

Rethinking Reasoning Models

Harman Singh, Dennis Jacob

Oct 2nd

Recap of reasoning models

State of the field...

- Last year saw a flurry of models that can “reason”

arXiv > cs > arXiv:2501.09388

Computer Science > Computation and Language

Computer Science

OpenAI o1

arXiv > cs > arXiv:2501.12948

Computer Science > Computation and Language

DeepSeek-R1: Incentivizing Reas

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhar Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, B Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Z Yuan, Junjie Qiu, Junlong Li, J.L. Cai, Jiaqi Ni, Jian Li Mingchuan Zhang, Minghua Zhang, Minghui Tang, R.J. Chen, R.L. Jin, Ruyi Chen, Shanghao Lu, Shang

We introduce our first-generation reasoning models, De remarkable reasoning capabilities. Through RL, DeepSee issues and further enhance reasoning performance, we introduce DeepSee support the research community, we open-source DeepSeek-R1-Zero, De

Subjects: Computation and Language (cs.CL)
Cite as: arXiv:2501.09388 [cs.CL]
(or arXiv:2501.09388v1 [cs.CL] for this version)
<https://doi.org/10.48550/arXiv.2501.09388>

Yichang Zhang, Yu Wan, Yaqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu (additional authors not shown)

In this report, we introduce Qwen2.5, a comprehensive series of large language models (LLMs) designed to meet diverse needs. Compared to previous iterations, Qwen 2.5 has been significantly improved during both the pre-training and post-training stages. In terms of pre-training, we have scaled the high-quality pre-training datasets from the previous 7 trillion tokens to 18 trillion tokens. This provides a strong foundation for common sense, expert knowledge, and reasoning capabilities. In terms of post-training, we implement intricate supervised finetuning with over 1 million samples, as well as multistage reinforcement learning. Post-training techniques enhance human preference, and notably improve long text generation, structural data analysis, and instruction following. To handle diverse and varied use cases effectively, we present Qwen2.5 LLM series in rich sizes. Open-weight offerings include base and instruction-tuned models, with quantized versions available. In addition, for hosted solutions, the proprietary models currently include two mixture-of-experts (MoE) variants: Qwen2.5-Turbo and Qwen2.5-Plus, both available from Alibaba Cloud Model Studio. Qwen2.5 has demonstrated top-tier performance on a wide range of benchmarks evaluating language understanding, reasoning, mathematics, coding, human preference alignment, etc. Specifically, the open-weight flagship Qwen2.5-72B-Instruct outperforms a number of open and proprietary models and demonstrates competitive performance to the state-of-the-art open-weight model, Llama-3.405B-Instruct, which is around 5 times larger. Qwen2.5-Turbo and Qwen2.5-Plus offer superior cost-effectiveness while performing competitively against GPT-4o-mini and GPT-4o respectively. Additionally, as the foundation, Qwen2.5 models have been instrumental in training specialized models such as Qwen2.5-Math, Qwen2.5-Coder, QwQ, and multimodal models.

Subjects: Computation and Language (cs.CL)
Cite as: arXiv:2412.15115 [cs.CL]
(or arXiv:2412.15115v4 [cs.CL] for this version)
<https://doi.org/10.48550/arXiv.2412.15115>

Search: All Fields Search

Help Advanced Search

Access Paper:
View PDF
Text Source
Other Formats

Current browse context:
cs.CL
< prev | next >
new | recent | 2025-05
Change to browse by:
cs

References & Citations
NASA ADS
Google Scholar
Semantic Scholar

Export BibTeX Citation

Bookmark per:
text:

References & Citations
NASA ADS
Google Scholar
Semantic Scholar

Export BibTeX Citation

Bookmark



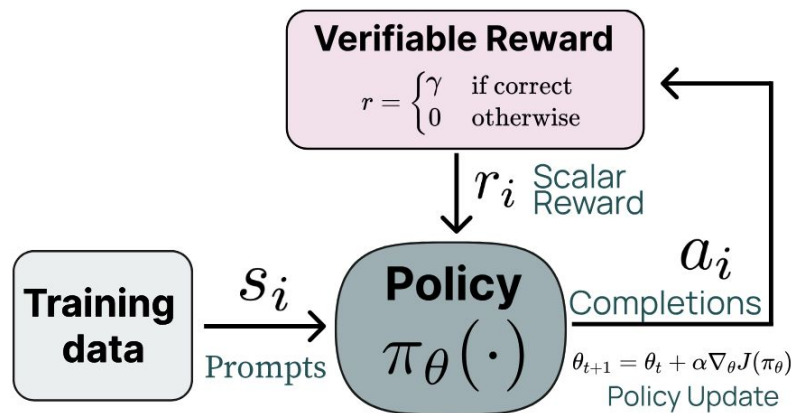
The Qwen model family

- An open source family of models from Alibaba
- Strong performance on math benchmarks that require “symbolic manipulation and multi-step logical deduction”
 - **MATH-500**: verifiable problems that vary in subject and difficulty
 - **AIME & AMC**: math competitions for high schoolers



Context and Primer on RLVR/Reasoning Models

- RLVR = train LLMs with verifiable outcome rewards (e.g., correctness checks for math).
- *are we teaching new reasoning, or just surfacing latent capabilities?*



Lambert et. al. 2024

Spurious rewards

Shao et. al.

Spurious Rewards

TL;DR: Spurious rewards (i.e. other than answer correctness rewards) can produce large MATH gains in Qwen2.5 — but effects are model-dependent.

This Presentation:

1. Main Claims
2. Experiments Performed
3. Ablations (Deep Dive)

Experimental setup

Models: Qwen2.5-Math (1.5B & 7B), Qwen general variants, Llama3, OLMo2.







RL algorithm: GRPO training for ~300 steps

Dataset: DeepScaleR subset. **Eval:** Math500, AMC, AIME

Experimental setup

How are Weak Rewards and Spurious Rewards applied in training

- Data: $\{P_i, G_i\}, i \in \{1, 2, \dots, D\}$
- Rollouts: $R_j, j \in \{1, 2, \dots, N\}$ obtained from a Prompt
- Spurious/Weak Rewards:

Correct answer		Ground Truth	Standard
Most common label from 64 rollouts		Majority Vote	
1 example w/ correct label (Wang et al., 2025)		One-Shot RL	Weak
Response contains <code>\boxed{ }?</code>		Format Reward	
Use an incorrect rollout as ground truth		Incorrect Label	Spurious
Reward = 1 50% of the time		Random Reward	

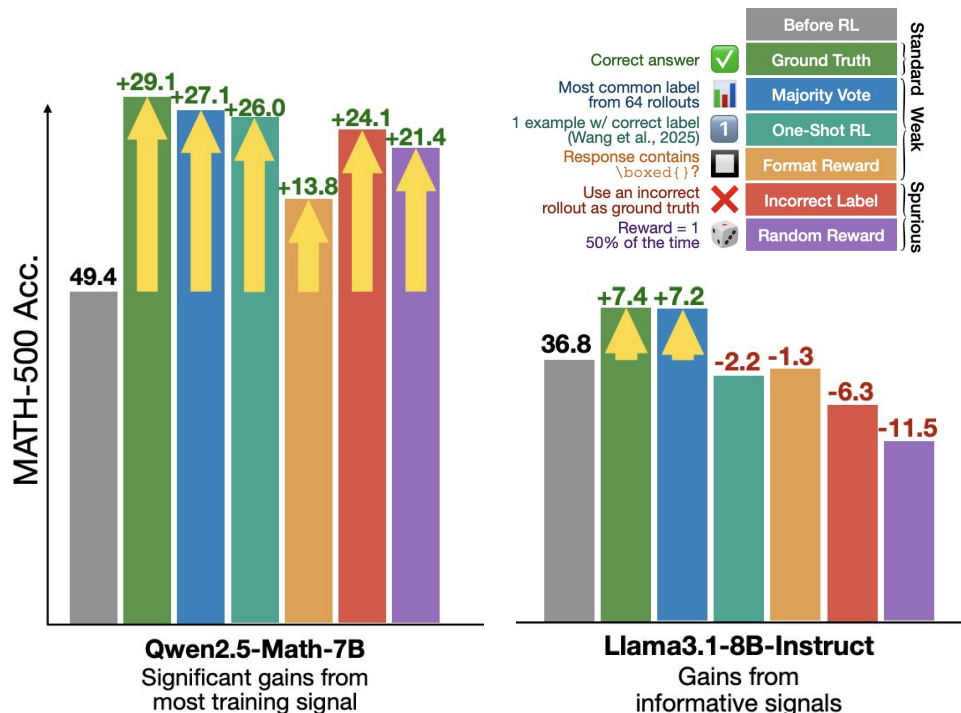
Do not depend on Ground Truth (G_i)

Main Claims

Main claim: Even random/incorrect/format rewards can yield large improvements on MATH-500 for Qwen2.5-Math models (e.g., +21% for random rewards).

Nuance: Effect is model-dependent — does not generalize fully to Llama3 or OLMo2 in their experiments.

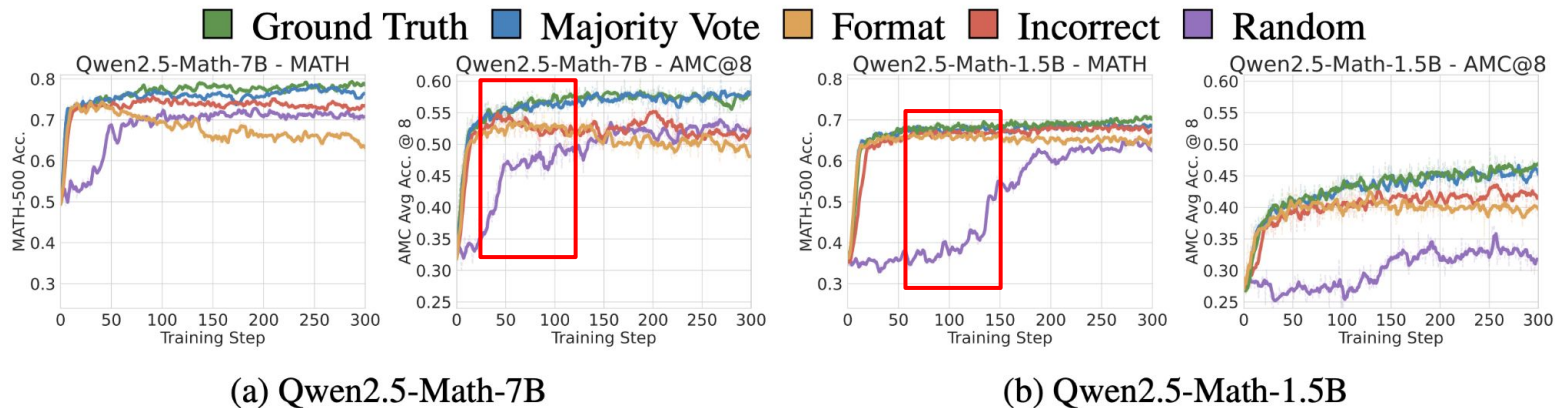
Key results and Learning Curves



- Qwen models show large improvements in performance with spurious rewards
- This effect does not generalize to other models like Llama.

Key results and Learning Curves

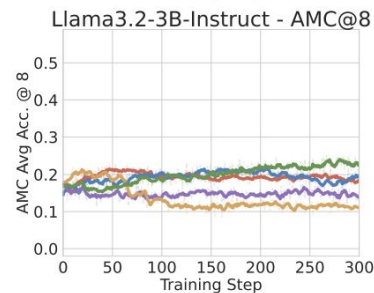
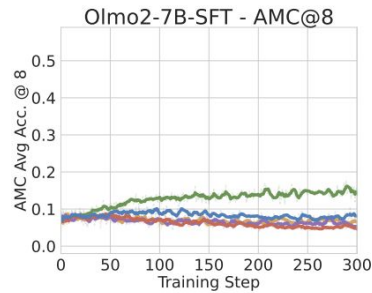
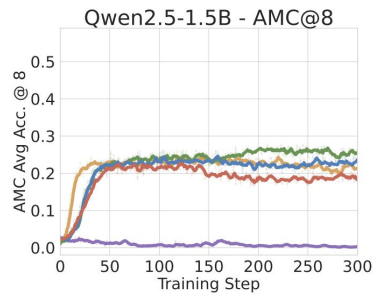
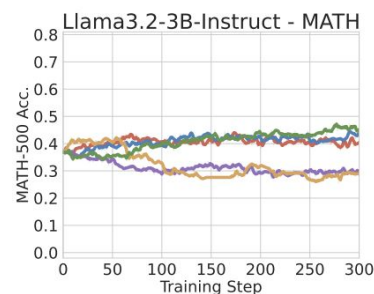
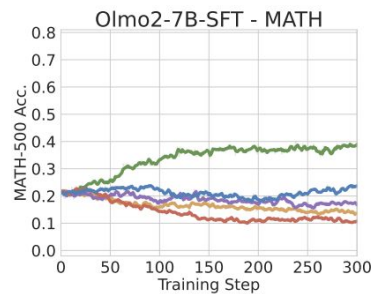
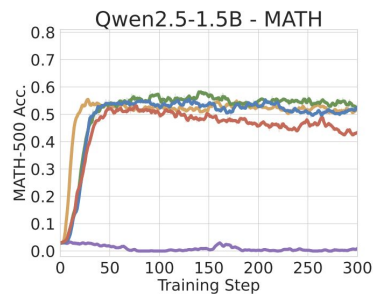
Learning curves:



- Qwen models show fast gains across reward types
- Random reward converges slower but still gives large gains
- Support for hypothesis: RLVR may only elicit reasoning learned in pre- and post-training

Key results and Learning Curves - Generalization to Other models

■ Ground Truth ■ Majority Vote
■ Format ■ Incorrect ■ Random



Qwen2.5-1.5B

Olmo-2-1124-7B-SFT

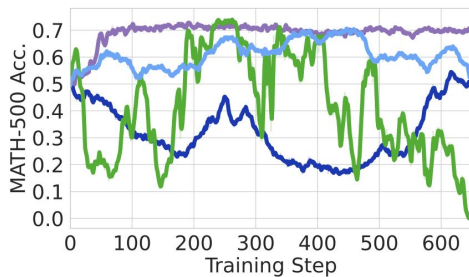
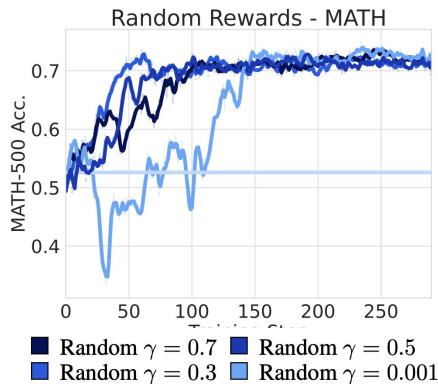
Llama3.2-3B-Instruct

13

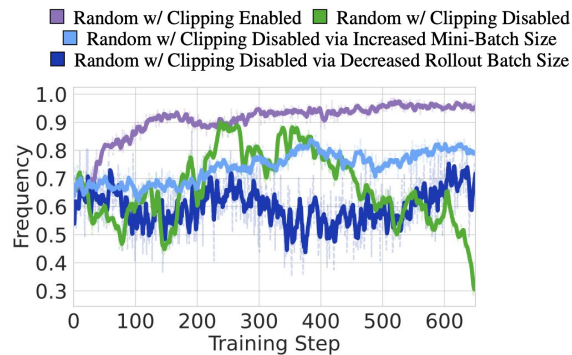
- Non-MATH Qwen models improve with spurious rewards
- Non-Qwen models often do not improve or even degrade.

Ablations Deep Dive - 1. Random reward + clipping

- Train with Bernoulli(γ) random rewards (varying γ) independent of correctness \rightarrow Improved MATH Performance
- **GRPO's clipping mechanism** \rightarrow introduces nonzero expected gradient bias toward high-prior (already probable) outputs (e.g. Code)
- Disabling Clipping \rightarrow reduces consistency of gains



(a) Performance on MATH-500



(b) Frequency of Code Reasoning

Ablations Deep Dive - 1. Code Reasoning (RLVR upweights pre-training biases)

MATH Question:

What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form.

Qwen2.5-Math-7B Solution (correct):

To find the distance between two points (x_1, y_1) and (x_2, y_2) in a Cartesian plane...
Let's break this down step-by-step and compute the result using Python.

```
1 import math
2 ...
3 # Calculate the distance using the distance formula
4 distance = math.sqrt(dx**2 + dy**2)
5 print(distance)
```

output: 10.816653826391969

...

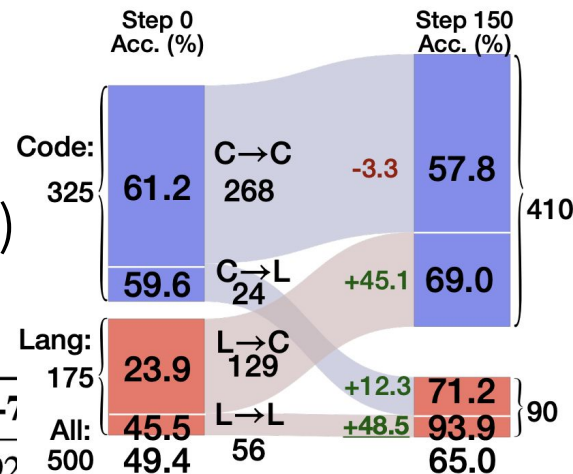
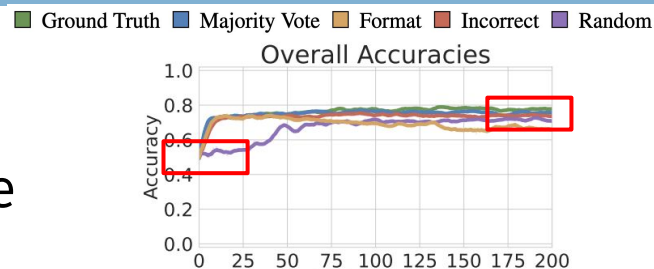
Thus, the final answer is: $3\sqrt{13}$

Ablations Deep Dive - 1. Case study: Code Reasoning (RLVR upweights pre-training biases)

Observations:

- Qwen2.5-Math frequently produce Python-like code as CoT, answers that include code have much higher accuracy.
- RLVR (even with spurious rewards) increases the frequency of code reasoning (from ~65% \rightarrow >90%) and this correlates with improved performance.

Model	Qwen2.5-Math-7B	Qwen2.5-Math-1.5B	Qwen2.5-72B
Code Frequency	65.0	53.6	92.0
Acc. w/ Code	60.9	52.6	39.0
Acc. w/ Lang	35.0	17.2	61.0



(e) Random

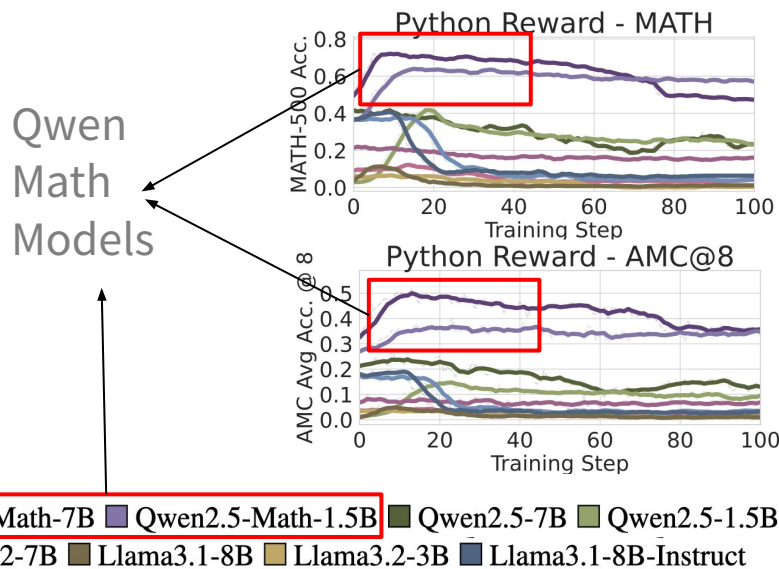
Ablations Deep Dive - 2. Causal Intervention on code reasoning

Hypothesis: Code Reasoning drives Qwen2.5-Math-7B's performance when doing RLVR w/ Spurious rewards

Causal Induction of Code Reasoning → increases MATH accuracy

- Prepend “Let’s solve this using Python.”
- RL reward = 1 whenever generated output contains “python”
 - code reasoning in > 99% of answers after 20 training steps for Qwen2.5-Math-7B

Model	Original	Prompting	Abs. Diff.
Qwen2.5-Math-1.5B	36.2%	60.4%	+24.2%
Qwen2.5-Math-7B	49.4%	64.4%	+15.0%
Qwen2.5-1.5B	3.0%	13.0%	+10.0%
Qwen2.5-7B	41.6%	22.2%	-19.4%
Llama3.2-3B-Instruct	36.8%	8.2%	-28.6%
Llama3.1-8B-Instruct	36.8%	15.2%	-21.6%

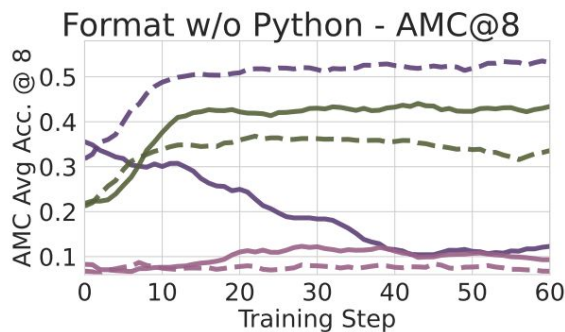
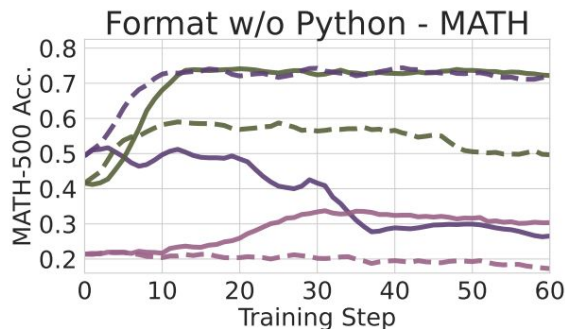


Ablations Deep Dive - 2. Causal Intervention on code reasoning

Hypothesis: Code Reasoning drives Qwen2.5-Math-7B's performance when doing RLVR w/ Spurious rewards

Causal Inhibition of Code Reasoning → decreases MATH accuracy

- compound reward = (**original reward**) \wedge (**no occurrence of “python” in output**)
- Penalizing code reasoning degrades Qwen-Math but improves other models.



(a) Format w/o Python

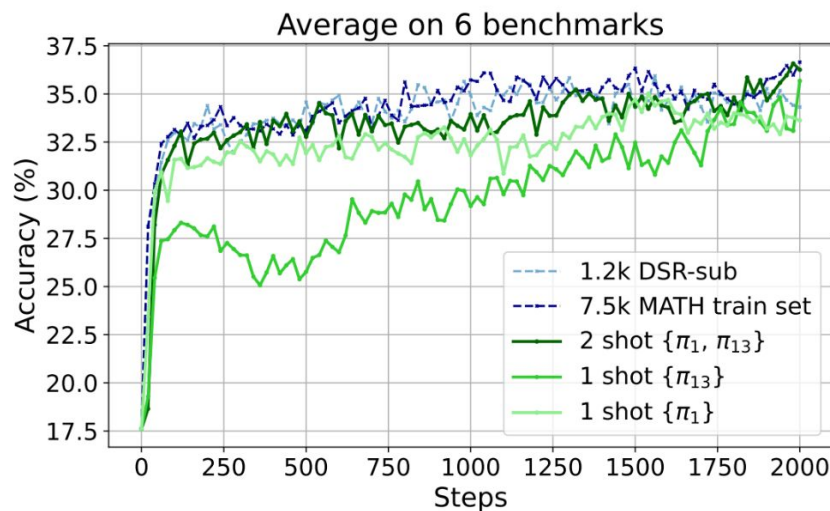
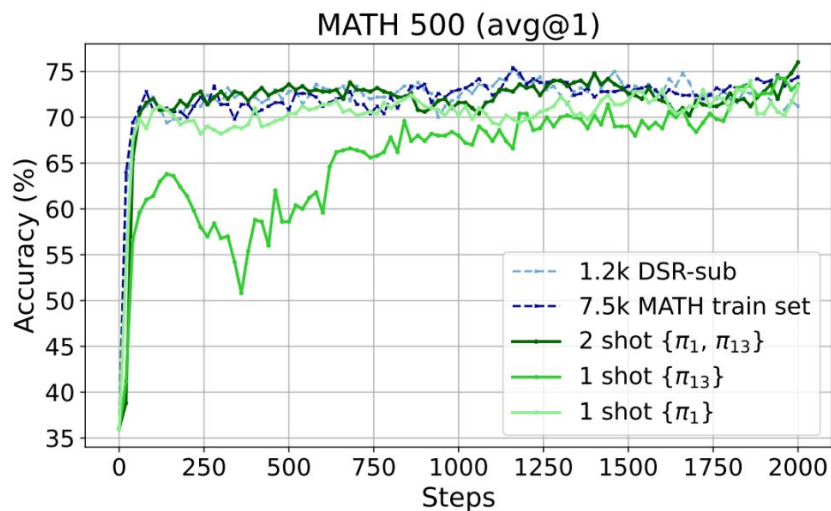
■ Qwen-Math-7B ■ Qwen-7B ■ Olmo2-7B-SFT
— Compound Reward - - Original Reward

Incorrect Rewards - Why do they work?

- Incorrect labels are obtained from majority voting, and may still be close to ground truth answer.
- Incorrect labels might function like format rewards (if you can't extract it, you can't evaluate it)

(Connecting the Dots) RL from 1 example + Other Similar Works

RL from 1 example (one-shot RLVR \rightarrow big gains from 1 example)



Other Similar works:

- Maximizing Confidence Alone Improves Reasoning

Impact on the Field

Promoted Numerous Community Opinions & discussions

E.g. Comments: “Incorrect Baseline Evaluations”, Reproducibility checklist is necessary, high variance with prompting, etc.

→ Enabled the field to reach at some conclusions

+ Also Potentially Improved Spurious Rewards Paper (compared to it's first version)

Curious to know how you ran baseline: Qwen-2.5? As per official numbers: Instruct, Math variant ****already**** performs at 75% & 83.6% on MATH. Is the performance gain only from getting right format in accord with your eval script?

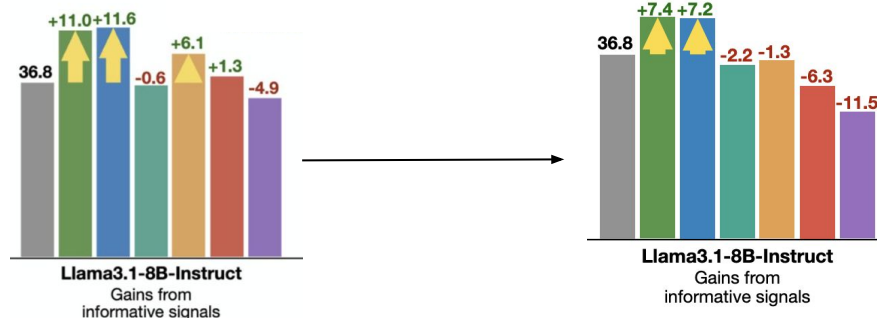
Incorrect Baseline Evaluations Call into Question Recent LLM-RL Claims

Authors*: Nikhil Chandak, Shashwat Goel, Ameeya Prabhu

*author order is alphabetic.

TL;DR

There has been a flurry of recent papers proposing new RL methods that claim to improve the “reasoning abilities” in language models. The most recent ones, which show improvements with random or no external rewards have led to a lot of surprise and excitement.



Key Takeaways

1. RLVR with random/incorrect/format rewards can substantially boost MATH performance, notably in Qwen2.5-Math. Effect is not universal across model families.
2. RLVR appears to amplify pre-existing strategies (code-style chain-of-thought)
3. Optimizer mechanics (GRPO clipping) can turn noise into a directional training signal.
4. **Don't trust single-model RLVR gains without strict parity controls and cross-model replication, Validate widely and match baselines:** control prompts, gen settings, and model families, Optimizer ablations

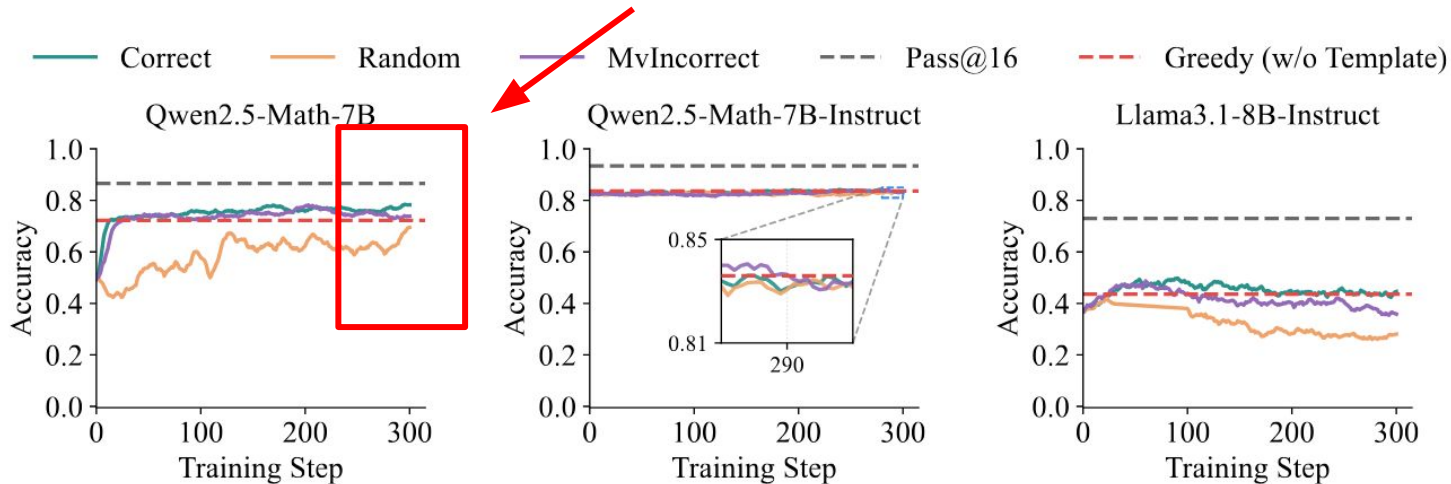
Reasoning or Memorization?

Wu et. al. 2025

Spurious rewards...are helpful?

- Models can be further improved through some RL
 - Rewards don't have to be accurate to improve Qwen performance!

But why?



But why?

Hypothesis #1 -> Data contamination

Parts of evals (i.e., MATH-500) leaked into pre-training data

Hypothesis #2 -> Strong math capacity

Qwen is a strong math model and can deal with “noisy” updates

But why?



Hypothesis #1 -> Data contamination

Parts of evals (i.e., MATH-500) leaked into pre-training data

~~**Hypothesis #2** -> Strong math capacity~~

~~Qwen is a strong math model and can deal with “noisy” updates~~

Qwen memorizes prompts!

Example from MATH-500

Problem: Suppose a and b are different prime numbers greater than 2. How many number divisors are there for the integer $a(2a + b) - 2a^2 + ab$?

Answer: 8



Partial Prompt

Prompt: Suppose a and b are different prime numbers greater than

Qwen2.5-Math-7B Output

2. How many whole-number divisors are there for the integer $a(2a + b) - 2a^2 + ab$?

To determine the number of whole-number divisors of the integer $a(2a + b) - 2a^2 + ab$, we first need to simplify the expression.

Given the expression:

Memorization metrics

1. **Partial-Prompt Completion Rate:**

Percentage of problems where model re-generates the remainder of prompt based on truncated prompt

2. **Partial-Prompt Answer Accuracy:**

Percentage of problems where model's re-generated output contains correct answer

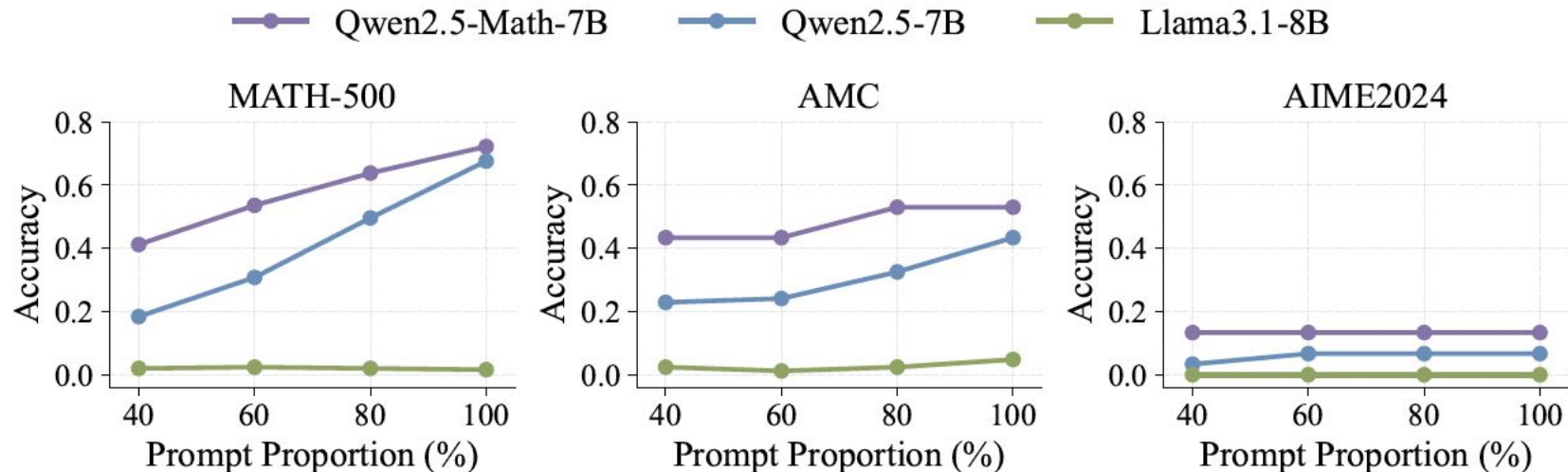
**There should be no way to complete these tasks
without the full context of the problems!**

Memorization results

Model	Dataset	Size	80%-Problem		60%-Problem		40%-Problem	
			RougeL	EM	RougeL	EM	RougeL	EM
Qwen2.5-Math-7B	MATH-500	500	81.25	65.80	78.06	54.60	69.01	39.20
	AMC	83	77.38	55.42	70.25	42.17	75.17	36.14
	AIME2024	30	74.04	56.67	55.31	20.00	57.72	16.67
Llama3.1-8B	MATH-500	500	48.33	17.80	40.55	3.80	32.07	0.60
	AMC	83	44.54	4.82	30.62	0.00	27.10	0.00
	AIME2024	30	50.50	13.33	30.80	0.00	26.08	0.00

Partial-prompt completion rate

Memorization results



Partial-prompt answer accuracy

Hypothesis #1 is correct!

RandomCalculation dataset

5-Step Calculation

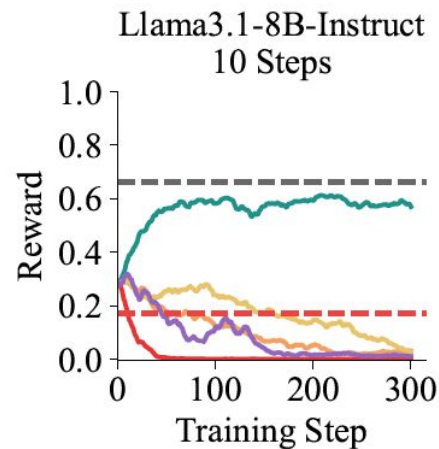
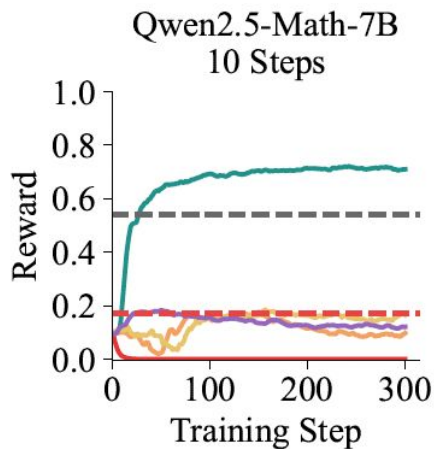
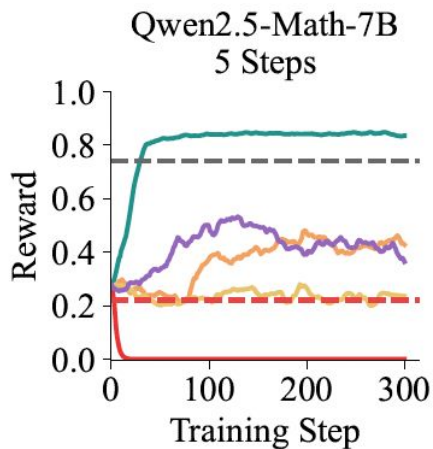
Problem: Evaluate this LaTeX numerical expression step-by-step and give the final value within `\boxed{}`:

$$45^2 - \frac{94}{6} / \left(\frac{76}{4} / \frac{19}{5} - 35^3 \right) + 81^2$$

Answer: `8586.00036544592`

Reward $r \in [0, 1]$, not $\{0, 1\}$

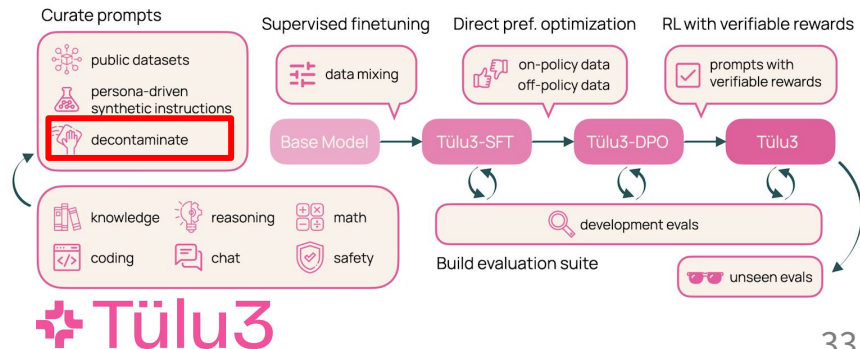
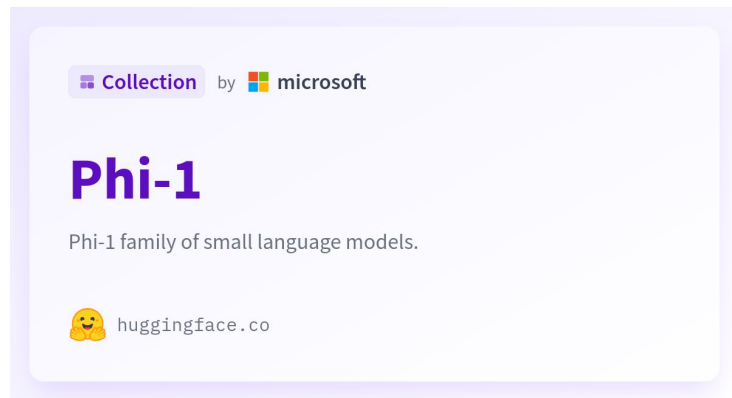
RandomCalculation results



Hypothesis #2 is incorrect!

Impact on the field...

- Quite a recent paper! Difficult to say...
 - RandomCalculation seems to be a bit contrived as a dataset (i.e., it is just arithmetic)
 - Hopefully the work serves as an additional reminder about data contamination!



Takeaways and follow-ups

1. No free lunch! Real rewards -> better performance
 - **Follow-up:** can we check for memorization within closed-source models?
2. Decontaminate datasets before evaluation!
 - **Follow up:** RandomCalculation only deals with math (i.e., just arithmetic). How would observations change for domains like physics, logic, etc.?

Critic - Spurious Rewards

#1.1 The baseline is not as high as it should be

Base results are much worse than standardized evaluations from [Sober Reasoning](#) ([blog post](#))
→ Is spurious reward RL just recovering the lost performance?

■ Reported Base Accuracy

■ Standard Eval Base Accuracy

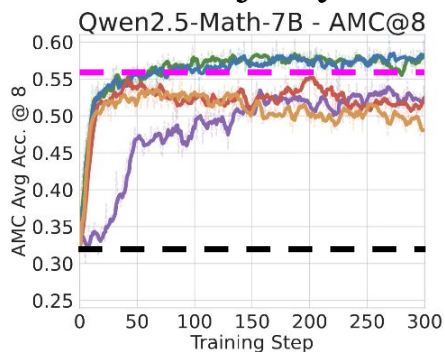
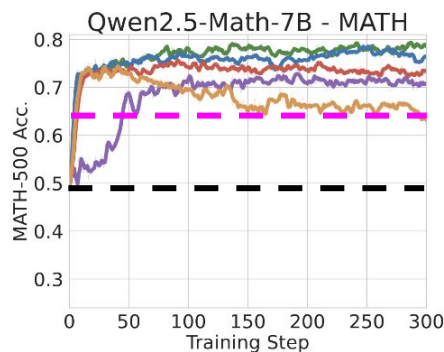
■ Ground Truth

■ Majority Vote

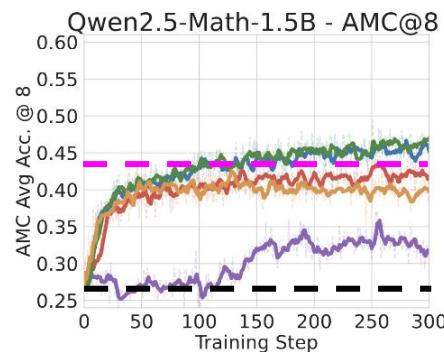
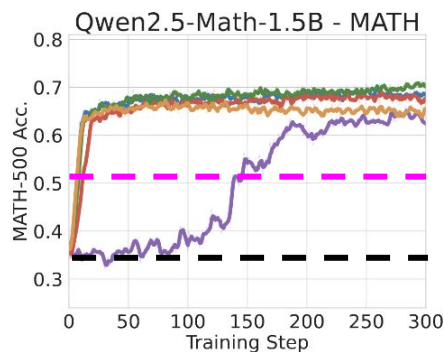
■ Format

■ Incorrect

■ Random



(a) Qwen2.5-Math-7B



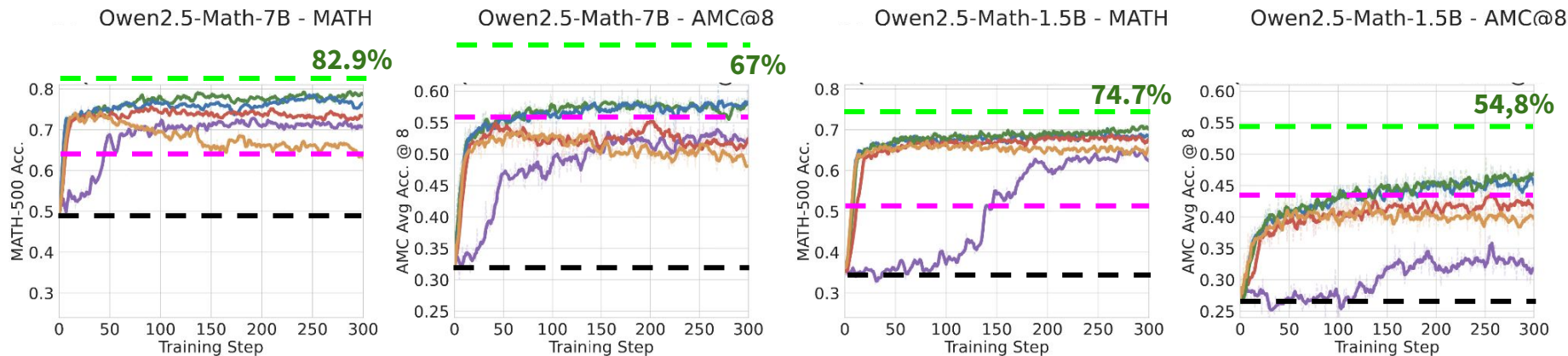
(b) Qwen2.5-Math-1.5B

#1.2 Did not use Qwen2.5-Math-Instruct

Instruct version of Qwen2.5-Math model performs far better than Base + SFT

→ Would spurious reward RL still work in this case?

■ Reported Base Accuracy ■ Standard Eval Base Accuracy ■ Math Instruct Accuracy
■ Ground Truth ■ Majority Vote ■ Format ■ Incorrect ■ Random



(a) Qwen2.5-Math-7B

(b) Qwen2.5-Math-1.5B

#1.3 Code reasoning not causally related to performance

Claim: “Increased code reasoning is one primary driver of performance gains for spurious rewards”

- “Reasoning or Memorization” paper points out performance come from memorization/contamination
- Code reasoning is a behavior in the base Qwen2.5-Math model
- RL with spurious rewards reinforce high-prior pre-trained behavior (like code reasoning)

My take: Code reasoning and performance gain could be artifact of pre-training data and memorization. **Inhibiting and inducing code reasoning is just pushing memorization**

Critic for “Spurious Rewards” – Overview

- The baseline is not as high as it should be ([blog post](#))
- Didn't test Qwen2.5-Math-Instruct models
- Code reasoning frequency might not be causally related to performance, but just a correlation through memorization

Prop - Spurious Rewards

Contribution 1: Fake Rewards Have Positive Effect

- In the abstract, the authors mention that “spurious rewards have little, no, or even negative correlation with the correct answer”.
- The paper does not claim spurious rewards are superior to correct ones, but demonstrates that:
 - Compared to an untrained baseline, even spurious rewards can provide positive learning gains.
 - This counter-intuitive finding challenges traditional assumptions about reward signal quality in RL.
- The Reasoning or Memorization paper is based on the conclusion of this one.

Contribution 1: Fake Rewards Have Positive Effect

- Reward Design:
 - Multiple types of spurious reward signals were used for training.
- General Observations: Effectiveness is model-dependent.
 - Works reasonably well on Qwen, but often fails on other models like OLMo and Llama 3.2.
 - Even on Qwen, not all spurious signals are effective.

Contribution 2: Why Qwen Successfully Converge?

- The foundation is pre-training: The effectiveness of spurious rewards is highly dependent on the model's inherent, pre-existing internal reasoning capabilities.
- Hypothesis: Qwen acquired the powerful strategy of code-augmented reasoning during pre-training.
- Parallel Insight from CodeIO:
 - This finding strongly aligns with the insight from CodeIO: Using code as an intermediate representation can robustly enhance model reasoning and generalization.
 - Related Work: CodeIO: <https://arxiv.org/pdf/2502.07316v2>

1.1 The baseline is not as high as it should be

- Few shot is probably the key of high eval values
- From my own evaluation the baseline is authentic (standard evaluation, 0 shot).

```
(run_inference_one_model pid=1381941) 2025-10-01 05:26:17,115 INFO compiled_dag_node.py:2198
(run_inference_one_model pid=1381941) 2025-10-01 05:26:17,115 INFO compiled_dag_node.py:2201
Splits: 100%|
[2025-10-01 05:26:34,494] [ INFO]: --- COMPUTING METRICS --- (pipeline.py:498)
[2025-10-01 05:26:44,664] [ INFO]: --- DISPLAYING RESULTS --- (pipeline.py:540)
| Task | Version | Metric | Value | Stderr |
|-----|-----|-----|-----|-----|
| all | | math_pass@1:1_samples | 0.5580 | ± 0.0222 |
| | | math_pass@1:4_samples | 0.5555 | ± 0.0169 |
| lighteval:math_500:0 | 2 | math_pass@1:1_samples | 0.5580 | ± 0.0222 |
| | | math_pass@1:4_samples | 0.5555 | ± 0.0169 |
[2025-10-01 05:26:44,682] [ INFO]: --- SAVING AND PUSHING RESULTS --- (pipeline.py:530)
[2025-10-01 05:26:44,682] [ INFO]: Saving experiment tracker (evaluation_tracker.py:196)
[2025-10-01 05:26:46,595] [ INFO]: Saving results to /rscratch/yuezhohu/data/evals/Qwen/Qw
```


1.2 Did not use Qwen2.5-Math-Instruct

- Instruct models are already trained with RL. It's meaningless to start from instruction-following models.
 - Shown in “reasoning or memorization” paper
- Starting RL from base model is a standard operation by many following works (e.g. DeepSeek R1 Zero).
- Even with instruction-following models, relative study can steadily reproduce the results.

nano-r1 Public

Watch 2 Fork 0 Star 5

main 1 Branch 0 Tags

Go to file

Add file

<> Code

About

No description, website, or topics provided.

Readme
Activity
5 stars
2 watching
0 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

Jupyter Notebook 100.0%

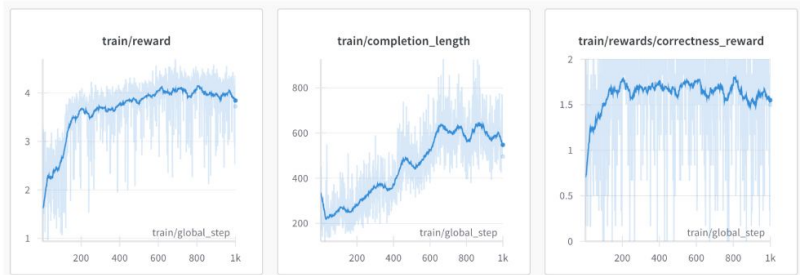
TruffleClock	gitignore	b8750f6 · 8 months ago	2 Commits
images	init	8 months ago	
.gitignore	gitignore	8 months ago	
README.md	init	8 months ago	
nano-r1.ipynb	init	8 months ago	

README

nano-r1

Taste the "aha moment" for \$12 in 3 hours

- Uses [Unsloth](#) to run reinforcement learning on [Qwen 2.5 3B](#) to create a mini reasoning model
- The model also forgets all its censorship



```
prompt = "How many r's in strawberry?"
```

TinyZero Public

Watch 125

Fork 1.5k

Star 12.2k

main

6 Branches

0 Tags

🔍 Go to file

Add file

<> Code

Jiayi-Pan-xai	Update README.md	6652a63 · 6 months ago	🕒 154 Commits
.github/workflows	[perf] feat: Support dynamic batch size (#118)	9 months ago	
docker	[docker] megatron: add TE to ngc dockerfile (#88)	9 months ago	
docs	[misc] fix: fix license (#110)	9 months ago	
examples	fix instruct train	9 months ago	
patches	[init] feat: upload first open source version of verl	last year	
scripts	Fix Ray OOM	7 months ago	
tests	[algo] feat: support GRPO algorithm (#124)	9 months ago	
verl	fix instruct train	9 months ago	
.gitignore	[example] docs: add getting started notebook with free G...	9 months ago	
.readthedocs.yaml	[init] feat: upload first open source version of verl	last year	
.style.yapf	[init] feat: upload first open source version of verl	last year	
LICENSE	[init] feat: upload first open source version of verl	last year	
Notice.txt	[init] feat: upload first open source version of verl	last year	
OLD_README.md	add readme	9 months ago	
README.md	Update README.md	6 months ago	
cover.png	Add files via upload	9 months ago	
pyproject.toml	[misc] feat: s:port rmpad/data-packing in FSDP with tran...	9 months ago	
requirements.txt	[example] docs: add getting started notebook with free G...	9 months ago	

About

Minimal reproduction of DeepSeek R1-Zero

- Readme
- Apache-2.0 license
- Activity
- 12.2k stars
- 125 watching
- 1.5k forks

Report repository

Contributors 4

- xingyaoww Xingyao Wang
- Jiayi-Pan Jiayi Pan
- junzhang Junjie ZHANG
- lifan-yuan Lifan Yuan

Languages



3B+ model In this case, the base model is able to develop sophisticated reasoning skills.

```
export N_GPU=2
export BASE_MODEL={path_to_your_model}
export DATA_DIR={path_to_your_dataset}
export ROLLOUT_TP_SIZE=2
export EXPERIMENT_NAME=countdown-qwen2.5-3b
export VLLM_ATTENTION_BACKEND=XFORMERS

bash ./scripts/train_tiny_zero.sh
```

Instruct Ablation

We experiment with [Qwen-2.5-3B Instruct](#) too. **Data Preparation** To follow chat template, we need to reprocess the data:

```
conda activate zero
python examples/data_preprocess/countdown.py --template_type=qwen-instruct --local_dir={path}
```

Training

```
export N_GPU=2
export BASE_MODEL={path_to_your_model}
export DATA_DIR={path_to_your_dataset}
export ROLLOUT_TP_SIZE=2
export EXPERIMENT_NAME=countdown-qwen2.5-3b-instruct
export VLLM_ATTENTION_BACKEND=XFORMERS

bash ./scripts/train_tiny_zero.sh
```

Acknowledge

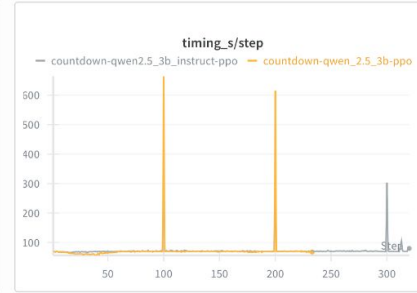
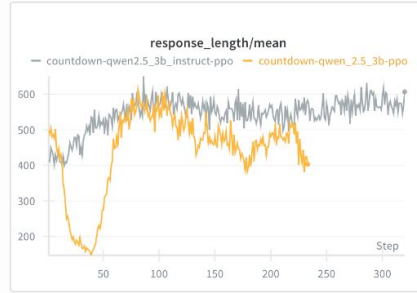
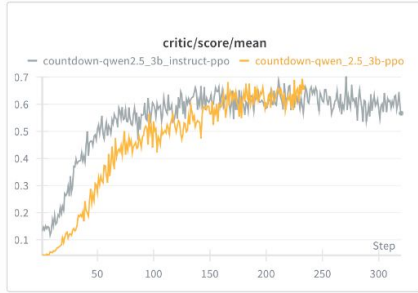
- We run our experiments based on [veRL](#).
- We use Qwen2.5 series base model [Qwen2.5](#).

Citation

```
@misc{tinyzero,
  author = {Jiayi Pan and Junjie Zhang and Xingyao Wang and Lifan Yuan and Hao Peng and
  title = {TinyZero},
  howpublished = {https://github.com/Jiayi-Pan/TinyZero}.
```

Search panels with regex

Core 3



> val 1

> actor 6

> critic 22

> mfu 2

> prompt_length 4

> response_length 3

> timing_per_token_ms 6

> timing_s 8

Blog

Train your own R1 reasoning model with Unsloth (GRPO)

Feb 6, 2025 • By Daniel & Michael

Feb 20, 2025 Update: You can now train your own reasoning model with just 5GB VRAM (down from 7GB VRAM) + 10x longer context lengths with Unsloth! [Read update here!](#)

View our step-by-step [Tutorial in our docs here!](#)

Today, we're excited to introduce reasoning in [Unsloth](#)! DeepSeek's R1 research revealed an "aha moment" where R1-Zero autonomously learned to allocate more thinking time without human feedback by using Group Relative Policy Optimization (GRPO).

We've enhanced the entire GRPO process, making it use 80% less VRAM than Hugging Face + FA2. This allows you to reproduce R1-Zero's "aha moment" on just 7GB of VRAM using Qwen2.5 (1.5B).

[Join Our Discord](#)

Llama3.1 (8B)-GRPO.ipynb

File Edit View Insert Runtime Tools Help



Share



Q Commands + Code + Text ▶ Run all Copy to Drive

Connect T4 ^



To run this, press "Runtime" and press "Run all" on a **free** Tesla T4 Google Colab instance!



Join our Discord

Documentation

Join Discord if you need help + ⭐ Star us on [Github](#) ⭐

To install Unsloth on your own computer, follow the installation instructions on our Github page [here](#).

You will learn how to do [data prep](#), how to [train](#), how to [run the model](#), & [how to save it](#)

News

Unsloth now supports [gpt-oss RL](#) with the fastest inference & lowest VRAM. Try our [new notebook](#) which automatically creates kernels!

[Vision RL](#) is now supported! Train Qwen2.5-VL, Gemma 3 etc. with GSPO or GRPO.

Introducing Unsloth [Standby for RL](#): GRPO is now faster, uses 30% less memory with 2x longer context.

Unsloth now supports Text-to-Speech (TTS) models. Read our [guide here](#).

Visit our docs for all our [model uploads](#) and [notebooks](#).

Installation

```
[ ] %capture
import os
os.environ["UNSLOTH_VLLM_STANDBY"] = "1" # [NEW] Extra 30% context lengths!
if "COLAB_" not in "".join(os.environ.keys()):
    # If you're not in Colab, just use pip install or uv pip install
    !pip install unsloth vllm
else:
    pass # For Colab / Kaggle, we need extra instructions hidden below \
```

> Colab Extra Install

[] [Show code](#)

Unsloth



) trainer.train()										
229	0.000700	1.666667	1.290994	168.833344	0.017366	0.000000	0.000000	0.000000	0.333333	1.333333
230	0.000800	1.727500	1.213482	193.333344	0.019479	-0.105833	0.000000	0.000000	0.500000	1.333333
231	0.000500	2.083333	1.020621	192.000000	0.013044	0.000000	0.000000	0.000000	0.416667	1.666667
232	0.000700	2.500000	0.000000	149.333344	0.018301	0.000000	0.000000	0.000000	0.500000	2.000000
233	0.001900	0.538833	0.095122	68.166672	0.048356	0.038833	0.000000	0.000000	0.500000	0.000000
234	0.003600	1.250000	1.369306	142.666672	0.090407	0.000000	0.000000	0.000000	0.250000	1.000000
235	0.000700	2.500000	0.000000	175.500000	0.016705	0.000000	0.000000	0.000000	0.500000	2.000000
236	0.001800	1.254167	1.399144	238.333344	0.044271	-0.079167	0.000000	0.000000	0.333333	1.000000
237	0.001500	2.083333	1.020621	136.666672	0.037116	0.000000	0.000000	0.000000	0.416667	1.666667
238	0.000700	2.500000	0.000000	107.000000	0.018739	0.000000	0.000000	0.000000	0.500000	2.000000
239	0.002100	2.013000	1.000507	140.333344	0.053014	-0.070333	0.000000	0.000000	0.416667	1.666667
240	0.001100	2.500000	0.000000	177.666672	0.027495	0.000000	0.000000	0.000000	0.500000	2.000000
241	0.000500	2.500000	0.000000	166.833344	0.012520	0.000000	0.000000	0.000000	0.500000	2.000000
242	0.001200	2.036000	0.815215	191.500000	0.029168	-0.130667	0.000000	0.000000	0.500000	1.666667
243	0.003400	2.500000	0.000000	93.000000	0.084111	0.000000	0.000000	0.000000	0.500000	2.000000
244	0.001100	2.500000	0.000000	107.166672	0.028535	0.000000	0.000000	0.000000	0.500000	2.000000
245	0.000600	1.750000	1.172604	229.666672	0.015354	0.000000	0.000000	0.000000	0.416667	1.333333
246	0.000400	0.416667	0.204124	239.166672	0.009761	0.000000	0.000000	0.000000	0.416667	0.000000
247	0.000600	2.500000	0.000000	169.500000	0.014920	0.000000	0.000000	0.000000	0.500000	2.000000
248	0.000300	0.500000	1.000000	226.000000	0.008604	0.000000	0.000000	0.000000	0.166667	0.333333
249	0.002100	1.666667	0.816497	166.333344	0.051596	0.000000	0.000000	0.000000	0.000000	1.666667
250	0.001100	1.472833	1.633226	291.666687	0.027454	-0.193833	0.000000	0.000000	0.333333	1.333333

Streaming output truncated to the last 5000 lines.

The cost of Joe's new HVAC system is \$20,000. It includes 2 conditioning zones, each with 5 vents. In dollars, what is the cost of the system per vent?

Answer:
2,000

Response:
<to find the cost per vent, we need to know the total number of vents and the total cost of the system.

1.3 Code reasoning not causally related to performance

- Chain of thought can be whatever the model feels “necessary” to do reasoning.
- “Code reasoning is just pushing memorization”: not necessarily
-

Further analysis hints at the origins of this reasoning behavior: Qwen2.5-Math-7B maintains accuracy when faced with numerical variations in questions from common math benchmarks—correctly predicting answers when different numbers are substituted in the original problems, echoing existing literature (Huang et al., 2025). Furthermore, the model can often produce complex numerical answers with high precision when predicting code outputs, as shown in Figure 21, 20. However, when those questions are reformulated with an alternative narrative, the model stops utilizing code reasoning approaches, as shown in Figures 26, 27, 28. Hence, we conjecture that Qwen2.5-Math-7B has seen many code-assisted math reasoning traces during pre-training.

Prop - Reasoning or Memorization

Proponent for “*Reasoning or Memorization*” – Overview

The paper is well-structured and conveys two points well.

1. **Data contamination, or strong math capabilities?** – The data contamination experiment is well-carried out and persuasive
 - a. First demonstrated that the issue exists
 - b. Then showed that spurious rewards’ observation no longer holds with a new clean dataset
2. **Why does random reward retrieve memorized answers?**
 - Good mathematical intuition proposed in Section 4.4
 - GRPO is the cause of eliciting memorized answer; along the line of “*Spurious Rewards*” Section 4.4.2; but instead of code-reasoning elicitation (Shao et al., 2025), it is memory elicitation
- Question: why does memorization and code reasoning appear together?
See figures 12-14

#1.1 Data contamination – Proving this issue exists

- Figure 1 and Table 2 should be enough to prove it

Example from MATH-500

Problem: For how many positive integers $n > 1$ is it true that 2^{24} is a perfect n^{th} power?

Answer: 7

Partial Prompt

Prompt: For how many positive integers $n > 1$ is

Qwen2.5-7B Output

it true that 2^{24} is a perfect n^{th} power?

To determine for how many positive integers $n > 1$ the number 2^{24} is a perfect n -th power...

First, we need to find...

...

Let's count these values. There are 7 such values...

...

The final answer is 7.

Llama3.1-8B Output

the set of positive divisors of n (including 1 and n) less than 50 ?

(A) 15 (B) 16 (C) 17 (D) 18 (E) 19

The number of positive divisors of n is given by...

...

The number of factors of each of these numbers is...

...

The answer is (B) 16

Table 2: Accuracy (Exact Match, EM) and ROUGE-L scores on several datasets (lower scores in gray) under different prompt prefix ratios in greedy decoding mode without applying chat template, namely *Greedy (w/o Template)* configuration.

Model	Dataset	Size	80%-Problem		60%-Problem		40%-Problem	
			RougeL	EM	RougeL	EM	RougeL	EM
Qwen2.5-Math-7B	MATH-500	500	81.25	65.80	78.06	54.60	69.01	39.20
	AMC	83	77.38	55.42	70.25	42.17	75.17	36.14
	AIME2024	30	74.04	56.67	55.31	20.00	57.72	16.67
	AIME2025	30	54.71	16.67	34.88	0.00	27.43	0.00
	MinervaMath	272	36.08	2.94	31.22	0.37	29.35	0.00
	LiveMathBench	100	42.76	5.00	32.78	0.00	29.97	0.00
Qwen2.5-7B	MATH-500	500	66.42	40.20	60.98	21.20	50.36	8.20
	AMC	83	73.24	49.40	64.42	33.73	63.79	28.92
	AIME2024	30	59.80	30.00	48.69	13.33	44.65	10.00
	AIME2025	30	54.61	10.00	37.59	0.00	30.30	0.00
	MinervaMath	272	35.24	2.94	32.35	0.37	27.89	0.00
	LiveMathBench	100	41.15	4.00	32.74	0.00	27.95	0.00
Llama3.1-8B	MATH-500	500	48.33	17.80	40.55	3.80	32.07	0.60
	AMC	83	44.54	4.82	30.62	0.00	27.10	0.00
	AIME2024	30	50.50	13.33	30.80	0.00	26.08	0.00
	AIME2025	30	47.04	10.00	33.49	0.00	25.20	0.00
	MinervaMath	272	36.24	2.21	29.52	0.00	27.11	0.00
	LiveMathBench	100	35.55	5.00	31.93	0.00	26.88	0.00

#1.1 Data contamination – experiments on a clean dataset

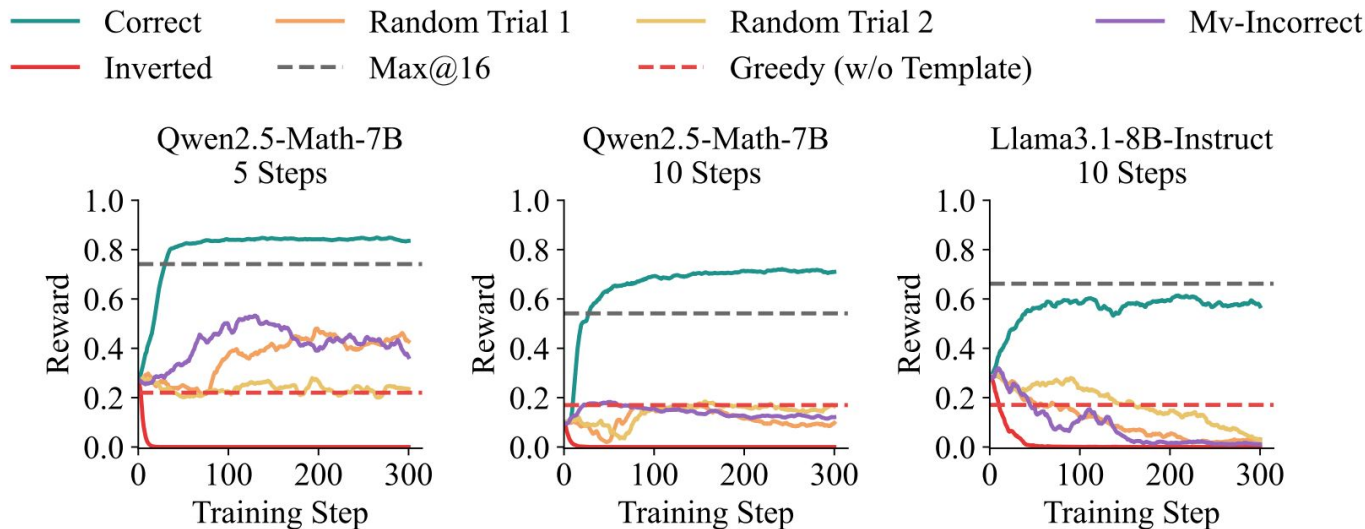


Figure 7: Reward of Qwen2.5-Math-7B and Llama3.1-8B-Instruct on *RandomCalculation*. Results are presented for datasets with 5-step and 10-step calculations.

Created a new dataset RandomCalculation, and spurious reward observation no longer holds

#1.1 Data contamination – some points worth mentioning

- Admittedly, RandomCalculation is a bit contrived – only arithmetics
 - But regardless, with correct label, RL demonstrates increase in performance as expected, so it is still a valid experiment
- Indeed, it would be more persuasive if they run experiments on say, *LiveMathBench*, which does not have much contamination

#2.1 Random reward exposes memorization – Maths

- The authors gave good intuition for understanding why random reward makes the model retrieve memorized content
- Full Maths in Section 4.4, but some quotes. With clipped objective:
 - *“high-probability tokens continue to be up-weighted without penalty”*
 - *“GRPO with random reward can retrieve these answers after learning format and leads to sharp accuracy jump”*
 - *“clipped objective introduces systematic exploitation bias for high-probability tokens, whereas mid-probability tokens are less optimized”*

#2.2 Random reward exposes memorization – KL

- Similarity in MATH-500 is higher than in RandomCalculation after RL
- *“Performance surge under spurious rewards arises because GRPO inadvertently triggers Qwen to retrieve memorized answers, rather than stimulating Qwen’s existing reasoning patterns like codes as explained in Shao et al. (2025).”*
- Do we see code reasoning after RL on RandomCalculation?

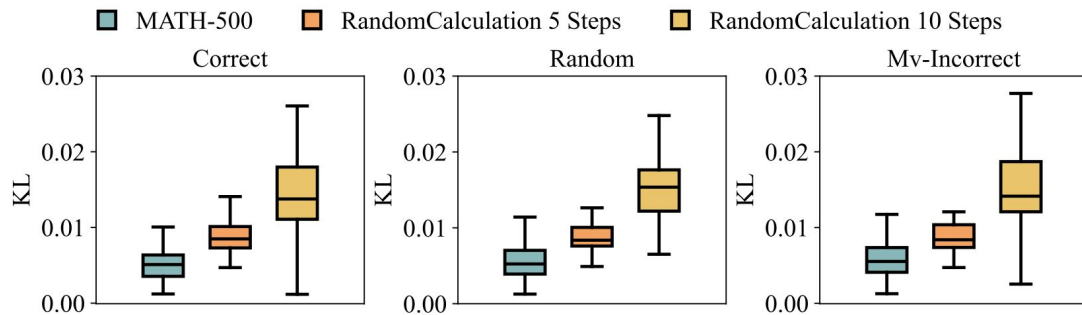


Figure 8: KL distance of model outputs before and after RL.

#3 Side note: Why do memorization and code reasoning appear together?

- Figures 12-14 demonstrate the same observation in MATH-500, AMC, AIME 2024

Example from AMC

Problem: What is the value of

$$2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3?$$

Answer:

Partial Prompt

Prompt: What is the value of

$$2^3 - 1^3$$

Qwen2.5-Math-7B Output

$$+4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3?$$

To solve the given problem, we need to evaluate the alternating sum of cubes:

$$2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3.$$

We can break this down into pairs of terms and compute the difference for each pair. Specifically, we need to compute:

$$(2^3 - 1^3) + (4^3 - 3^3) + (6^3 - 5^3) + \cdots + (18^3 - 17^3).$$

Let's write a Python code to compute this sum.

```
1 # Initialize the sum
2 result = 0
3
4 # Iterate through the pairs (2,1), (4,3), (6,5), ..., (18,17)
5 for i in range(1, 18, 2):
6     result += ( (i+1)**3 - i**3 )
7
8 print(result)
```

output

```
1 3159
```

The value of the given expression $2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3$ is .

Figure 13: **Memorization Example of Qwen2.5-Math-7B on AMC.** In this case, the first 40% of the original problem is used as the prompt, and the generation is performed under the *Greedy (w/o Template)* configuration.

Critic - Reasoning or Memorization (prasann)

Critic: “Reasoning or Memorization” (overview)

- The paper’s claim about memorization is interesting, but the investigation only shows this at a surface level, and doesn’t go as deep as it could
- “Why”: Mathematical intuition can be misleading and isn’t a substitute for experiments
- *RandomCalculation* is extremely toy, performance on it doesn’t build evidence for their hypothesis, and the task is not really a contribution
- Narrow eval scope (only 2 model families), common problem in reasoning

1.1: “Data contamination or math capabilities”

- KL and self-bleu is pretty weak evidence for memorization (e.g. SFT on the test set would prob have higher values)
- What counts as contamination?
 - E.g. if there’s similar (but non-identical) data in training, does that count?
 - The paper could’ve been much stronger if they could re-create a case where we *know* of contamination and show similar patterns

Dataset	Reward Signal	ROUGE-L
MATH-500	Correct	0.555
	Random	0.601
	Mv-Incorrect	0.563
RandomCalculation 5 Steps	Correct	0.225
	Random	0.247
	Mv-Incorrect	0.251
RandomCalculation 10 Steps	Correct	0.193
	Random	0.251
	Mv-Incorrect	0.279

Table 3: Similarity of model outputs before and after RL.

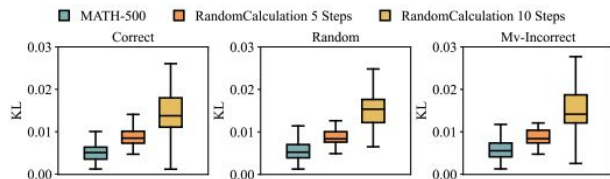


Figure 8: KL distance of model outputs before and after RL.

1.1: “RandomCalculation”

- This setting is very artificial, and may be completely independent of the other reasoning settings
- Prior RLHF work has examined extremely similar settings, so this is not a contribution (they frame this as 2 out of 4 of their contributions)
- They don’t give actual evidence of it not being contaminated
 - We propose an automatic generator that creates arbitrarily long arithmetic expressions. Zero-shot evaluation on this dataset exposes the absence of memorization, enabling fair assessment of RL methods.
 - Using this clean dataset, we conduct RL experiments and demonstrate that only *correct* reward yields stable improvement, whereas spurious rewards provide no benefit.

Other Problems

- They don't actually connect why memorization is supported by random rewards with experiments, despite having some “mathematical intuition”
- They only experiment with 2 model families. This leaves an incomplete picture of how much this affects the field.

4.4 MORE EVIDENCE FOR MEMORIZATION

Here, we provide more detailed analyses of Qwen's sudden performance gains on MATH-500 under random reward. Let $\mathcal{J}_{\text{CLIP}} = \mathbb{E}_{\hat{A}_{i,t}} \left[\min \left(r_{i,t} \hat{A}_{i,t}, \text{clip}(r_{i,t}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right) \right]$, where $\hat{A}_{i,t}$ is a random variable under the setup of random reward. Referring to Appendix B of Shao et al. (2025), the gradient of the clipped policy has the following format:

$$\nabla_{\theta} \mathcal{J}_{\text{CLIP}} = \nabla_{\theta} r_{i,t} \cdot G(r_{i,t}), \quad (2)$$

$$G(r_{i,t}) = \begin{cases} \mu, & r_{i,t} < 1 - \epsilon, \\ 0, & 1 - \epsilon \leq r_{i,t} \leq 1 + \epsilon, \\ -\mu, & r_{i,t} > 1 + \epsilon, \end{cases} \quad (3)$$

where $\mu > 0$ is a positive coefficient, $r_{i,t} = \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}$.

Overall

- The blog brings up issues in baselines w.r.t formats and sampling hyper-params. These concerns are extremely important / valid
- Both papers show signs of counterintuitive results and weaknesses of limited evaluation in reasoning
- They align with the same overall takeaway: Reasoning is a very narrow subfield of NLP with a lot of bad practices! **Proceed with caution**

Plannings

Spurious Rewards (Critique)

- The baseline is not as high as it should be ([blog post](#))
- didn't use instruct model
 - From the blog: *"However, for Llama the instruct model was tested, while for Qwen it was the base — which might be a critical difference, as instruction-tuning already elicits base model capabilities to a large extent. The equivalent instruct numbers in Qwen models are also significantly higher than the performance their methods elicit."*
- Code reasoning frequency might not be causally related to performance, but just a sign of memorization (second paper's figures 12-14)

Reasoning or Memorization (Defend)

- Data contamination experiment
 - The fact that they can complete the problem (Table 2)
 - The fact it only works for datasets released before Qwen2.5 was released
 - The fact that spurious reward does not generalize to RandomCalculation (Figure 7)
- Trying to justify GRPO eliciting memorized answer (Figure 8)
 - *"Therefore, performance surge under spurious rewards arises because GRPO inadvertently triggers Qwen to retrieve memorized answers, rather than stimulating Qwen's existing reasoning patterns like codes as explained in Shao et al. (2025)."*
 - 4.4 maths intuition – along the line of spurious rewards' maths
- Not really conflicting – look at figures 12-14, memorized + code
- Shared: spurious rewards upweight pre-existing characteristics (memorized content vs. reasoning strategies)
 - Both used maths to justify (spurious reward page 12)
- For some reason, memorization and code reasoning appear together – why is that? The pre-train data uses code to solve maths?

Questions and Moderation

- Refer Tweets?
- Refer blogs?
-

Reasoning models

- From Slack: Not everyone in class may be familiar with the general reasoning model training process, so try to give a short overview about it. (The previous class covers reasoning models already, so I don't think it has to be long, but it can be a short recap.)
- A small animation here might be useful

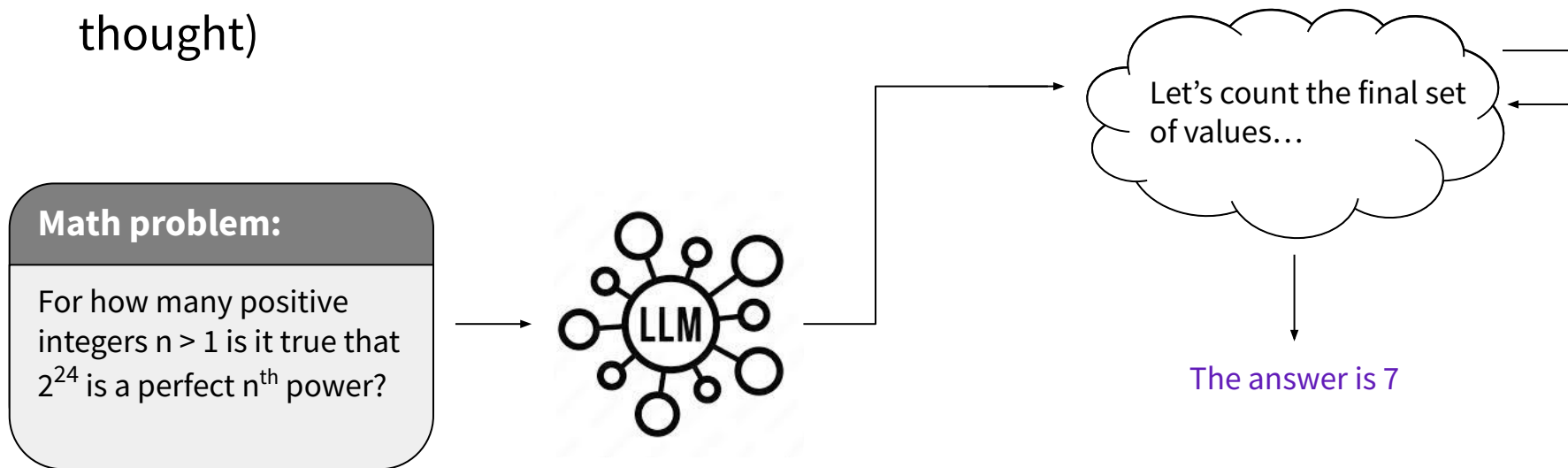
Notes for discussion

- Ask critics of spurious:
 - [Harman]
- Ask prop of spurious:
 - [Harman]

- Ask critics of memorization:
 - [Dennis]
- Ask prop of memorization:
 - [Dennis]

Reasoning models summary

- One implementation of test-time compute...
- Use RL to “reason” at test time (essentially an internal chain of thought)



Impact on the Field

Don't trust single-model RLVR gains without strict parity controls and cross-model replication

1. Match pre/post evaluation exactly
2. Test across multiple base models
3. Optimizer ablations: report runs with clipping on/off, PPO vs other optimizers
4.