

Universal Reasoner: A Single, Composable Plug-and-Play Reasoner for Frozen LLMs

Jaemin Kim*

Hangeol Chang*

Hyunmin Hwang*

Choonghan Kim

Jong Chul Ye

Korea Advanced Institute of Science and
Technology (KAIST)

*Equal contributions



Paper

Current Monolithic Fine-Tuning is Inefficient

- Recently, LLMs have improved their reasoning performance through RL methods such as PPO and GRPO.
- To specialize a reasoning LLM for a specific domain, a significant amount of memory is required.
- Therefore, parameter-efficient methods (PEFTs) such as LoRA are used, but they have the following two limitations:
 1. **Architecture-dependent:** A LoRA trained on a 3B model cannot be used on a 14B model.
 2. **Inability to modularize:** LoRAs trained for different tasks cannot be combined.

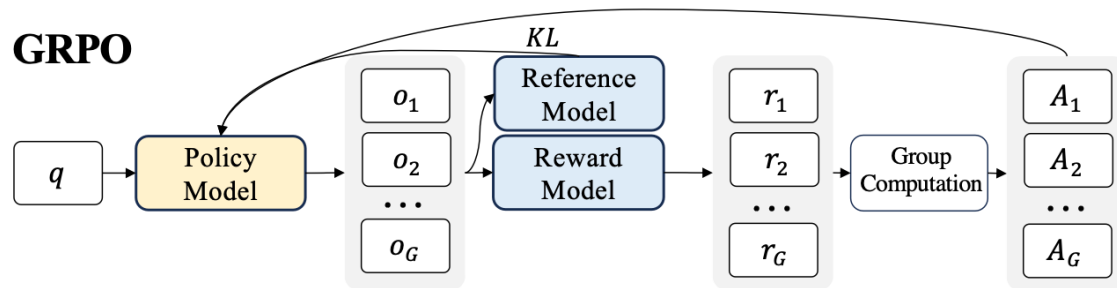


Figure 1. GRPO

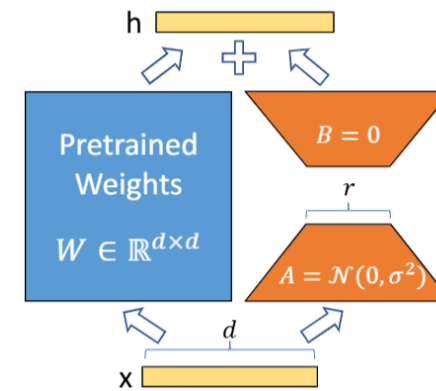


Figure 2. LoRA(Low-Rank Adaptation)

Key Question

Is it possible to modularize the reasoning capability separately while keeping the LLM frozen?

We Propose a Modular Method, UniR (Universal Reasoner)

- UniR does not train the Backbone LLM (π_b), but instead trains a lightweight, modular reasoning module (π_r).
- The reasoning module is trained to maximize verifiable rewards via GRPO.
 - e.g., Mathematical task \rightarrow Answer accuracy / Translation \rightarrow Translation quality
- At inference, the reasoning module guides the backbone LLM's reasoning at the logit level.

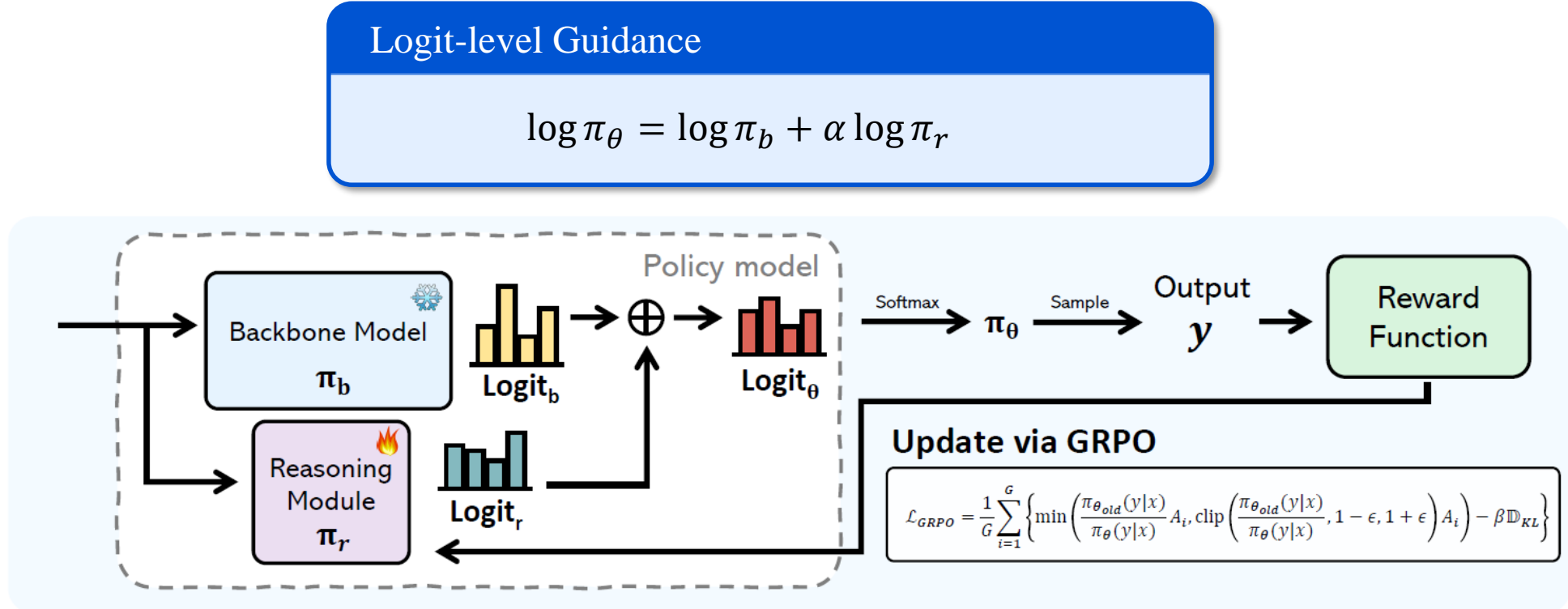


Figure 3. UniR Pipeline

From Trajectory Rewards to Token Guidance

- We can model the trajectory-level reward as the sum of token-level log-probabilities.

Assumption: Reward Decomposition

Verifiable rewards are trajectory-level. We propose:

$$\frac{1}{\beta} r(x, y) = \sum_{t=1}^{|y|} \log \pi_r(y_t | x, y_{<t}; \phi).$$

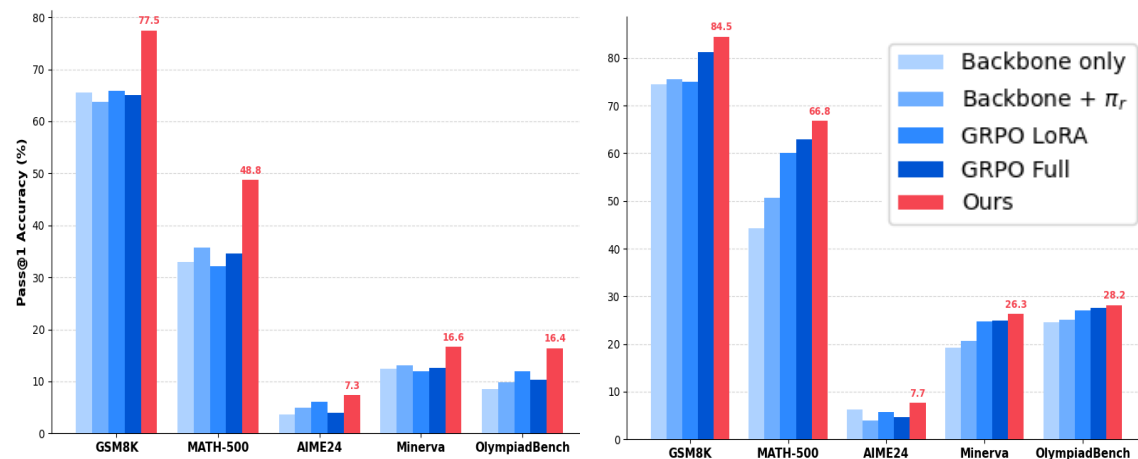
- The optimization policy for KL-regularized is expressed in closed form as follows : $\pi_{\theta}^*(y|x) = \frac{1}{Z(x)} \pi_b(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$
- Substituting the assumptions into the above equation, the optimal next-token probability is expressed as follows:

$$\log \pi_{\theta}^*(y_t | \cdot) = \log \pi_b(y_t | \cdot) + \underbrace{\log \pi_r(y_t | \cdot)}_{\text{Reasoning Signal}} - \log Z'$$

- **Intuitive interpretation:** Training $\log \pi_r$ to maximize the reward is equivalent to learning the optimal soft Q-function.
(See Theorem 1).

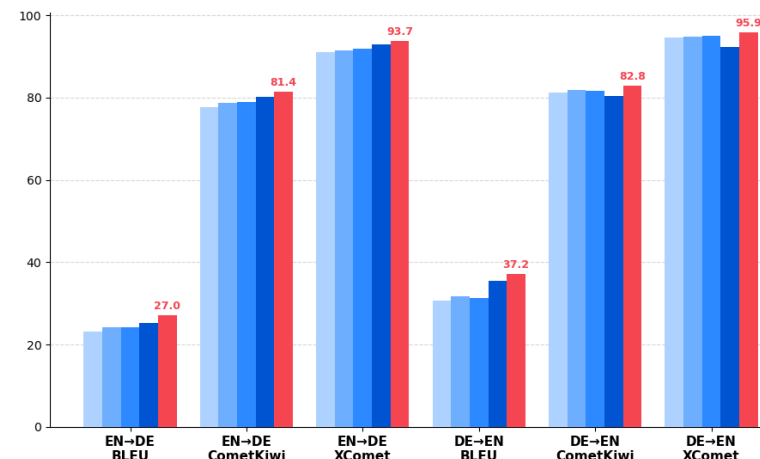
UniR Achieves Higher Accuracy than Monolithic Fine-Tuning

- UniR demonstrates higher performance compared to GRPO Full Fine-Tuning and LoRA baselines.
- The Backbone LLM uses 3B, while the Reasoning Module uses 1B or 1.5B.
 - **Math Benchmark:** The Qwen and Llama models trained on GSM8k and MATH-12k demonstrate high performance.
 - **Translation task:** The Llama model demonstrates high translation quality.



Method	Llama3.2-3B Family	Qwen2.5-3B Family
Backbone only	24.7	33.7
GRPO LoRA	25.6	38.5
GRPO Full	25.3	40.3
UniR (Ours)	33.3	42.7

Table 1. Performance Comparison for Mathematical Benchmarks



Method	EN → DE	DE → EN
Backbone only	63.96	68.82
GRPO LoRA	65.03	69.36
GRPO Full	66.09	69.43
UniR (Ours)	67.39	71.97

Table 2. Performance Comparison for Machine Translation

UniR Can Transfer to Larger Models at Inference Time

- We demonstrate that a reasoning module trained using the UniR method can effectively guide larger backbone LLMs.
- **Weak-to-Strong Generalization (Transferability):**
 - **Experiment setup:** Train π_r (1.5B) with π_b (3B) \rightarrow Inference with π_b (**14B**).
 - **Experiment results:** It demonstrates higher performance than existing Backbone LLMs without additional training, proving that a plug-and-play approach is feasible.

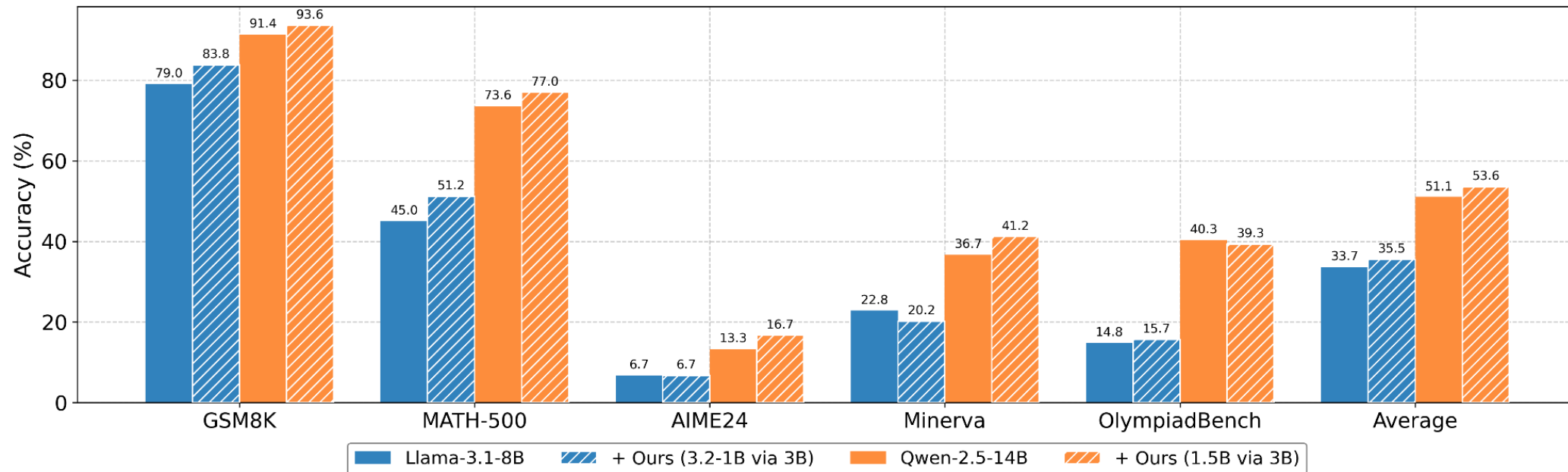


Figure 4. Performance comparison after transferring π_r , trained using the UniR method, to a larger π_b

UniR Can Combine to Solve Complex, Multi-Objective Task

- We demonstrate that complex tasks can be effectively handled by linearly combining the modules trained for each task.
- **Multi-Objective Generalization (Composability):**
 - **Goal Task:** Solving Math Problems Written in German in English (German-to-English Math Solving)
 - **Experiment setup:** Infer by linearly combining π_r^{Math} , π_r^{Trans} which have been trained on mathematics and translation, respectively

Linear Composability

$$\log \pi_{\theta} \propto \log \pi_b + \alpha \log \pi_r^{Math} + (1 - \alpha) \log \pi_r^{Trans}$$

- **Results:**
 - The larger α is, the better the math performance, but the worse the translation performance.
 - The smaller α is, the better the translation performance, but the worse the math performance.
 - Values in between show the Pareto frontier.

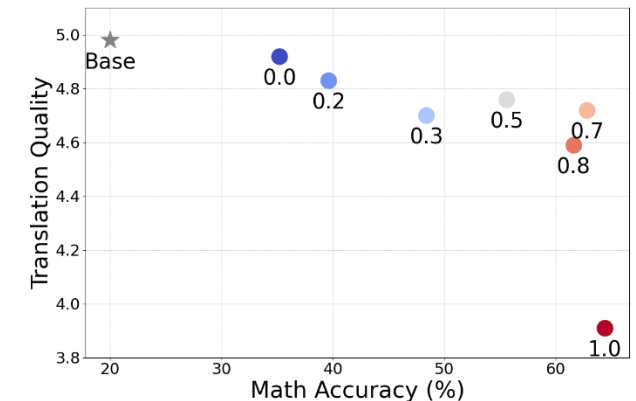


Figure 5. Comparison of performance when solving German math problems in English

Extensions: Multimodal and Domain Generalization

■ UniR's plug-and-play generalization can also be applied to the vision-language and medical domains.

■ **1. Vision Language Model (VLM):**

- **Backbone:** Qwen2.5-VL-3B (Frozen)
- **Setup:** We applied π_r^{Math} which was trained on a math text benchmark, to the VLM backbone.
- **Results:** Performance has also improved in MathVerse and Geometry 3k, which require Vision data.

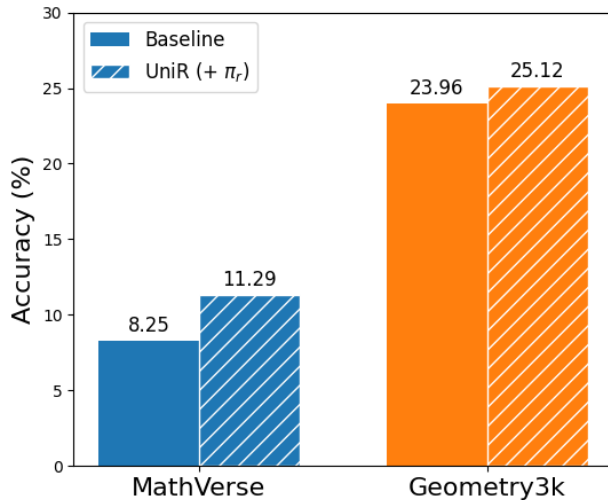


Figure 6. Comparison of performance in math benchmarks requiring visual information

■ **2. Medical Reasoning (MIMIC-IV):**

- **Task:** Predicting ICU Readmission and Length of Stay
- **Setup:** Inference is performed by combining the π_r learned from each task.
- **Results:** This demonstrates a Pareto frontier, and performance is also improved for the new task (mortality prediction).

Task	Method	Reasoning module	Acc.	F1
Readmission	Backbone only	-	39.46	25.87
	UniR	Readmission	52.91	26.77
	UniR	Readmission + LOS	44.88	28.79
Length of stay (LOS)	Backbone only	-	9.64	5.92
	UniR	LOS	67.77	16.25
	UniR	Readmission + LOS	61.75	18.32
Mortality	Backbone only	-	39.76	28.27
	UniR	Readmission + LOS	63.65	35.18

Table 3. Comparison of ICU Prediction Performance Using the MIMIC Benchmark

Analysis: UniR Allows Memory-Efficient Optimization

- Since UniR trains only a relatively small reasoning module, it enables memory-efficient and stable training.
 - **VRAM:** Since gradients do not flow through the large backbone LLM, VRAM usage is reduced.
 - **Train Stability:** By freezing the backbone LLM, UniR effectively prevents high reward variance—one of the issues with GRPO—and ensures a stable increase in rewards.

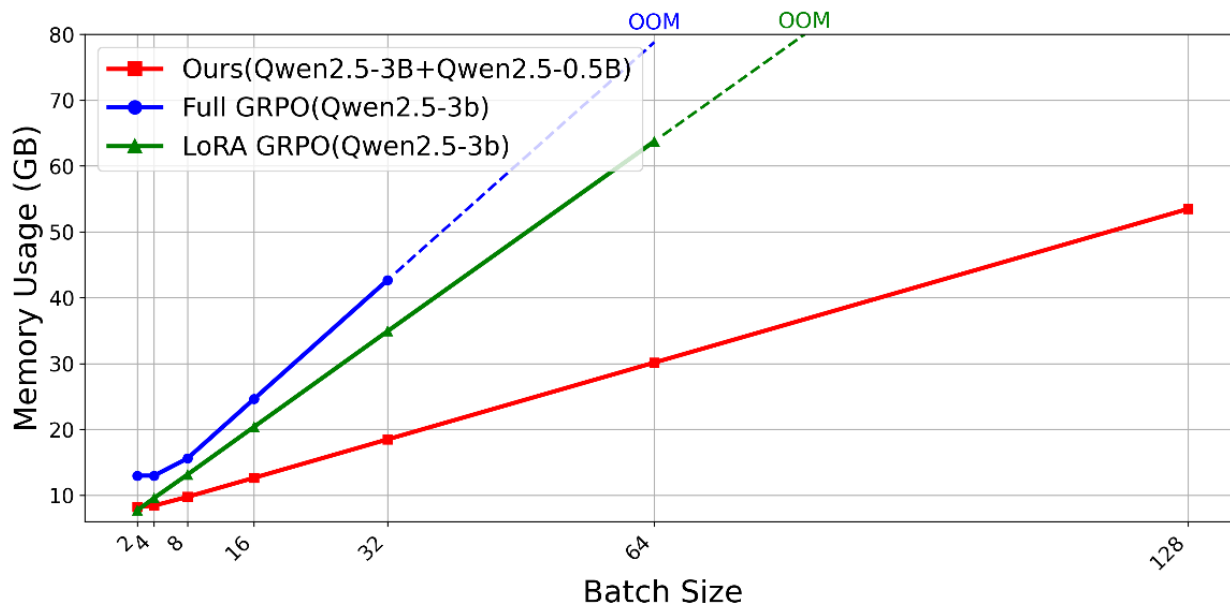


Figure 7. Comparison of VRAM usage by batch size during training

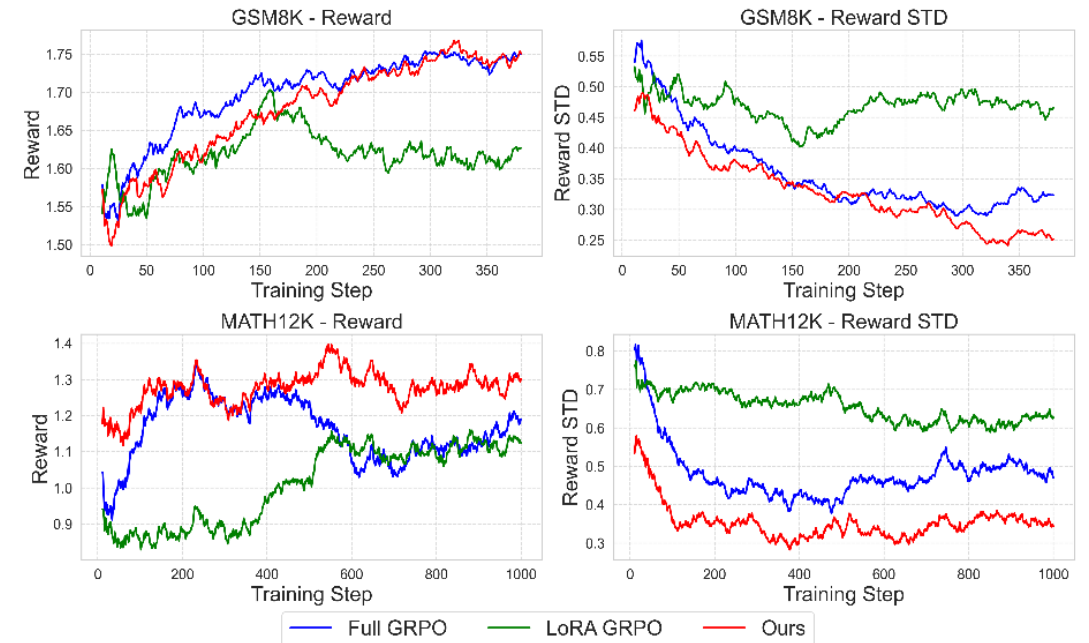


Figure 8. Comparison of the mean and standard deviation of rewards during training

Analysis: UniR Enables Efficient Inference

■ The following demonstrates that UniR is a methodology that can be effectively applied to inference as well.

- **Pre-trained Reasoning Model:**

- When the DeepSeek 1.5B model, which is capable of reasoning, was applied as the Reasoning Module, performance on math benchmarks improved even without additional training, and performance improved further when additional training was performed using the UniR method.

- **Inference Speed:**

- **Issue:** The existing UniR approach requires an additional model, which slows down inference.
- **Solution:** Applying Collaborative Speculation (CoS)[1], based on Speculative Decoding, mitigates this performance degradation.

Backbone Model (π_b)	Reasoning Module (π_r)	Train with UniR	Avg.
DeepSeek 1.5B	–	X	32.8
Qwen2.5 3B	DeepSeek 1.5B	X	43.7
Qwen2.5 3B	DeepSeek 1.5B	O	46.9

Table 4. Performance Comparison of the Reasoning Model and UniR with Additional Training

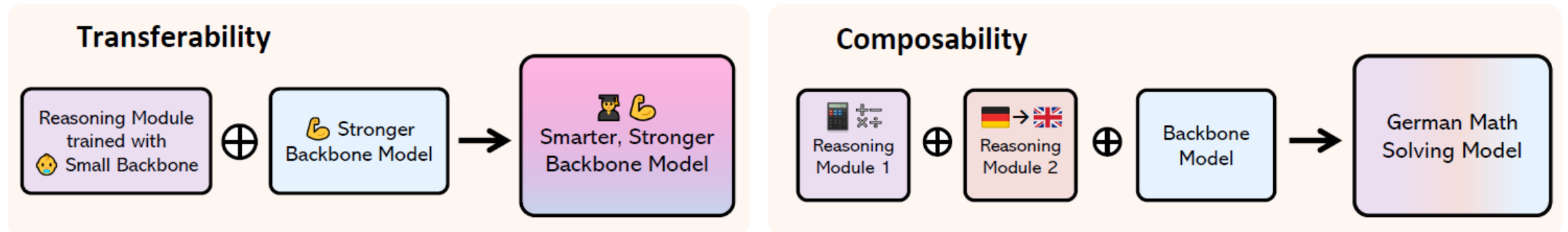
Method	Sec./Token (\downarrow)
Backbone only	0.0262
UniR (Naïve)	0.0414
UniR + CoS	0.0279

Table 5. Comparison of UniR’s Inference Speed

[1] Fu, Jiale, et al. “Fast Large Language Model Collaborative Decoding via Speculation.”, *ICML 2025*.

Conclusion

- UniR modularizes reasoning capabilities so they can be learned separately, replacing the approach of fine-tuning a single model to specialize in a specific domain.
 1. **Efficient:** Since only small modules are trained, memory-efficient training is possible.
 2. **Effective:** Outperforms the performance of existing LoRA and GRPO full fine-tuning on math and translation benchmarks.
 3. **Transferable:** The reasoning module effectively controls larger backbone LLMs without additional training.
 4. **Composable:** Combines the necessary modules for tasks requiring diverse capabilities to enhance performance during inference.



Appendix

Theoretical Justification

- Lemma 1. In a KL-normalized RL objective, the optimal policy remains unchanged even if any trajectory-level reward $r(x,y)$ is replaced with a reward in the form of a log probability. [1]
- This is because the optimal policy remains the same even if the reward is shifted by a constant $f(x)$ that is independent of the output (y).

$$\hat{r}(x, y) = \beta \log p_r(y|x) \qquad p_r(y|x) = \frac{\exp(r(x, y)/\beta)}{\sum_{y'} \exp(r(x, y')/\beta)}$$

- Corollary 1. Corollary 1. The trajectory-level log-probability reward $\log p_r(x,y)$ can be expressed as the sum of token-level log-probability rewards.

$$\log p_r(y|x) = \sum_t \log p_r(y_t | x, y_{<t})$$

- Corollary 2. The optimal policy can be described as follows.

$$\pi^*(y|x) \propto \pi_b(y|x) \exp\left(\sum_t \log p_r(y_t|x, y_{<t})\right)$$

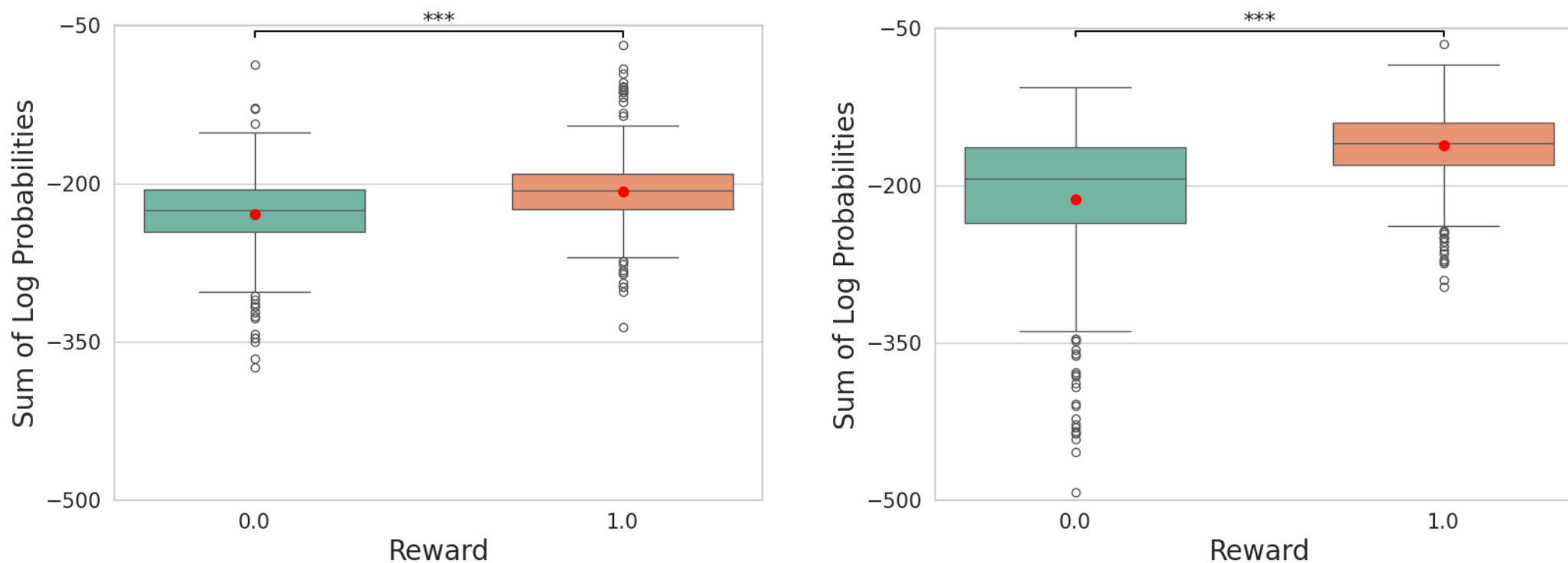
- Theorem 1. If the learning converges, by the definition of the optimal policy, the optimized inference module is equal to a scaled version of the optimized soft-Q function.

$$\log \pi^*(y_t|x, y_{<t}) = \log \pi_b(y_t|x, y_{<t}) + \frac{1}{\beta} Q^*(y_t|x, y_{<t}) - \log Z'$$

$$\log p_r(y_t|x, y_{<t}) = \frac{1}{\beta} Q^*(y_t|x, y_{<t})$$

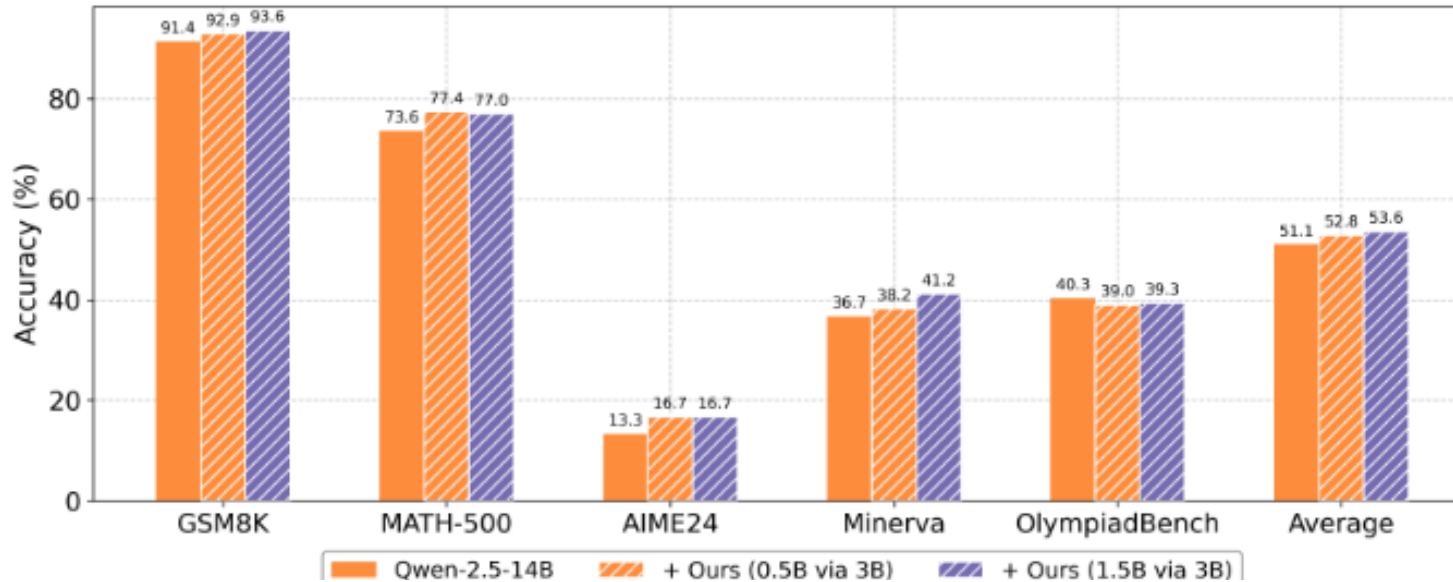
[1] Xu, Yuancheng, et al. "GenARM: Reward guided generation with autoregressive reward model for test-time alignment.", ICLR 2025

Analysis of the Reward Decomposition



- The sum of log probabilities calculated by the Reasoning Module consistently remains high for the correct answer (Reward = 1).
- This difference becomes even more pronounced after training with the UniR method.
- This indicates that the Reasoning Module, modeled using the UniR method, is internally incorporating the reward signal.

Analysis of the Reward Decomposition



Method	Trained Model	In-distribution		Out-of-distribution			Avg.	
		GSM8K	MATH-500	AIME24	Minerva	OlympiadBench		
Qwen2.5 3B	Backbone only	-	74.4 ± 1.0	44.2 ± 1.3	6.3 ± 3.7	19.2 ± 1.7	24.6 ± 0.9	33.7
	Backbone + 0.5B	-	70.9 ± 0.9	43.1 ± 1.2	6.0 ± 3.1	16.5 ± 1.9	22.5 ± 0.9	31.8
	Ours	0.5B	81.1 ± 0.4	62.2 ± 1.4	5.0 ± 3.2	24.7 ± 2.0	26.3 ± 4.8	39.9
	Ours	1.5B	84.5 ± 0.7	66.8 ± 1.4	7.7 ± 2.7	26.3 ± 2.0	28.2 ± 0.8	42.7

- As the size of the Reasoning Module increases, the performance improvement achieved through training with UniR also increases.

Example of Output

Prompt: Hannah's city is having a big display of fireworks for the 4th of July. They're going to set off 15 boxes of 20 fireworks each. Hannah's house is at the right angle to see 40% of the city's fireworks. Hannah will also set off 3 boxes of 5 fireworks each in her backyard. How many fireworks will Hannah see in total?

Backbone Model:

<think> First, calculate the total number of fireworks in the display. There are 15 boxes of 20 fireworks each, so the total number of fireworks is $15 * 20$. Then, calculate the number of fireworks Hannah can see in the display, which is 40% of the total number of fireworks. Next, calculate the number of fireworks in the 3 boxes she sets off in her backyard, which is $3 * 5$. Finally, add the number of fireworks she can see in the display to the number of fireworks she sets off in her backyard to get the total number of fireworks Hannah sees. </think>
<answer> The total number of fireworks in the display is $15 * 20 = 300$. Hannah can see 40% of these, which is $0.40 * 300 = 120$ fireworks. The number of fireworks in the 3 boxes she sets off in her backyard is $3 * 5 = 15$. Therefore, the total number of fireworks Hannah sees is $120 + 15 = 135$. </answer>

GRPO:

<think> First, let's calculate the total number of fireworks in the display. There are 15 boxes of 20 fireworks each, so the total number of fireworks is $15 * 20$. Then, we need to find out how many of these fireworks Hannah can see, which is 40% of the total. After that, we'll add the 3 boxes of 5 fireworks each that Hannah sets off in her backyard. The total number of fireworks Hannah sees will be the sum of the fireworks she can see in the display and the fireworks she sets off in her backyard. </think> <answer> The total number of fireworks in the display is $15 * 20 = 300$. Hannah can see 40% of these, which is $0.40 * 300 = 120$ fireworks. She also sets off 3 boxes of 5 fireworks each, which is $3 * 5 = 15$ fireworks. Therefore, the total number of fireworks Hannah sees is $120 + 15 = 135$. </answer>

Ours:

<think> First, let's calculate the total number of fireworks in the display. There are 15 boxes of 20 fireworks each, so the total number of fireworks is $15 * 20 = 300$. Hannah's house is at the right angle to see 40% of the city's fireworks, so the number of fireworks she will see is $300 * 0.4 = 120$. In her backyard, she will set off 3 boxes of 5 fireworks each, which totals $3 * 5 = 15$ fireworks. Therefore, the total number of fireworks Hannah will see is $120 + 15 = 135$. </think> <answer> 135 </answer>
