

Extended Abstract

Motivation Arithmetic reasoning is a useful controlled setting for studying post-training in language models because solutions are easy to verify but difficult to discover reliably. In the Countdown task, a model is given a set of input numbers and a target value, and it must produce an arithmetic expression that uses each number exactly once and evaluates to the target. This project studies how far a small Qwen2.5-0.5B model can be pushed on Countdown using a modern post-training pipeline. The most important question to be answered is not only whether post-training improves single-sample accuracy, but whether it preserves the diversity of solution paths — a model could become sharper in distribution of correct answers under reinforcement learning while losing coverage over valid reasoning trajectories, limiting the benefit of test-time sampling.

Method We build on a standard post-training pipeline consisting of supervised fine-tuning (SFT), preference optimization with IPO, and on-policy reinforcement learning with RLOO. We then introduce two extensions. First, we use on-policy distillation (OPD), where a frozen 7B teacher provides token-level distributional guidance on student-generated rollouts. Unlike ordinary offline distillation, the student is trained on its own on-policy samples, so the teacher corrects and reshapes the distribution in regions that the student actually visits. Second, we introduce an anti-collapse version of OPD designed to preserve output diversity: since strong RL and distillation objectives over-sharpen the student distribution (raising pass@1 while reducing pass@16), we increase the entropy coefficient and soften the teacher distribution with a higher distillation temperature. Finally, because Countdown has a deterministic verifier, we use test-time self-verification: sample multiple candidate solutions and return the first one that satisfies the verifier.

Implementation The student model is Qwen2.5-0.5B, and the teacher is a frozen 7B Qwen2.5-Instruct model fine-tuned on Countdown. The dataset is `countdown_tasks_3to4`, containing 3- and 4-number problems. Evaluation uses a held-out set of 50 problems with 16 samples per prompt under a fixed sampler (temperature 0.6, top- p 0.95, top- k 20). The verifier reward is sparse and rule-based: invalid or missing answers receive low reward; valid equations using exactly the input numbers and reaching the target receive 1.0. Headline numbers (pass@1 / pass@16): SFT 0.354 / 0.800; IPO 0.409 / 0.760; RLOO 0.598 / 0.740 — revealing diversity collapse; plain OPD 0.640 / 0.780; anti-collapse OPD 0.680 / 0.840, the best overall, matching the 7B teacher’s pass@16 while far exceeding its single-sample accuracy. Test-time self-verification converts this coverage into 84.0% realized accuracy at only 3.80 samples per problem on average.

Results Three findings stand out. (1) *Diversity collapse is real and measurable*: every sharpening method trades coverage for precision — pass@16 drops from SFT’s 0.800 to RLOO’s 0.740 even as pass@1 climbs from 0.354 to 0.598. (2) *The collapse is reversible with two hyperparameters*: a $10\times$ entropy coefficient and a softened distillation temperature lift pass@16 by +6 points and pass@1 by +4 points simultaneously — diversity and sharpness improve together, not as a trade-off — largest where the policy is weakest (4-number pass@16, +7.7). (3) *Coverage converts to accuracy for free at test time*: verifier-guided best-of- N realizes 84.0% accuracy at an average of 3.80 samples per problem, and the anti-collapse policy dominates the small-model accuracy-compute frontier.

Discussion The main limitation is the binding constraint we identified but did not eliminate: 4-number coverage. The anti-collapse policy reaches 4-number pass@16 of only 0.692 — about 31% of 4-number problems are never solved in 16 tries because the 0.5B policy never discovers a valid path, and neither RL nor distillation can reinforce a trajectory that is never sampled. Results are single runs on 50 test problems, so small deltas warrant care. We also implemented tool-integrated reasoning and multi-agent self-play; neither outperformed this pipeline, and the main paper analyzes why.

Conclusion A 0.5B model, post-trained well, can match a 7B teacher’s coverage on Countdown (pass@16 = 0.84) while nearly doubling its single-try accuracy (0.680 vs. 0.535). The broader lesson: *diversity is a first-class training objective* — pass@1 gains from RL can silently destroy the coverage that test-time scaling depends on, cheap regularization recovers it, and on verifiable tasks self-verification converts that coverage into realized accuracy at negligible cost. The clearest next step: rejection-sample correct 4-number traces from a stronger teacher, injecting solution paths the student cannot find alone.

Diversity-Preserving On-Policy Distillation and Test-Time Verification for Countdown Reasoning

Juntao Cheng

Institute of Computational and Mathematical Engineering
Stanford University
jtcheng3@stanford.edu

Qi Wu

Department of Computer Science
Stanford University
q1wu@stanford.edu

Zhuoang Tao

Institute of Computational and Mathematical Engineering
Stanford University
zhuoang@stanford.edu

Abstract

We study how far a small Qwen2.5-0.5B model can be pushed on the Countdown arithmetic reasoning task. Our pipeline follows modern post-training practice — supervised fine-tuning (SFT), preference optimization (IPO), and on-policy reinforcement learning (RLOO) — followed by two extensions: on-policy distillation (OPD) from a frozen 7B teacher, and verifier-guided test-time self-verification. Our key observation is that methods sharpened by policies (IPO, RLOO, OPD) improve single-sample accuracy (pass@1: 0.354 \rightarrow 0.640) while *collapsing* sampling diversity (pass@16: 0.800 \rightarrow 0.740 under RLOO), which limits the benefit of test-time sampling. We diagnose this collapse and reverse it with an *anti-collapse* OPD variant that changes exactly two knobs — a $10\times$ larger entropy bonus and a softened distillation temperature ($T = 1.3$) — lifting pass@1 and pass@16 *together* to 0.680 and 0.840. The 0.5B student thereby matches its 7B teacher’s coverage while exceeding its single-try accuracy by 14.5 points. Because Countdown solutions are locally checkable, best-of- N self-verification converts this coverage into 84.0% realized accuracy at only 3.80 samples per problem. We also report negative results for tool-integrated reasoning and multi-agent self-play, which did not improve over this pipeline. Relevant codes and configurations are included in our project’s Github repository.

1 Introduction

Post-training has become the dominant recipe for turning base language models into reliable problem solvers: imitate curated demonstrations (SFT), align to preferences (DPO/IPO), then optimize a reward directly with on-policy RL [1–4]. Tasks with *verifiable* rewards — where a deterministic checker scores any candidate solution — have emerged as the cleanest setting for studying this pipeline, because reward hacking is impossible by construction and progress is unambiguous [5, 6].

We work in exactly this setting. **Countdown** gives the model a list of numbers (e.g., [44, 19, 35]) and a target (e.g., 98); the model must produce an arithmetic equation that uses each number exactly once and evaluates to the target, formatted as `<answer>(35 + 19) + 44</answer>`. A rule-based verifier assigns reward 1.0 to a correct equation, 0.1 to a well-formatted but wrong one, and 0.0 otherwise. Our policy is deliberately small — Qwen2.5-0.5B [7] — which makes the task genuinely hard (the base model solves only 32% of problems in one try) and makes every architectural and algorithmic choice visible in the metrics.

The standard pipeline question is “how high can pass@1 go?” We argue this is only half the question. The other half is what happens to **pass@ k** — the probability that at least one of k samples is correct — which measures the *coverage* of the policy’s output distribution over valid solution paths.

Coverage is usually not an academic curiosity: on any task with a cheap verifier, pass@ k is *realizable* accuracy: sample k candidates and return the first that verifies. A policy that gains pass@1 by concentrating probability mass on a few decoding paths may simultaneously lose pass@16, trading away exactly the headroom that test-time compute could exploit [8, 9].

This trade-off is not hypothetical. In our experiments, RLOO lifts pass@1 from 0.354 (SFT) to 0.598 — the strongest required method — but pass@16 *falls* from 0.800 to 0.740. We call this *diversity collapse*, and making it a first-class training concern is the organizing idea of this report. Concretely, we contribute:

1. **A complete, tuned implementation of the required pipeline** (SFT \rightarrow IPO \rightarrow RLOO) on Countdown, with pass@1 improving monotonically from 0.354 to 0.598, and an analysis of the training dynamics of each stage.
2. **On-policy distillation (OPD) from a 7B teacher** [10]: a frozen Countdown-fine-tuned 7B Qwen2.5-Instruct teacher provides token-level forward-KL guidance *on the student’s own rollouts*, combined with the RLOO policy gradient in a hybrid objective. OPD reaches pass@1 = 0.640.
3. **A diagnosis of diversity collapse and a two-knob fix.** Warm-starting from the best OPD checkpoint, we raise the entropy coefficient $10\times$ and soften the distillation temperature to $T = 1.3$. This *anti-collapse* variant improves pass@1 and pass@16 simultaneously (0.680 / 0.840), matching the 7B teacher’s coverage with $14\times$ fewer parameters.
4. **Test-time self-verification.** Because the target is part of the input, the model can check its own candidates with no reference solution. Best-of- N with early stopping realizes pass@ N exactly: 84.0% accuracy at an average cost of 3.80 samples per problem.
5. **Negative results on two further extensions.** We also implemented tool-integrated reasoning and multi-agent self-play (the directions of our original proposal); neither improved over the pipeline above. We analyze why in §5.3.

The through-line of our results is that *sharpness and diversity need not trade off*. The naive view is that pass@1 and pass@16 sit on a Pareto frontier and regularization simply slides along it. Instead we find that targeted diversity pressure (entropy + soft targets) recovers coverage that converts into *higher* pass@1 — because on hard problems, a policy that explores more paths during training discovers solutions that sharpening alone can never find.

2 Related Work

Post-training pipelines for reasoning. Modern post-training follows InstructGPT’s workflow [1]: SFT on demonstrations, then preference or reward optimization. Direct Preference Optimization [2] removes the explicit reward model; IPO [3] replaces DPO’s logistic loss with a bounded squared loss that resists overfitting when preferences are nearly deterministic — our regime, since our preference pairs come from a hard verifier. For on-policy RL we use RLOO [4], which revisits REINFORCE with a leave-one-out baseline computed from the other rollouts in a group, avoiding a learned critic. It is simpler than PPO [11] and works well for single-turn generation with verifiable rewards.

DeepSeek-R1 [5] demonstrated that pure RL against verifiable rewards can elicit reflection and self-verification behaviors; TinyZero [6] reproduced this on Countdown specifically, establishing the task as a standard small-scale testbed. Our required pipeline follows this line; our contribution is to measure and repair what RL does to the *distributional* properties of the policy.

Knowledge distillation, on- and off-policy. Classical distillation [12] matches teacher logits on a fixed dataset — off-policy with respect to the student. Agarwal et al. [10] (GKD) showed that distilling on *student-generated* sequences fixes the train-inference distribution mismatch: the teacher corrects the student precisely in the regions the student actually visits. Our OPD implements this with a forward (mode-covering) KL, which Gu et al. [13] contrast with the reverse (mode-seeking) KL; we choose forward KL deliberately because coverage preservation is our goal — reverse KL would

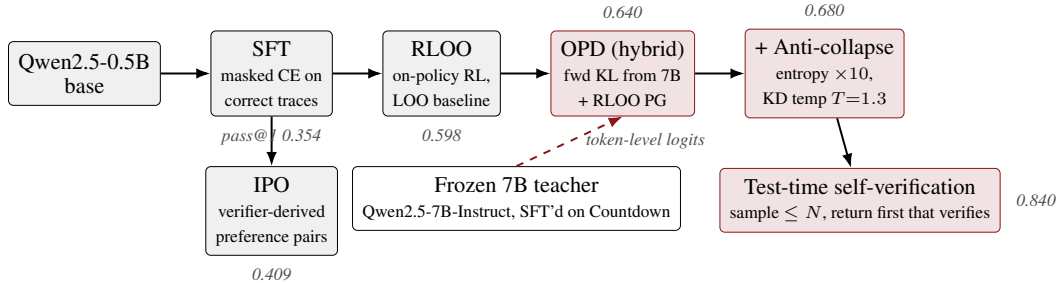


Figure 1: **Method overview.** Gray: the required pipeline. Red: our extensions — on-policy distillation from a frozen 7B teacher with an anti-collapse configuration, and verifier-guided best-of- N at inference. Numbers are pass@1 on the held-out test set (the final number is realized accuracy at budget $N=16$).

sharpen toward teacher modes and worsen the collapse we are trying to fix. Distillation temperature as a softness knob dates to Hinton et al. [12]; we repurpose it not for optimization stability but explicitly as a *diversity* regularizer, and combine it with an entropy bonus inside an RL+KD hybrid objective. Therefore, fixing pass@ k collapse as the target of distillation-temperature tuning is the novelty of this project.

Pass@ k , verifiers, and test-time scaling. Pass@ k and its unbiased estimator come from code generation [8]. Sampling-and-selection at inference is well studied: trained verifiers rerank math solutions [14], and self-consistency selects by majority vote [15]. Snell et al. [9] argue test-time compute can beat parameter scaling when coverage exists to exploit.

Countdown admits a stronger primitive than any of these. Its verifier is *exact and intrinsic* (the target is in the prompt), so best-of- N selection realizes pass@ N with zero verifier error and no learned components. Majority voting is in fact inapplicable: the “answer” (the target value) is given, and the variation across samples lies in the equation — precisely what the verifier checks. This makes Countdown a clean environment for the coverage-vs-sharpness question: pass@ k is not a proxy for capability, it *is* deployable accuracy. Our finding that RL silently erodes it connects the RL-for-reasoning line to the test-time-scaling line in a way neither typically measures.

Tool-integrated reasoning. Recent work extends RL-trained LLMs with tool calls during inference: Search-R1 [16] trains models to invoke a search engine mid-rollout and masks retrieved tokens from the policy gradient; Tool-Star [17] uses a two-phase pipeline (cold-start SFT then multi-tool RL) with hierarchical rewards. Both demonstrate that even small models can learn when to call a tool through RL, but evaluation has focused on knowledge-intensive or multi-step math tasks. Whether tool use helps on a constrained arithmetic-search task like Countdown — where the bottleneck may be *search* rather than *computation* — has not been tested.

3 Method

3.1 Task and verifier reward

Each Countdown instance is a prompt containing numbers $\mathbf{c} = (c_1, \dots, c_m)$ ($m \in \{3, 4\}$) and a target g . The model emits a chain-of-thought inside `<think>` tags and a final equation inside `<answer>` tags. The rule-based verifier (`evaluation/countdown.py`) extracts the equation and assigns

$$R(y) = \begin{cases} 1.0 & \text{equation uses exactly } \mathbf{c}, \text{ evaluates to } g, \\ 0.1 & \text{<answer> extracted but invalid or wrong result,} \\ 0.0 & \text{no <answer> tags found.} \end{cases}$$

The 0.1 format reward is important early in training: it produces gradient signal for incorrect outputs before the policy can reliably solve anything.

3.2 Required pipeline: SFT, IPO, RLOO

SFT. We minimize the masked next-token loss on response tokens of demonstrated correct traces:

$$\mathcal{L}_{\text{SFT}}(\theta) = - \sum_t \mathbb{1}[\text{resp}_t] \log \pi_\theta(y_t | y_{<t}, x), \quad (1)$$

where the mask restricts the loss to completion tokens so prompt conditioning is preserved. Demonstrations come from the `cog_behav_all_strategies` corpus of structured search traces, teaching the `<think>`-then-`<answer>` format and basic search behaviors (trying pairs, backtracking, verifying intermediate results).

IPO. From verifier-scored sample pairs we build preferences (x, y_w, y_l) (chosen y_w correct, rejected y_l incorrect) and minimize the IPO objective [3]:

$$\mathcal{L}_{\text{IPO}}(\theta) = \mathbb{E}_{(x, y_w, y_l)} \left(\log \frac{\pi_\theta(y_w | x) \pi_{\text{ref}}(y_l | x)}{\pi_\theta(y_l | x) \pi_{\text{ref}}(y_w | x)} - \frac{1}{2\beta} \right)^2, \quad (2)$$

with the SFT model as the frozen reference π_{ref} . The squared (bounded) loss avoids DPO’s failure mode of driving the preference margin to infinity on deterministic preferences — exactly our regime, since the verifier never flips a label.

RLOO. We optimize the verifier reward directly with REINFORCE leave-one-out [4]. For each prompt we draw a group of G rollouts y_1, \dots, y_G from a vLLM sampler μ (a recent snapshot of the policy), score them, and form per-rollout advantages with the leave-one-out baseline:

$$A_i = R_i - \frac{1}{G-1} \sum_{j \neq i} R_j, \quad \mathcal{L}_{\text{PG}}(\theta) = - \frac{1}{G} \sum_{i=1}^G w_i A_i \log \pi_\theta(y_i | x), \quad (3)$$

where $w_i = \text{clip}(\exp(\log \pi_\theta(y_i) - \log \mu(y_i)), w_{\text{max}}=10)$ is a detached, clipped importance weight correcting the small off-policyness between the sampling snapshot and the current policy. We add an entropy bonus and a KL penalty to the sampling policy for stability.

The LOO baseline is unbiased and critic-free. Within a group it cancels prompt difficulty, so only *relative* success among the G rollouts drives the gradient: prompts where all rollouts succeed or all fail contribute nothing, which automatically focuses learning on problems at the edge of the policy’s ability.

3.3 Extension 1: on-policy distillation with an anti-collapse configuration

RLOO has a structural ceiling: it reinforces only trajectories the policy itself discovers. If the 0.5B model never samples a valid path for a hard 4-number problem, no amount of RL can reward it. On-policy distillation [10] attacks exactly this gap: a stronger teacher supplies *dense, token-level* information about which continuations are promising, evaluated *on the student’s own rollouts* so the guidance lands on-distribution.

Each step samples G rollouts per prompt from the student (vLLM), then both models score the student-generated response tokens. The hybrid objective over masked response tokens is

$$\mathcal{L}(\theta) = \lambda_{\text{KL}} \underbrace{\text{KL}(p_T^{(1/T)} \| p_S^{(1/T)})}_{\text{teach}} + \lambda_{\text{RL}} \underbrace{\mathcal{L}_{\text{PG}}}_{\text{verifier}} - \lambda_H \underbrace{H[\pi_\theta]}_{\text{explore}}, \quad (4)$$

where $p_T^{(1/T)} = \text{softmax}(z_T/T)$ and $p_S^{(1/T)} = \text{softmax}(z_S/T)$ are the temperature-scaled teacher and student token distributions, and \mathcal{L}_{PG} is the same RLOO gradient as above, computed from raw ($T=1$) student log-probs. Three design choices matter:

- **Forward (mode-covering) KL.** $\text{KL}(p_T \| p_S)$ penalizes the student for assigning low mass to *any* token the teacher finds plausible. This is the coverage-preserving direction; reverse KL would let the student collapse onto a single teacher mode [13], reproducing the very failure we are repairing.

Algorithm 1 Anti-collapse on-policy distillation (one step)

Require: student π_θ , frozen teacher p_T , sampler snapshot μ , batch of B prompts, group size G

- 1: $\{y_i^{(b)}\} \leftarrow$ sample G rollouts per prompt from μ (vLLM; temp 1.0, no truncation)
- 2: $R_i^{(b)} \leftarrow$ verifier score of each rollout; $A_i^{(b)} \leftarrow R_i^{(b)} - \frac{1}{G-1} \sum_{j \neq i} R_j^{(b)}$
- 3: **for** each rollout (with gradient accumulation over micro-batches) **do**
- 4: $z_S, z_T \leftarrow$ student / teacher logits on response tokens (truncate to shared vocab)
- 5: $\mathcal{L}_{\text{KL}} \leftarrow \text{mean}_t \text{KL}(\text{softmax}(z_T/T) \parallel \text{softmax}(z_S/T))$ $\triangleright T = 1.3$, forward KL
- 6: $w_i \leftarrow \text{clip}(\exp(\log \pi_\theta(y_i) - \log \mu(y_i)), 10)$, detached
- 7: $\mathcal{L}_{\text{PG}} \leftarrow -w_i A_i \log \pi_\theta(y_i)$ \triangleright raw $T=1$ log-probs
- 8: $H \leftarrow \text{mean}_t \text{entropy}(\text{softmax}(z_S))$
- 9: accumulate $\nabla_\theta [0.5 \mathcal{L}_{\text{KL}} + 0.5 \mathcal{L}_{\text{PG}} - 0.01 H]$
- 10: **end for**
- 11: clip global grad-norm to 1.0; AdamW step (lr 10^{-5} , constant)

- **Hybrid, not pure distillation.** The teacher (pass@1 = 0.535) is imperfect; the verifier is exact. $\lambda_{\text{KL}} = \lambda_{\text{RL}} = 0.5$ lets the RL term overrule the teacher when the verifier disagrees.
- **Shared tokenizer.** Student and teacher are both Qwen2.5, so token-level KL is well-defined; vocab-padding differences (151,936 vs. 152,064) are truncated to the shared min before the KL.

Diagnosing diversity collapse. Plain OPD (entropy $\lambda_H=0.001$, $T=1.0$) reaches pass@1 = 0.640, but the pass@16 history across the pipeline reveals the problem: SFT 0.800 \rightarrow RLOO 0.740 \rightarrow OPD 0.780. Sharpening methods raise pass@1 by concentrating mass on few decoding paths, shrinking the set of problems the policy can solve *at all*. The cost concentrates on 4-number problems (pass@16 \approx 0.62; \sim 38% never solved in 16 tries), while 3-number problems are nearly saturated. Moreover the OPD training reward had plateaued (\approx 0.57), so simply training longer changes nothing — the lever has to be distributional.

The anti-collapse fix. We warm-start from the best OPD checkpoint and change *exactly two* hyperparameters, keeping everything else fixed:

Knob	OPD	Anti-collapse	Mechanism
Entropy coef. λ_H	0.001	0.01 (10 \times)	direct pressure against mode collapse
Distill temp. T	1.0	1.3	softer targets transfer more “dark knowledge”

Both methods target re-expanding the output distribution through different terms of Eq. 4. Note that without T^2 rescaling, raising T also gently shrinks the KL gradient magnitude; the two effects are entangled (see §6). Algorithm 1 summarizes one training step.

3.4 Extension 2: test-time self-verification

Countdown has a property most benchmarks lack: *the target is part of the input*. A solver can therefore check any candidate equation locally — does it use exactly the provided numbers, and does it evaluate to the target? — with no reference solution, no learned verifier, and no voting heuristic. Our test-time method is simply: **sample up to N candidates; return the first that verifies; stop early on success**. This realizes pass@ N exactly: a problem is solved iff at least one of the N samples is correct.

For a problem with c correct among n recorded samples, realized accuracy at budget k is the standard unbiased pass@ k estimator [8], and the expected number of samples drawn under sequential early stopping is

$$\text{pass}@k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \quad \mathbb{E}[\min(T, k)] = \sum_{j=0}^{k-1} \frac{\binom{n-c}{j}}{\binom{n}{j}}, \quad (5)$$

where T is the index of the first verified sample. The second quantity is the honest compute axis: easy problems verify on the first draw and stop, so the average cost sits far below the budget. We validated

Table 1: Main results on held-out Countdown (50 problems, 16 samples/prompt, fixed sampler). The required pipeline raises pass@1 monotonically but *erodes* pass@16; the anti-collapse OPD variant is the only method that is strongest on both axes, matching the 7B teacher’s coverage at 14× fewer parameters. Subscripts $n3/n4$ denote 3-/4-number problem splits.

Model (0.5B)	Overall		pass@1		pass@16	
	pass@1	pass@16	$n3$	$n4$	$n3$	$n4$
Base (zero-shot)	0.321	0.740	–	–	–	–
SFT	0.354	0.800	0.513	0.207	1.000	0.615
IPO ($\beta=0.05$)	0.409	0.760	0.547	0.281	0.958	0.577
RLOO	0.598	0.740	0.826	0.387	0.917	0.577
OPD (hybrid)	0.640	0.780	0.875	0.423	0.958	0.615
OPD + anti-collapse	0.680	0.840	0.914	0.464	1.000	0.692
7B teacher (ref.)	0.535	0.840	0.745	0.341	0.958	0.731

the self-verifier against the evaluation scorer on all 50×16 stored samples: **zero mismatches**, confirming the model is checking its own answers with the same fidelity as the grader. Note this is a deliberately higher-compute inference setting than the fixed single-sample grading protocol; we always report single-sample numbers alongside it.

4 Experimental Setup

Data. Training and evaluation use `countdown_tasks_3to4` (3- and 4-number problems). SFT uses the `cog_behav_all_strategies` demonstration corpus; IPO uses the verifier-labeled preference split (`countdown_tasks_3to4-dpo`); RLOO and OPD train directly on prompts with the verifier as reward. The held-out test set has 50 problems (24 three-number, 26 four-number).

Models. Student: Qwen2.5-0.5B (base). Teacher: Qwen2.5-7B-Instruct, SFT’d on Countdown for 6 epochs (pass@1 = 0.535, pass@16 = 0.840) and kept frozen. The shared Qwen tokenizer makes token-level distillation well-defined.

Evaluation protocol. All numbers use the *fixed* project sampler — temperature 0.6, top- p 0.95, top- k 20, 16 samples per prompt, max 1024 generated tokens — so methods differ only in their weights. We report pass@1 (mean per-sample accuracy, the official metric), pass@16 (coverage), and 3-/4-number splits. pass@ k for intermediate k uses the unbiased estimator in Eq. 5. Metric choice: pass@1 measures deployable single-shot quality; pass@16 measures what the policy is *capable* of and upper-bounds any test-time selection method; the per- m split isolates the difficulty axis (4-number problems have a much larger search space).

Baselines. The base model and SFT anchor the pipeline; IPO and RLOO are the required preference/RL stages and the natural ablations for OPD (does the teacher add anything beyond RL?); plain OPD is the ablation for the anti-collapse knobs (same objective, two hyperparameters apart); the 7B teacher bounds what distillation alone could hope to copy.

Key hyperparameters. SFT: lr 5×10^{-5} , 6 epochs, effective batch 512, cosine schedule. IPO: lr 5×10^{-6} , $\beta = 0.05$, 1 epoch (selected over $\beta = 0.1$ by pass@1). RLOO: lr 10^{-5} constant, batch 128 prompts \times group 8 (=1024 rollouts/step), 200 steps, entropy 0.001, KL 0.001, IS clip 10, sampler temp 1.0. OPD: $\lambda_{KL} = \lambda_{RL} = 0.5$, forward KL, lr 10^{-5} , 100 steps, same batch geometry. Anti-collapse: +150 steps from the OPD checkpoint with $\lambda_H = 0.01$, $T = 1.3$. All runs on single H100s (Modal); full tables in Appendix A.

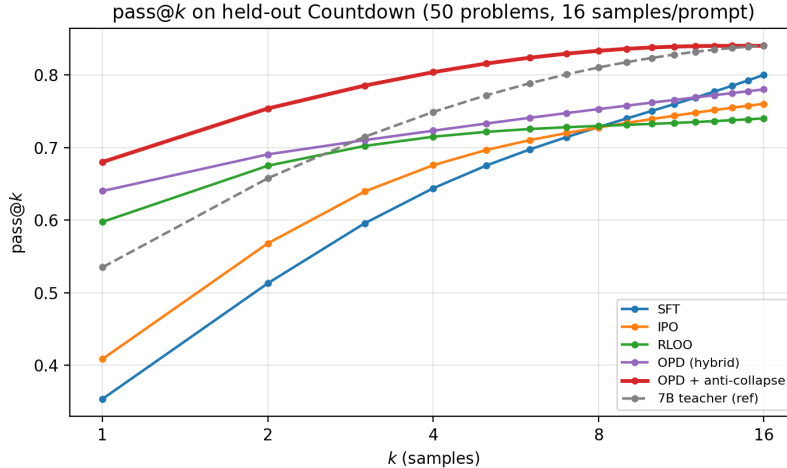


Figure 2: **pass@k across the pipeline** (unbiased estimator, Eq. 5). The curves *cross*: SFT starts lowest but climbs steepest (high diversity); RLOO starts high and flattens earliest (collapsed diversity); the anti-collapse policy (red) both starts highest and keeps climbing, dominating every 0.5B alternative at every budget. The reference 7B teacher is dashed.

5 Results

5.1 Quantitative evaluation

The required pipeline works, but pays in coverage. Table 1 and Figure 2 give the complete comparison. Each stage of the required pipeline improves pass@1 — SFT +3.3 points over base, IPO +5.5 over SFT, RLOO +18.9 over IPO — and the gains are largest where the methods can exploit the verifier (RLOO nearly doubles 4-number pass@1 over IPO, 0.281 \rightarrow 0.387). But pass@16 tells the opposite story: 0.800 (SFT) \rightarrow 0.760 (IPO) \rightarrow 0.740 (RLOO). RLOO’s 16-sample coverage is no better than the *base model*’s. The policy got dramatically sharper while the set of problems it can solve at all stopped growing — on 4-number problems, pass@16 is identical for IPO and RLOO (0.577) and below SFT (0.615).

Why the numbers move the way they do. The pass@1/pass@16 gap measures how much correct probability mass exists but is not prioritized. For SFT the gap is 44.6 points — correct solutions are *in* the distribution but rare. RL closes the gap almost entirely from the wrong side on easy problems (3-number pass@1 0.826 vs. pass@16 0.917) while leaving 4-number coverage untouched: RLOO can only reinforce paths it samples, and on hard problems those samples never arrive. This is the structural ceiling that motivates distillation.

OPD lifts the ceiling; anti-collapse converts it. Adding the teacher’s token-level guidance (OPD) buys +4.2 pass@1 and +4 pass@16 over RLOO — the first method to improve both at once, because forward KL explicitly preserves mass on teacher-plausible paths. The anti-collapse configuration then adds +4.0 pass@1 and +6.0 pass@16 *on top*, with the largest single gain exactly at the diagnosed bottleneck: 4-number pass@16 +7.7 points (0.615 \rightarrow 0.692). Three-number Countdown is fully solved at the coverage level (pass@16 = 1.000). The final policy matches the 7B teacher’s overall coverage (0.840) while exceeding its pass@1 by 14.5 points, and its formatting failures drop from 1.6% of samples to 0.1%.

Training dynamics corroborate the mechanism. Figure 3 shows the on-policy training reward for the three RL-style stages. The OPD hybrid’s plateau at ≈ 0.57 is the key diagnostic: the optimizer was not starved of steps but of *exploration*. Switching on the anti-collapse knobs (same objective, same data, same lr) re-opens progress to a peak of ≈ 0.665 . The cost is measurable off-policy drift — the importance-weight mean ends near 12.4 (clip 10) versus ≈ 1 early in training (Appendix B)

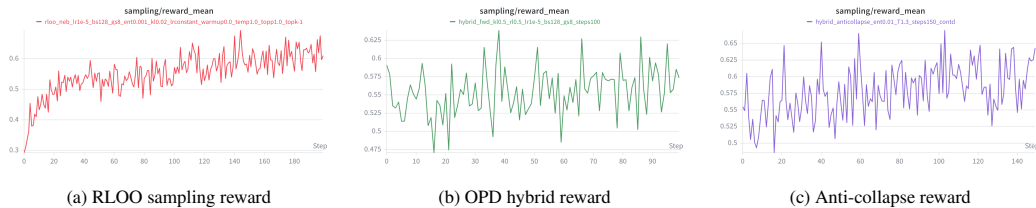


Figure 3: **Training reward dynamics.** (a) RLOO climbs steadily from the SFT initialization. (b) The OPD hybrid run plateaus near reward ≈ 0.57 — training longer with the same recipe is exhausted. (c) Warm-started anti-collapse breaks the plateau (peak ≈ 0.665 around step 70); the diversity knobs, not more steps, unlocked further progress. Full loss/KL/IS-weight diagnostics in Appendix B.

Table 2: Test-time self-verification. Left: realized accuracy vs. budget N for the best policy (OPD + anti-collapse). Right: cross-model comparison at $N=16$. “avg n ” is the expected number of samples actually drawn under early stopping (Eq. 5).

N	acc.	4-num	avg n	Model	best-of-16	4-num	avg n
1	68.0%	46.4%	1.00	SFT	80.0%	61.5%	5.71
2	75.4%	56.1%	1.32	RLOO	74.0%	57.7%	5.26
4	80.4%	63.8%	1.78	OPD (hybrid)	78.0%	61.5%	4.88
8	83.3%	68.3%	2.51	OPD + anti-collapse	84.0%	69.2%	3.80
16	84.0%	69.2%	3.80	<i>7B teacher (ref.)</i>	84.0%	73.1%	4.36

— the policy moves further from the sampling snapshot within each step, which is exactly what re-expanding the distribution looks like from the sampler’s perspective.

Test-time verification converts coverage into accuracy. Table 2 and Figure 4 quantify Extension 2. Three observations:

- *Test-time compute is cheap here.* 80%+ accuracy arrives by $N=4$ at an average cost of 1.78 samples; the full budget of 16 costs only 3.80 samples on average, because easy problems stop after one draw.
- *It helps most where the policy is weakest.* 4-number accuracy rises from 46.4% to 69.2%.
- *It makes diversity visible.* SFT, the most diverse policy, gains +44.6 points from $N=16$ while RLOO gains only +14.2 — and the anti-collapse policy uniquely combines a high intercept with a steep curve.

Training-time diversity and test-time compute are complements, not substitutes.

5.2 Qualitative analysis

What a successful search looks like. The anti-collapse policy retains SFT’s structured search style — propose, evaluate, backtrack — but searches longer and more systematically. On the 4-number problem $[67, 21, 31, 20] \rightarrow 98$ (solved in 3 of 16 samples; the live selector found a solution on draw 4), the successful rollout tries 16 distinct partial combinations before finding a path:

```

<think> ...11. Let's try: 67 + 21 = 88, 88 + 31 = 119 (too high) ... 14. Final attempt: 20 *
21 = 420, 420 / 67  $\approx$  6.3 ... Final try: 21 - 20 = 1; 1 * 31 = 31; 31 + 67 = 98 (our target!)
</think>
<answer> (21 - 20) * 31 + 67 </answer>
[reward 1.0]

```

The decisive move — manufacturing a 1 from $21 - 20$ to neutralize two numbers — is a trick that appears in teacher rollouts; it is rare in SFT-only generations. This is the kind of solution *path* (not just solution) that token-level distillation can transfer but sparse RL cannot.

The dominant failure mode: constraint anchoring. All 8 never-solved test problems are 4-number. The failures are not arithmetic errors — they are search failures under the use-every-number-exactly-once constraint. Representative example ($[83, 78, 1, 39] \rightarrow 82$):

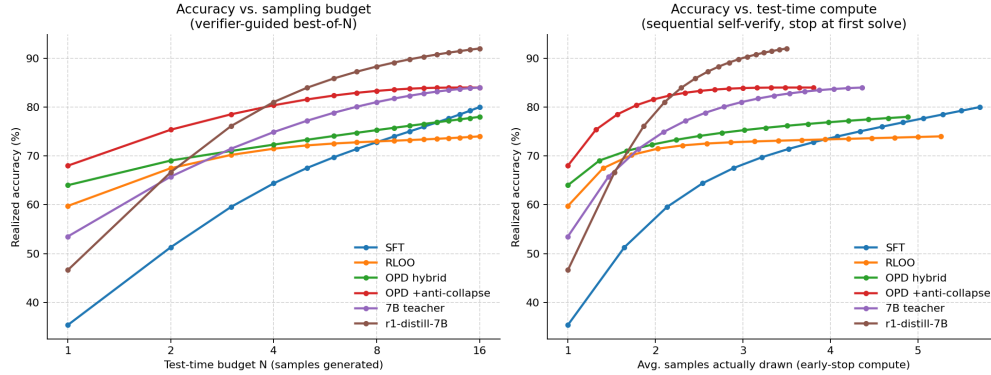


Figure 4: **Test-time scaling.** Realized accuracy vs. sampling budget (left) and vs. *actual* early-stop compute (right). The anti-collapse policy dominates the small-model frontier on both axes: it starts highest at $N=1$ and keeps climbing, while RLOO’s collapsed distribution flattens by $N \approx 8$. On the compute axis, sharp-but-diverse means cheap: 84% accuracy at ~ 3.8 samples/problem.

```

<think> 83 - 1 = 82! This is exactly what we need. ... We can incorporate 39 and 78 in a way
that doesn't affect this result. ... [12 more attempts, none revisiting the division 78 / 39]
</think>
<answer> (83 - 78) * 1 + 39 </answer>
[= 44 ≠ 82; reward 0.1]

```

The model finds the seductive two-number solution $83 - 1 = 82$ immediately, anchors on it, and never explores the multiplicative structure that actually solves the problem ($83 - 78/39 + 1 = 82$). All 16 samples for this problem share the anchor. This is residual mode collapse in miniature: the failure is not that no path exists, but that the policy’s search distribution never visits it — evidence that the remaining headroom is still a *coverage* problem.

Where each training stage leaves its fingerprint. SFT rollouts are diverse but unfocused (long meanders, frequent arithmetic slips late in the trace). RLOO rollouts are short and confident, reusing a small set of opening moves — visibly sharper, and on hard problems, repetitively wrong in the same way. OPD restores teacher-style moves (working backwards from factorizations of the target). Anti-collapse rollouts are the only ones where the 16 samples of a hard problem explore visibly different opening strategies; formatting failures also nearly vanish (0.1% of samples), so the entropy bonus did not degrade output discipline — exploration went into the *search*, not into noise.

5.3 What did not work: tool-integrated reasoning and multi-agent self-play

Alongside the extensions above, we implemented the two extensions from our original proposal — tool-integrated reasoning and multi-agent self-play — and evaluated both under the same fixed protocol. Neither outperformed the pipeline of §3.3–3.4. We report them here as negative results, because *why* they fail on Countdown at the 0.5B scale is itself informative.

Tool-integrated reasoning. We built a complete TIR pipeline that trains the RLOO checkpoint to autonomously call an arithmetic calculator during chain-of-thought rollouts, drawing on Tool-Star [17] and Search-R1 [16] but introducing several task-specific innovations. Although the final numbers are a negative result on Countdown, the infrastructure contributions are reusable and the failure diagnosis is itself informative.

Training pipeline. The pipeline has two phases. **Phase 1 (cold-start SFT):** Starting from the RLOO checkpoint, we fine-tune on ~ 200 synthetic demonstrations that teach the `<calc>/<result>/<answer>` tool-calling protocol. **Phase 2 (TIR-RLOO):** We run online RL with the same RLOO objective, replacing the single-turn sampler with a multi-turn `TIRSamplingWorker` that executes tool calls mid-rollout (up to 3 turns per trajectory).

Innovation 1: AST-sandboxed zero-overhead evaluator. Rather than launching a subprocess per tool call (as in the standard code-executor pattern), we implemented a pure-Python AST-whitelist evaluator that restricts the computation graph to numeric nodes (`ast.Add`, `ast.Mult`, `ast.Pow`,

etc.) and rejects all imports, I/O, and function calls at parse time. This eliminates process-creation overhead entirely, enabling thousands of concurrent evaluations during RL rollouts with zero latency and absolute safety — a critical requirement when every training step executes $\text{batch} \times \text{group_size} \times \text{max_turns} = 4 \times 8 \times 3 = 96$ tool calls.

Innovation 2: Constructive rich feedback. Unlike standard calculator tools that return only the numeric result, our environment injects *structured natural-language feedback* inside each `<result>` block: (i) a **constraint check** reporting whether each given number was used exactly once (e.g. “INVALID: 56 missing; 40 used 2×”), and (ii) a **target-alignment signal** stating whether the result equals the target (e.g. “result 42 \neq target 82 — please try a different expression”). This gives the 0.5B model a much richer intermediate supervision signal than a bare scalar, significantly lowering the reasoning burden for self-correction.

Innovation 3: Curriculum training on harder problems with $O(1)$ problem generation. Our initial experiments on standard 3–4 number Countdown showed that the RLOO baseline already solves $\sim 76\%$ of problems via pure CoT — the task is too easy for a calculator tool to provide meaningful leverage. We hypothesized that tool use becomes valuable only when the arithmetic complexity exceeds what the model can reliably compute in its head. To test this, we introduced a **mixed-difficulty curriculum** that blends easy (3-number), medium (4-number), and hard (5-number) problems during RL training (default mix: 30/50/20%). The harder 5-number problems involve a combinatorial search space $\sim 25\times$ larger than 3-number ones ($5! \times 4^4$ vs. $3! \times 4^2$), making arithmetic verification and iterative refinement far more relevant. To supply this curriculum with unlimited hard problems at negligible cost, we generate solvable 5-number instances by *forward construction*: randomly permute 5 numbers, randomly insert operators, and evaluate the expression to obtain the target. This guarantees a valid solution exists in $O(1)$ time per problem (vs. $O(5! \times 4^4)$ for brute-force verification), enabling scalable hard-problem generation without any search.

Innovation 4: Synthetic retry demonstrations for self-correction. Approximately 50% of cold-start demonstrations deliberately include a *wrong first attempt*: the model proposes an expression that evaluates to \neq target, observes the mismatch in the `<result>` feedback, reasons about the error in a second `<think>` block, and then produces the correct expression. This “fail-then-fix” pattern teaches the model *before RL* how to consume negative feedback and iterate — a capability that standard single-attempt demonstrations never provide.

Innovation 5: Precision token masking (Search-RL style). Tokens inside `<result>...</result>` blocks are injected by the environment, not generated by the policy. We build a per-token binary mask by mapping character-level `<result>` spans back to token positions via the tokenizer’s offset mapping, and set `is_response_token = 0` for all masked positions. This ensures the policy gradient’s credit assignment falls exclusively on the model’s own decisions (`<think>`, `<calc>`, `<answer>`).

Results. Table 3 compares the TIR model (v13, 300 RL steps after cold-start) against the RLOO baseline under the fixed evaluation sampler ($K=8$).

Table 3: TIR vs. RLOO baseline on easy (3–4 number) and hard (5-number) Countdown problems.

Model	Mode	pass@8	pass@1
<i>Easy problems (3–4 numbers)</i>			
RLOO baseline	no tool	0.76	0.60
RLOO baseline	w/ tool	0.74	0.58
TIR (v13)	no tool	0.44	0.10
TIR (v13)	w/ tool	0.56	0.18
<i>Hard problems (5 numbers)</i>			
RLOO baseline	no tool	0.00	—
TIR (v13)	no tool	0.04	—
TIR (v13)	w/ tool	0.04	—

On easy problems the TIR model *under-performs* the RLOO baseline by a wide margin (pass@8: 0.56 vs. 0.76 with tool, 0.44 vs. 0.76 without). On hard 5-number problems neither model solves more than a single instance. However, the within-model comparison shows that tool access consistently *helps*: pass@8 rises from 0.44 to 0.56 (+27% relative) on easy problems when tools are enabled,

confirming that the pipeline’s innovations (rich feedback, retry training, token masking) do produce a functional tool-using policy.

Diagnosis: why tools do not beat pure CoT. The negative headline result has four root causes, all traceable to the task–model–tool interaction at 0.5B scale:

1. **Arithmetic is not the bottleneck.** On 3–4 number Countdown, the RLOO baseline already reaches $\sim 76\%$ pass@8 with pure chain-of-thought. The dominant failure mode is *combinatorial search* — not finding the right arrangement of operations — and a calculator that *verifies* a proposed expression does not expand the search space. The rich feedback (Innovation 2) correctly identifies constraint violations, but the 0.5B model cannot leverage this signal to redirect its search.
2. **Cold-start SFT erodes task ability.** Teaching the `<calc>` format via SFT on 200 synthetic examples degrades the model’s CoT quality acquired during RLOO (no-tool pass@8 drops from 0.76 to 0.44). The 0.5B model lacks the capacity to maintain both the tool-calling protocol and fluent arithmetic reasoning simultaneously — a capacity bottleneck that would not arise at larger scales.
3. **Tool tokens consume the context budget.** Each `<calc>...<result>...` round adds ~ 50 –100 tokens of overhead (expression + rich feedback). With a 1024-token generation limit, tool-using rollouts have fewer tokens available for the combinatorial search that actually solves the problem.
4. **The tool-use reward bonus is a lose-lose dilemma.** We swept the tool-use bonus across three runs: at +0.05 (Run 1) the signal is too weak and the model never emits `<calc>` tags at all (tool_rate = 0%); at +0.30 (Run 2) the model calls the tool on every problem (tool_rate = 100%) but pass@8 drops from 0.66 to 0.42 because the inflated bonus distorts the outcome reward — the policy is rewarded for *calling* the tool regardless of whether the call helps, crowding out the signal for actually solving the problem. At the intermediate value +0.15 (Run 3) tool_rate falls back to 0%, suggesting no sweet spot exists: tool calling is either absent or accuracy-destroying, because the underlying task does not reward tool use with better outcomes.

Multi-agent self-play. Our first attempt used a traditional multi-agent judge to verify the model outputs, but this was not a good fit for our setting. Many tasks were already easy to verify directly, so adding a judge did not provide much extra useful signal and mostly added cost and noise, which is a known risk in LLM-as-a-judge evaluation (Zheng et al., 2023[18]). To make the method more novel, we then implemented a proposer agent: after each batch, it summarized the failures from the previous batch and proposed similar new problems that were expected to have high reward in the next batch based on previous rollouts. The hope was that this would create a useful feedback loop, where the system would automatically focus training on the kinds of cases where the model still failed. However, this also did not work well, as the model was already trained for the task of reasoning and it could hardly be changed due to the limited size of the model. In the end, the failure seems to come from a mismatch between the method and the task: the task was too easy for judge-based verification to matter, but the proposer-based novelty was not strong enough to create a better learning signal.

Take-away. Both extensions attack constraints that are not the binding bottleneck at 0.5B scale on Countdown. Tool-integrated reasoning eliminates arithmetic errors and provides rich, constructive mid-rollout feedback, but arithmetic was never the primary failure mode — combinatorial search coverage was. Similarly, our multi-agent curriculum extension attempts to create harder and more useful training data by using a proposer agent to summarize failures from the previous batch and generate similar problems expected to yield high reward in the next batch. However, this mechanism also fails to address the core bottleneck. Countdown solutions are easy to verify once proposed, so an additional judge-style verification loop provides little new signal; meanwhile, the 0.5B proposer cannot reliably turn past failures into valid, usefully harder problems. In practice, the generated problems often repeat surface patterns from prior failures without expanding the student’s solution distribution in a meaningful way. In contrast, on-policy distillation from a frozen 7B teacher (§3.3) directly injects new solution *paths* into the student’s distribution, which is precisely the resource the policy is missing.

Nevertheless, these failed extensions contribute reusable infrastructure and clarify when such methods are likely to help. The TIR pipeline would transfer to settings where tools *do* address the bottleneck: the AST-sandboxed zero-overhead evaluator, the Search-R1-style precision token masking, the constructive-feedback environment, and the synthetic retry demonstrations all generalize beyond Countdown. The multi-agent proposer likewise suggests a potential curriculum mechanism for domains where failures are harder to verify directly and where a generator can reliably produce valid follow-up tasks. The key lesson is that auxiliary mechanisms at small scale must target the *right* missing resource: at 0.5B on Countdown, the model needs new search paths, which distillation provides, not arithmetic verification from a calculator or near-neighbor problem proposals from a weak generator.

6 Discussion

Limitations.

1. The test set is 50 problems and each configuration is a single training run; deltas of ± 2 –4 points are meaningful, but error bars from seed/temperature sweeps would strengthen the claims.
2. The anti-collapse ablation is entangled: without T^2 rescaling of the KL, raising the distillation temperature both softens targets and shrinks the KL gradient. A compensated ablation would isolate the two effects.
3. The final importance-weight mean (≈ 12.4 against a clip of 10) shows the diversity knobs increase within-step off-policy drift. Our run was stable, but pushing λ_H or T further likely requires more frequent sampler refreshes.
4. Test-time self-verification relies on the task’s intrinsic verifiability. The protocol is honest — it uses only the prompt’s own numbers and target, never a reference solution — but it does not transfer to tasks without checkable outputs.
5. The 4-number ceiling stands: 31% of 4-number problems are never solved in 16 tries.

Difficulties encountered. The recurring engineering theme was off-policyness between the vLLM sampler and the HF policy: unclipped importance weights destabilized early RLOO runs (hence the clip-at-10 fix), and mismatched vocab padding between the 0.5B and 7B Qwen variants silently corrupted the first distillation KL implementation until truncated to the shared vocabulary. The recurring scientific theme was that single-number evaluation misleads: several configurations looked like clear wins on pass@1 while regressing pass@16, which we only caught because the evaluation pipeline always recorded all 16 samples.

Broader impact. The result pattern — small model + teacher distribution + diversity regularization + cheap verifier \approx large-model coverage — is a favorable recipe for deploying capable small models, with corresponding energy/cost benefits. The methods are task-agnostic wherever verifiable rewards exist (code, math, formal proofs), though the same recipe could sharpen models on less benign verifiable objectives; nothing here is specific to that risk beyond standard dual-use considerations.

7 Conclusion

We pushed a 0.5B model through the full modern post-training pipeline on Countdown and found that the standard story — each stage raises accuracy — hides a distributional regression: RL sharpens the policy at the expense of the sampling diversity that test-time compute needs. Treating diversity as a first-class objective, we showed that on-policy distillation with two targeted knobs (a $10\times$ entropy bonus and a softened KD temperature) recovers the lost coverage and converts it into additional single-shot accuracy, reaching pass@1 = 0.680 / pass@16 = 0.840 — teacher-level coverage at $14\times$ fewer parameters — and 84% realized accuracy under verifier-guided best-of- N at 3.8 samples per problem. The take-home message generalizes beyond Countdown: *measure pass@k, not just pass@1, whenever you intend to spend test-time compute — and if pass@k is eroding, cheap distributional regularizers can buy it back without sacrificing sharpness.* Our negative-result extensions reinforce this message: tool-integrated reasoning and self-play attack axes (arithmetic accuracy, curriculum

difficulty) that are not the binding constraint at this scale — further evidence that the coverage gap is the central bottleneck. The clearest future direction is to attack the residual 4-number coverage gap by rejection-sampling correct traces from a stronger teacher and retraining the student on solution paths it cannot currently discover, closing the loop between distillation and exploration.

8 Team Contributions

- **Qi Wu:** Led the SFT and IPO stages: implemented the masked next-token objective and the IPO loss (policy/reference log-probabilities, β sweep), ran hyperparameter sweeps on Modal, and produced the SFT/IPO evaluations and training-dynamics plots. Co-designed the anti-collapse configuration (entropy/temperature knobs) and ran the anti-collapse training and evaluation.
- **Juntao Cheng:** Led evaluation: the pass@k analysis framework, cross-method comparison and figures, the verifier-guided self-verification module. Implemented the on-policy distillation trainer (teacher loading, temperature-scaled forward KL, hybrid objective) and the OPD experiments. Implemented and tested multiagent workflow for proposer and verifier.
- **Zhuoang Tao:** Led the RLOO pipeline: per-sample log-probability computation, leave-one-out advantage estimation, importance-weight clipping, and entropy/KL regularization; owned rollout generation, verifier integration, and RL stability tuning. Led the tool-integrated reasoning extension: the two-phase TIR trainer (cold-start SFT with synthetic retry demonstrations + TIR-RLOO with multi-turn rollouts), the AST-sandboxed calculator, the Search-R1-style token masking, and the constructive-feedback environment; exploring different hyperparameter configurations and evaluating the models on both easy and hard problem sets. Coordinated the poster and final report.

Changes from Proposal Our proposal described a multi-agent self-play system with tool-integrated reasoning, in which a generator agent creates progressively harder verifiable tasks and a solver agent learns to use external tools (Python, calculator, terminal). We did implement and evaluate both components — but neither improved over the required pipeline on Countdown at the 0.5B scale (§5.3 reports the setups, numbers, and our diagnosis). Meanwhile, our milestone results surfaced a concrete, measurable open problem inside the default project itself: RL was eroding the sampling diversity that test-time methods depend on. We therefore shifted the project’s main contribution to (1) on-policy distillation with explicit diversity preservation and (2) verifier-guided test-time self-verification, and we report the proposal’s original directions as negative results. The final report thus retains the proposal’s core themes — verifiable rewards, signals beyond final-answer correctness, and adaptive difficulty (our gains concentrate on the hardest 4-number problems) — while replacing the multi-agent curriculum with a teacher-student curriculum that we could implement, ablate, and evaluate rigorously.

References

- [1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [2] 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.
- [3] 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.
- [4] 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.

- [5] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [6] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Zhang. Tinyzero: Reproducing deepseek r1-zero on the countdown task. <https://github.com/Jiayi-Pan/TinyZero>, 2025.
- [7] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.
- [8] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [9] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- [10] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.
- [11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [13] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.
- [14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [15] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [16] Bowen Jin et al. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. 2025.
- [17] Junjie Guo et al. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. 2025.
- [18] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL <https://arxiv.org/abs/2306.05685>.

Appendix

A Full Hyperparameter Tables

Table 4: SFT and IPO configuration.

Hyperparameter	SFT	IPO
Initialization	Qwen2.5-0.5B (base)	SFT checkpoint
Dataset	cog_behav_all_strategies	countdown_tasks_3to4-dpo
Learning rate	5×10^{-5}	5×10^{-6}
LR schedule	cosine, warmup 0.05	cosine, warmup 0.05
Epochs	6	1
Batch size \times grad. accum.	64×8 (eff. 512)	64×16
IPO β	—	0.05 (selected over 0.1)
Weight decay / grad clip	0.01 / 1.0	0.01 / 1.0
Max prompt / response length	512 / 1024	512 / 1024
Precision / hardware	bf16, 1 \times H100 (Modal)	bf16, 1 \times H100 (Modal)

Table 5: RLOO, OPD, and anti-collapse configuration. Bold marks the only two values that differ between OPD and anti-collapse.

Hyperparameter	RLOO	OPD (hybrid)	Anti-collapse
Initialization	SFT checkpoint	RLOO-style init	best OPD checkpoint
Teacher	—	7B SFT (frozen)	7B SFT (frozen)
λ_{KL} / KL direction	—	0.5 / forward	0.5 / forward
Distillation temperature T	—	1.0	1.3
λ_{RL}	1.0	0.5	0.5
Entropy coefficient λ_H	0.001	0.001	0.01
KL-to-sampler coefficient	0.001	—	—
IS-weight clip w_{max}	10	10	10
Learning rate (constant)	10^{-5}	10^{-5}	10^{-5}
Batch \times group size	128×8	128×8	128×8
Training steps	200	100	150
Rollout sampler	temp 1.0, top- p 1.0, top- k off	same	same
Max rollout tokens	1024	1024	1024
Weight decay / grad clip	10^{-4} / 1.0	10^{-4} / 1.0	10^{-4} / 1.0

Evaluation sampler (fixed for all reported numbers). Temperature 0.6, top- p 0.95, top- k 20, min- p 0.0, 16 samples per prompt, max 1024 generated tokens, vLLM backend.

Table 6: TIR cold-start and RL configuration (best run: v13).

Phase	Hyperparameter	Value
Cold-start SFT	Base model	RLOO checkpoint
	# demonstrations	200 (50% with retry)
	Learning rate	5×10^{-6}
	Epochs	4
TIR-RLOO	Batch \times group	4×8
	Learning rate	1×10^{-6}
	Tool-use bonus	+0.15
	Error penalty	-0.10
	Max tool turns	3
	Training steps	300
	Token masking	Search-R1 style

B Training Dynamics: Full Diagnostics

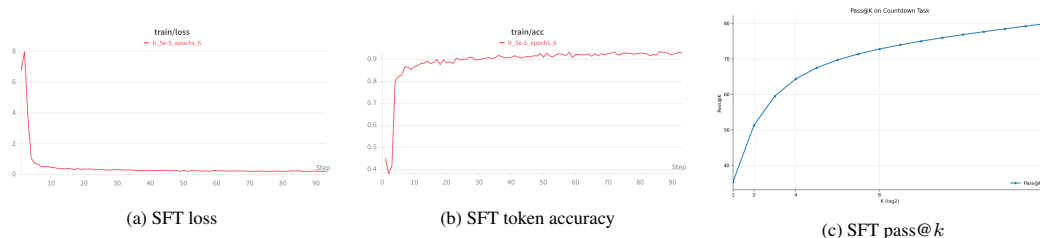


Figure 5: SFT dynamics: loss falls from $\sim 7\text{--}8$ to ~ 0.15 and token accuracy reaches ~ 0.93 within ~ 92 optimizer steps; the wide pass@1–pass@16 gap (35.4 vs. 80.0) shows correct solutions are present but not prioritized.

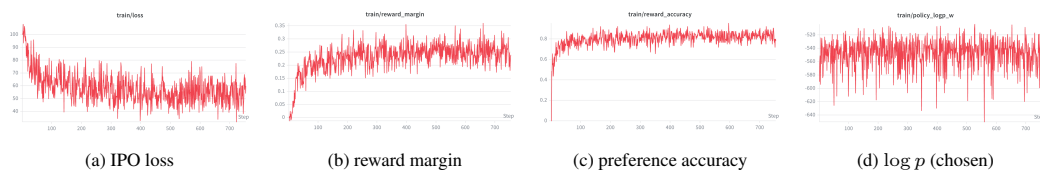


Figure 6: IPO dynamics: the bounded loss keeps the chosen/rejected margin (b) growing smoothly without the runaway divergence DPO exhibits on deterministic preferences, while preference accuracy (c) saturates. Chosen log-probabilities (d) stay near the reference — IPO reweights rather than re-learns. (Rejected log p and pass@ k : plots/ipo_plots/.)

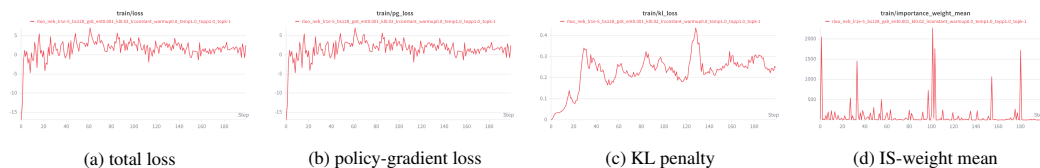


Figure 7: RLOO diagnostics. The clipped importance weight (d) is the key stability monitor: it stays near 1 through most of training, indicating the policy remains close to the vLLM sampling snapshot between refreshes.

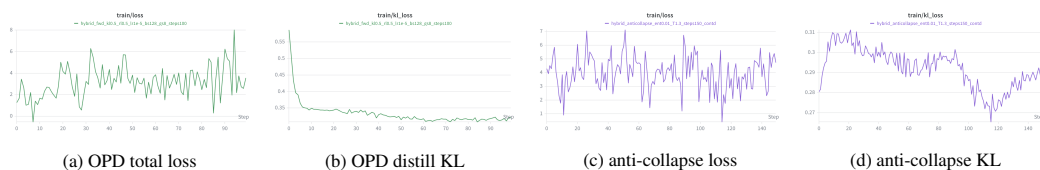


Figure 8: OPD and anti-collapse diagnostics. The distillation KL (b, d) decreases as the student absorbs the teacher’s per-token distribution; under $T = 1.3$ the KL sits lower in scale (softened targets) while the training reward (Fig. 3c) breaks the hybrid run’s plateau. The anti-collapse run’s IS-weight mean drifts to ≈ 12.4 by the final step — stable here, but the monitor to watch before pushing the diversity knobs further.

C Sample Rollout (Full Text)

Full successful rollout from the anti-collapse policy on $[67, 21, 31, 20] \rightarrow 98$, illustrating the structured search style (abridged mid-trace attempts marked with . . .):

<think>

Let me analyze this step by step:

```
1. First, let's try to work backwards from 98:
- 98 is  $2 * 49$ , so getting close to 98 might be useful
- 98 could be reached by adding a small number to something close to 98

2. Looking at the numbers:
- 67 is closest to 98
-  $67 + 21 = 88$  (too low)
-  $67 + 20 = 87$  (too low)
-  $21 + 20 = 41$  (too low)

3. Let's try:
 $67 + 21 = 88$ 
 $88 + 31 = 119$  (too high)
...
12. Let's try:
 $21 - 20 = 1$ 
 $67 + 31 = 98$ 
 $98 - 1 = 97$  (still not our target)
...
Final try:
 $21 - 20 = 1$ 
 $1 * 31 = 31$ 
 $31 + 67 = 98$  (our target!)
</think>
<answer>  $(21 - 20) * 31 + 67$  </answer>
```

Note the near-miss at attempt 12: the model has all the pieces ($21 - 20 = 1$ and $67 + 31 = 98$) but combines them subtractively; four attempts later it finds the multiplicative composition. Under the collapsed RLOO policy, the 16 samples for this problem repeat early additive attempts and never reach this composition.

TIR rollout example (characteristic failure). Problem: [44, 56, 40, 38], target 82.

```
<think>I need to find:  $44 \text{ op } 56 \text{ op } 40 \text{ op } 38 = 82$ . Let me try adding
some numbers:  $44 + 38 = 82!$  Let me verify.</think>
<calc> $44 + 38$ </calc>
<result>82 [INVALID: 56 missing; 40 missing]</result>
<think>The result is 82 but I haven't used 56 and 40. I need to
incorporate them without changing the result...  $56 - 40 = 16$ ...  $44$ 
+  $38 + 16 = 98$ , too high.</think>
<answer> $44 + 38$ </answer>
[reward 0.1 — correct value but constraint violated]
```

The calculator correctly verifies the arithmetic, and the constructive feedback identifies the missing numbers, but the model cannot find a way to incorporate them. The tool solves the *verification* sub-problem but not the *search* sub-problem.

D Reproducibility

All training runs execute on Modal (single H100) via the launchers in the project repository: `sft_trainer/train_sft_modal.sh`, `ipo_trainer/train_ipo_modal.sh`, `rloo_trainer/train_rloo_modal.sh`, `distill_trainer/train_distill_modal.sh`, and `distill_trainer/train_anticollapse_modal.sh` (whose defaults encode the anti-collapse configuration exactly). Evaluation uses `evaluation/countdown_eval.py` with the fixed sampler; passing `--test_time` additionally runs the self-verification selector and writes the accuracy/compute curves. Figure 2 and Figure 4 regenerate from the stored evaluation dumps via `evaluation/test_time_inference.py`; the verifier reproduces all stored scores with zero mismatches under `--recompute`.