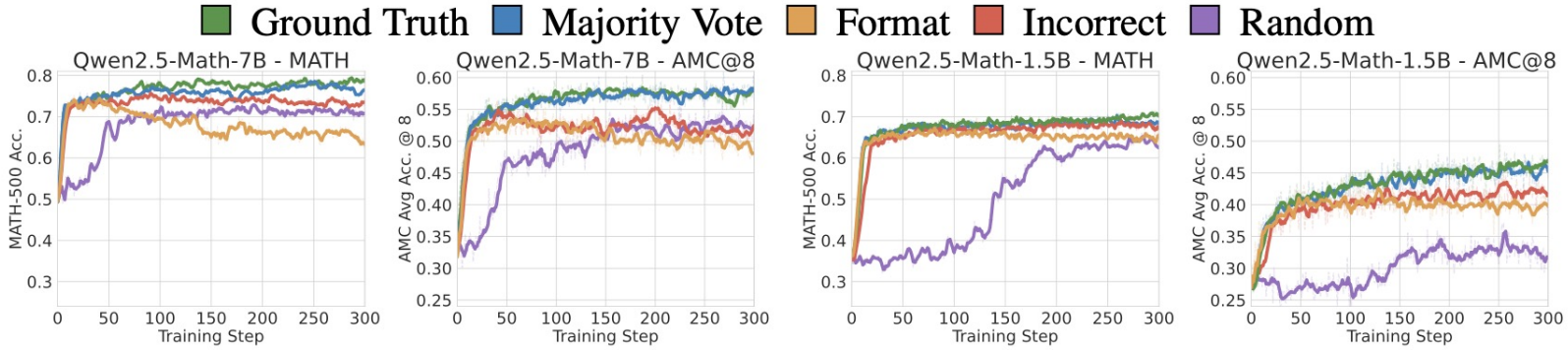
# Spurious Rewards and TinyLoRA

Bae Sun Woo

Feb 23, 2026

# Table of Contents

# Spurious Rewards: Rethinking Training Signals in RLVR




(a) Qwen2.5-Math-7B (b) Qwen2.5-Math-1.5B

| | Reward Types | Description | Gains on AMC |
|---|---|---|---|
| | Ground Truth | Ground Truth Label, Verifiably Correct Answer | 29% |
| Weak | Majority Vote | Pseudo-Label by Selecting Majority Answer | 27% |
| | Format | Reward all responses containing at least one non-empty \boxed{} expression (Disregard the responses' mathematical correctness) | 13.8% |
| Spurious | Incorrect | Reward only Incorrect Answers | 24.1% |
| | Random | Give Reward of 1 by probability hyperparameter γ ∈{0.001,0.3,0.7} | 21.4% |

# A Few Questions

**“Ground Truth ~ Majority vote => No need ground truth=> no rule based? => RL for Open Ended question?”**

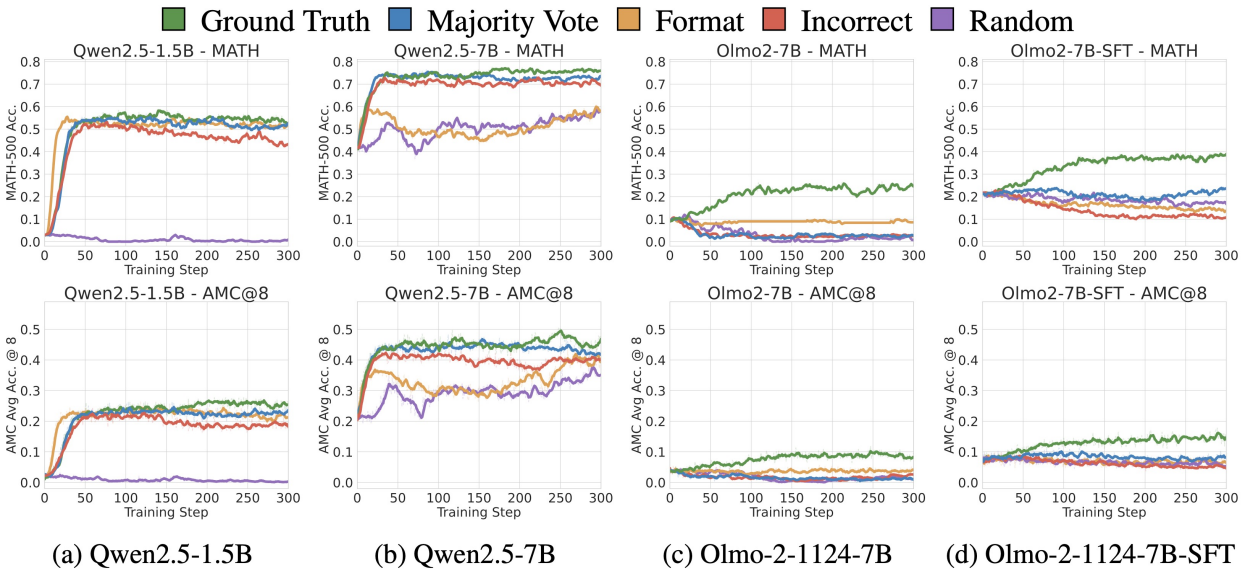
**“If Answer(lable) is not that important, then can we do in unsupervised rl?”**

=> Self-Consistency Preference Optimization, Test-Time Reinforcement Learning

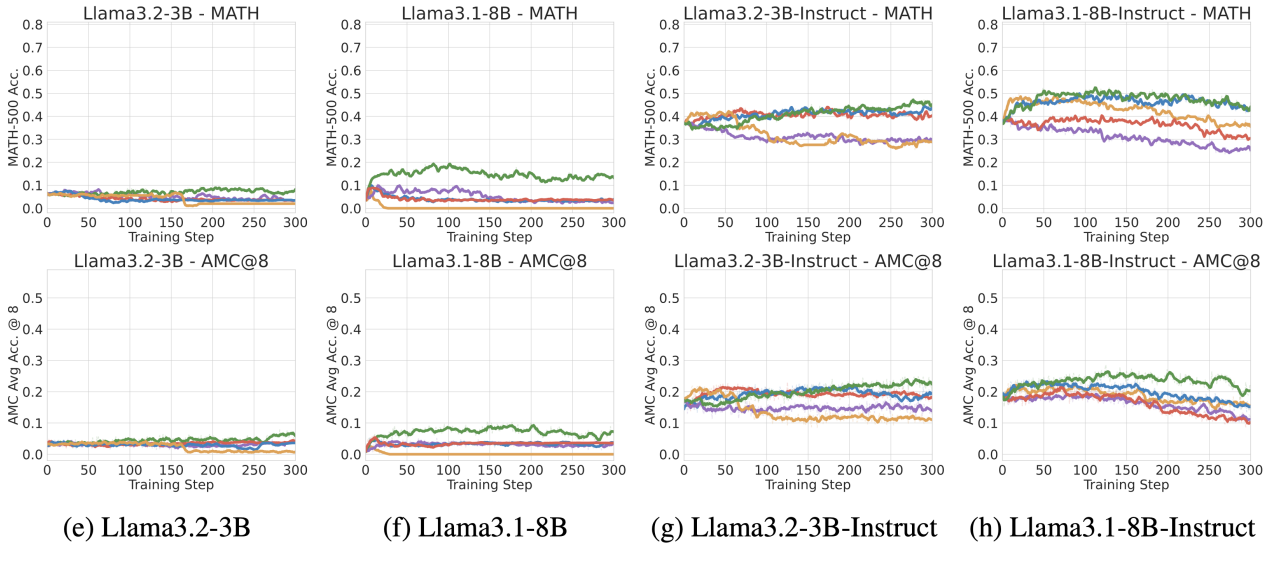
**“Incorrect is better that Random? How did incorrect rewards work?”**

=> Mentioned in later part of the paper

# Spurious Rewards: Lack of Generalization to Other Models




(a) Qwen2.5-1.5B (b) Qwen2.5-7B (c) Olmo-2-1124-7B (d) Olmo-2-1124-7B-SFT




(e) Llama3.2-3B (f) Llama3.1-8B (g) Llama3.2-3B-Instruct (h) Llama3.1-8B-Instruct

1. Models within the same family generally exhibit similar trends
2. Reward signals which work well for one model family do not necessarily generalize to other model families

"Qwen models are uniquely robust to reward signal strength"

# What Makes RLVR with Spurious Rewards Work?

## "Code Reasoning – generating code to assist in math reasoning"

**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.

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

output: 10.816653826391969

...

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

(NO Access to Code Execution)

# Code Reasoning – Accuracy & Frequency

Table 1: We track the percentage of model-generated MATH-500 responses that contain Python code before RL training, as well as their accuracy on (1) responses with code and (2) responses with only natural language. Overall, Qwen2.5-Math models achieve higher performance when using code than when not, while other models do not benefit from code reasoning. The unlisted models (Qwen2.5-1.5B, OLMo2-7B, Llama3.1-8B-Instruct, Llama3.2-3B-Instruct) *never* generated code.

| Model | Qwen2.5-Math-7B | Qwen2.5-Math-1.5B | Qwen2.5-7B | OLMo2-7B-SFT |
|---|---|---|---|---|
| **Code Frequency** | 65.0 | 53.6 | 92.2 | 98.0 |
| **Acc. w/ Code** | 60.9 | 52.6 | 39.9 | 21.0 |
| **Acc. w/ Lang** | 35.0 | 17.2 | 61.5 | 40.0 |

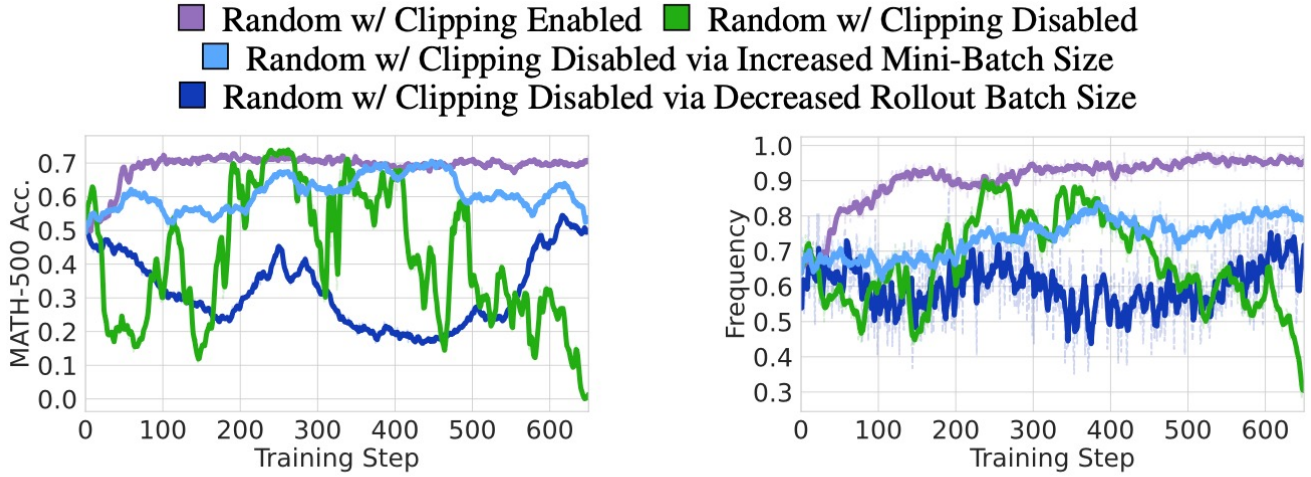
Effective Code Reasoning  Bad Code Model  Bad Code Model

"*Effective* code reasoning is a unique pre-existing capability of the Qwen2.5-Math models before RLVR training"

# How did Incorrect Rewards work? - Hypothesis

1. Many incorrect labels remain close to ground truth values
2. Incorrect labels may function like format rewards

# How did Random Rewards work? – Clipping Bias




(a) Performance on MATH-500

(b) Frequency of Code Reasoning

Expected value of Advantage in GRPO is 0 => There should be no learning signal in random rewards theoretically

$$A = \frac{r_i - \bar{r}}{\sigma} \quad \sum_{i=1}^{G} A = \frac{\sum_{i=1}^{G} r_i - G * \bar{r}}{\sigma} = \frac{G * \bar{r} - G * \bar{r}}{\sigma} = 0$$

⇒ However, clipping bias reinforces high-prior behaviors under random rewards

⇒ Model was initially smart => becomes smarter

# Spurious Rewards, 3 Main Implications

1. Base model **pretraining** significantly affects RLVR outcomes

2. Even corrupted or spurious supervision can enhance reasoning when it triggers useful **existing behaviors**

3. Effects observed in one model family may not **generalize** to others

# Learning to Reason in 13 Parameters

| W: d*k | Update | Trained (dimension) | Number of Trained Parameter | Complexity |
|---|---|---|---|---|
| LoRA | $W' = W + AB$ | $A$ (from Gaussian)<br>$B$ (from zero matrix) | $dr + rk (10^6 \sim)$ | $O(dr)$ |
| LoRA-XS | $W' = W + U\Sigma R V^T$ | $R$ (from zero matrix) | $r^2 (10^3 \sim 10^6)$ | $O(r^2)$ |
| **TinyLoRA** | $\boldsymbol{W' = W + U\Sigma(\sum_{i=1}^{u} v_i P_i)V^T}$<br>**$P_i$:randomly fixed** | **$v_i$(from zero vector)** | $\boldsymbol{u(10^0 \sim 10^3)}$ | $\boldsymbol{O(u)}$ |

# Learning to Reason in 13 Parameters - results (GRPO vs SFT)

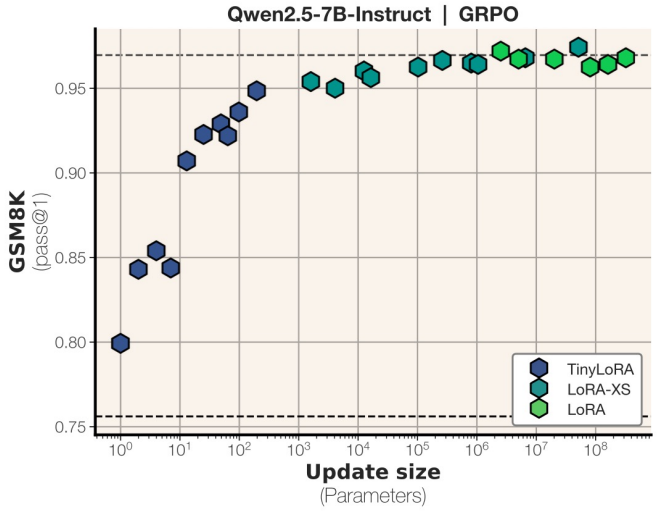



**Figure 1** Using Qwen2.5-7B-Instruct as a base model, our TinyLoRA achieves performance within 5% of full finetuning on GSM8K with only 13 parameters. Dashed lines indicate untrained and full-FT baselines.

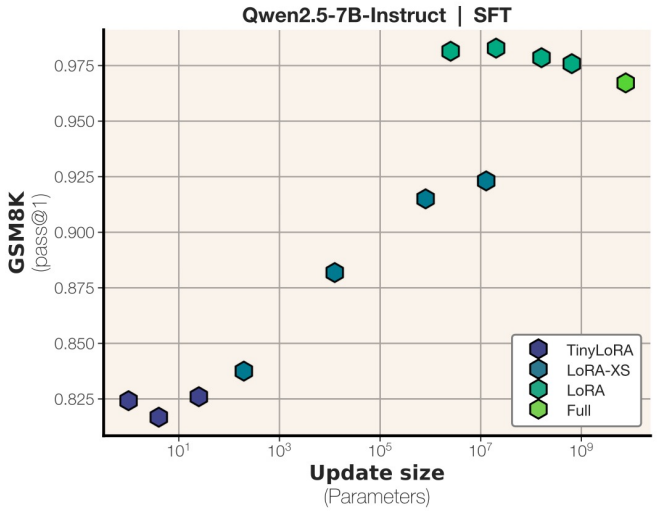



**Figure 2** Using Qwen2.5-7B-Instruct as a base model, SFT works best with larger update sizes of at least $1M$ parameters.

RL separates noise by reward annotation
(Focus on the reward : correct or not)

SFT includes noise => inefficient
(Also consider irrelevant details)

# Learning to Reason in 13 Parameters – Scaling

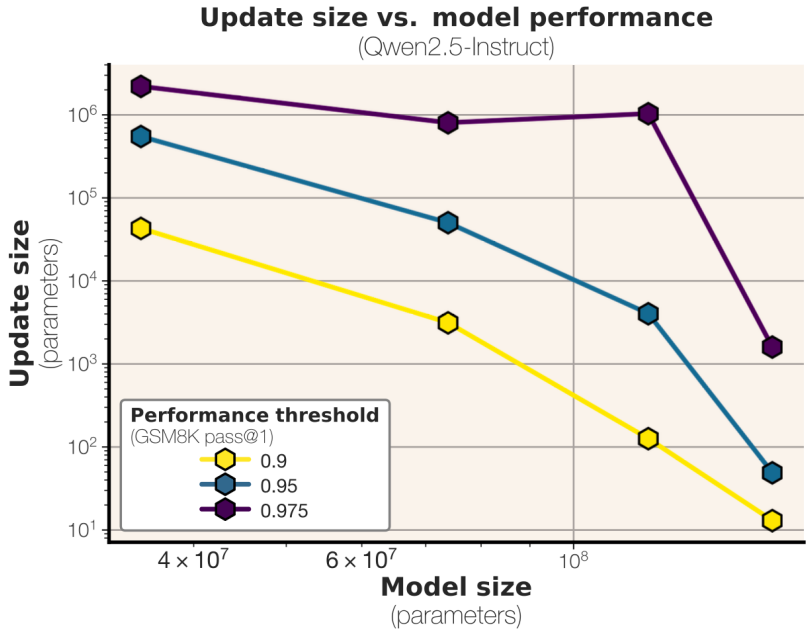



**Figure 3** Minimal-sized parameter update to hit threshold of maximum performance vs. backbone model size. Larger models require smaller updates to reach e.g. 95% of peak performance.

**The Bigger the model, the Less paramters are needed**

(seems to be counterintuitive)

How is this possible?

One theory is that the knowledge required to solve the task is **already stored in the parameters** of the model, and only the **style has to change** for task success.

# Learning to Reason in 13 Parameters
## – Selection of frozen rank (r)

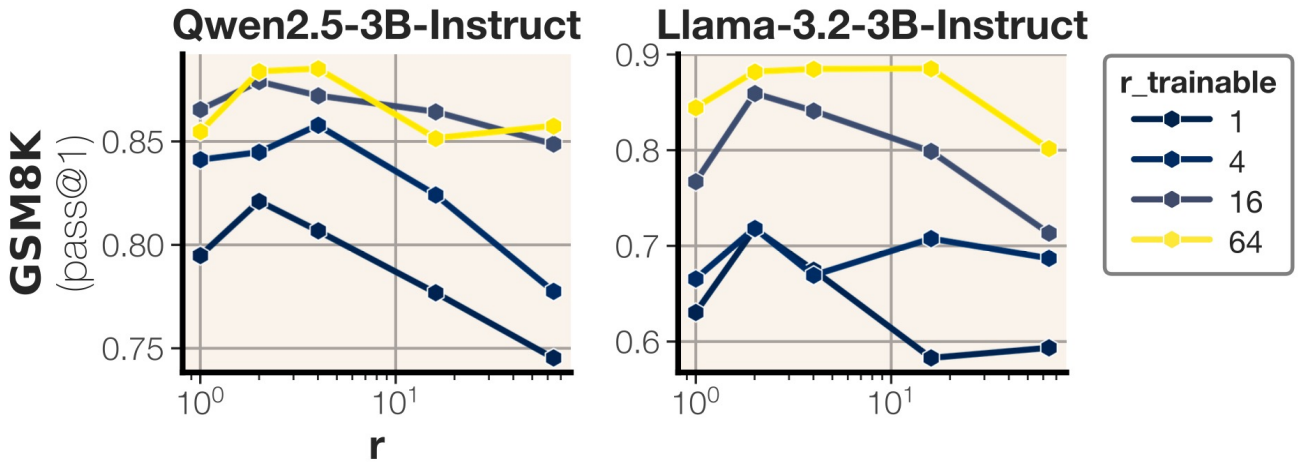



**Figure 7** TinyLoRA performance across frozen rank $r$ and trainable rank $r_{trainable}$.

Truncate SVD in rank 2

$$\boldsymbol{U\Sigma}(\sum_{i=1}^{u} \boldsymbol{v_i}\, \boldsymbol{P_i})\boldsymbol{V^T}$$

$(\sum_{i=1}^{u} \boldsymbol{v_i}\, \boldsymbol{P_i})$ becomes 2*2 matrix

"We hypothesize that higher ranks introduce more degrees of freedom in the frozen U, Σ, V components, making optimization of the small trainable vector v more difficult"