Post-training (III): Reinforcement Learning from Verified Rewards

Zheng Fang, Jayden Li, Liyi Chen

Reinforcement Learning from Verified Rewards (RLVR)

- Definition: RL where reward signals come from automatic verification (e.g., math correctness, code unit tests, output format).
- This session – Four Perspectives:
  - DeepSeek-R1: RLVR alone can drive emergent reasoning and enable distillation
  - DAPO: An open-source RLVR system to improve reproducibility
  - Yue et al.: RLVR mainly re-weights existing abilities rather than creating new ones
  - Chu et al.: Compares SFT vs RLVR -> RLVR generalizes better

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

Paper Overview

- Goal: Enhance reasoning in large language models
- Key Contributions
  - DeepSeek R1-Zero: RL alone → emergent reasoning
  - DeepSeek R1: Cold Start + multi-stage RL → usable model
  - Distillation: Transfer reasoning to smaller models

Core Algorithm: Group Relative Policy Optimization (GRPO)

- GRPO vs PPO
  - PPO variant, no critic → cheaper and simpler training
- Eq. (3): Relative Advantage
  - Reward compared to group mean, normalized
- Eq. (1): Training Objective
  - Ratio × advantage, with clipping for stability
- Eq. (2): KL Regularization
  - Keeps policy close to reference, prevents drift

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O \mid q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{old}}(o_i \mid q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{old}}(o_i \mid q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}(\pi_\theta \| \pi_{ref})\right), \tag{1}$$

$$\mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}) = \frac{\pi_{ref}(o_i \mid q)}{\pi_\theta(o_i \mid q)} - \log\frac{\pi_{ref}(o_i \mid q)}{\pi_\theta(o_i \mid q)} - 1, \tag{2}$$

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

DeepSeek-R1-Zero: RL on Base Model (No SFT)

- **Training Setup**: Base model trained with RL only, no supervised fine-tuning
- **Algorithm**: GRPO (no critic, group-based rewards)
- **Tasks**: Mix of math (AIME 2024, MATH), coding challenges (Codeforces, LiveCodeBench), and QA benchmarks – all with automatically checkable answers (for reward signals)
- **Reward Modeling**
  - Accuracy: correct final answer / passed test cases
    - Rewards only depend on final answer → model must explore its own reasoning steps
  - Format: reasoning inside <think>…</think>, final answer in <answer>…</answer>

DeepSeek-R1-Zero: Performance

- Steady Improvement (Figure 2)
  - 15.6% → 71% pass@1
- Majority Voting Boost (Table 2)
  - 86.7% (surpasses OpenAI o1-0912)

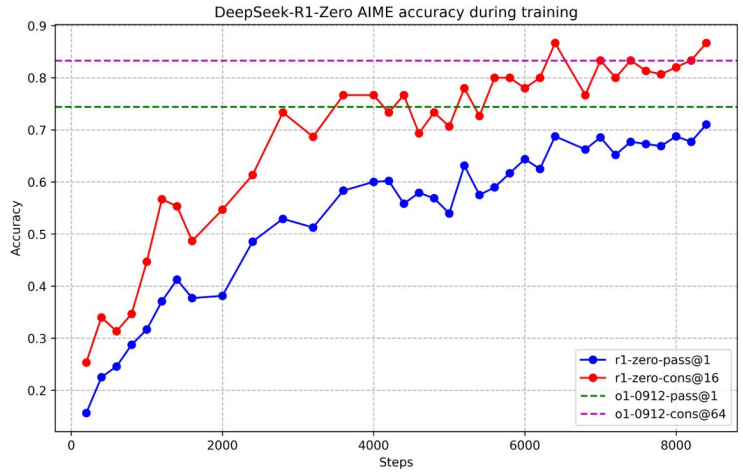


Figure 2: AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

Table 2: Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

DeepSeek-R1-Zero: Observations

- Emergent Behaviors
  - Longer “thinking chains” (self-extended reasoning length)
  - “Aha moments”: stop, reflect, restart with a new path
- Limitations
  - Poor readability (messy reasoning steps)
  - Language mixing (Chinese + English in the same response)
  - Not user-friendly → motivates DeepSeek-R1 with Cold Start

DeepSeek-R1: Adding “Cold Start” + Multi-Stage RL

- **Motivation**: Improve readability & alignment beyond R1-Zero
- Stage 1: Cold Start
  - Fine-tune base model with curated long CoT samples
  - Advantages: Readability & Potential
- Stage 2: RL for Reasoning
  - Rewards: accuracy + language consistency
- Stage 3: Rejection Sampling + SFT
  - Generate reasoning data via RL checkpoints
  - Filter & retain only correct outputs
  - Combine with curated non-reasoning tasks (QA, writing, translation)
  - Scale: ~800k total samples → 2 epochs fine-tuning
- Stage 4: RL for All Scenarios
  - Extend beyond reasoning to general helpfulness & harmlessness
  - Refines overall alignment + reasoning capability

DeepSeek-R1: Evaluation

- Reasoning: near o1-1217
- Math: AIME-24 79.8%, MATH-500 97.3%
  - Stronger than DeepSeek-V3, close to o1-1217
- Coding: Codeforces ~2029 rating
  - Competitive with o1-mini, above open baselines
- Knowledge: MMLU 90.8%, GPQA 71.5%
  - Solid results, near o1-1217
- Writing/QA: AlpacaEval2 87.6%, ArenaHard 92.3%
  - Outperforms DeepSeek-V3
- Limitations: prompt-sensitivity, safety trade-offs

	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	**91.8**	90.8
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-	**92.9**
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-	**84.0**
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	**92.2**
	IF-Eval (Prompt Strict)	**86.5**	84.3	86.1	84.8	-	83.3
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	**75.7**	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	**47.0**	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	**82.5**
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-	**87.6**
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-	**92.3**
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	63.4	**65.9**
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	**96.6**	96.3
	Codeforces (Rating)	717	759	1134	1820	**2061**	2029
	SWE Verified (Resolved)	**50.8**	38.8	42.0	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	**61.7**	53.3
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	79.2	**79.8**
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	96.4	**97.3**
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	**78.8**
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-	**92.8**
	C-Eval (EM)	76.7	76.0	86.5	68.9	-	**91.8**
	C-SimpleQA (Correct)	55.4	58.7	**68.0**	40.3	-	63.7

Table 4: Comparison between DeepSeek-R1 and other representative models.

Distillation

- Supervised fine-tuning of smaller models using outputs generated by DeepSeek-R1 (671B)
- Data Source: ~800K samples (reasoning + non-reasoning) distilled from DeepSeek-R1
- Target Models: Qwen (1.5B–32B) and Llama (8B, 70B) models.
- Key idea: small models fine-tune on R1’s reasoning + answers → inherit reasoning at lower cost and higher stability

Distilled Model Evaluation

- 7B (Qwen): AIME 55.5%, surpasses GPT-4o-0513
- 14B (Qwen): AIME 69.7%, better than QwQ-32B Preview
- 32B / 70B: AIME 72.6% / 70.0%, competitive with o1-mini
- Comparison: RL on Qwen-32B reaches only 47.0%, while distillation lifts it to 72.6%, showing that distillation is far more effective than direct RL on smaller models

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633

Table 5: Comparison of DeepSeek-R1 distilled models and other comparable models on reasoning-related benchmarks.

Significance of Distillation

- Efficiency: avoids costly RL
- Effectiveness: distilled > direct RL (Qwen-32B: 72.6% vs 47.0%)
- Transferability: Works across multiple families (Qwen, Llama, Gemma)
- Stability: SFT more reliable than RL
- Limitations: reasoning focus, uncertain generalization

Q&A

ByteDance | Seed

清華大學 智能产业研究院
Institute for AI Industry Research, Tsinghua University

DAPO: An Open-Source LLM Reinforcement Learning System at Scale

[1]ByteDance Seed [2]Institute for AI Industry Research (AIR), Tsinghua University
[3]The University of Hong Kong
[4]SIA-Lab of Tsinghua AIR and ByteDance Seed

Full author list in Contributions

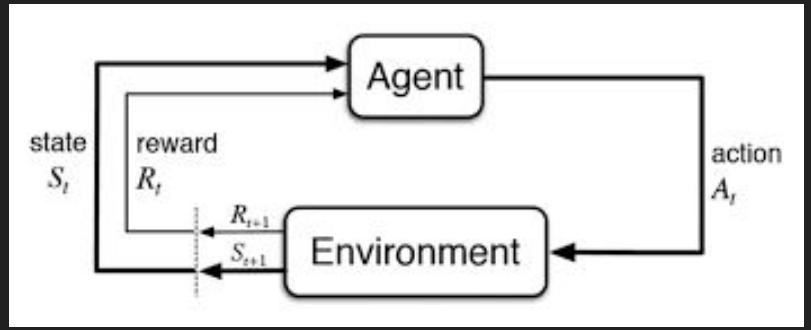
Problem Background & Motivation

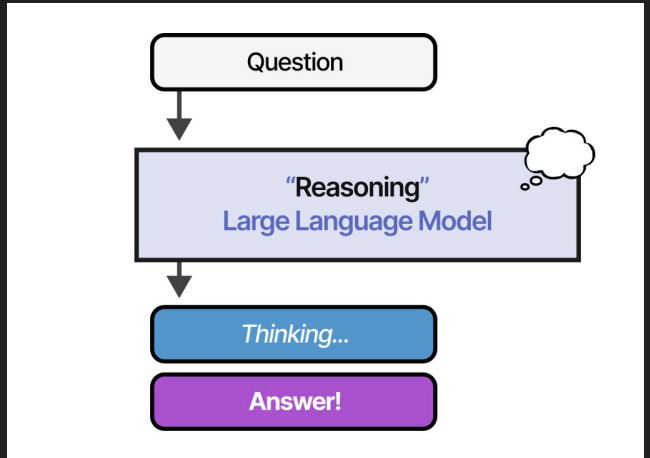
Why Reinforcement Learning for LLM Reasoning?

Test-time scaling (e.g., OpenAI o1, DeepSeek R1) enables longer chain-of-thought (CoT) reasoning.

Core driver: Large-scale Reinforcement Learning (RL).

RL elicits complex behaviors: self-verification, iterative refinement, reflective reasoning.



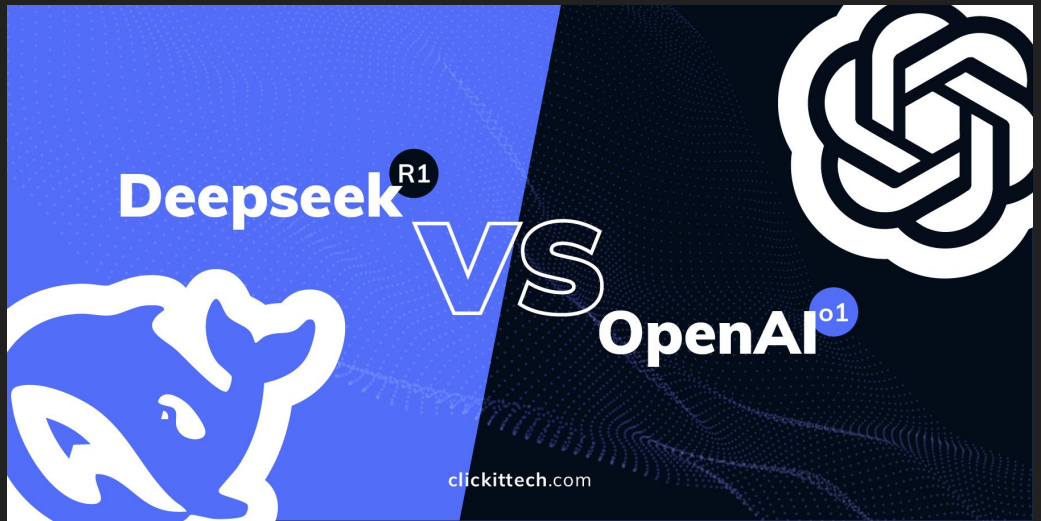


The Reproducibility Challenge

Problem: Key technical details of state-of-the-art RL systems are not disclosed.

OpenAI o1 and DeepSeek R1 reports omit critical training recipes.

Community faces difficulty reproducing results even with similar base models.



Limitations of Naive GRPO/PPO

- Naive GRPO/PPO baselines struggle on long-CoT tasks:
  - Entropy collapse → loss of exploration
  - Reward noise → unstable training
- Zero-gradient cases when all samples are correct/incorrect
  - Training instability overall

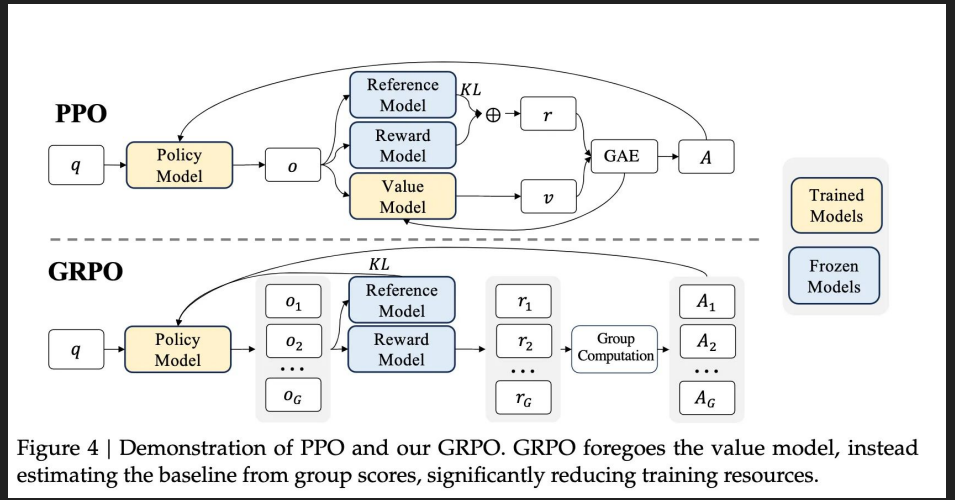
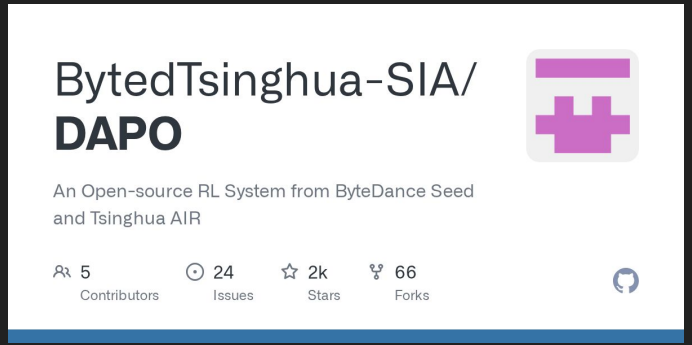


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

The Motivation for DAPO

- Need a reproducible, open-source RL recipe for reasoning LLMs.
- Must directly address four system-level issues:
  - Entropy collapse
  - Reward noise
  - Zero-gradient prompts
  - Instability with long CoT
- Goal: Develop an open, scalable RL algorithm that solves these problems.



Preliminary: PPO & GRPO

PPO: A Classic RL Algorithm

- Uses clipped surrogate objective to stabilize training.
- Constraints updates with ε clipping range.
- Works well in general RLHF, but struggles in long-CoT reasoning.
- Key limitation: leads to entropy collapse in reasoning tasks

PPO [21] introduces a clipped surrogate objective for policy optimization. By constraining the policy updates within a proximal region of the previous policy using clip, PPO stabilizes training and improves sample efficiency. Specifically, PPO updates the policy by maximizing the following objective:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},o_{\leq t}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\min\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}\hat{A}_t,\ \text{clip}\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right], \quad (1)$$

where (q, a) is a question-answer pair from the data distribution $\mathcal{D}$, ε is the clipping range of importance sampling ratio, and $\hat{A}_t$ is an estimator of the advantage at time step t. Given the value function V and the reward function R, $\hat{A}_t$ is computed using the Generalized Advantage Estimation (GAE) [22]:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}, \quad (2)$$

where

$$\delta_l = R_l + \gamma V(s_{l+1}) - V(s_l), \quad 0 \leq \gamma, \lambda \leq 1. \quad (3)$$

GRPO: Group Relative Policy Optimization

- Removes value function, computes advantages relative to group rewards.
- For each prompt, sample G outputs and normalize rewards.
- Simpler, effective in some settings.
- Limitation: still faces zero-gradient issues and training instability

Compared to PPO, GRPO eliminates the value function and estimates the advantage in a group-relative manner. For a specific question-answer pair (q, a), the behavior policy $\pi_{\theta_{\text{old}}}$ samples a group of G individual responses $\{o_i\}_{i=1}^{G}$. Then, the advantage of the i-th response is calculated by normalizing the group-level rewards $\{R_i\}_{i=1}^{G}$:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}. \tag{4}$$

Similar to PPO, GRPO adopts a clipped objective, together with a directly imposed KL penalty term:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \right) \right], \tag{5}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}. \tag{6}$$

Two More Design Choices

- Removing KL Divergence:
  - In reasoning, model distribution can diverge significantly from base model.
  - KL penalty is unnecessary, so DAPO drops it.
- Rule-based Reward Modeling:
  - Instead of learned reward models (prone to reward hacking), use task accuracy as reward.
  - For math: reward = +1 if correct, –1 if wrong

The use of reward model usually suffers from the reward hacking problem [24–29]. Instead, we directly use the final accuracy of a verifiable task as the outcome reward, computed using the following rule:

$$R(\hat{y}, y) = \begin{cases} 1, & \texttt{is\_equivalent}(\hat{y}, y) \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

DAPO Algorithm Overview

From GRPO to DAPO

- GRPO (Group Relative Policy Optimization):
  - Samples G outputs per prompt.
  - Normalizes rewards within the group to compute advantage.
- Problem: Still suffers from entropy collapse, zero gradients, and instability in long-CoT RL.

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right) \right] \tag{8}$$

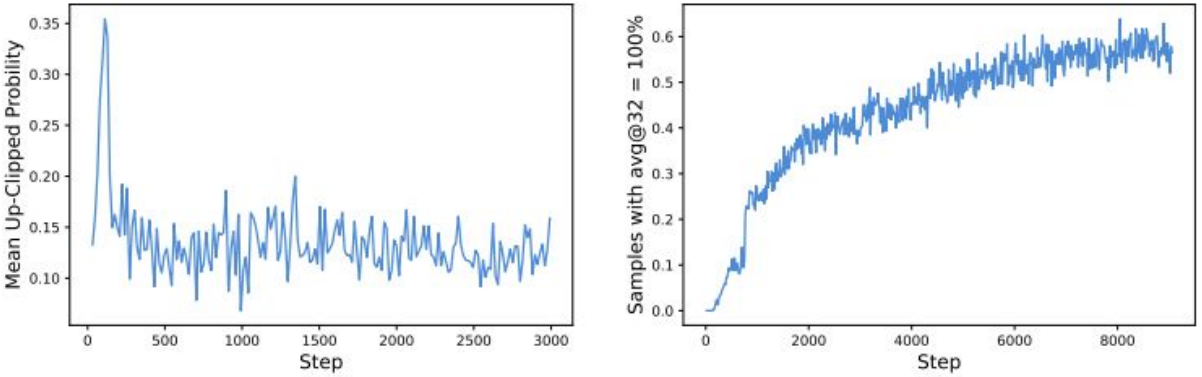
$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G,$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, \quad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}. \tag{9}$$

Core Idea of DAPO

- Decoupled Clip: separate lower and higher clipping bounds (ε_low, ε_high).
- Dynamic Sampling: filter out zero-gradient prompts (all correct or all wrong).
- Token-Level Loss: assign gradient at token level, not just sequence level.
- Overlong Reward Shaping: handle truncated outputs with filtering and soft penalties



(a) Mean up-clipped probability. **(b)** The proportion of samples with an accuracy of 1.

Figure 3 The mean up-clipped probability as well as the ratio of prompts with accuracy=1.

Algorithm Workflow

- Sample batch of prompts.
- For each prompt, sample G responses with old policy.
  Compute rewards using rule-based correctness.
- Dynamic Sampling: keep only non-trivial samples (not all correct/incorrect).
- Compute token-level advantages.
- Update policy with decoupled clipping + reward shaping.

Algorithm 1 **DAPO**: **D**ecoupled Clip and **D**ynamic s**A**mpling **P**olicy **O**ptimization

Input initial policy model π_θ; reawrd model R; task prompts $\mathcal{D}$; hyperparameters $\varepsilon_{\text{low}}, \varepsilon_{\text{high}}$

for step = 1,...,M **do**
  Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
  Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
  Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|q)$ for each question $q \in \mathcal{D}_b$
  Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running R
  Filter out o_i and add the remaining to the dynamic sampling buffer (**Dynamic Sampling** Equation (11))
  if buffer size $n_b < N$:
    continue
  For each o_i in the buffer, compute $\hat{A}_{i,t}$ for the t-th token of o_i (Equation (9))
  for iteration = 1, ..., μ **do**
    Update the policy model π_θ by maximizing the DAPO objective (Equation (8))

Output π_θ

One-Sentence Summary

- DAPO = GRPO backbone + engineered fixes for long-CoT RL.
  - Exploration preserved (Clip-Higher).
  - Efficiency improved (Dynamic Sampling).
  - Gradients assigned fairly (Token-Level Loss).
    Stability ensured (Overlong Reward Shaping).

Four Key Techniques in DAPO

Technique 1: Clip-Higher (Prevent Entropy Collapse)

- Problem: In PPO/GRPO, clipping ratio limits low-probability “exploration tokens” more severely than high-probability tokens.
- Solution: Decouple clipping bounds:
  - Keep ε_low (stability).
  - Raise ε_high (allow exploration tokens to increase).
- Effect: Higher entropy, more diverse samples.

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right)\right] \quad (10)$$

$$\text{s.t.} \quad 0 < \left|\{o_i \mid \texttt{is\_equivalent}(a, o_i)\}\right| < G.$$

Technique 2: Dynamic Sampling (Avoid Zero Gradients)

- Problem: If all G responses for a prompt are correct or incorrect → advantage = 0, no learning signal.
- Solution: Oversample & filter to keep only prompts with mixed outcomes.
- Effect: Every batch contains useful gradients.

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right) \right] \quad (11)$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G.$$

Technique 3: Token-Level Policy Gradient Loss

- Problem: GRPO averages loss per sequence, so long responses get diluted signal; encourages gibberish and repetition.
- Solution: Compute loss at token level → each token's impact counted directly.
- Effect:
- Long reasoning steps reinforced properly.
  Redundant text penalized effectively.

We introduce a **Token-level Policy Gradient Loss** in the long-CoT RL scenario to address the above limitations:

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right) \right], \tag{12}$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G.$$

Technique 4: Overlong Reward Shaping

- Problem: Truncated outputs at max length get harsh penalties → introduces reward noise.
- Solution:
  - Overlong Filtering: mask losses for truncated samples.
  - Soft Punishment: apply gradual, length-aware penalty.
  - Effect: Stable training, better AIME accuracy.

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \le L_{\max} - L_{\text{cache}} \\ \frac{(L_{\max} - L_{\text{cache}}) - |y|}{L_{\text{cache}}}, & L_{\max} - L_{\text{cache}} < |y| \le L_{\max} \\ -1, & L_{\max} < |y| \end{cases}$$

Experiments & Ablations

Experimental Setup

- Task: Mathematical reasoning (AIME 2024 benchmark).
- Base model: Qwen2.5-32B.
- Framework: verl (open-source RLHF/RL framework).
- Evaluation: avg@32 (repeated sampling ×32 for stability).
- Training highlights:
- Learning rate = 1e-6, AdamW optimizer.
- Prompt batch size = 512, 16 responses per prompt.
- Max generation length = 20,480 tokens (with 4,096-token soft cache).

Main Results

- DAPO reaches 50 points on AIME 2024, outperforming DeepSeek-R1-Zero (47).
- Achieves this with 50% fewer training steps.
- Starting from Naive GRPO (30 points):
  - Overlong Filtering → 36
  - Clip-Higher → 38
  - Soft Overlong Punishment → 41
  - Token-Level Loss → 42
  - Dynamic Sampling (DAPO) → 50

Table 1 Main results of progressive techniques applied to **DAPO**

Model	$\text{AIME24}_{\text{avg@32}}$
DeepSeek-R1-Zero-Qwen-32B	47
Naive GRPO	30
+ Overlong Filtering	36
+ Clip-Higher	38
+ Soft Overlong Punishment	41
+ Token-level Loss	42
+ Dynamic Sampling (**DAPO**)	**50**

Training Dynamics

- Metrics monitored:
  - Response length (exploration space).
  - Reward score (training stability)
  - Generation entropy (exploration balance)
  - Mean probability (distribution sharpness)
- Observation: DAPO maintains stable upward trends across metrics

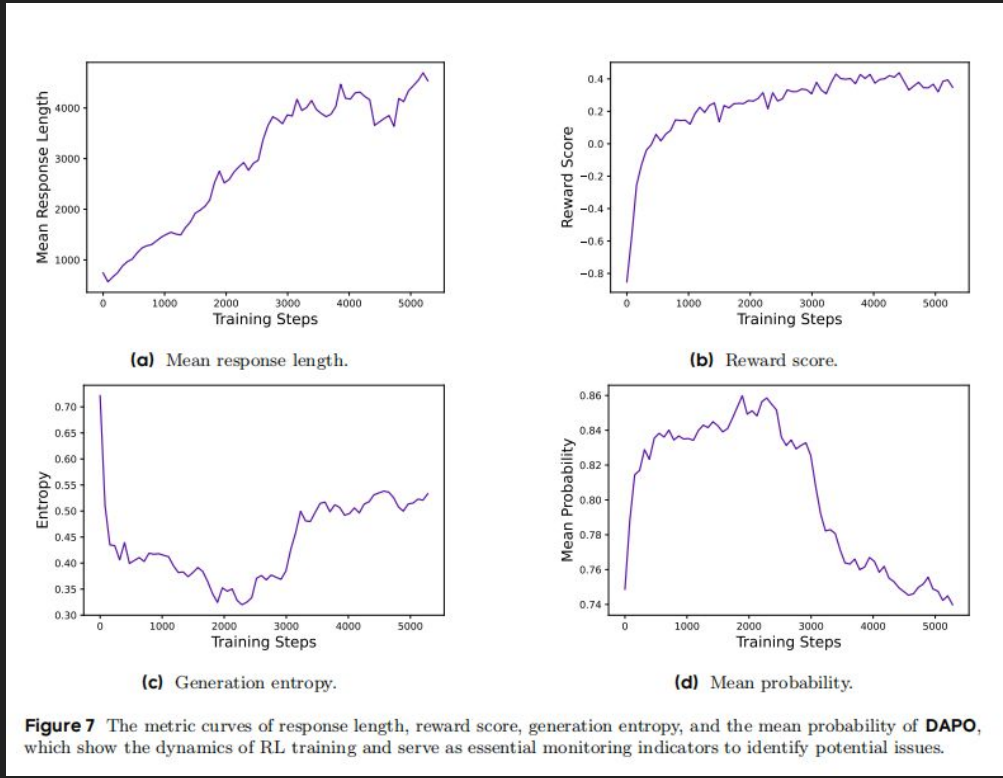


Figure 7 The metric curves of response length, reward score, generation entropy, and the mean probability of **DAPO**, which show the dynamics of RL training and serve as essential monitoring indicators to identify potential issues.

Dataset & Reproducibility

Dataset Construction

- Source: Web scraping + official math competition problems.
- Challenge: Math answers appear in diverse formats (fractions, radicals, formulas).
- Solution: Answer transformation → convert outputs into integers for consistent rule-based evaluation.
- Example:
- Original: 11−2611 - 2\sqrt{6}11−26
- Transformed: ask for k+m+nk+m+nk+m+n → final answer = 19

DAPO-Math-17K Dataset

- Contains 17K prompts, each with an integer-form answer.
- Designed to minimize reward hacking and parser errors.
- Provides a clean benchmark for mathematical reasoning RL.
- Released publicly for community use.

Reproducibility & Open-Source Release

- Codebase: released via verl framework DAPO An Open-Source LLM Reinfor…
- Dataset: DAPO-Math-17K available for public use.
- Training details fully disclosed (hyperparameters, evaluation protocol).
- Goal: lower the barrier for large-scale LLM RL research.

Training Dynamics & Emergent Behaviors

Training Dynamics: Key Metrics

- Response length → exploration space for reasoning.
- Reward score → stability of learning process.
- Entropy → balance between exploration vs. exploitation.
- Mean probability → sharpness of policy distribution.
- Observation: DAPO maintains healthy upward trends across metrics

Emergent Reasoning Behaviors

- Early training: no reflection or backtracking in reasoning.
- Later training: model begins to self-check and revise steps.
- Indicates RL can induce new reasoning modes, not present in the base model.
- Table Example: model gradually learns reflection in solving geometry problems

Question:
Given that the base ABC of the tetrahedron $S-ABC$ is an equilateral triangle, the projection H of point A on the face SBC is the orthocenter of $\triangle SBC$, the dihedral angle $H-AB-C$ is 30°, and $SA=2$, find the volume of the tetrahedron. The answer is in the form $\frac{k}{m}$, so give the value of $k+m$.

Response:
Let us consider the given problem: We need to determine ...
...
Next, let S have coordinates (x_0, y_0, z_0). Since $SA = 2$, the distance formula gives us $\sqrt{(x_0)^2+(y_0)^2+(z_0)^2} = 2$, so we can write $x_0^2 + y_0^2 + z_0^2 = 4$. Now, remember that H is the orthogonal projection of
...
However, wait a moment, let's rethink about the dihedral angle involving planes in a more thoughtful geometric way. Consider the plane $\alpha_1 = ABC$, which lies entirely on the xy coordinate plane (as $z=0$). Let the plane $\alpha_2 = SBC$. The point A projected perpendicularly to plane α_2 lands on H. The line $l = AB$ ...
...

Table 2 Emergence of Reflective Behavior in Reinforcement Learning

Conclusion & Takeaways

Key Contributions of DAPO

- Algorithm: Decoupled Clip & Dynamic Sampling Policy Optimization.
- Techniques: Four innovations — Clip-Higher, Dynamic Sampling, Token-Level Loss, Overlong Reward Shaping.
- Performance: Achieves 50 points on AIME 2024 with 50% fewer steps than DeepSeek-R1-Zero.
- Reproducibility: Fully open-sourced code, dataset, and training details

Broader Impact & Outlook

- Reinforcement learning for LLMs is both a research challenge and a systems engineering problem.
- DAPO lowers the barrier for future research: math, code, theorem proving, reasoning agents.
- Opens the door for the community to replicate, benchmark, and extend reasoning-focused RL.
- DAPO = A reproducible recipe for scaling LLM reasoning with reinforcement learning.

Q&A

Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?

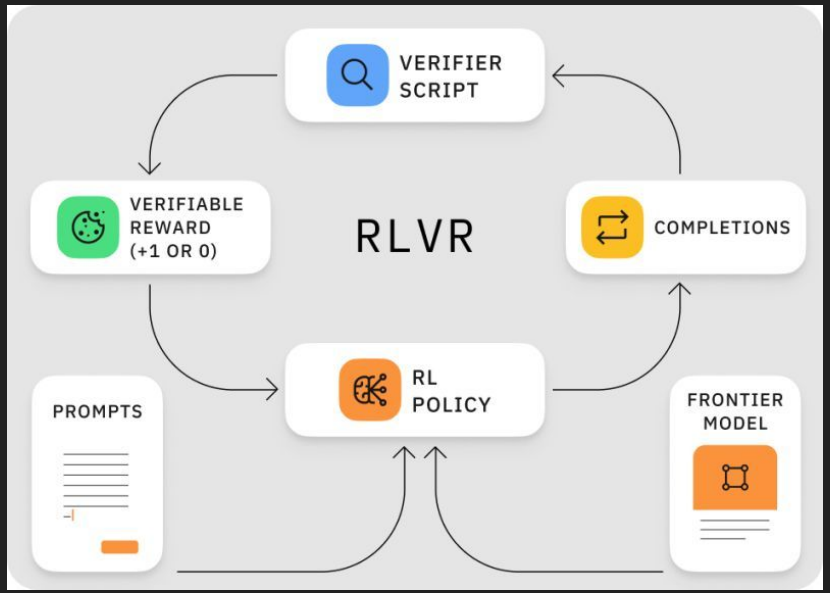
Yang Yue[1*†], **Zhiqi Chen**[1*], **Rui Lu**[1], **Andrew Zhao**[1], **Zhaokai Wang**[2], **Yang Yue**[1], **Shiji Song**[1], **and Gao Huang**[1✉]

[1] LeapLab, Tsinghua University [2] Shanghai Jiao Tong University

* Equal Contribution † Project Lead ✉ Corresponding Author

Introduction & Background

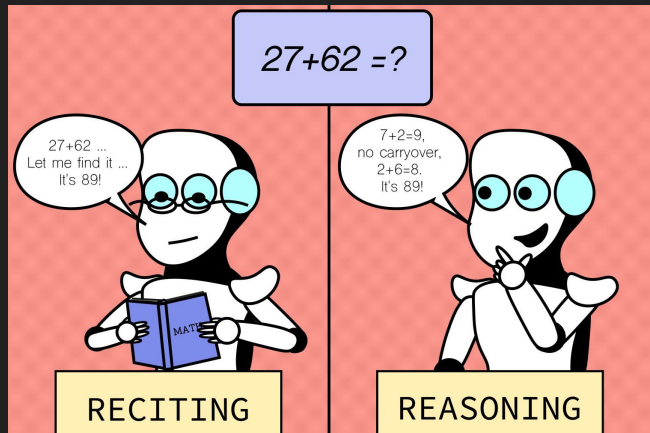
Motivation



- LLMs are increasingly applied to reasoning tasks (math, coding, logical reasoning).
- Traditional instruction-tuning relies on human annotations.
- RL with Verifiable Rewards (RLVR): scalable, automatic, reward-based training.
- Belief in the field: “RLVR enables LLMs to develop novel reasoning patterns, similar to how RL discovered new strategies in AlphaGo.”

Research Question

- Key Question:
  - Does RLVR actually create new reasoning abilities for LLMs?
  - Or… does it only make models sample existing reasoning paths more efficiently?
  - Answering this requires examining the reasoning capacity boundary of both base models and RLVR-trained models.

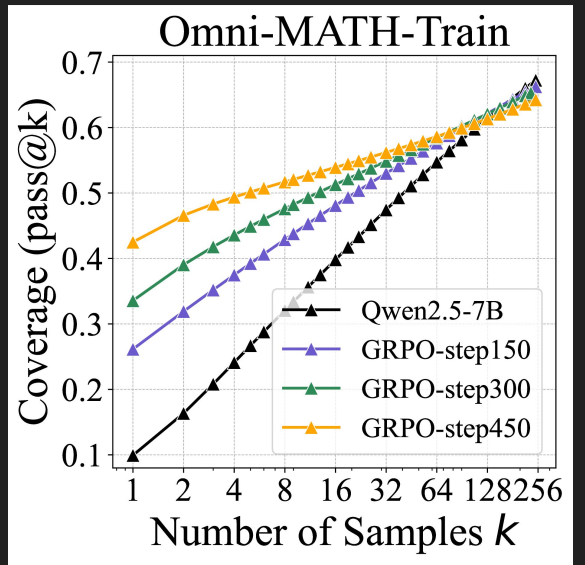


Methodology & Experimental Setup

Evaluation Metric: pass@k

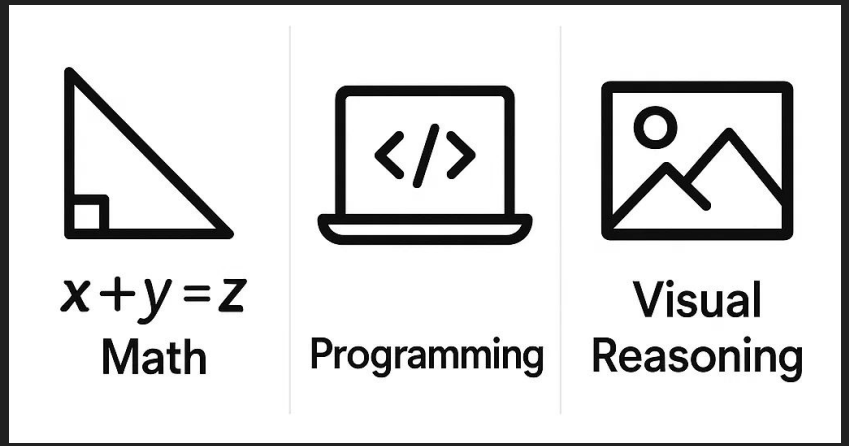
- pass@k: A problem is considered solved if any of k sampled outputs is correct.
- Small k (e.g., k = 1): reflects average-case accuracy.
- Large k (e.g., k = 128, 256): reveals the reasoning capacity boundary.
- More robust than greedy decoding or best-of-N.

$$\text{Pass@}k = \mathbb{E}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)$$



Tasks & Benchmarks

- Mathematics: GSM8K, MATH500, Minerva, Olympiad, AIME24, AMC23
- Code Generation: HumanEval+, MBPP+, LiveCodeBench
- Visual Reasoning: MathVista, MathVision
- Covers multiple domains to rigorously test reasoning abilities.



Models & Algorithms

- Models: Qwen2.5 (7B, 14B, 32B), LLaMA-3.1 (8B)
- RL Algorithms: PPO, GRPO, Reinforce++, RLOO, ReMax, DAPO
- Setup:
  - Base models vs RLVR-trained models
  - Zero-shot prompts for fairness (no few-shot examples)
  - Consistent hyperparameters (temp = 0.6, top-p = 0.95, max length

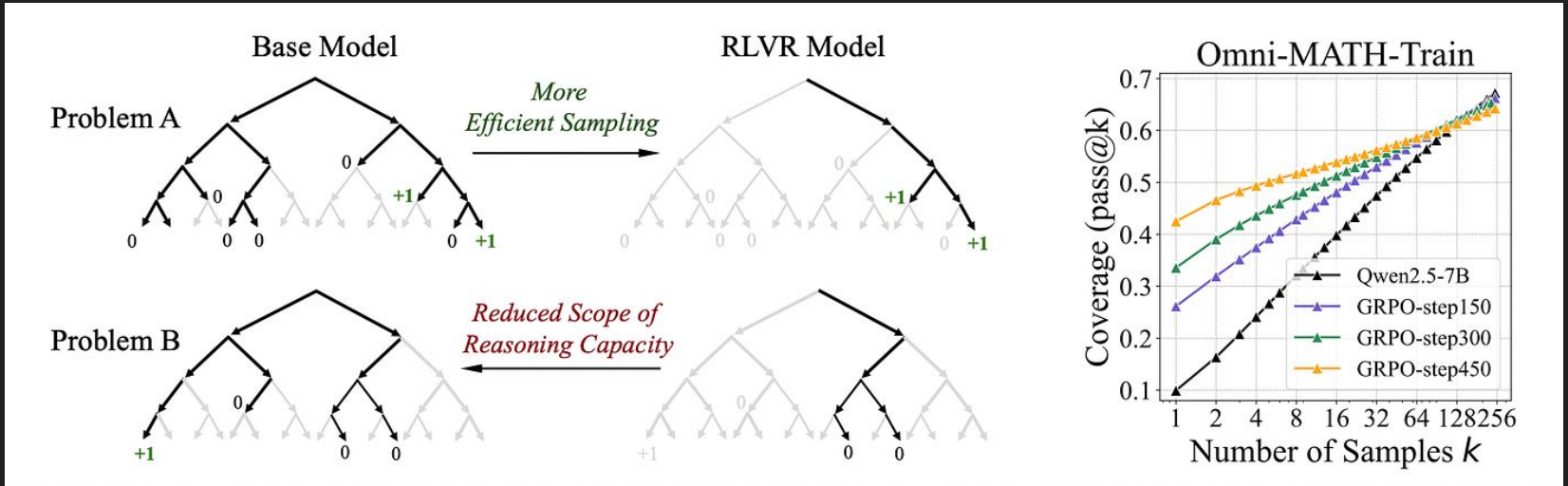
Table 1: Experimental setup for assessing RLVR's effect on the reasoning boundaries of LLMs.

Task	Start Model	RL Framework	RL Algorithm(s)	Benchmark(s)
Mathematics	LLaMA-3.1-8B Qwen2.5-7B/14B/32B-Base Qwen2.5-Math-7B	SimpleRLZoo Oat-Zero DAPO	GRPO	GSM8K, MATH500 Minerva, Olympiad AIME24, AMC23
Code Generation	Qwen2.5-7B-Instruct DeepSeek-R1-Distill-Qwen-14B	Code-R1 DeepCoder	GRPO	LiveCodeBench HumanEval+
Visual Reasoning	Qwen2.5-VL-7B	EasyR1	GRPO	MathVista MathVision
Deep Analysis	Qwen2.5-7B-Base Qwen2.5-7B-Instruct DeepSeek-R1-Distill-Qwen-7B	VeRL	PPO, GRPO Reinforce++ RLOO, ReMax, DAPO	Omni-Math-Rule MATH500

Core Findings

RLVR boosts small-k but narrows reasoning boundary

- At small k (e.g., k=1), RLVR-trained models outperform base models.
- At large k (e.g., k=128/256), base models surpass RLVR-trained models.
- RLVR = more efficient sampling, but with reduced coverage of solvable problems.



Reasoning paths already exist in the base model

- Manual inspection: base models can already generate correct chains-of-thought.
- Perplexity analysis: RLVR outputs lie within base model's distribution.
- RLVR does not create new reasoning paths.

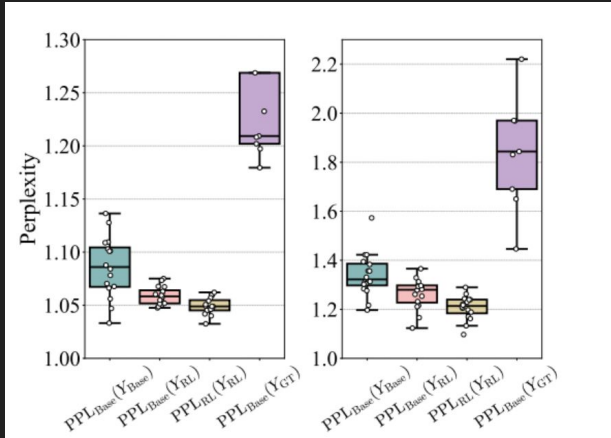


Figure 6: Perplexity distribution of responses. The conditioning problem x is omitted in the figure.

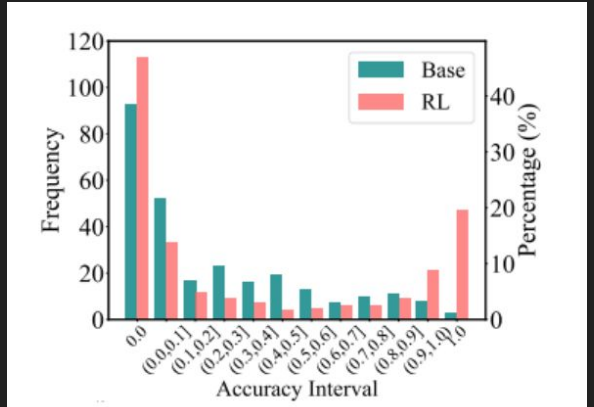


Figure 5: Qwen2.5-7B Accuracy Histogram on Minerva.

Different RL algorithms show similar limitations

- Compared algorithms: PPO, GRPO, Reinforce++, RLOO, ReMax, DAPO.
- Defined metric: Sampling Efficiency Gap (ΔSE) = pass@1 (RL) – pass@256 (base).
- All algorithms → similar performance; ΔSE remains large.
- Conclusion: Current RL methods are far from optimal.

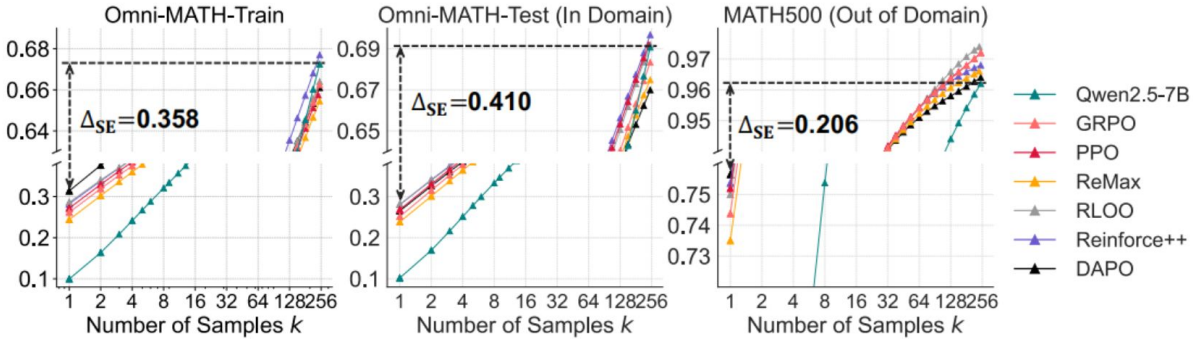


Figure 8: **Different RL algorithms.** We use a *folded y-axis* range to better highlight the details at $k = 1$ and 256. Unfolded version can be found in Figure 14. The detailed values for each point at pass@1 and pass@256 are provided in Table 2 and Table 3.

Distillation genuinely expands reasoning ability

- RLVR: improves efficiency but bounded by base model capacity.
- Distillation: transfers new reasoning patterns from stronger teacher.
- Distilled models show higher pass@k curves across all k.
- Key difference: Distillation expands reasoning scope, RLVR does not.

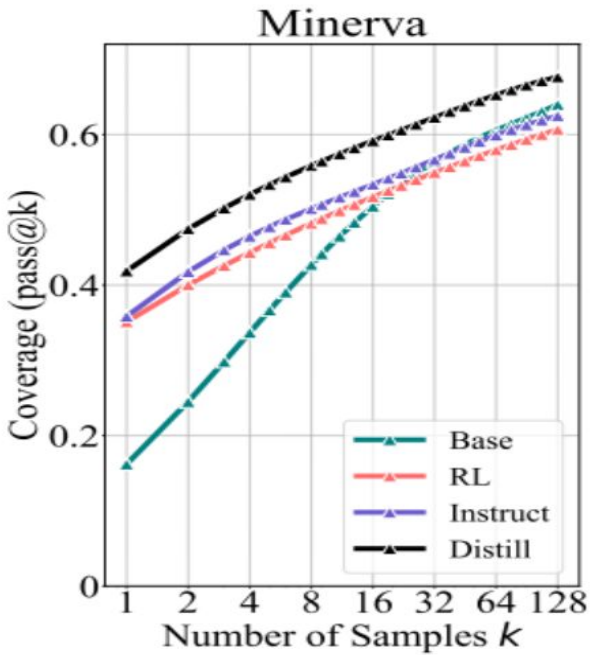
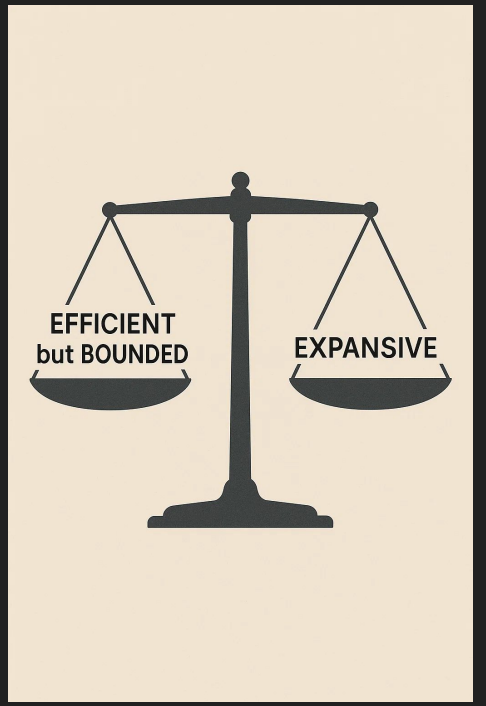


Figure 7: Coverage comparison of base, Instruct, RLVR, and distilled models.

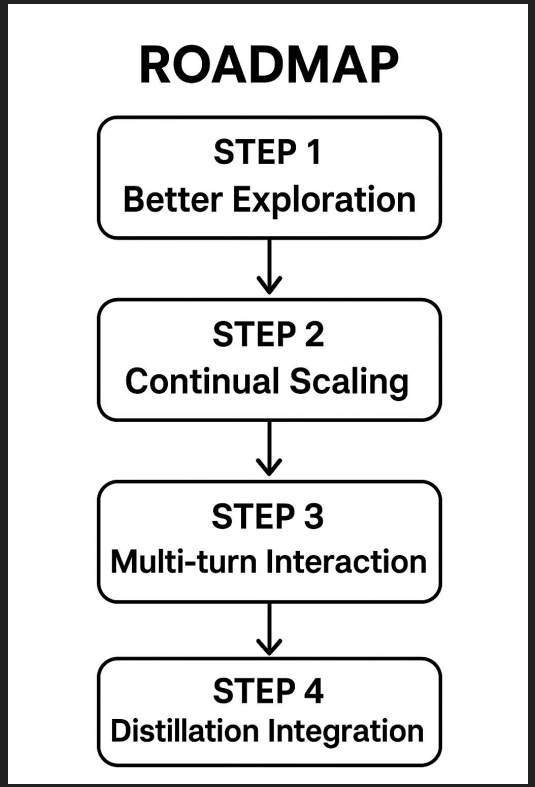
Conclusion & Discussion

RLVR boosts small-k but narrows reasoning boundary

- RLVR improves sampling efficiency at small k.
- But it does not introduce novel reasoning abilities.
- RLVR-trained models remain bounded by their base models.
- Key limitation: reduced reasoning boundary as training progresses.



Reasoning paths already exist in the base model



- Improved exploration strategies in the vast language space.
- Continual scaling of training to avoid capacity collapse.
- Multi-turn agent–environment interactions to simulate real exploration.
- Integration with distillation, which can expand reasoning boundaries

Q&A

SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training

Tianzhe Chu * Yuexiang Zhai * Jihan Yang Shengbang Tong
Saining Xie Dale Schuurmans Quoc V. Le Sergey Levine Yi Ma

Problem Background & Motivation

Background

Post-training is crucial for foundation models:

- **Supervised Fine-Tuning (SFT)** adapts to downstream tasks
- **Reinforcement Learning (RL)** aligns with outcomes/preferences

Their roles in **memorization vs. generalization** remain unclear, especially in rule-based and vision-language reasoning.

Research Questions:

1. Does SFT mainly memorize training data instead of learning transferable rules?
2. Can RL drive genuine generalization across unseen tasks or domains?
3. Is SFT still necessary for RL training to be effective?
4. How does RL enhance the visual recognition capabilities of VLMs?
5. Does scaling inference-time computation (verification steps) further improve generalization?

Related Works:

- Prior studies typically focus only on SFT or RL, or on only LLMs or VLMs; few works directly compare both methods across modalities
- Existing research improved VLMs mainly via SFT data/recipes or encoder design; our work shows that RL also enhances visual perception and generalization

Recap: Supervised Fine-Tuning (SFT)

What is it?

- Fine-tunes a pre-trained model using labeled data (prompt → expected answer)

Role

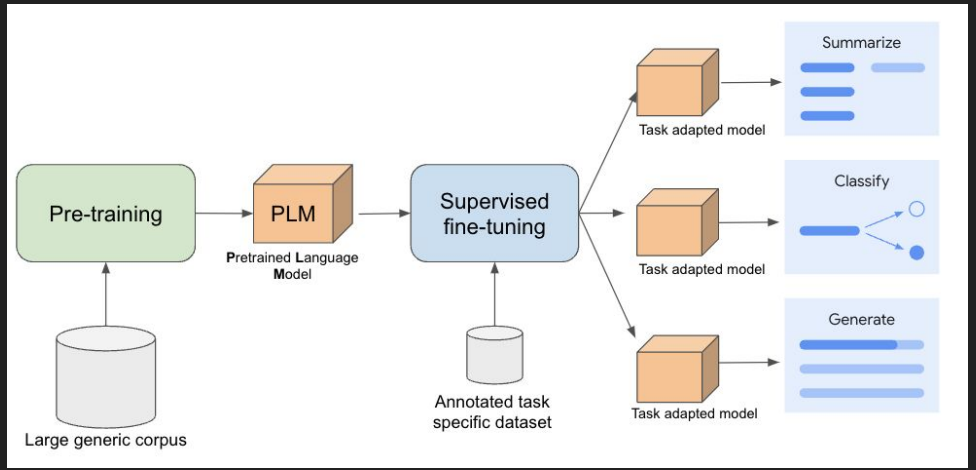
- Teaches the model how to adapt to downstream tasks (translation, summarization, QA, ...)
- Enforces consistent format and style in outputs

Strengths

- Stabilizes output structure
- Easy to implement with curated dataset

Limitations

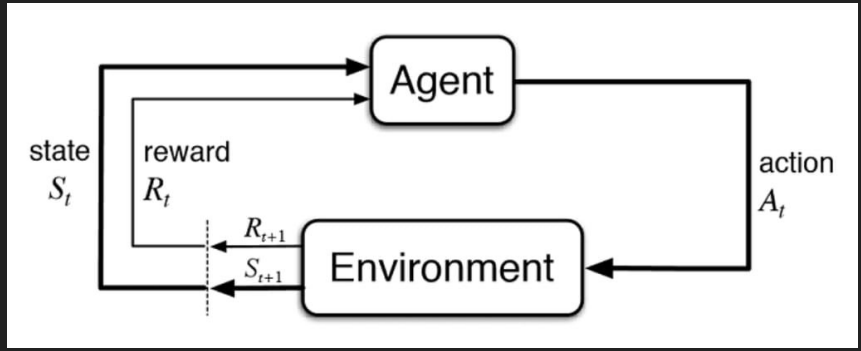
- Prone to memorization of training examples
- Fails to generalize to unseen rules or domains



Recap: Reinforcement Learning (RL)

What is it?

- Trains the model with a reward signal, not fixed labels
- Example: correct answer -> +10 reward, wrong answer -> -1



Role

- Aligns the model with preferences or outcomes
- Encourages learning of transferable rules and strategies

Strengths

- Promotes generalization across unseen tasks/domains
- Can improve underlying capabilities (e.g., visual recognition, reasoning)

Limitations

- Unstable when applied directly (needs SFT as a foundation)
- Requires careful design of reward functions

Preliminaries

Standard RL Setup:

$\pi^* = \arg\max_{\pi \in \Pi} E_\pi \left[\sum_{t=0}^{T} r_t \right]$, where $r : S \times A \to R$

S and A are state space and action space respectively

T is horizon (max step per episode)

$\pi(a|s)$ is probability policy that chooses action a in state s

RL for LLMs / VLMs with Verifier:

Vocabulary space: V

Input Token: V^m, Output Token: V^n

State:

- LLM: $S = V^m$
- VLM: $S = V^m \times O$, with $O =$ image space

Action: $A = V^n$

Verifier: $\text{VER}(v_{out,t}) \mapsto (r_t, v_{\text{ver},t})$

Ex: verified v would be "illegal number used"

Sequential Revision (multi-turn RL):

At step t:

$v_t^{\text{in}} = v_0^{\text{in}} \parallel \left[v_k^{\text{out}}, v_k^{\text{ver}} \right]_{k=0}^{t-1}$, where v_0^{in} is system prompt

Each new input = prompt + all previous outputs & verifier feedback

General Points (Points24 Game)

Goal: Test **rule generalization** in arithmetic reasoning

Set up:

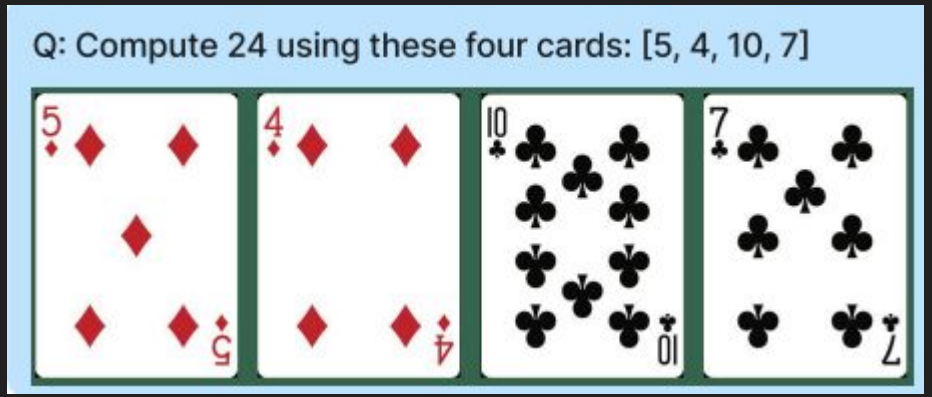
- Input: 4 cards (Text-only **GP-L** or Vision-Language **GP-VL**)
- Task: use all 4 cards exactly once to form an equation = 24
- Output: Valid equation string

Variants (for OOD tests):

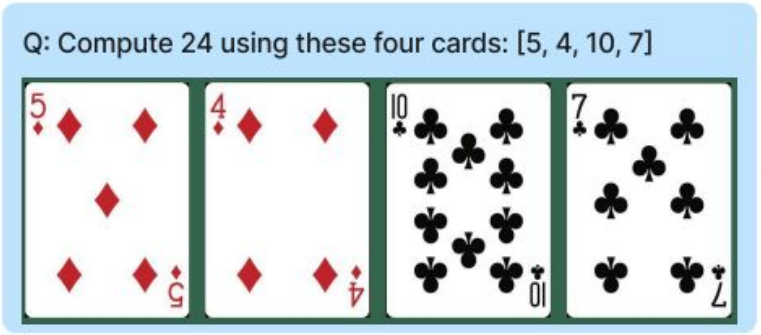
- Rule Variations (Text OOD):
  - Train: J/Q/K = 10
  - Test: J=11, Q=12, K=13

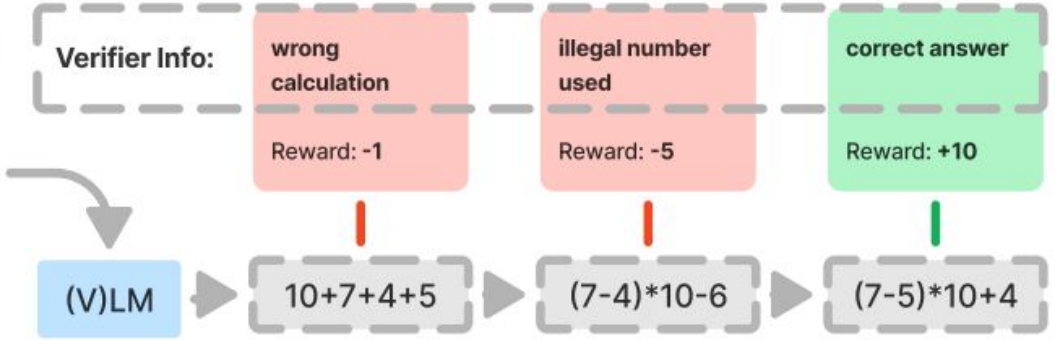
Visual Variations (Vision OOD):

- Train: Black suits (♠, ♣)
- Test: Red suits (♥, ♦)



General Points (Points24 Game)





System Prompt (v_0^{in})

[Task Description] You are an expert in {task name}, you are observing {purely language/vision-language inputs + <image>}. You are currently at {state related info}. Please follow {tasks rules}.

[Output] Your response should be a valid json file in the following format:
{task related information and answer}

Appending previous model and verifier outputs to obtain v_t^{in}

$$\boldsymbol{v}_t^{\text{in}} = [\boldsymbol{v}_0^{\text{out}}, \boldsymbol{v}_0^{\text{ver}}, \boldsymbol{v}_1^{\text{out}}, \boldsymbol{v}_1^{\text{ver}}, \ldots, \boldsymbol{v}_{t-1}^{\text{out}}, \boldsymbol{v}_{t-1}^{\text{ver}}] \qquad \triangleright\ \boldsymbol{v}_t^{\text{in}} = \text{concat}\left(\boldsymbol{v}_0^{\text{in}}, [\boldsymbol{v}_k^{\text{out}}, \boldsymbol{v}_k^{\text{ver}}]_{k=0}^{t-1}\right)$$

Model output (v_t^{out}) and Verifier Output (v_t^{ver})

{Task related json outputs}, {You success/fail}. $\qquad \triangleright\ \boldsymbol{v}_{t+1}^{\text{in}} = \text{concat}(\boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}}, \boldsymbol{v}_t^{\text{ver}})$

V-IRL (Visual Navigation Task)

Defn: A real-world navigation task that focuses on the model's spatial reasoning capabilities

Goal: Test **spatial reasoning** and **visual generalization**

Set up:

- Environment: Real-world street-view images + text navigation instructions
- Task: Navigate step by step to the target location

Input:

- Text-only instructions (V-IRL-L)
- Street-view images + text (V-IRL-VL)

Variants (for OOD tests):

- Rule Variations (Action Space OOD):
  - Train: Absolute directions (west, east, northwest)
  - Test: Relative directions (left, right, slightly left)
- Visual Variations (Domain OOD)
  - Train: Routes in New York City
  - Test: Routes in other cities (London, Milan, …)

Results

GP = General Point

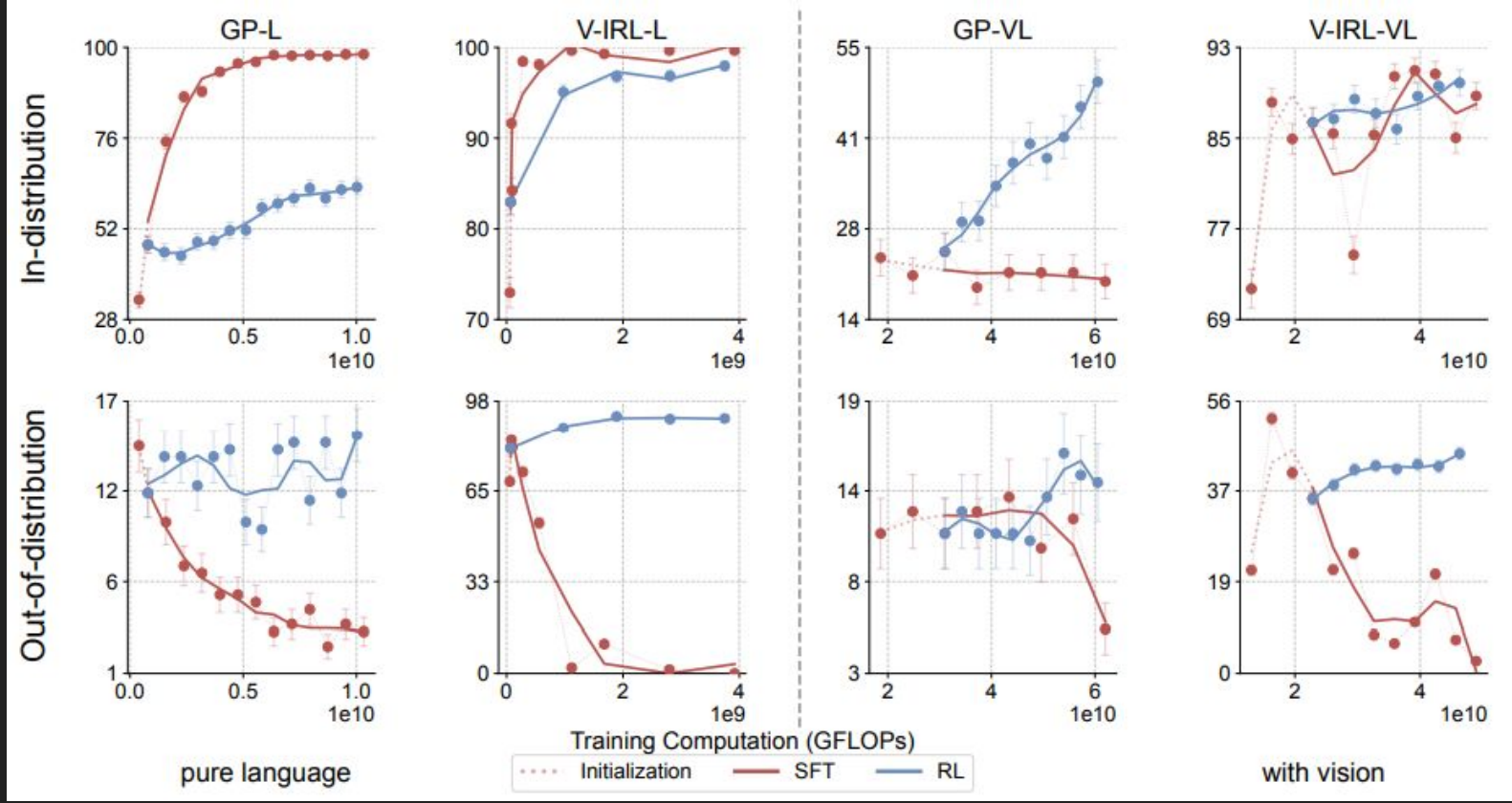
-L = Language (text)

V-IRL = Navigation Task

-VL = visual + language tasks

- - - = Initial (Llama-3.2-Vision-11B)

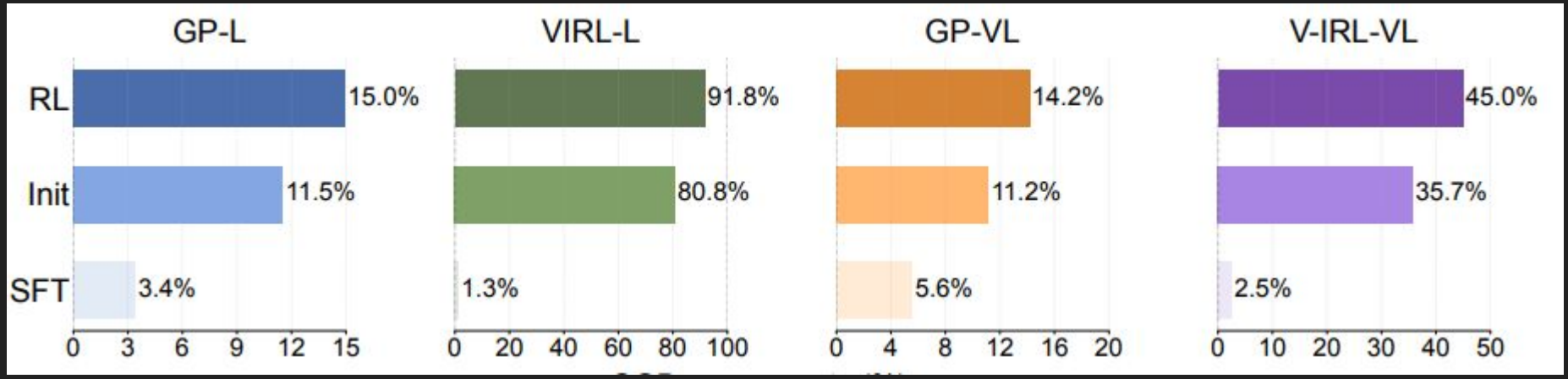
It's Pretrained Model + SFT One epoch



Language-Only (Left): In-sample accuracy -> RL < SFT ; Out-sample accuracy -> RL > SFT

Vision-Language (Right): In-sample accuracy -> SFT <= RL (RL higher robustness) ; Out-sample accuracy -> RL > SFT

Comparison of out-of-distribution performance under rule variants



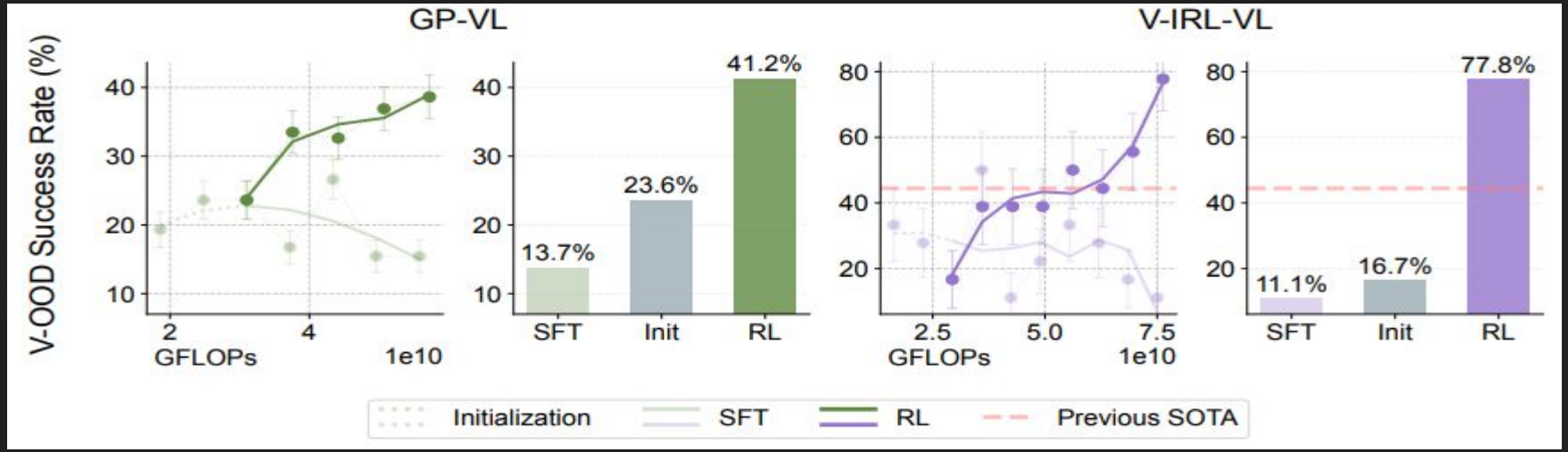
RL Performance:

- GP-L: 11.5% -> 15%, VIRL-L: 80.8% -> 91.8%, GP-VL: 11.2% -> 14.2%, V-IRL-VL: 35.7% -> 45%
- **All Outperform Initial Baseline!**

SFT Performance:

- GP-L: 11.5% -> 3.4%, VIRL-L: 80.8% -> 1.3%, GP-VL: 11.2% -> 5.6%, V-IRL-VL: 35.7% -> 2.5%
- **All Underperform Initial Baseline! Memorizes only, fails on OOD**

Comparison of out-of-distribution performance for visual variants



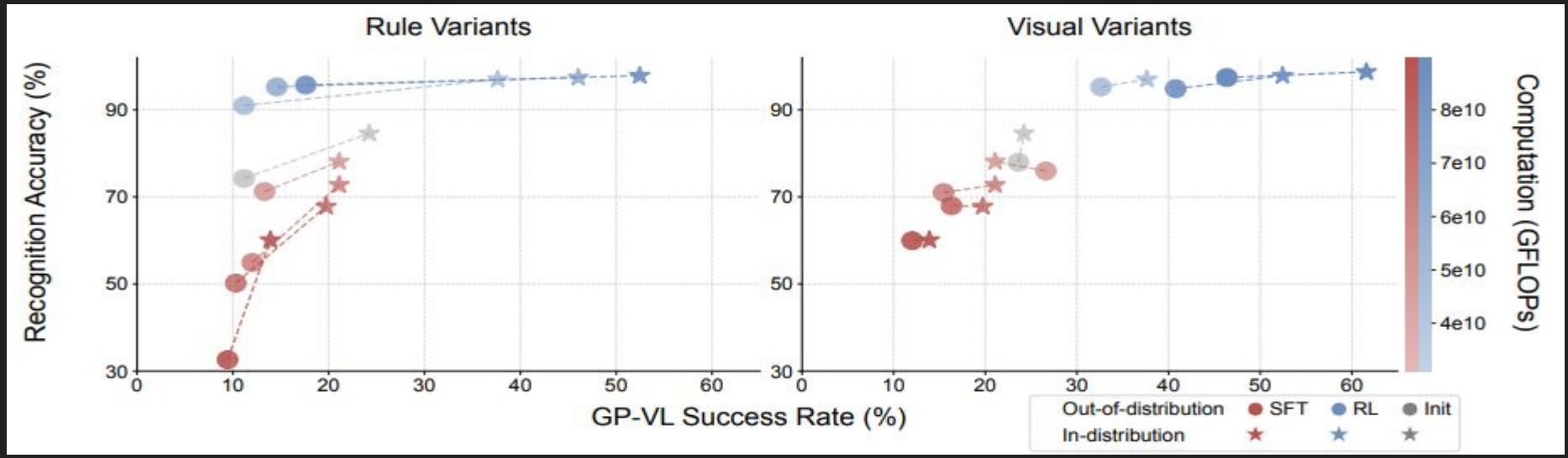
RL Performance:

- GP-VL: 23.6% -> 41.2%, V-IRL-VL: 16.7% -> 77.8%
- **All Outperform Initial Baseline!**

SFT Performance:

- GP-VL: 23.6% -> 13.7%, V-IRL-VL: 16.7% -> 11.1%
- **All Underperform Initial Baseline! Shows that it memorizes only again, fails on OOD**

Recognition vs. Success Rate: RL Improves, SFT Degrades (GP-VL)



- RL improves both recognition & success rate
  - The blue curve shows that as training compute increases, RL improves both success rate and recognition accuracy.
- SFT shows opposite effect
  - The red curve shows that as training compute increases, it decreases both recognition accuracy and recognition accuracy

RL experiments on GP-L without SFT initialization

SFT is still necessary !

Key result:

All RL runs fail when trained directly from the base model without SFT

Why?: The base Llama-3.2 model has poor **instruction-following ability**. Without SFT, it tends to produce long, irrelevant, and unstructured outputs, which prevents meaningful reward signals for RL

Different Learning Rate: all curves quickly drop toward zero success rate as computation increases

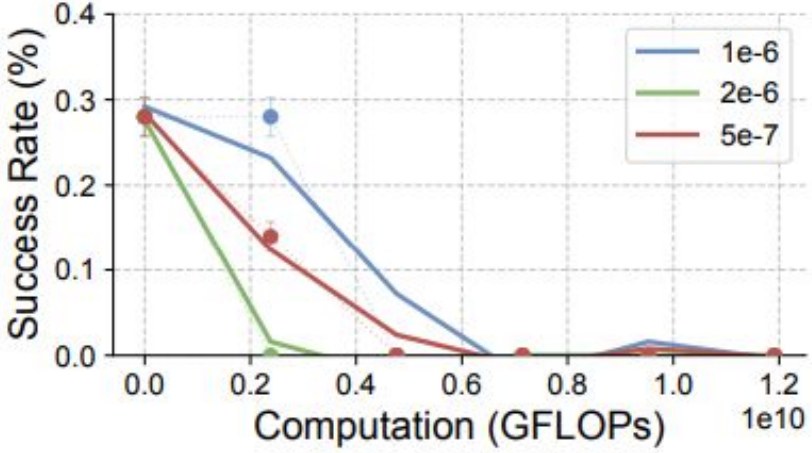


Figure 9: **RL experiments on `GP-L` without SFT initialization.** All trials fail due to poor instruction following capability of the base model.

SFT is necessary as a warm-up stage to make RL training feasible

More Verification Iterations → Better Generalization

What is VIter?

It represents the maximum number of times the model can use the verifier to revise its output in one episode

- **As computation increases(from transparent to solid), the Both the OOD Growth and IDG increases**
- **As vIter increases, the curve shift from left to right (blue to red) -> OOD Growth increases**

Takeaways:

Higher VIter allows the model to not only remember rules but also adapt to new ones

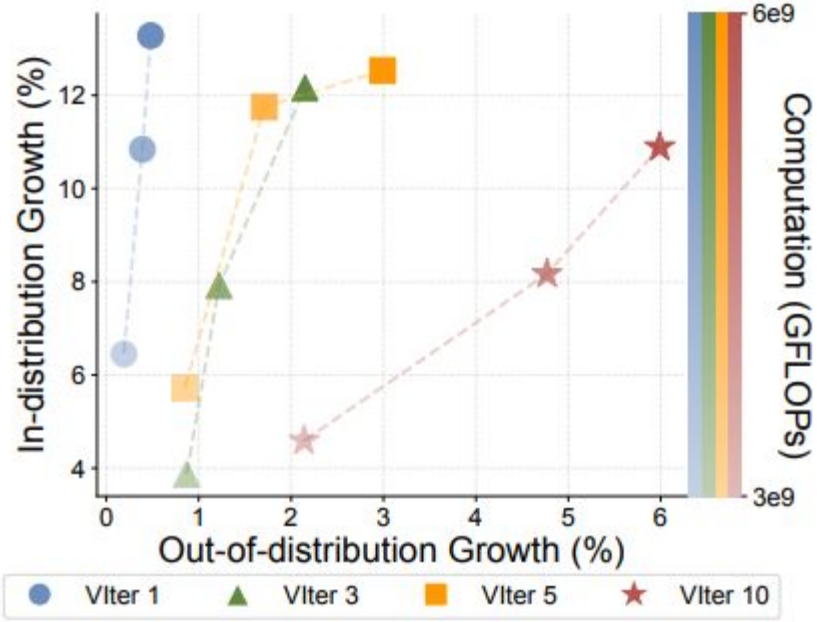


Figure 10: **In-distribution vs. OOD performance growth on GP−L.** We record RL experiments with different number of verification iterations (VIter) as scaling up training compute (color transparency).

Conclusion and key findings

1. SFT Memorizes, RL Generalizes
   a. SFT increases in-distribution performance, but significantly degrades performance on OOD tasks
   b. RL consistently boosts OOD performance across both rule-based and visual variations
2. RL Enhances Visual Recognition
   a. Scaling RL improves both reasoning and recognition accuracy
   b. SFT scaling harms recognition, showing overfitting to reasoning tokens
3. SFT is Necessary for RL Initialization
   a. Without SFT, base models fail due to poor instruction following
   b. SFT provides the structure needed for RL to work effectively
4. Verification Iterations Matter
   a. More verification steps (multi-turn correction) → stronger OOD generalization
   b. Iterative refinement is key to RL's success

Wrap and summary

1. **DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning**
   a. RLVR (R1-Zero) alone already shows emergent reasoning with long, structured “thinking chains”
   b. Multi-stage training (Cold Start (SFT) → RL → SFT → RL) stabilizes format, improves readability, and keeps reasoning power
   c. Large RL-trained R1 model distills its reasoning traces into smaller models, significantly boosting their reasoning benchmarks compared to baseline small models
2. **DAPO: An Open-Source LLM Reinforcement Learning System at Scale**
   a. Naive PPO/GRPO unstable on long chain-of-thought tasks
   b. Contributions: Clip-Higher, Dynamic Sampling, Token-Level Loss, Overlong Reward Shaping
   c. Results: Outperforms DeepSeek-R1-Zero-Qwen-32B on AIME with **50% fewer steps**
3. **Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?**
   a. RLVR boosts **pass@1** (sampling efficiency), but not reasoning capacity
   b. At large k (pass@128/256), base models often surpass RLVR models
   c. Reasoning paths already exist in base models; RLVR just samples them better
   d. Distillation, not RLVR, expands reasoning boundary
4. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training
   a. **SFT** → improves in-distribution, but degrades OOD performance (memorization)
   b. **RL** → consistently improves OOD generalization (rules & visual tasks)
   c. SFT is necessary for RL initialization (warm-up for instruction following)

Thank You for listening !
Questions?