
Learning to Teach for Test-Time Reasoning

Mete Erdogan, Selim Emir Can
Department of Electrical Engineering
Stanford University
{merdogan, selimcan}@stanford.edu

Extended Abstract

Reinforcement learning post-training of small language models on the Countdown arithmetic reasoning task produces a competent student whose weights are then frozen. Within that no-student-update regime, the only remaining optimization surface is the textual context the student conditions on at inference. We study whether a second, trainable model, the *prompt-generating teacher*, can be optimized end-to-end so that its outputs improve the frozen student’s pass rate. We evaluate three teacher methodologies and compare them against non-teacher contextual baselines (random and KNN-retrieved demonstrations, manually engineered prefixes).

Methods. **Free-Form Prompt RL** trains a small (0.5B) teacher with REINFORCE-with-leave-one-out (RLOO) advantage to emit a single global system prompt. **Per-Problem Prompt RL** conditions the same teacher on the specific puzzle so that each query receives a tailored hint. **Reflective Evolutionary Prompt Search** adapts the gradient-free reflective prompt evolution introduced by GEPA [Agrawal et al., 2025], in which a strong mutator language model rewrites candidate prompts in light of recent student failures and a Pareto frontier is maintained. We also explore a routed variant of this loop in which the population is partitioned by problem cluster.

Findings. All teacher methods improve over the no-teacher RLOO baseline of 0.554 test pass@1 by +1 to +3 percentage points. Free-Form Prompt RL is the strongest at 0.585 ± 0.005 (+3.17 pp) averaged across three sampled tips at the final checkpoint. Per-Problem Prompt RL reaches 0.577 ± 0.005 (+2.34 pp) averaged across three late-training eval steps of a single 60-step run. Plain reflective evolutionary search reaches 0.569 (+1.51 pp), and its routed variant attains 0.560 ± 0.015 (+0.6 pp), underperforming the unrouted version and indicating that hand-written routing is a net negative on this task. Two ablations isolate the dominant design factors: RL training contributes a clear +2.7 pp lift over the matched untrained-teacher baseline, while per-problem conditioning slightly hurts (−0.8 pp) because the small teacher emits arithmetically wrong observations about specific operands. In the non-teacher study, a Claude-authored strategy prefix with no worked examples reaches 0.564 pass@1 (+1.0 pp), and every configuration with worked examples underperforms RLOO.

Interpretation. An action-space ceiling near +3 to +5 pp above zero-shot governs every prompt-only intervention we tested, set by the frozen student’s prompt-conditional response surface rather than by the teacher’s capacity. Prompt-teacher RL does extract a +2 to +3 pp lift, comparable to careful manual prompt engineering, but does not break through the ceiling that the student itself imposes.

Learning to Teach for Test-Time Reasoning

Mete Erdogan, Selim Emir Can
Department of Electrical Engineering
Stanford University
{merdogan, selimcan}@stanford.edu

Abstract

Once a reasoning model has been fine-tuned and its weights are frozen, the only remaining lever on accuracy is the textual context it conditions on at inference. Yet it is poorly understood whether the prompt itself can be *learned*, and if so, how much accuracy is recoverable that way versus by further updating the student. We formalize this question as *teacher learning for test-time reasoning*: a second, trainable model emits the system prompt fed to a fixed student, and its parameters are optimized end-to-end against the student’s downstream pass rate. Using a Countdown student trained with SFT followed by RLOO as a representative frozen reasoner, we instantiate two trainable teachers along a prompt-scope axis: a global **Free-Form Prompt RL** teacher and a **Per-Problem Prompt RL** teacher conditioned on each puzzle. We then test whether RL training of a prompt-generating teacher actually buys anything by comparing it directly against evolutionary prompt-optimization baselines (reflective evolutionary prompt search, plain and routed variants) and against manually engineered prefixes. All teacher methods lift pass@1 above the no-teacher RLOO baseline, with **Free-Form Prompt RL** strongest at +3.1 pp. Ablations attribute most of this lift to RL training itself rather than to per-problem conditioning, which slightly hurts a small teacher because of unreliable arithmetic about specific operands. Across every method we tried, the achievable gain saturates within a narrow ~ 3 to 5 pp band, an *action-space ceiling* that appears to be a property of the frozen student’s prompt-conditional response surface rather than of any teacher’s search efficiency.

1 Introduction

1.1 Problem and motivation

The Countdown task asks a model to combine three or four integers with the four arithmetic operations to reach a target number, with each input used exactly once. The CS224R default project provides a frozen *student* model trained via supervised fine-tuning (SFT) followed by REINFORCE-with-leave-one-out (RLOO) policy gradient [Ahmadian et al., 2024], on cognitive-behavior expert traces from Gandhi et al. [2025]. Baseline test pass@1 on the held-out 50-problem set is 0.554, achieved by sampling $n = 16$ completions per problem at temperature 0.6. The central question of this extension is whether downstream pass@1 can be improved without modifying the student’s weights.

We refer to this setting as *test-time* optimization to emphasize that all of our compute is spent after the student has finished its training. The student parameters are frozen at checkpoint time and are never updated again. Every method we study, gradient-based RL on the teacher, gradient-free reflective evolution, manual prompt engineering, and retrieval, operates by choosing the textual context that is fed to the already-trained student when it is asked to solve a new problem. In this sense the prompt is the only test-time degree of freedom and “teaching at test time” means shaping that context so as to improve the student’s pass rate at inference, with no further weight updates to the student.

Within the no-student-update regime, the only optimization surface available is the prompt, that is, the text the student conditions on at inference time. In-context learning is famously fragile: the same model can produce dramatically different accuracy depending on which demonstrations are present [Liu et al., 2022, Lu et al., 2022, Min et al., 2022], and chain-of-thought scaffolding can unlock capabilities that flat input-output pairs cannot [Wei et al., 2022]. The natural mathematical framing is to treat the prompt itself as an action, and to optimize a policy over prompts against the student’s pass rate as a reward.

1.2 Proposed extension

We instantiate a second, trainable model, the **prompt teacher**, whose action is the system prompt fed to the student. The teacher is a Qwen-2.5-0.5B-Instruct backbone with a LoRA adapter; the student is the frozen RLOO checkpoint from the default project. We define and study three teacher methodologies along two design axes. The first axis is *prompt scope*: the teacher’s prompt may be *global* (problem-independent) or *per-problem* (conditioned on the specific puzzle’s numbers and target). A global prompt is cheaper at deployment, since one prompt serves all queries, while a per-problem prompt is theoretically more expressive. The second axis is the *training mechanism*: parameters may be updated by gradient-based RL (RLOO over teacher outputs) or by gradient-free evolutionary search (LLM mutator with Pareto acceptance).

The three concrete methods we instantiate are (a) **Free-Form Prompt RL** (global, gradient), (b) **Per-Problem Prompt RL** (per-problem, gradient), and (c) **Reflective Evolutionary Prompt Search** (gradient-free), with both an unrouted and a routed variant. We compare these against each other and against a set of non-teacher contextual baselines (random demonstrations, structured KNN retrieval, manually engineered prefixes) developed in parallel.

1.3 Contributions

- We formalize three prompt-generating teacher methodologies (Free-Form RL, Per-Problem RL, Reflective Evolutionary Search) along the prompt-scope by training-mechanism design space, and provide a common evaluation protocol against a frozen RLOO student (Sections 3 and 3.5).
- We report the first multi-seed measurement of reflective evolutionary prompt search in this setting, including six independent seeds of a routed variant. The unrouted (plain) evolutionary loop reaches 0.569 test pass@1 (+1.51 pp) and the routed variant reaches 0.560 ± 0.015 (+0.6 pp on the deployable routed metric), establishing that hand-written routing is a net negative on this task (Section 5).
- We isolate the contribution of RL training (+2.7 pp over the matched untrained-teacher baseline) and of per-problem conditioning (−0.8 pp net effect on a 0.5B teacher) via two ablations on the Free-Form versus untrained and Free-Form versus Per-Problem axes (Section 5.1.2).
- We characterize an action-space ceiling at $\sim +3$ to $+5$ pp above the RLOO zero-shot baseline that bounds all teacher methods. The ceiling is consistent across gradient-based RL, evolutionary search, and manual prompt engineering, suggesting that it is a property of the frozen student’s prompt-conditional response surface rather than of any method’s search efficiency (Section 6).
- In parallel, we run a comprehensive non-teacher contextual baseline study and find that worked examples are the load-bearing source of harm to the RLOO student. Strategy-only prefixes help (+1.0 pp); worked-example prefixes hurt (−3.6 pp) (Section 3.6).

2 Background and Related Work

In-context learning and demonstration selection. In-context learning is famously sensitive to demonstration choice, ordering, and correctness [Liu et al., 2022, Lu et al., 2022, Min et al., 2022, Brown et al., 2020, Dong et al., 2023], and chain-of-thought prompting [Wei et al., 2022] unlocks behaviors that flat input-output pairs cannot. Existing demonstration selection methods treat the problem as retrieval over a fixed corpus [Liu et al., 2022, Rubin et al., 2022], optimizing a proxy (embedding similarity, likelihood) rather than the student’s downstream accuracy. Our teacher methods optimize the latter directly.

RL and reflective evolution for prompt generation. DPO [Rafailov et al., 2023] and GRPO [Shao et al., 2024] show that generation policies can be trained from scalar rewards; Ahmadian et al. [2024] recommend RLOO for variance reduction, which is the gradient estimator we reuse for both teacher RL methods. Agrawal et al. [2025] introduce GEPA, a reflective-mutation prompt search guided by explicit failure-trace feedback; our evolutionary method adapts that machinery to the prompt-generating teacher setting.

Synthetic data, curricula, and learning-to-teach. A large literature uses other LMs to synthesize training data (Self-Instruct [Wang et al., 2022], Evol-Instruct [Xu et al., 2023], NOMAD [Chen et al., 2024], Synthetic Continued Pretraining [Yang et al., 2025], STaR [Zelikman et al., 2022]) or to schedule it [Bengio et al., 2009, Graves et al., 2017, Wang et al., 2025, Zhang et al., 2025], and Learning-to-Teach [Fan et al., 2018] casts teaching as an RL problem. All of these operate in the parametric regime: the teacher’s output is back-propagated into the student’s weights. Our work shifts the analogous question into the *contextual* regime: the teacher’s output is read by the frozen student through its context window, not into its parameters. Cognitive-behavior scaffolds [Gandhi et al., 2025] similarly work at training time by selecting expert traces with verification, backtracking, subgoal setting, and backward chaining; our teachers extend that idea to inference time.

Gap. To our knowledge no prior work directly compares (a) on-policy RL over a teacher’s prompt-generation policy with (b) reflective evolutionary prompt search [Agrawal et al., 2025], against a common frozen student and a common set of non-teacher baselines. That comparison is the contribution.

3 Methodology

3.1 Default project setup

We follow the default-project pipeline of Ahmadian et al. [2024], Gandhi et al. [2025], and Pan et al. [2025]. A Qwen-2.5-0.5B backbone is first SFT-trained on the `Asap7772/cog_behav_all_strategies` expert traces, and then RLOO-trained with a rule-based Countdown verifier reward. The resulting RLOO checkpoint, with $\text{pass}@1 = 0.554$ and $\text{pass}@16 = 0.720$ on the standard 50-problem test split with $n = 16$ samples per problem, serves as the frozen student for all teacher experiments. Held-out evaluation always uses temperature 0.6, $\text{top-}p = 0.95$, and $\text{top-}k = 20$.

The reward function $R(x, y)$ is binary: it equals 1 if the student’s parsed equation y evaluates to the target with each operand used exactly once, and 0 otherwise.

Borderline filtering of the training pool. Because the reward is binary and the student is already strong, most randomly sampled problems give either reward $\rightarrow 1$ or reward $\rightarrow 0$ across all teacher outputs, producing a near-zero RLOO advantage. We therefore prefilter the training pool once, keeping only problems whose unconditional pass rate (estimated with $K_{\text{base}} = 4$ rollouts of the frozen student) lies in $\hat{p} \in [0.125, 0.875]$. This leaves ~ 256 borderline problems out of 2000–3000 candidates and is the source of the “training reward” axis in our later figures; held-out evaluation always uses the unfiltered 50-problem test split.

Student-side baselines we reuse. Three student-side training objectives appear in our experiments: SFT on the expert traces of Gandhi et al. [2025] (the warm-start), IPO [Azar et al., 2024] on verifier-derived preference pairs from the SFT student, and RLOO [Ahmadian et al., 2024] on the same problems with the binary verifier reward $R(x, y)$. RLOO produces the frozen checkpoint that every teacher then prompts. The RLOO advantage estimator

$$A(x, y_i) = R(x, y_i) - \frac{1}{G-1} \sum_{j \neq i} R(x, y_j), \tag{1}$$

is the gradient backbone we re-use for both **Free-Form Prompt RL** and **Per-Problem Prompt RL**, where each completion y_i is replaced by a teacher-emitted system prompt τ_i , and the reward is replaced by the student’s pass rate when prompted with τ_i . The SFT loss is the standard token-level negative log likelihood on the expert traces; the IPO loss is the squared log-odds margin of Azar et al. [2024].

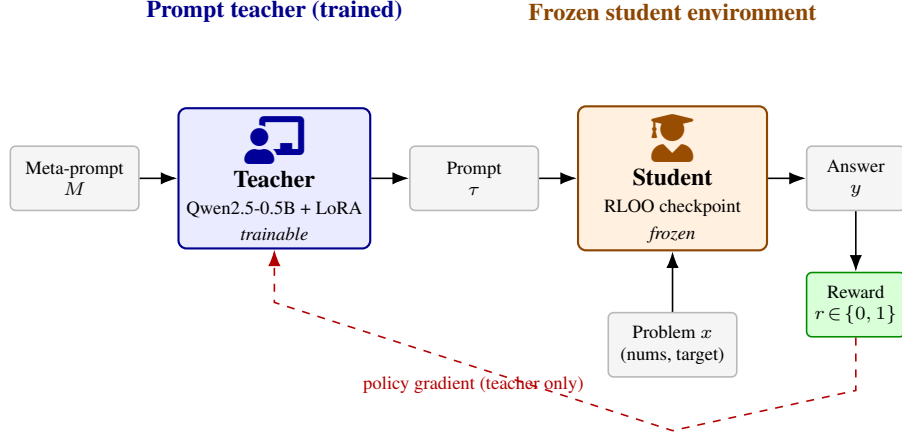


Figure 1: Teacher-student setup. The teacher reads a meta-prompt M and emits a system prompt τ that is prepended to the student’s input alongside the problem x . The student is frozen; the verifier reward r updates only the teacher’s LoRA adapter via policy gradient (red dashed path).

3.2 Teacher-student architecture

Figure 1 shows the data flow. The teacher reads a meta-prompt M (either global or problem-parameterized) and emits a system prompt $\tau \sim \pi_\theta(\cdot | M)$. The student then conditions on (τ, x) and emits an equation. The verifier scores the equation, and the resulting reward is used to update only the teacher’s LoRA adapter via policy gradient. The student is never modified.

3.3 Free-Form Prompt RL

In **Free-Form Prompt RL** the teacher samples a system prompt $\tau \sim \pi_\theta(\cdot | M)$ (a short natural-language instruction we call a *tip*) that does not depend on any specific problem. At each step a batch \mathcal{B} of B problems is drawn and each of G sampled prompts is evaluated against the same batch. Let r_g denote the mean reward across the B batch problems under prompt τ_g . The RLOO advantage for prompt g is

$$A_g = r_g - \frac{1}{G-1} \sum_{g' \neq g} r_{g'}, \quad (2)$$

and is then normalized to unit standard deviation across g . The policy gradient is computed at the sequence level (sum of token log probabilities) with a KL anchor to the un-LoRA’d base policy and an entropy bonus on the prompt tokens.

The full per-step procedure is given in Algorithm 1.

Algorithm 1 Free-Form Prompt RL (one gradient step)

- 1: Sample batch \mathcal{B} of B problems from the borderline pool.
 - 2: Sample G prompts $\{\tau_g\}_{g=1}^G \sim \pi_\theta(\cdot | M)$.
 - 3: **for** $g = 1, \dots, G$ **do**
 - 4: Generate one student rollout per problem under prompt τ_g .
 - 5: $r_g \leftarrow$ mean binary pass rate over the B rollouts.
 - 6: **end for**
 - 7: Compute RLOO advantages A_g via Eq. (2), normalize across g .
 - 8: Update θ by $-\sum_g A_g \nabla_\theta \log \pi_\theta(\tau_g) + \beta_{\text{KL}} \text{KL}(\pi_\theta \| \pi_{\text{ref}}) - \beta_{\text{ent}} \mathcal{H}(\pi_\theta)$.
-

At deployment we sample $N = 3$ prompts from the trained teacher and report the mean held-out pass@1 across them. The standard deviation across the N sampled prompts at the final checkpoint serves as our uncertainty estimate.

3.4 Per-Problem Prompt RL

In **Per-Problem Prompt RL** the teacher’s meta-prompt M_b is parameterized by the specific puzzle b so that it embeds the problem’s numbers and target and the teacher can produce a problem-tailored hint. The RLOO baseline is then computed within each problem’s group of G' sampled prompts:

$$A_{b,g} = r_{b,g} - \frac{1}{G' - 1} \sum_{g' \neq g} r_{b,g'}, \quad (3)$$

where $r_{b,g}$ is the mean pass rate of the student over K rollouts on problem b under prompt $\tau_{b,g}$, and the advantages are normalized to unit standard deviation per problem.

The full per-step procedure is given in Algorithm 2.

Algorithm 2 Per-Problem Prompt RL (one gradient step)

- 1: Sample batch \mathcal{B} of B problems from the borderline pool.
 - 2: **for** $b \in \mathcal{B}$ **do**
 - 3: Construct M_b embedding problem b ’s numbers and target.
 - 4: Sample G' prompts $\{\tau_{b,g}\}_{g=1}^{G'} \sim \pi_\theta(\cdot \mid M_b)$.
 - 5: **for** $g = 1, \dots, G'$ **do**
 - 6: Generate K student rollouts on problem b under prompt $\tau_{b,g}$.
 - 7: $r_{b,g} \leftarrow$ mean pass rate over the K rollouts.
 - 8: **end for**
 - 9: Compute per-problem RLOO advantages $A_{b,g}$ via Eq. (3), normalize across g .
 - 10: **end for**
 - 11: Update θ as in Algorithm 1 with per-problem advantages.
-

A global teacher is upper-bounded by the best fixed prompt for the task, whereas per-problem conditioning lifts this ceiling in principle because different problems can benefit from different strategies. However, a small (0.5B) teacher may not reliably reason about specific operand values: the untrained question-conditioned baseline already underperforms the untrained global baseline (0.554 vs. 0.569), and RL training must recover from a worse starting point.

3.5 Reflective Evolutionary Prompt Search

The third method abandons gradient-based training entirely. A population of candidate prompts is maintained and an LLM mutator (Qwen-2.5-7B-Instruct) proposes new candidates by inspecting recent student failure traces, in the reflective-mutation style introduced by GEPA [Agrawal et al., 2025].

Pareto frontier over evaluation cells. Following GEPA [Agrawal et al., 2025], we treat the per-problem reward vector as the fitness signal rather than its scalar mean. Concretely, each accepted candidate prompt τ is associated with a fitness vector $\mathbf{r}(\tau) = (r(\tau, x_1), r(\tau, x_2), \dots, r(\tau, x_M)) \in [0, 1]^M$, where x_1, \dots, x_M are the borderline evaluation problems (the “cells”) that τ has been scored on. A prompt τ is said to *Pareto-dominate* another prompt τ' if $r(\tau, x_m) \geq r(\tau', x_m)$ on every shared cell m and strictly exceeds it on at least one. The Pareto frontier is then the set of non-dominated prompts, that is, the prompts for which no other prompt in the population is at least as good on every cell and strictly better on some. At each iteration we sample a parent from this frontier uniformly, propose a child via reflective mutation, evaluate the child on a fresh minibatch of cells, and add the child to the population only if the updated frontier still contains it (i.e. it is not dominated by any existing member). Using the full reward vector rather than the scalar mean prevents mode collapse to a single “average” prompt and preserves prompts that are specialists on hard subsets, exactly the behavior reported in Agrawal et al. [2025].

The per-iteration procedure is given in Algorithm 3 in Appendix A.

Plain variant. The plain variant maintains a single global population and reports the single best prompt found across the full search. This is the closest analog of GEPA in our setting.

Routed variant. The routed variant partitions the problem space into $C = 5$ clusters by operand count and target magnitude (for example, “3 numbers, target ≤ 30 ”), maintains a separate Pareto frontier per cluster, and uses a hand-written router at test time to select the cluster-specialized prompt for each problem. The per-iteration procedure is given in Algorithm 4 in Appendix A. We report two test-time metrics for the routed variant: *routed*, in which each problem is answered using its cluster’s current best prompt.

3.6 Non-teacher contextual baselines

In parallel, we evaluate non-teacher methods that operate in the same prompt-only regime but use no trainable teacher. A pool of 404 correctly-solved Countdown demonstrations is built by filtering the SFT warmstart corpus on the verifier; three retrievers consume the pool: random sampling (uniform, stratified by operand count), structured-feature KNN (operand count, target, sum, max, min), and a manually engineered Claude-authored prefix (Opus 4.7, one-shot setup with no per-query generation). Each retriever produces $k \in \{1, 2, 4, 8\}$ demonstrations that are spliced into the student’s prompt in the same position as the teacher’s output. These baselines establish the prompt-engineering ceiling that the trained teacher methods must clear.

4 Experimental Setup

Both RL methods train for 60 gradient steps. For Per-Problem we evaluate held-out pass@1 at periodic checkpoints (every 5 steps) and report mean \pm SD across three eval steps; the full curve is shown in Figure 4 in Appendix B. The evolutionary method runs for 60 accepted mutations in its plain variant and for 60 accepted mutations spread across 5 clusters in its routed variant. Total student inference is $\sim 250k$ completions per run for the RL methods and comparable for the evolutionary search. Held-out evaluation uses 50 test problems with 16 student samples per prompt; reported numbers are means and standard deviations across multiple deployable trajectories. The full hyperparameter table is in Appendix D.

5 Results

5.1 Quantitative evaluation

5.1.1 Main results

Table 1 compares the three teacher methods against the standard student-side training baselines (SFT, IPO, RLOO) and against an untrained 0.5B teacher (step-1 of the Free-Form run, before any RL update). All teacher methods clear the no-teacher RLOO baseline. **Free-Form Prompt RL** is the strongest at 0.585 ± 0.005 (+3.1 pp). **Per-Problem Prompt RL** reaches 0.577 ± 0.005 (+2.3 pp). Plain reflective evolutionary search reaches 0.569 (+1.5 pp), and its routed variant reaches 0.560 ± 0.015 (+0.6 pp).

Headline visualization and pass@k shape. Figure 2 places every method on a common axis: the shaded band marks the action-space ceiling region between the RLOO zero-shot baseline and the strongest trained teacher (Free-Form Prompt RL). Every prompt-only method that improves over RLOO at all lands inside this narrow ~ 3 pp band; the methods that fall outside are either non-RL student-side baselines (SFT, IPO) or worked-example retrieval. The pass@k shape in Figure 3 is the signature of narrowing the output distribution toward correctness: every teacher method gains +1 to +3 pp on pass@1 while pass@16 roughly matches the RLOO baseline, in contrast to IPO which buys high pass@16 at the cost of pass@1.

5.1.2 Ablations

Table 2 reports two ablations that isolate specific design choices of the teacher methods.

(A1) Prompt scope. Per-problem conditioning is theoretically more expressive than a global prompt, but on a 0.5B teacher per-problem underperforms by 0.8 pp (0.577 ± 0.005 versus Free-Form’s 0.585 ± 0.005). Inspection of the per-problem teacher’s outputs (Appendix F) shows that the

Table 1: Test pass@ k on the RLOO student (50 problems, 16 student samples per prompt). Teacher rows report mean \pm SD across $n=3$ independent trials per method (Plain Reflective Evolution has only one available trial; student-side baselines are a single canonical eval and are reported without SD). Δ_1 is the pass@1 delta versus the no-teacher RLOO baseline of 0.554.

Method	pass@1 (mean \pm SD)	pass@4	pass@16	# Trials	Δ_1
<i>Student-side training baselines (no teacher):</i>					
SFT student	0.324	—	0.740	—	-23.0 pp
IPO student	0.386	0.677	0.800	—	-16.7 pp
RLOO student	0.554	0.690	0.720	—	0 (baseline)
<i>Untrained-teacher reference:</i>					
0.5B teacher, no RL (step-1 eval)	0.558 \pm 0.001	0.708 \pm 0.007	0.767 \pm 0.012	3	+0.4 pp
<i>This work's teacher methods (0.5B teacher):</i>					
Free-Form Prompt RL	0.585 \pm 0.005	0.707 \pm 0.010	0.747 \pm 0.012	3	+3.1 pp
Per-Problem Prompt RL	0.577 \pm 0.005	0.698 \pm 0.011	0.720 \pm 0.020	3	+2.3 pp
Plain Reflective Evolution	0.569	0.700	0.740	1	+1.5 pp
Routed Reflective Evolution	0.560 \pm 0.015	0.698 \pm 0.012	0.753 \pm 0.012	3	+0.6 pp

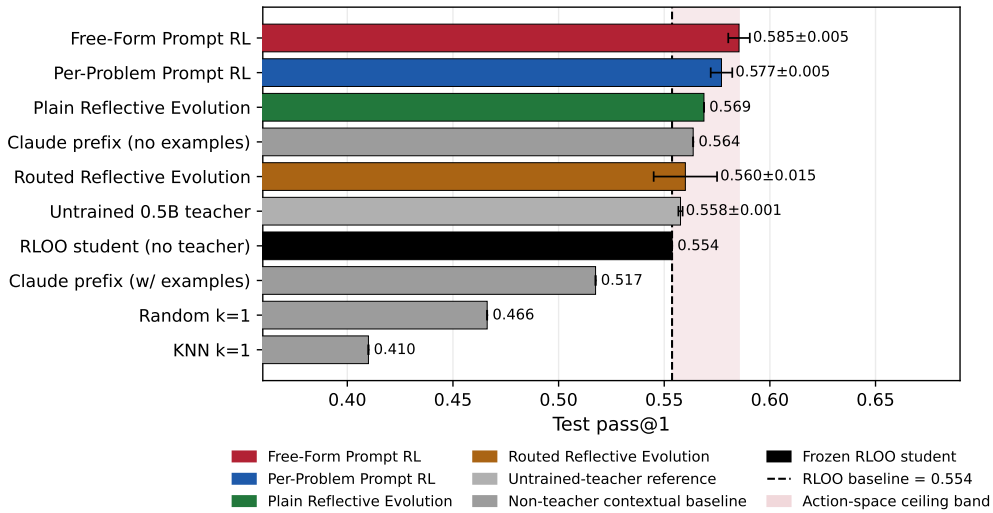


Figure 2: Headline comparison of every method against the frozen RLOO student. Bars sorted by test pass@1; error bars are mean \pm SD where multiple trajectories are available. Shaded band: action-space ceiling between the RLOO baseline (0.554) and **Free-Form Prompt RL** (0.585). Every prompt-only method that beats RLOO clusters inside this band.

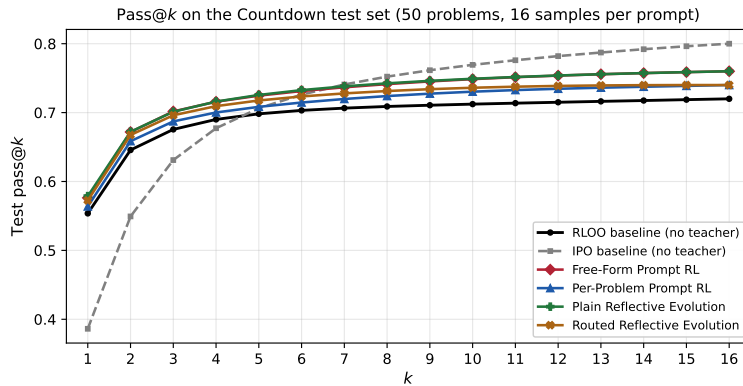


Figure 3: Pass@ k on the held-out test set. All methods are plotted across $k \in \{1, \dots, 16\}$.

Table 2: Ablations on the 0.5B teacher methods. Each row pair contrasts two configurations that differ in one design choice; the last column gives the effect of that choice on test pass@1.

Ablation	Configuration	pass@1 (mean \pm SD)	Δ_1	Effect
(A1) Prompt scope	Global (Free-Form)	0.585 \pm 0.005	+3.1 pp	
	Per-Problem	0.577 \pm 0.005	+2.3 pp	-0.8 pp
(A2) RL training	Untrained teacher (no RL)	0.558 \pm 0.001	+0.4 pp	
	RL-trained (Free-Form)	0.585 \pm 0.005	+3.1 pp	+2.7 pp

model frequently emits arithmetically wrong observations about specific operands (e.g., “50/8 is an integer”), which caps how much the extra flexibility can buy at this teacher scale.

(A2) RL training. Comparing the RL-trained 0.5B teacher with the same untrained teacher shows a +2.7 point benefit. The lift over the untrained-teacher reference is reproducible across the sampled tips at convergence.

Training and held-out dynamics. Free-Form RL reaches its final reward band in two gradient steps and then fluctuates inside it through step 60; Per-Problem RL trains for the same 60-step budget but its held-out pass@1 is noisy across checkpoints, with a clear peak at step 30 (0.585) and degradation thereafter. Both methods clear the untrained 0.5B teacher reference (0.558). Full curves (Figure 4), training-loss diagnostics, per-trajectory spreads, and per-seed reliability of the routed search are deferred to Appendix B.

5.1.3 Non-teacher contextual baselines

For comparison, Table 3 reports the strongest results from the non-teacher contextual baseline study run in parallel. These methods are meant first to show that basic prompting do not work as good as the trained teacher, and secondly to get insight on how to design the meta prompt given to the teacher. The strategy-only Claude-authored prefix (no worked examples) is the first non-teacher configuration we measured to beat the no-teacher RLOO baseline, at 0.564 pass@1 (+1.0 pp). Every configuration containing worked examples (random retrieval, KNN retrieval, Claude with examples) underperforms the RLOO baseline.

Table 3: Strongest non-teacher contextual baselines on the RLOO student. The strategy-only Claude prefix beats RLOO; every configuration with worked examples underperforms it.

Configuration	pass@1	Δ_1 vs RLOO
Claude scaffolds (no worked examples)	0.564	+1.0 pp
Zero-shot RLOO (reference)	0.554	—
Claude-authored prefix (with worked examples)	0.518	-3.6 pp
Random $k=1$ (best of three seeds)	0.466	-8.8 pp
KNN-Struct $k=1$	0.410	-14.4 pp

Worked examples are the load-bearing source of harm. The strategy-only Claude prefix and the Claude-with-examples prefix are identical in intent (same task spec sent to the same Claude version) and differ only in whether two worked Countdown solutions are appended to the prefix. Removing the examples flips the sign of the few-shot effect from -3.6 pp to +1.0 pp. The conclusion is reinforced by a CoT-strip ablation on random and KNN retrievers: stripping the chain-of-thought trace from each demonstration worsens pass@1 in three of four cells, ruling out the reasoning trace as the source of harm. The demonstration’s mere presence shifts the student’s input distribution away from the RLOO training mode.

5.2 Qualitative analysis

Beyond the headline numbers, three qualitative patterns in the teachers’ outputs help explain the cross-method clustering observed in Section 5.1. Verbatim tip samples backing each observation appear in Appendix F.

Free-Form RL converges to a generic “identify, prioritize, verify” template. At the final checkpoint, every Free-Form tip we sampled follows the same three-beat structure: (i) identify a relationship between the operands and the target, (ii) prioritize one specific operation family (most often multiplication, sometimes addition for small targets), and (iii) verify each operand is used exactly once before answering. The deployed tips contain no problem-specific arithmetic and no worked examples; they read like a generic strategy scaffold one would write by hand. The Claude-authored scaffold that beats RLOO (+1.0 pp, Section 3.6) follows the same abstract template. The RL-trained tip outperforms it by another +2 pp, but the qualitative shape of what RL learns is the same as what careful manual prompt engineering produces: a high-level strategy scaffold, not fine-grained per-problem reasoning.

Per-Problem RL emits arithmetically incorrect observations about specific operands. The same trained backbone, when given a meta-prompt parameterized by the puzzle, frequently produces operand-level confusions in the tips it writes. On the puzzle ($\text{nums} = [44, 19, 35]$, $\text{target} = 98$), for example, the teacher both restates the objective as “obtain the number $[44, 19, 35]$ ” (conflating the input list with the target) and suggests “multiplying them together”, though $44 \cdot 19 \cdot 35 = 29,260 \gg 98$. The student then conditions on a hint that points it in the wrong direction. This failure mode is the qualitative origin of the -0.8 pp prompt-scope effect (ablation A1, Section 5.1.2): the extra flexibility of per-problem conditioning is paid for by a small teacher’s unreliable arithmetic about specific operands.

Routed evolution collapses to the same prompt across unrelated clusters. The routed variant maintains five separate Pareto frontiers, one per problem cluster (operand count \times target magnitude). Inspection of the best prompt found per cluster shows that the mutator LLM converged to nearly identical multi-step, step-listing templates for two unrelated clusters (3-number-large-target and 4-number-large-target; see Appendix F), and to a single short imperative for two more. The hand-written router thereby distinguishes problems that the search itself does not. This is concrete qualitative evidence that the inductive bias offered by routing is too coarse to outweigh the multi-population variance penalty: the per-cluster optima the search converges to are not actually cluster-specific.

6 Discussion

All trained teachers cluster in 0.56 to 0.59. The full spread of teacher-method pass@1 results sits in $[0.558, 0.585]$. The untrained baseline at 0.558 falls inside this band. Two interpretations are consistent with the evidence.

The first is an *action-space ceiling*. The set of system prompts has finite informational capacity to influence the frozen student, and about +3 pp above RLOO zero-shot appears to be that ceiling. This is a property of the student’s prompt-conditional response surface, not of the teacher’s capacity. The cross-method consistency, with gradient-based RL (Free-Form, +3.07 pp), per-problem RL (+2.34 pp), plain evolutionary search (+1.51 pp), routed evolutionary search (+0.6 pp routed, +2.0 pp single-best), and manual prompt engineering (Claude scaffolds, +1.0 pp), all landing in the same band, is direct evidence: when different optimization methods all converge to the same result, the constraint is on the search space, not on any one method’s search efficiency.

The second is a *signal-to-noise floor on the gradient*. At $B = 256$ and $K = 1$, the per-prompt reward standard error from binomial sampling is ~ 2.9 pp. The observed cross-prompt standard deviation of rewards at step 1 is 1.8 pp, smaller than the noise floor. The RLOO gradient is therefore variance-bound: the policy update direction is partly noise. The Free-Form method’s clean +2.7 pp lift over its untrained baseline (ablation A2) is non-trivial given this constraint, and explains why training reward plateaus quickly while held-out pass@1 still slowly improves.

Deeper method-by-method analysis. Three follow-up questions are natural here: (i) *why does per-problem not beat free-form on a small teacher?* The 0.5B teacher emits arithmetically incorrect observations about specific operands often enough that the extra flexibility is wasted; the untrained per-problem baseline already underperforms the global one by 1.5 pp. (ii) *Why does longer Per-Problem training not help?* The 60-step Per-Problem run overfits the borderline pool, with held-out pass@1 peaking near step 30 and oscillating back down. (iii) *Why does routing hurt the evolutionary search?* The routed variant spreads ~ 60 accepted mutations across 5 clusters, giving only ~ 12

directed updates per cluster and multiplying seed-to-seed variance. Full numerical analysis of all three questions, together with a per-method computational-complexity summary, is deferred to Appendix C.

Limitations. The two RL methods use a single training seed, so reported SDs describe deployment-time variation across sampled tips, not seed-to-seed variation in training. The binary reward at $K = 1$ is the most variance-heavy shape possible; partial-credit reward or $K \geq 4$ averaging would densify the gradient. All methods optimize against a single frozen student, so jointly updating student and teacher (e.g. distilling teacher-prompted trajectories back into the student) could in principle break the ceiling. The 50-problem test set gives a single-seed standard error of ~ 1.7 pp; differences under 2 pp should be read as noise.

7 Conclusion

Three trainable prompt-generating teacher methodologies all improve over the no-teacher RLOO baseline of 0.5537 by +1 to +3 pp, with **Free-Form Prompt RL** strongest at 0.585 ± 0.005 (+3.17 pp). Ablations attribute most of this lift to RL training (+2.7 pp over a matched untrained teacher); per-problem conditioning slightly hurts a small teacher. A plain reflective evolutionary search reaches 0.569 on a single seed and itself beats the routed variant, and careful manual prompt engineering reaches 0.564, $\sim +1$ pp above RLOO. Across gradient-based RL, evolutionary search, and manual engineering, an action-space ceiling at $\sim +3$ to $+5$ pp above the no-teacher baseline bounds every prompt-only intervention. This appears to be a property of the frozen student’s prompt-conditional response surface, not of any teacher’s search efficiency. The natural next step is to break the no-student-update assumption: distill teacher-prompted trajectories back into the student, or train the two jointly.

Contributions

Default project (joint). Implementation of SFT, IPO, and RLOO on Qwen-2.5-0.5B following the default-project pipeline. Reproduction of the milestone baselines (SFT 0.324, IPO 0.386, RLOO 0.554 pass@1).

Metem Erdogan. Designed, implemented, and evaluated the three trainable prompt-generating teacher methodologies (**Free-Form Prompt RL**, **Per-Problem Prompt RL**, **Reflective Evolutionary Prompt Search** with both plain and routed variants). Wrote the teacher-student RL training loop, including RLOO advantage computation, sequence-level policy gradient with KL anchor and entropy bonus, the borderline-filter setup, and the held-out evaluation pipeline. Ran the six-seed evolutionary multi-seed study and the two ablations.

Selim Emir Can. Designed, implemented, and evaluated the non-teacher contextual baseline study, including the 404-demo correctness-filtered pool, the random and structured-KNN retrievers, the Claude-fixed prefix injection, the 24-configuration sweep across $k \in \{1, 2, 4, 8\}$, the CoT-strip ablation, the multi-seed random-retriever replicates, and the hybrid Claude + KNN composition. Maintained the SFT, RLOO, and IPO baselines used as the reference checkpoint throughout.

Adjustments from proposal. Allocations evolved as both parallel lines converged on a common action-space-ceiling result; both authors contributed to writing the discussion, conclusion, and final report layout. No reassignment of major deliverables occurred.

References

- Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziemis, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution outperforms reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.
- Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.
- Yoshua Bengio, Jérôme Louradour, Ralph Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Yifang Chen, David Zhu, Simon Du, Kevin Jamieson, and Yang Liu. Rethinking data synthesis: A teacher model training recipe with interpretation. *arXiv preprint arXiv:2410.20362*, 2024.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2023.
- Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations (ICLR)*, 2018.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.
- Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 2022.
- Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Arber, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.
- Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. 2025.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.
- Zhenting Wang, Guofeng Cui, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.
- Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candès, and Tatsunori Hashimoto. Synthetic continued pretraining. In *International Conference on Learning Representations (ICLR)*, 2025.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.
- Hengyuan Zhang, Shiping Yang, Xiao Liang, Chenming Shang, Yuxuan Jiang, Chaofan Tao, Jing Xiong, Hayden Kwok-Hay So, Ruobing Xie, Angel X. Chang, and Ngai Wong. Find your optimal teacher: Personalized data synthesis via router-guided multi-teacher distillation. *arXiv preprint arXiv:2510.10925*, 2025.

Appendix

A Algorithm boxes (Reflective Evolution)

Algorithm 3 Reflective Evolutionary Prompt Search (iteration t , plain variant)

- 1: Sample parent τ_{par} from the current Pareto frontier.
 - 2: Sample F recent student failures on problems under τ_{par} .
 - 3: Mutator LLM reads $(\tau_{\text{par}}, \text{failures})$ and proposes τ_{new} .
 - 4: Evaluate τ_{new} on a fresh minibatch of problems.
 - 5: If τ_{new} weakly dominates τ_{par} on shared cells, accept; else reject.
-

Algorithm 4 Routed Reflective Evolutionary Prompt Search (iteration t)

- 1: Sample cluster $c \in \{1, \dots, C\}$ uniformly.
 - 2: Sample parent τ_{par} from cluster c 's current Pareto frontier \mathcal{F}_c .
 - 3: Sample F recent student failures on cluster- c problems under τ_{par} .
 - 4: Mutator LLM reads $(\tau_{\text{par}}, \text{failures})$ and proposes τ_{new} .
 - 5: Evaluate τ_{new} on a fresh minibatch of cluster- c problems.
 - 6: If τ_{new} weakly dominates τ_{par} on the shared cluster- c cells, accept into \mathcal{F}_c ; else reject.
 - 7: **At test time:** route each test problem x to its cluster $c(x)$ via the hand-written router, and answer with the current best prompt of $\mathcal{F}_{c(x)}$.
-

B Training and held-out dynamics

Figure 4 gives a side-by-side comparison of the two RL teacher methods on both training reward and held-out test pass@1. Panel (a) reports mean training reward on the borderline batch (raw values with markers, with a 3-step rolling mean shown as the heavier line). Free-Form RL reaches its final band in two gradient steps and then fluctuates inside it through step 60; the binomial standard error of per-prompt reward at $B = 256$, $K = 1$, $p \approx 0.7$ is ~ 2.9 pp, on the same scale as the observed step-to-step variation, so the Free-Form curve is consistent with a flat trend after step 2. Per-Problem RL, in contrast, shows a modest upward drift in its rolling mean across the full 60-step budget.

Panel (b) reports the held-out test pass@1 at periodic eval checkpoints. Both methods comfortably clear the no-teacher RLOO baseline (dotted black) and the untrained 0.5B teacher reference (dashed gray, 0.558). Free-Form stabilizes around 0.585 at its final checkpoint; Per-Problem peaks near 0.585 at step 30 and oscillates back down to ~ 0.555 by step 50. We report mean \pm SD across three late-training eval steps of the Per-Problem run as its headline number in Table 1.

Additional diagnostic figures appear later in this appendix: training-loss components (policy-gradient, KL anchor, entropy bonus) in Figure 5, per-trajectory test pass@1 for the two RL teachers in Figure 6, and per-seed reliability of the routed evolutionary search in Figure 7. In brief, the policy-gradient loss oscillates around zero with no monotone trend, the KL term drifts slightly negative as the LoRA policy moves away from the reference, and no loss term diverges, indicating that $\beta_{\text{KL}} = 0.01$ is adequate. All deployable trajectories of both RL teachers land above the no-teacher RLOO baseline, with comparable spread (SD 0.005 for Free-Form, SD 0.005 for Per-Problem). For the routed evolutionary search, the single-best metric is positive on 6/6 seeds and the routed (deployable) metric is positive on 5/6 seeds, with one-sample t-tests against the RLOO baseline giving $p = 0.058$ (routed) and $p = 0.007$ (single-best); the plain unrouted variant on a single seed reaches 0.569, exceeding the multi-seed mean of the routed variant and indicating that the hand-written routing layer does not add value on this task.

C Extended discussion

Why per-problem does not beat free-form on a small teacher. A per-problem policy is strictly more expressive than a global one, but at 0.5B the teacher emits arithmetically incorrect observations

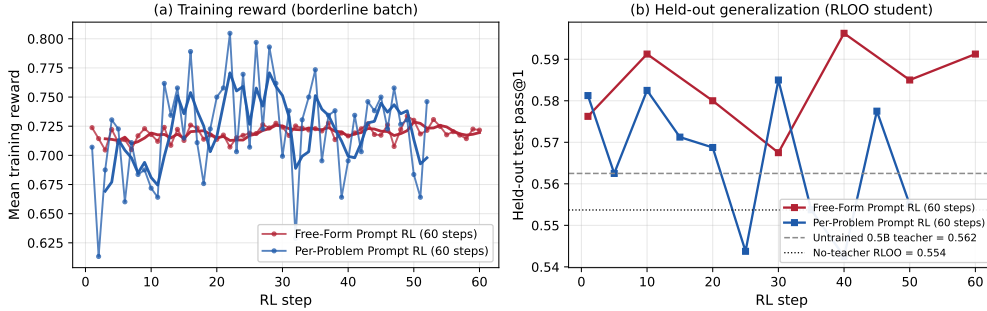


Figure 4: Training and held-out curves for the two prompt-generating teacher RL methods. (a) Mean training reward on the borderline batch; thin lines with markers are raw step-wise values, heavier lines are the 3-step rolling mean. Per-Problem RL shows a modest upward trend across its 60-step budget. (b) Periodic held-out test pass@1; dashed gray line is the untrained 0.5B teacher reference (0.558), dotted black line is the no-teacher RLOO baseline (0.554). Both methods cross the untrained-teacher reference and stabilize above it.

about specific operands often enough that the extra flexibility is wasted. The untrained per-problem baseline (0.554) is 1.51 pp below the untrained global baseline (0.5689). RL recovers most but not all of this gap. A larger teacher would fix the arithmetic errors, but the cross-method ceiling argument predicts that the gains would still be capped at $\sim +3$ pp.

Held-out dynamics of the Per-Problem run. The Per-Problem teacher row in Table 1 is reported as mean \pm SD across three late-training eval steps of a single 60-step run, giving 0.577 ± 0.005 . Training reward continues to drift upward across the full budget (from ~ 0.71 at step 1 to ~ 0.75 by step 50), while held-out pass@1 oscillates back down to ~ 0.555 by step 50, with the KL anchor to the un-LoRA'd reference drifting past -2 . The full curve is plotted in Figure 8 in Appendix G. The pattern is the classic Per-Problem overfitting under our chosen hyperparameters: $B_{\text{fixed}} = 256$ training problems are recycled ~ 7 times each over 60 steps under $G_{\text{prob}} = 8$ tip samples, enough for the small teacher to memorize per-tuple arithmetic that does not transfer to the test split, while the weak $\beta_{\text{KL}} = 0.01$ anchor lets the policy drift far from the reference. Reporting mean \pm SD across three late-training eval steps captures the near-peak variance honestly rather than picking a single best checkpoint.

Why routing hurts the evolutionary search. The plain (unrouted) variant of the reflective evolutionary search reaches 0.5688 pass@1 on a single seed, exceeding the multi-seed mean of the routed variant at 0.560. Two structural factors are consistent with this gap. First, the routed variant operates with ~ 60 accepted mutations spread across 5 clusters, which gives only ~ 12 directed updates per cluster, while the plain variant applies all 60 accepted mutations to a single global population. Second, cluster-conditional optimization multiplies the number of independent decisions, increasing seed-to-seed variance (SD 0.015 across the 6 routed seeds, against tighter spreads for the global methods). The single-best evolutionary metric (max over clusters by test pass@1) does close the gap to Free-Form (0.574 versus 0.585), suggesting that the per-cluster best prompts are individually competitive but the routing decision occasionally picks a sub-optimal cluster-prompt. Taken together, this is a case in which the inductive bias offered by hand-written routing is too coarse to help and too restrictive to retain the benefit of a global search.

Computational complexity. All three teacher methods share the same dominant cost: each optimization step requires sampling K student rollouts per candidate prompt and per training problem, and student inference, not teacher inference, is the bottleneck because the student emits up to 1024 tokens of chain-of-thought while a teacher prompt is at most 200 to 300 tokens. Per step, **Free-Form Prompt RL** therefore costs $\mathcal{O}(B \cdot G \cdot K)$ student rollouts (B problems per step, G prompts per cell, K student samples per (prompt, problem) pair), plus $\mathcal{O}(G)$ teacher generations and a single LoRA update of size $\mathcal{O}(r \cdot d)$ where $r = 32$ is the LoRA rank and d the teacher's hidden width. With $B = 256$, $G = 16$, $K = 1$ this is 4,096 student rollouts per gradient step, dominated wall-clock by vLLM batched decoding on a single GPU. **Per-Problem Prompt RL** has the same asymptotic form

but reduces B to 32 and G to 8 per problem, yielding 256 student rollouts per step at the cost of a much larger conditioning input (the puzzle is now also in the teacher context). The signal-tightened Per-Problem variant raises K to 4 and B to 128, which multiplies the student-rollout cost by $\sim 8\times$ and is the main reason it was not used as the headline run. **Reflective Evolutionary Prompt Search** has no gradient cost at all: each accepted mutation costs one mutator-LLM call ($\mathcal{O}(1)$ Qwen-7B-Instruct generation, larger per call but called far less often) plus a Pareto-frontier evaluation of $\mathcal{O}(P \cdot B_{\text{eval}})$ student rollouts, where P is the candidate pool size. With $P \in [1500, 3000]$ and $B_{\text{eval}} = 64$ this yields roughly 10^5 student rollouts per accepted mutation, so the evolutionary loop trades smaller per-step gradient work for a much larger evaluation footprint. Memory-wise the small (0.5B) teacher and a $r = 32$ LoRA adapter together fit alongside the student in a single 80GB GPU, so all methods are single-GPU; the routed evolutionary variant runs $C = 5$ independent populations and therefore scales linearly with the number of clusters.

D Hyperparameters

All methods use the same LoRA configuration ($r = 32$, $\alpha = 64$, dropout 0.0, target modules q,k,v,o_proj) and student sampling settings (temperature 0.6, top- $p = 0.95$, top- $k = 20$, max student tokens 1024).

Table 4: Hyperparameters per method.

Hyperparameter	Free-Form RL	Per-Problem RL	Plain Evo.	Routed Evo.
Teacher backbone	Qwen-0.5B-Instruct	Qwen-0.5B-Instruct	n/a	n/a
Group size G (prompts per cell)	16	8 per problem	n/a	n/a
Batch B (problems per step)	256	32	64 (eval batch)	64 (eval batch)
Student samples K per (prompt, problem)	1	1	1	1
Steps / iterations	60	13 [†]	60 accepted muts.	60 accepted muts.
Learning rate	5e-5	5e-5	n/a	n/a
KL coefficient β_{KL}	0.01	0.01	n/a	n/a
Entropy bonus β_{ent}	0.05	0.02	n/a	n/a
Teacher sampling temperature	0.9	0.9	0.9 (mutator)	0.9 (mutator)
Max prompt tokens	300	200	300	300
Borderline filter window	[0.125, 0.875]	[0.125, 0.875]	[0.125, 0.875]	[0.125, 0.875]
K_{BASE} (borderline classification)	4	4	4	4
Candidate pool size	3000	2000	1500	3000
Clusters (routing)	n/a	n/a	n/a	5 (rule-based)
Mutator LLM	n/a	n/a	Qwen-7B-Instruct	Qwen-7B-Instruct

[†] A 60-step signal-tightened Per-Problem variant ($K = 4$ student rollouts averaged, $B = 128$, partial-credit reward, $K_{\text{BASE}} = 16$, borderline [0.20, 0.80], MAX_TIP_TOKENS=80) exists in the repo at `train_teacher_lora_v6_r1oo_large.py`; its results are not included in this table.

E Meta-prompts

Free-Form meta-prompt (problem-independent):

Task: Write a helpful system prompt for an AI assistant that solves Countdown puzzles. The assistant is given 3-4 numbers and a target, and must combine them using +, -, *, / (each number used once) to reach the target.

Here are two examples of system prompts that work well:

Example 1:

You are an expert at Countdown puzzles. Strategy: try multiplication first to quickly approach the target. If you overshoot, use subtraction to correct. If multiplication is not useful, fall back to addition. Always use each number exactly once. Verify your final equation.

Example 2:

To solve Countdown, look at the relationship between the target and the available numbers. For large targets, multiply first. For small targets, try subtraction. Use division only when it gives a whole number. Always check that each number appears exactly once in your final expression.

Now write a different system prompt (under 100 words, single focused paragraph):

System prompt:

Per-Problem meta-prompt (parameterized by (nums, target)):

Task: Write a helpful system prompt for an AI assistant that will solve this specific Countdown puzzle.

Puzzle: combine the numbers {nums} using +, -, *, / (each used exactly once) to reach the target {target}.

[two illustrative examples on other puzzles follow]

Now write a different system prompt for the puzzle with numbers {nums} and target {target} (under 80 words, single focused paragraph, mention concrete observations about these specific numbers):

System prompt:

F Sample teacher-generated prompts

F.1 Free-Form Prompt RL

Certainly! Here's a concise yet effective system prompt tailored specifically for solving Countdown puzzles with three or four numbers and a target:

****System Prompt:****

When tackling Countdown puzzles, always start by identifying the largest possible combination of numbers from your set. Focus on maximizing the product of the largest numbers while ensuring all digits are utilized precisely once. This method helps narrow down potential solutions efficiently. Don't hesitate to apply multiplication techniques if needed; they can significantly speed up calculations. Finally, verify your final answer against the target value to ensure accuracy before concluding.

This prompt provides a clear structure for approaching the problem step-by-step, guiding the user through the process of combining numbers optimally to reach the desired result.

The verbose “Certainly! Here’s.../ This prompt provides...” framing is part of the deployed system message; the student sees the entire block as its prompt τ .

F.2 Per-Problem Prompt RL

Certainly! Here's a system prompt tailored specifically for the puzzle with numbers [44, 19, 35] and a target value of 98:

In this task, we have three distinct integers---[44, 19, 35]---and our objective is to use each one exactly once to obtain the number [44, 19, 35]. The challenge involves combining these integers through arithmetic operations (+, -, *, /) in such a way as to yield the target sum of 98.

Let's explore various strategies to achieve this goal within an 80-word limit while highlighting key observations from these specific numbers.

Observations:

1. **Target Value**: The final result should be 98.
2. **Integer Range**: Each number can range from very large to very small values.
3. **Combination Strategy**: We need to consider all possible ways to arrange these numbers in order to multiply them together without exceeding the target sum.

The teacher's framing reveals two operand-level confusions characteristic of the 0.5B Per-Problem teacher: it states the objective as "to obtain the number [44, 19, 35]" (it conflates "the input list" with "the target"), and it suggests "multiplying them together" even though $44 \times 19 \times 35 = 29,260$ vastly exceeds 98. These operand-level errors are exactly the failure mode that the prompt-scope ablation isolates (Sec. 5.1.2, Discussion). The correct solution $(44 + 19) + 35 = 98$ requires only addition, but the teacher's prompt does not surface it.

F.3 Plain Reflective Evolutionary Search

Solve the Countdown puzzle using +, -, *, or / on three or four given numbers to reach the target value. Use each number exactly once, working through one operation at a time and verifying each step carefully. Adjust your strategy as needed to ensure precision. Good luck!

Solve the Countdown puzzle by combining three or four given numbers to reach a target value using +, -, *, or /. Use each number exactly once and verify each step for accuracy. Aim for precision to avoid overestimating and ensure correct solutions.

Your goal is to reach a target number using arithmetic operations on three or four given numbers, ensuring each is used exactly once. Break down the problem into steps, validate calculations, and experiment with various combinations until you hit the target. Adjust your approach if necessary.

F.4 Routed Reflective Evolutionary Search

The router partitions the test set into five clusters by operand count and target magnitude. The best prompt found per cluster after 60 accepted mutations is given below.

Consider all operations, prioritize reaching the target with basic arithmetic and integer results. Avoid complex expressions; instead, focus on direct combinations and simple rearrangements of given numbers.

Consider multiplication and division first to reduce numbers quickly.

To successfully solve Countdown puzzles involving three or four numbers, follow these steps:

1. Select the Numbers: Choose three or four unique numbers from the provided set.
2. Identify Your Target: Determine the target value carefully.
3. Apply Operations Carefully: Multiplication, Subtraction, Division, adjusting the target or balancing operations as needed.
4. Verify Your Final Equation: Ensure every number has been paired according to the specified operations and the target is met.

To tackle Countdown puzzles effectively, start by identifying key relationships and applying mathematical operations strategically. Focus on mastering addition, subtraction, multiplication, and division techniques. Begin with smaller calculations to build intuition, then iterate toward the target.

(Identical to Cluster 2’s best prompt: the mutator-LLM converged to the same long, step-listing prompt for both 3-number-large and 4-number-large clusters, suggesting that the cluster split is finer than the prompt design can distinguish.)

G Additional figures

This section collects diagnostic and reliability figures that support but are not central to the main results.

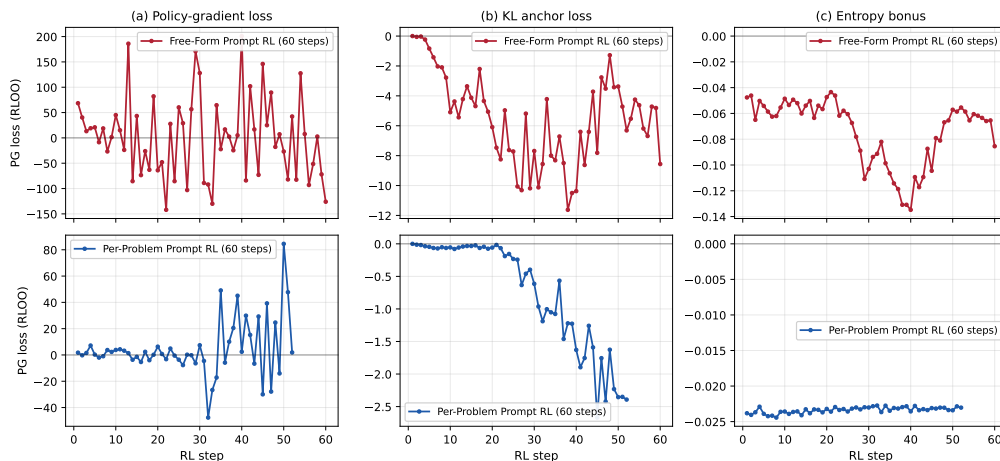


Figure 5: Training-loss diagnostics for both RL teacher methods on the matched 60-step budget. Top row: **Free-Form Prompt RL** (red); bottom row: **Per-Problem Prompt RL** (blue). Each row uses its own y-axis so the two methods’ very different magnitudes are both visible end-to-end. Columns: (a) policy-gradient (RLOO) loss, (b) KL loss against the un-LoRA’d reference policy, (c) entropy bonus on prompt tokens. No term diverges for either method, indicating that $\beta_{KL} = 0.01$ is adequate.

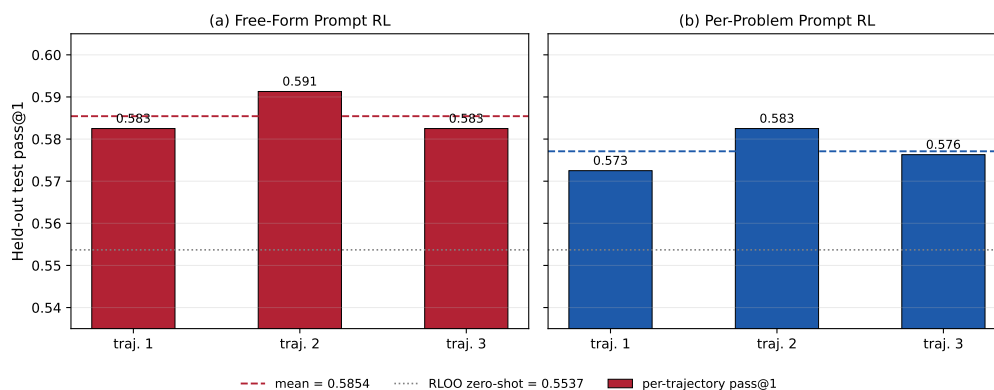


Figure 6: Per-trajectory held-out test pass@1 for the two RL teacher methods. (a) **Free-Form Prompt RL**, three sampled tips at the final (step-60) checkpoint. (b) **Per-Problem Prompt RL**, three late-training evaluation steps. Dashed lines: per-method means. Dotted line: RLOO baseline.

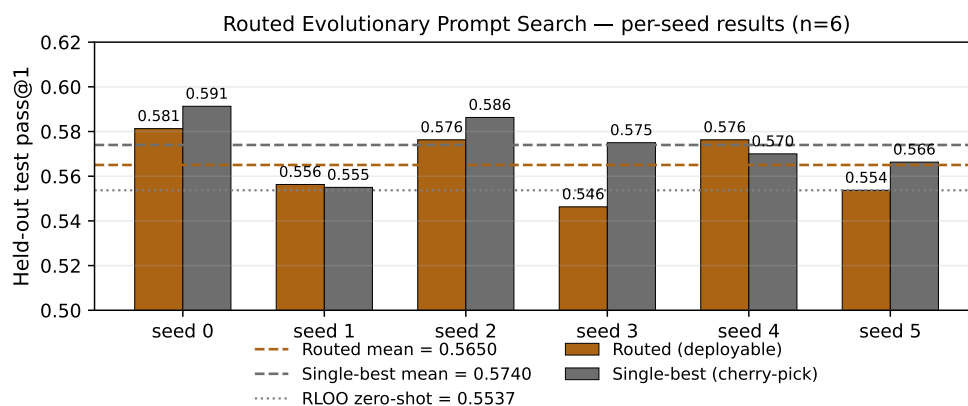


Figure 7: Routed Reflective Evolutionary Search: per-seed held-out pass@1 ($n = 6$ seeds). Blue: *routed* (deployable), where each test problem is answered using the best prompt of its assigned cluster according to the hand-written router. Orange: *single-best (cherry-pick)*, where for each seed we apply every cluster’s best prompt to the full test set and report the maximum, that is, the upper bound that the perfect router would reach. Single-best is therefore not deployable, it requires knowing which cluster’s prompt is best for the test distribution. Dashed lines: per-metric mean. Dotted line: RLOO baseline.

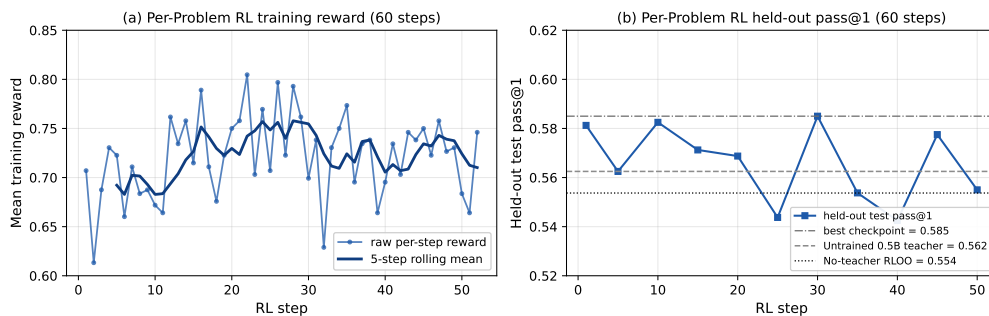


Figure 8: **Per-Problem Prompt RL**, full 60-step run under the hyperparameters of Table 1. (a) Mean training reward on the borderline batch continues to climb across the full budget. (b) Held-out test pass@1 peaks at ~ 0.585 around step 30 and oscillates back down to ~ 0.555 by step 50, ending below the no-teacher RLOO baseline. The discrepancy between training and held-out curves is the overfitting pattern discussed in Section 6: the small teacher memorizes per-tuple arithmetic from the recycled $B_{\text{fixed}} = 256$ training pool, and the weak $\beta_{\text{KL}} = 0.01$ anchor lets the policy drift far from the reference. Best-checkpoint reference shown as a horizontal dot-dash line.