Parameter-Efficient Reinforcement Learning using Prefix Optimization

Itamar Rocha Filho Rosie Zhao Sham M. Kakade Eran Malach† Samy Jelassi†

Harvard University **Kempner Institute**

Published at ICLR 2026

Code: github.com/ItamarRocha/Prefix-RL

Motivation: What does RLVR actually do?

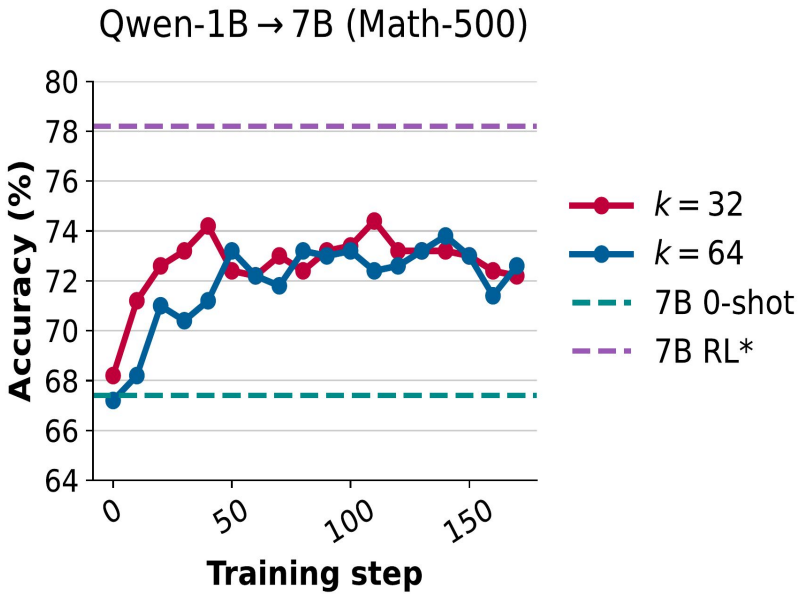
Does RL improve reasoning, or just steer the model toward formats that already work?

Key finding (Zhao et al., 2025):

RL post-training ***sharpens*** the output distribution — concentrating on one solution style that already had the highest accuracy.

The first few tokens determine the strategy:

```
"## Step 1: To find..." → step-by-step
"def solve():" → code solution
```



If RL is mainly selecting the right strategy,
we should capture most gains by optimizing only the prefix of the answer.

Prefix-RL: Our Approach

Prefix-RL

- Small ~1B adapter generates prefix (first k tokens)
- Large frozen target model completes the solution
- Train adapter with PPO + verifiable rewards
- Input-conditional — adapts prefix per question

Learned, parameter-efficient

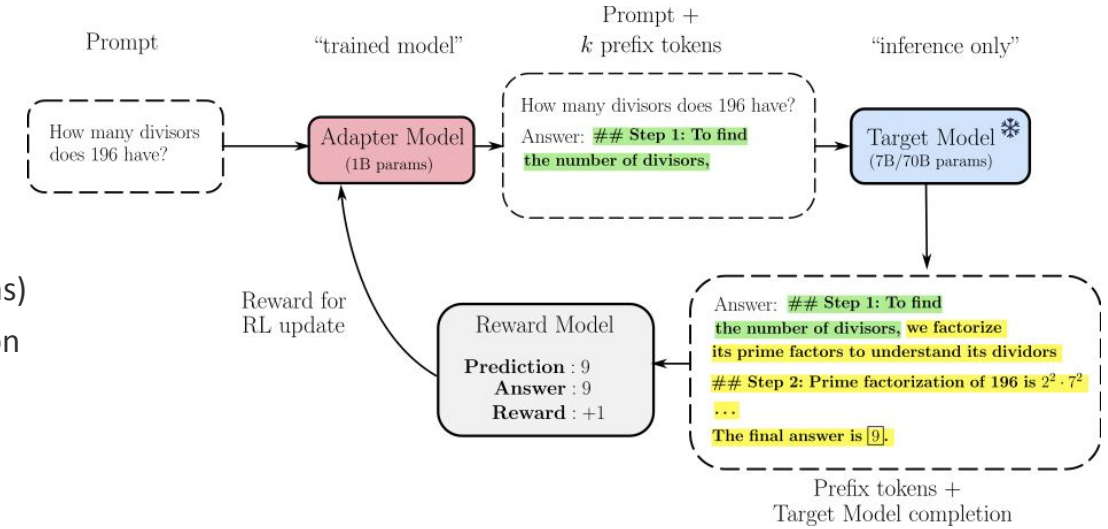


Figure 1: Prefix-RL architecture. The adapter generates k prefix tokens; the frozen target completes the solution.

Main Results

Target Model	Math-500	AIME	AMC23	Minerva
Qwen-7B	67.4	23.0	40.3	19.1
+Prefix-RL (k=32)	74.4 (+7.0)	25.8 (+2.8)	50.0 (+9.7)	22.4 (+3.3)
Llama-70B-FP8	62.0	32.8	45.0	29.0
+Prefix-RL (k=32)	67.8 (+5.8)	49.1 (+16.3)	46.2 (+1.2)	34.6 (+5.5)

Consistent improvements across model families (Qwen, Llama), scales (7B–72B), and quantization (FP8).

Compute Efficiency

Prefix-RL shifts the compute burden from training to inference:

Standard RL

- Train all 72B parameters
- Backprop through full model
- 32 GPUs required for 70B model

Prefix-RL

- Train only the ~1.5B adapter
- Target model in inference-only mode
- Only 8 GPUs (4 train + 4 serve)

Method	Training Compute	Inference Compute
Standard RL	$C_{\text{train}} N_t R T$	$C_{\text{inf}} N_t R T$
Prefix-RL (k)	$C_{\text{train}} N_a R k$	$C_{\text{inf}} R(N_t T + N_a k)$

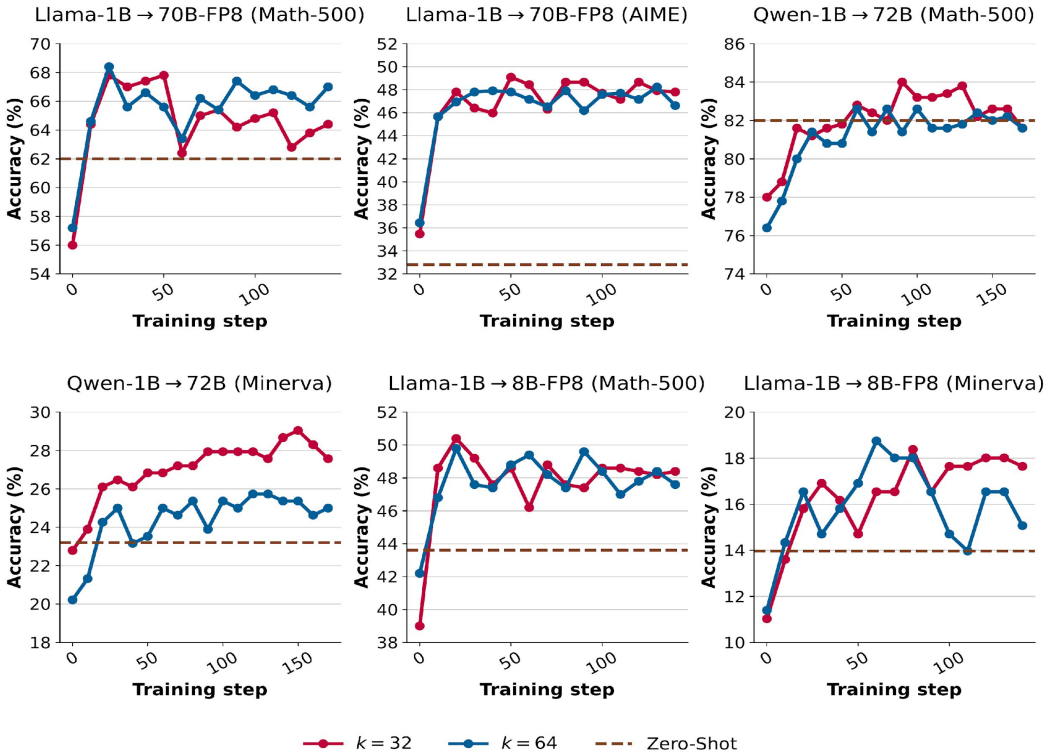
$N_t =$ # Params. of target model, $N_a =$ # Params. of adapter model
$R =$ # Rollouts, $T =$ # Tokens per rollout.
$C_{\text{train}} =$ Train compute constant, $C_{\text{inf}} =$ Inference compute constant.

Comparison in terms of FLOPs between Standard RL finetuning and Prefix-RL with prefix of length k.

Training FLOPs reduced by ~3,000× • GPU usage reduced by 4× (8 vs 32 for 70B)

- No catastrophic forgetting — target model weights unchanged
- Works on quantized (FP8) models
- Requires only inference access to the target model

Robustness & Generalization



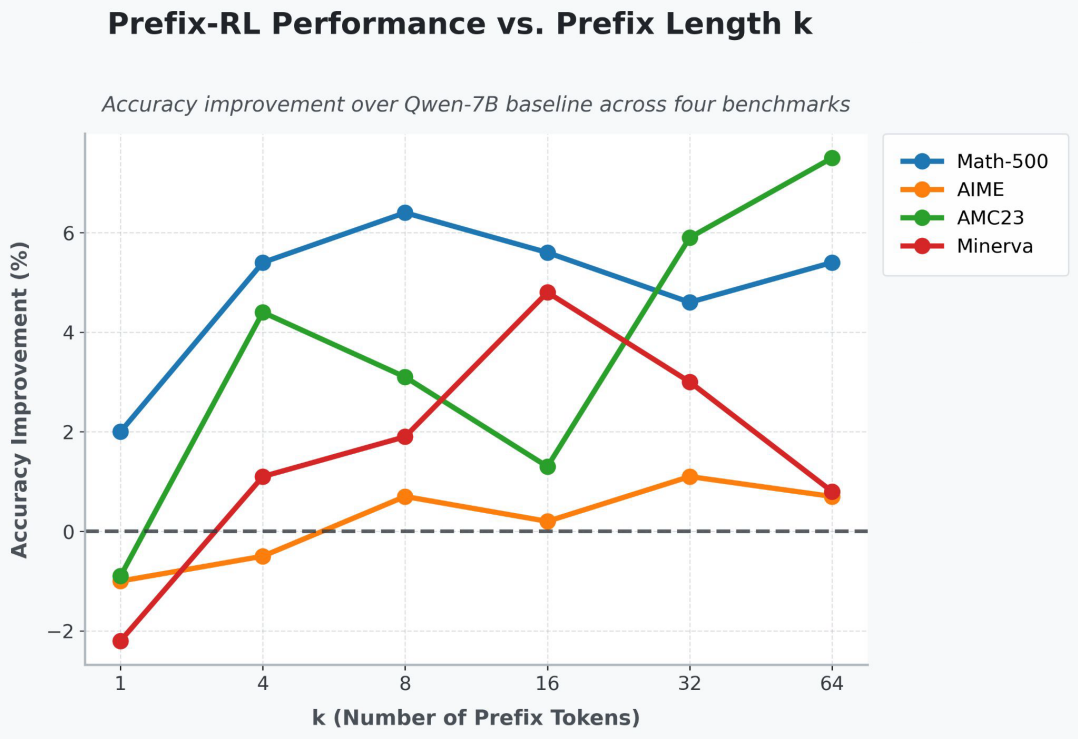
Works on quantized FP8 models
+16.3 on AIME for Llama-70B-FP8

Robust across random seeds
Effect sizes 2–25× larger than std dev

Transfers to physics tasks
OCW Courses & UGPhysics benchmarks

Outperforms LoRA & Prefix-Tuning
Larger and more consistent gains

Sensitivity to Prefix Length k



Moderate prefix lengths (k ≈ 8–32) consistently improve all benchmarks. Not sensitive to the exact choice of k — selecting k = 32 works well across models and tasks.

Summary

1. Much of RL's gain on math comes from upweighting good solution strategies.

2. Optimizing only the first k tokens (the prefix) is enough to capture a large fraction of these gains.

3. Prefix-RL uses a small ~1B adapter to steer a large frozen target model, achieving significant improvements at a fraction of the compute.

4. Results are consistent across model families (Qwen, Llama), scales (7B–72B), quantization (FP8), and even transfer to physics.

Code: github.com/ItamarRocha/Prefix-RL itamarfilho@g.harvard.edu