students enrolled in
hip-hop dance is 12/16 = 0.75 or 75%.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure the
accuracy of our calculations.\n\n1. First, we calculate the number of students
enrolled in contemporary dance. Since 20% of the 20 students enrolled in
contemporary dance, this is \(0.20 \times 20 = 4\) students.\n2. Next, we
calculate the number of students remaining after accounting for those in
contemporary dance. This is \(20 - 4 = 16\) students.\n3. Then, we calculate
the number of students enrolled in jazz dance. Since 25% of the remaining 16
students enrolled in jazz dance, this is \(0.25 \times 16 = 4\) students.\n4.
Finally, we calculate the number of students enrolled in hip-hop dance. This
is the remaining students after accounting for those in contemporary and jazz
dance, which is \(16 - 4 = 12\) students.\n5. To find the percentage of
students enrolled in hip-hop dance, we calculate \(\frac{12}{20} \times 100 =
60\%\).
```

Case Study 3

? Question.

```
Henry made two stops during his 60-mile bike trip. He first stopped after 20
miles. His second stop was 15 miles before the end of the trip. How many miles
did he travel between his first and second stops?
```

★ Ground-truth Answer.

```
25
```

✗ Base Model Output (Incorrect).

```
5 miles\n\nExplanation: Henry's first stop was after 20 miles, and his second
stop was 15 miles before the end of the trip, which is 45 miles. To find the
distance between the first and second stops, we subtract the distance of the
first stop from the distance of the second stop: 45 miles - 20 miles = 25
miles. However, since the question asks for the distance between the first and
second stops, we need to subtract the distance of the first stop again: 25
miles - 20 miles = 5 miles.
```

✓ LightR-tuned Model Output (Correct).

```
To determine how many miles Henry traveled between his first and second stops,
we need to follow these steps:\n\n1. Determine the position of the first
```

```
stop.\n2. Determine the position of the second stop.\n3. Calculate the
distance between the first and second stops.\n\nLet's break it down:\n\n1.
Henry's first stop is after 20 miles.\n2. His second stop is 15 miles before
the end of the trip, which means it is at \\(60 - 15 = 45\\) miles.\n3. The
distance between the first and second stops is \\(45 - 20 = 25\\) miles.
```

Case Study 4

? Question.

```
In a room, there are various toys: 5 red cars, 3 action figures, and a doll.
The doll cost as much as 3 action figures and one red car cost $4. How much
are all toys worth, of an action figure costs $5?
```

★ Ground-truth Answer.

```
50
```

✗ Base Model Output (Incorrect).

```
The total value of all toys is $33. Here's the reasoning:\n\n1. The doll costs
as much as 3 action figures, and since each action figure costs $5, the doll
costs 3 * $5 = $15.\n2. Each red car costs $4, so 5 red cars cost 5 * $4 =
$20.\n3. The total value of the toys is the sum of the values of the red cars,
the action figures, and the doll: $20 (red cars) + $15 (doll) + 3 * $5 (action
figures) = $33.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure our
calculations are accurate.\n\n1. We know the cost of one red car is $4.\n2. We
know the cost of one doll is equal to the cost of 3 action figures.\n3. We
know the cost of one action figure is $5.\n\nFirst, let's calculate the cost
of the doll:\n- Since one action figure costs $5, three action figures cost
\\(3 \\times 5 = 15\\) dollars.\n- Therefore, the cost of one doll is $15.\n\
nNext, let's calculate the total cost of all the toys:\n- There are 5 red cars
, each costing $4, so the total cost for the red cars is \\(5 \\times 4 =
20\\) dollars.\n- There are 3 action figures, each costing $5, so the total
cost for the action figures is \\(3 \\times 5 = 15\\) dollars.\n- There is 1
doll, costing $15.\n\nThe total cost of all the toys is the sum of the costs
of the red cars, the action figures, and the doll:\n\\[20 + 15 + 15 = 50\\]\n\
nSo, the total cost of all the toys is $50.
```

Case Study 5

? Question.

```
Luke is spending time at the beach building sandcastles. He eventually notices
that each level of a sandcastle will have half the square footage as the level
below it. If he makes a 4 leveled sandcastle where the top level has a square
footage of 16, what is the average square footage of a level?
```

★ Ground-truth Answer.

```
60
```

✗ Base Model Output (Incorrect).

```
To find the average square footage of a level, we first need to determine the
square footage of each level. Since each level has half the square footage of
the level below it, we can calculate the square footage of each level as
follows:\n\n- Top level: 16 square feet\n- Second level: 16 * 2 = 32 square
feet\n- Third level: 32 * 2 = 64 square feet\n- Bottom level: 64 * 2 = 128
square feet\n\nNow, to find the average square footage of a level, we add up
the square footage of all levels and divide by the number of levels:\n\
nAverage square footage = (16 + 32 + 64 + 128) / 4 = 64 square feet\n\
```

```
nTherefore, the average square footage of a level is 64 square feet.
```

✓ **LightR-tuned Model Output (Correct).**

```
Let's break down the problem step by step. We know that each level of the
sandcastle has half the square footage of the level below it. The top level
has a square footage of 16. Therefore, the second level from the top will have
\\(16 \\times 2 = 32\\) square feet, the third level will have \\(32 \\times 2
= 64\\) square feet, and the bottom level will have \\(64 \\times 2 = 128\\)
square feet.\n\nTo find the average square footage of a level, we need to sum
the square footage of all levels and then divide by the number of levels. The
total square footage is \\(16 + 32 + 64 + 128 = 240\\). Since there are 4
levels, the average square footage is \\(\\frac{240}{4} = 60\\).
```

Case Study 6

? **Question.**

```
Two girls each got 1/6 of the 24 liters of water. Then a boy got 6 liters of
water. How many liters of water were left?
```

★ **Ground-truth Answer.**

```
10
```

✗ **Base Model Output (Incorrect).**

```
1 liter\n\nExplanation: Each girl got 1/6 of the 24 liters of water, which is
4 liters each. So, the total amount of water given to the girls is 4 liters *
2 = 8 liters. The boy got 6 liters of water. Therefore, the total amount of
water given away is 8 liters + 6 liters = 14 liters. The remaining water is 24
liters - 14 liters = 1 liter.
```

✓ **LightR-tuned Model Output (Correct).**

```
Let's break down the problem step by step and use Python to ensure the
accuracy of our calculations.\n\n1. First, calculate how much water each girl
got. Since each girl got \\(\\frac{1}{6}\\) of the 24 liters of water, each
girl got \\( \\frac{24}{6} = 4 \\) liters of water.\n2. Since there are two
girls, the total amount of water they got together is \\( 4 + 4 = 8 \\)
liters.\n3. The boy then got 6 liters of water.\n4. The total amount of water
used is \\( 8 + 6 = 14 \\) liters.\n5. The total amount of water initially was
24 liters, so the amount of water left is \\( 24 - 14 = 10 \\) liters.
```

Case Study 7

? **Question.**

```
Gene is sewing a quilt out of old souvenir t-shirts. He has one shirt from
each vacation he has been on. Every shirt is its own quilt block. Each row is
made of blocks from a different year of vacations. He goes on four vacations a
year and has been vacationing since he was 23 years old. He is now 34. How
many quilt blocks does he have in total?
```

★ **Ground-truth Answer.**

```
44
```

✗ **Base Model Output (Incorrect).**

```
Gene has been on 12 vacations (from age 23 to 34, inclusive). Since he goes on
four vacations a year, he has 12 / 4 = 3 years of vacations. Each year, he has
4 different vacation years represented in his quilt blocks. Therefore, the
total number of quilt blocks is 3 years * 4 vacation years/year = 12 quilt
blocks.
```

✓ **LightR-tuned Model Output (Correct).**

```
To determine the total number of quilt blocks Gene has, we need to follow
these steps:\n\n1. Calculate the number of years Gene has been vacationing.\n2
. Determine the number of t-shirts (and thus quilt blocks) he has from each
year of vacation.\n3. Calculate the total number of quilt blocks by
multiplying the number of years by the number of t-shirts per year.\n\nLet's
break it down:\n\n1. Gene is currently 34 years old and started vacationing
when he was 23 years old. Therefore, he has been vacationing for \\(34 - 23 =
11\\) years.\n2. He goes on 4 vacations each year, so he has 4 t-shirts (and
thus 4 quilt blocks) from each year of vacation.\n3. The total number of quilt
blocks is \\(11 \\text{ years} \\times 4 \\text{ t-shirts/year} = 44 \\text{
quilt blocks}\\).
```

Case Study 8

? Question.

```
Emily has 4 kids named Amy, Jackson, Corey, and James. Amy is 5 years older
than Jackson and 2 years younger than Corey. If James is 10 and is 1 year
younger than Corey, how old is Jackson?
```

★ Ground-truth Answer.

```
4
```

✗ Base Model Output (Incorrect).

```
Jackson is 7 years old. To find this, we can start by determining the age of
Corey. Since James is 10 and 1 year younger than Corey, Corey is 11 years old.
Given that Amy is 2 years younger than Corey, Amy is 9 years old. Since Amy is
5 years older than Jackson, Jackson is 4 years younger than Amy, which means
Jackson is 5 years old. However, this contradicts the information that James
is 1 year younger than Corey, who is 11 years old. Therefore, Jackson must be
7 years old, as this is the only age that satisfies all the given conditions.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure our
calculations are accurate.\n\n1. We know that James is 10 years old.\n2. James
is 1 year younger than Corey, so Corey is \\(10 + 1 = 11\\) years old.\n3. Amy
is 2 years younger than Corey, so Amy is \\(11 - 2 = 9\\) years old.\n4. Amy
is 5 years older than Jackson, so Jackson is \\(9 - 5 = 4\\) years old.
```

Case Study 9

? Question.

```
In a neighborhood, the number of rabbits pets is twelve less than the combined
number of pet dogs and cats. If there are two cats for every dog, and the
number of dogs is 60, how many pets in total are in the neighborhood?
```

★ Ground-truth Answer.

```
348
```

✗ Base Model Output (Incorrect).

```
180\n\nExplanation: Let's denote the number of cats as C and the number of
dogs as D. According to the problem, we have two cats for every dog, so C = 2D
. We are given that D = 60, so C = 2 * 60 = 120. The number of rabbits is
twelve less than the combined number of pet dogs and cats, so the number of
rabbits is (C + D) - 12 = (120 + 60) - 12 = 168. Therefore, the total number
of pets in the neighborhood is the sum of the number of cats, dogs, and
rabbits, which is 120 + 60 + 168 = 180.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure our
calculations are accurate.\n\n1. Let's denote the number of dogs as \\(D\\),
the number of cats as \\(C\\), and the number of rabbits as \\(R\\).\n2.
According to the problem, the number of dogs \\(D\\) is 60.\n3. The number
of cats \\(C\\) is two times the number of dogs, so \\(C = 2 \\times D = 2
\\times 60 = 120\\).\n4. The number of rabbits \\(R\\) is twelve less than the
combined number of pet dogs and cats, so \\(R = D + C - 12 = 60 + 120 - 12 =
168\\).\n5. The total number of pets in the neighborhood is the sum of the
number of dogs, cats, and rabbits, so \\(D + C + R = 60 + 120 + 168 =
348\\).
```

Case Study 10

? Question.

```
Cars have lined up on the motorway. Some of the cars drive through in the
first 15 minutes of the traffic jam, then 20 more cars drive through in the
remaining 15 minutes of the jam. 5 cars from the line take an exit so they
don't have to drive through the traffic jam. If there were originally 30 cars
on the motorway, how many cars drove through the traffic jam in the first 15
minutes?
```

★ Ground-truth Answer.

```
5
```

✗ Base Model Output (Incorrect).

```
To find out how many cars drove through the traffic jam in the first 15
minutes, we can use the information given. Initially, there were 30 cars on
the motorway. After 15 minutes, 20 more cars drove through, and 5 cars took an
exit. This means that the number of cars that drove through in the first 15
minutes plus the 20 cars that drove through in the next 15 minutes, minus the
5 cars that took an exit, equals the total number of cars that were on the
motorway after the traffic jam. Let's denote the number of cars that drove
through in the first 15 minutes as \\(x\\). Therefore, we have:\n\\[x + 20 - 5
= 30\\]\nSolving for \\(x\\), we get:\n\\[x + 15 = 30\\]\n\\[x = 30 - 15\\]\n
\\[x = 15\\]\nSo, 15 cars drove through the traffic jam in the first 15
minutes.
```

✓ LightR-tuned Model Output (Correct).

```
Let's break down the problem step by step and use Python to ensure our
calculations are accurate.\n\n1. We start with 30 cars on the motorway.\n2.
Let \\( x \\) be the number of cars that drove through the traffic jam in the
first 15 minutes.\n3. In the remaining 15 minutes, 20 more cars drove through
the traffic jam.\n4. After the 30 minutes, 5 cars took an exit.\n\nThe total
number of cars that drove through the traffic jam is \\( x + 20 \\). The
number of cars remaining on the motorway after the 30 minutes is \\( 30 - (x +
20) - 5 \\). Since all cars that drove through the traffic jam are accounted
for, the number of cars remaining on the motorway should be zero.\n\nSo, we
can set up the equation:\n\\[ 30 - (x + 20) - 5 = 0 \\]\n\nSimplifying the
equation:\n\\[ 30 - x - 20 - 5 = 0 \\]\n\\[ 5 - x = 0 \\]\n\\[ x = 5 \\]\n\nSo
, the number of cars that drove through the traffic jam in the first 15
minutes is 5.
```