THE CATFISH EFFECT: HETEROGENEOUS COMPETITION IN GROUP-RELATIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning systems do not evolve in isolation: their progress is shaped by the populations against which they compete. Yet most reinforcement learning methods focus on improving the learner itself, while largely treating the composition of its competitive population as fixed. This raises a broader question: can introducing a heterogeneous competitor reshape learning without providing demonstrations or direct supervision? We investigate this question through Catfish GRPO, a group-relative reinforcement learning framework that introduces a frozen, moderately stronger model as a competitor rather than a teacher. The Catfish contributes only a scalar task outcome to reshape relative learning signals and guide rollout allocation; its generations, tokens, log probabilities, and parameters never enter the student loss. In our main comparison, Catfish GRPO exceeds the strongest evaluated baseline by up to 2.5 percentage points in-domain and 3.2 percentage points out-of-domain. Beyond performance, we study how heterogeneous competitors alter learning dynamics. Catfish encourages greater behavioral diversity early in training, increasing the chance that the student discovers correct on-policy trajectories on difficult problems and expanding the learning opportunities available in later stages. The quality of the competitor also plays a critical role: weaker or poorly matched competitors can substantially diminish the benefit, while stronger competitors are not necessarily better. Together, these findings position Catfish GRPO not only as an RL method, but also as a controlled testbed for studying how population composition and heterogeneous competition shape learning and co-evolution.

1 INTRODUCTION

What a learning system discovers depends not only on its optimization objective, but also on the population against which it competes. This principle is well established in population-based and multi-agent learning, where changing the opponents, co-players, or population composition can substantially alter the trajectories that agents discover and the behaviors they retain (Lanctot et al., 2017; Balduzzi et al., 2019; Vinyals et al., 2019; Bettini et al., 2024b). Yet reinforcement learning (RL) post-training of large language models (LLMs) remains largely base-model-centric and typically treats the composition of the comparison group as fixed.

This limitation is especially important in Group Relative Policy Optimization (GRPO) (Shao et al., 2024). GRPO normalizes outcome rewards within each group of rollouts to estimate a sequence-level scalar advantage; however, hard questions with zero successful rollouts produce zero advantage and are silently discarded (Le et al., 2026). Recent work has improved reinforcement learning with verifiable rewards through better sampling, advantage estimation, and external guidance (Yu et al., 2025; Lee et al., 2026; Feng et al., 2025; Wang et al., 2026b). These methods predominantly ask how to obtain a better learning signal from the base model's own trajectories or from additional supervision.

We instead ask a complementary question: *can introducing a heterogeneous competitor reshape learning without providing direct supervision*? We introduce Catfish GRPO, a group-relative reinforcement learning framework that introduces a heterogeneous competitor into the comparison population without using its response as a target. The Catfish contributes only a scalar task outcome to reshape relative learning signals and guide rollout allocation. The base model therefore learns en-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

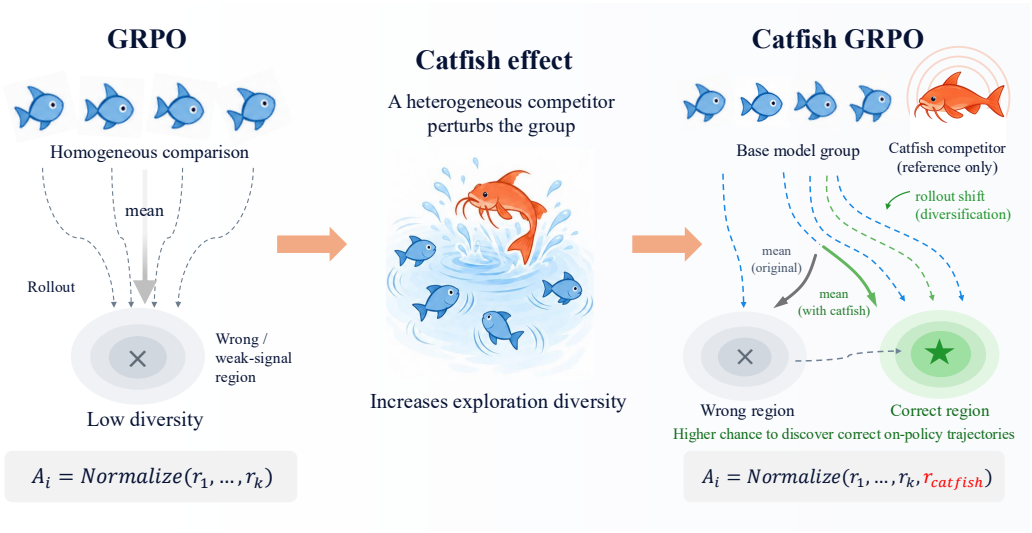


Figure 1: The Catfish effect in group-relative learning. A heterogeneous competitor perturbs an otherwise homogeneous comparison population, increasing behavioral diversity and the chance of discovering correct on-policy trajectories on difficult prompts.

tirely from its own on-policy trajectories. Our results show that the resulting gain depends on where the Catfish enters the update, inducing distinct trade-offs between reinforcing correct responses and suppressing incorrect ones. Moreover, a stochastic virtual Catfish replaces prompt-level competitor generation and verification with fixed conditional success probabilities. It removes competitor inference and maintains a controlled capability advantage as the base model improves. These findings establish comparison-population composition, rather than supervision transfer, as an important design axis for language-model reinforcement learning.

Our contributions are summarized as follows:

- We introduce **Catfish GRPO**, a simple competitor-augmented framework that improves group-relative RL without demonstrations, distillation, or direct teacher supervision, and achieves the strongest performance among the GRPO-style methods evaluated in our setting. Virtual competitors recover substantial gains without competitor inference.
- We show that heterogeneous competition changes the **temporal dynamics of learning**: Catfish increases early behavioral diversity and successful on-policy exploration on difficult problems, which subsequently translates into improved task accuracy.
- We provide a systematic analysis of **population composition as a training mechanism**, showing that the effect depends on competitor strength, signal placement, and training stage. We further show that the effect depends critically on the relative capability and placement of the competitor, establishing population composition as an important axis in reinforcement learning design.

2 Preliminaries

Given a prompt x, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples G responses $\{y_i\}_{i=1}^{G}$ from a policy π_θ and assigns each response a verifiable reward r_i. GRPO estimates each response's advantage relative to the other responses sampled for the same prompt:

$$\hat{A}_i = \frac{r_i - \mu(\mathbf{r})}{\sigma(\mathbf{r})}, \qquad \mathbf{r} = \{r_1, \ldots, r_G\}. \tag{1}$$

Here, μ and σ denote the population mean and standard deviation; flat reward sets are assigned zero advantage. For binary rewards, an all-wrong group has $\mathbf{r} = \{0, \ldots, 0\}$ and hence zero advantage. It

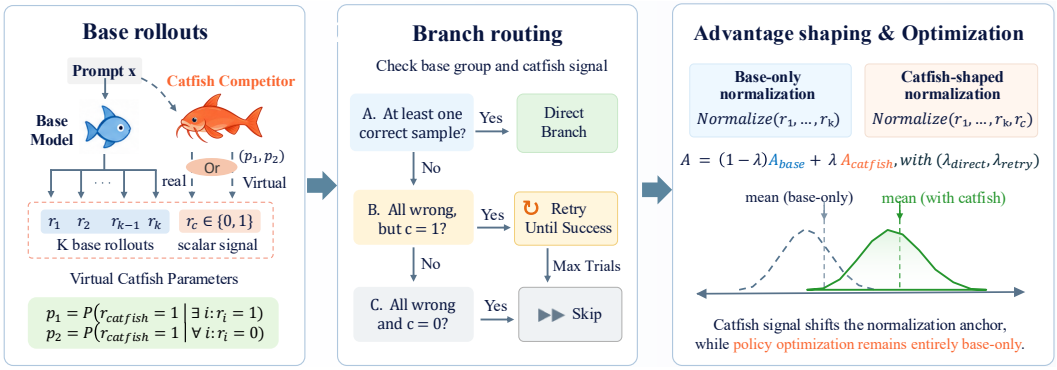


Figure 2: Overview of Catfish GRPO. A real or virtual Catfish supplies only a scalar outcome to the base-model rollout group. The signal determines whether all-wrong prompts receive additional retries and, through $(\lambda_{\text{direct}}, \lambda_{\text{retry}})$, whether Catfish-augmented normalization is applied on the direct and retry branches. Only base-model trajectories are optimized; Catfish generations, log probabilities, and parameters never enter the policy loss.

produces no policy-gradient signal precisely when the current base model fails consistently on the prompt (Yu et al., 2025).

3 Catfish GRPO

3.1 Catfish GRPO

As illustrated in Figure 2, Catfish GRPO augments the homogeneous group of base-model outcomes with the binary outcome of a heterogeneous competitor. Given $c(x) \in \{0, 1\}$, we form $\mathbf{r}^c = \{r_1, \ldots, r_G, c\}$ and define

$$\hat{A}_i^c = \frac{r_i - \mu(\mathbf{r}^c)}{\sigma(\mathbf{r}^c)}, \qquad A_i(\lambda) = (1-\lambda)\hat{A}_i + \lambda \hat{A}_i^c. \tag{2}$$

Only base-model trajectories enter the policy loss: the Catfish generation, tokens, log probabilities, and parameters are never optimized. The Catfish therefore changes the reference population without becoming a supervision target.

We separate the initial, or *direct*, path from the *retry* path. λ_{direct} applies when the initial group contains at least one correct response. If the initial group is all wrong, we do not optimize the Catfish-induced negative-only group. Instead, $c(x) = 1$ permits up to J additional groups. We use the first retry group containing a correct response, with λ_{retry} and the same Catfish outcome $c = 1$. If no retry succeeds, or the selected group has zero advantages, it contributes no loss. Thus, $(0, 0)$ still uses Catfish-guided retry allocation and is not Vanilla GRPO. In our placement ablations, each coefficient is binary; Equation 2 also permits continuous interpolation with $\lambda \in [0, 1]$.

The distribution shifts induced by λ. The two coefficients redistribute probability mass between correct and incorrect base-model responses on different paths. To see this, note that the augmented advantages sum to zero only after including the Catfish. Because its trajectory is excluded from optimization,

$$\sum_{i=1}^{G} \hat{A}_i^c = -\hat{A}_c^c, \qquad \sum_{i=1}^{G} A_i(\lambda) = -\lambda \hat{A}_c^c, \tag{3}$$

where $\hat{A}_c^c$ is the normalized Catfish advantage. For a mixed base group with a correct Catfish outcome, setting $\lambda = 1$ lowers the positive advantage of correct base-model responses and increases the magnitude of the negative advantage on incorrect responses, relative to base-only normalization. It therefore acts primarily by suppressing incorrect response modes. Setting $\lambda = 0$ instead assigns

comparatively more positive update to correct responses. When the Catfish is wrong, the direction reverses; the expected direct-path shift is therefore controlled by its conditional success rate.

For a correct Catfish outcome, this yields the following interpretation of the four placements:

- $(0, 0)$ emphasizes positive base-model discoveries on both paths. The Catfish changes which all-wrong prompts receive retries, but not their advantage normalization.
- $(0, 1)$ retains positive emphasis on direct groups, but suppresses incorrect responses more strongly within successful retry groups, at the cost of reducing their positive updates.
- $(1, 0)$ suppresses incorrect modes on ordinary direct groups, while allowing a newly discovered correct retry response to receive the larger base-only positive update.
- $(1, 1)$ applies error suppression on both paths. It produces the strongest avoidance pressure, but can also attenuate useful positive updates most broadly.

The appropriate balance is not universal. It depends on the task distribution, base-model capability, competitor strength, and training stage. The four settings should therefore be understood as a structured 2×2 trade-off over where the update distribution is shifted, rather than as unrelated algorithms.

3.2 Virtual Competitor Reference

A *virtual Catfish* eliminates competitor inference by sampling a conditional Bernoulli variable:

$$\Pr(\tilde{c} = 1 \mid \mathbf{r}) = \begin{cases} p_1, & \max_i r_i > 0, \\ p_2, & \max_i r_i = 0. \end{cases} \tag{4}$$

Here, p_1 is the competitor's success probability conditional on an initial base group containing at least one correct response; p_2 is its success probability conditional on an all-wrong group. Thus, p_1 controls the expected direct-path shift and p_2 controls how often all-wrong groups trigger retries. The outcome is sampled once per initial group and retained throughout retries.

Anchor competitor and reference settings. A *real Catfish* is a frozen policy π_c with prompt-matched outcome

$$c(x) = R(x, y_c), \qquad y_c \sim \pi_c(\cdot \mid x). \tag{5}$$

Here, R is the binary task verifier. Only $c(x)$ is retained. A real Catfish preserves prompt-specific competence, but requires online generation and verification or an offline outcome cache.

We define an *anchor competitor* by calibrating (p_1, p_2) from these real base-model–Catfish interactions. On GSM8K, Qwen2.5-7B provides the anchor $(0.85, 0.38)$ for the 3B base model, while Qwen2.5-14B provides $(0.975, 0.71)$ for the 7B base model. We also vary the two probabilities around these anchors to study weaker and stronger conditional reference settings; the Llama experiments use $(0.85, 0.30)$. Holding (p_1, p_2) fixed maintains a controlled conditional reference throughout training and removes the cost of generating and verifying competitor answers. We use $G = 4$ and at most $J = 3$ retry groups, with retry temperatures 0.9, 1.0, and 1.1.

This efficiency trades away prompt-level selection. When a real Catfish solves an all-wrong prompt, its success provides evidence that the prompt may lie near the base model's current learning frontier: unsolved by the base group but solvable by a stronger competitor. A virtual Catfish samples uniformly within the all-wrong pool conditional only on p_2; it can therefore trigger retries on extremely difficult prompts for which the base model is unlikely to discover a correct trajectory, wasting rollout computation. Conversely, conservative p_2 can miss learnable hard prompts. Selecting (p_1, p_2) consequently balances compute efficiency and stable relative strength against the prompt-specific precision of a real competitor.

3.3 Stage-Dependent Placement

Because the all-wrong pool and the relative capability gap evolve during training, a placement that is effective early need not remain optimal. We study a stage-wise schedule that moves Catfish

Algorithm 1 Catfish GRPO with stage-dependent placement

Require: Policy π_θ, prompt x, stage t, group size G, retry limit J, schedules $\lambda_{\text{direct}}(t)$ and $\lambda_{\text{retry}}(t)$

Sample $\{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x)$ and rewards $\mathbf{r} = \{r_i\}_{i=1}^{G}$
$c \leftarrow \textsc{CatfishScore}(x, \mathbf{r})$
if $\max_i r_i > 0$ **then**
    Compute $\{A_i(\lambda_{\text{direct}}(t))\}_{i=1}^{G}$
    if $\max_i |A_i| > 0$ **then**
        Update π_θ using the base-model trajectories
    end if
else if $c = 1$ **then**
    for $j = 1, \ldots, J$ **do**
        Sample a retry group $\{y_i^{(j)}\}_{i=1}^{G}$ and rewards $\mathbf{r}^{(j)}$
        if $\max_i r_i^{(j)} > 0$ **then**
            Compute $\{A_i^{(j)}(\lambda_{\text{retry}}(t))\}_{i=1}^{G}$ and w_j
            if $\max_i |A_i^{(j)}| > 0$ **then**
                Update π_θ with $\{w_j A_i^{(j)}\}_{i=1}^{G}$
            end if
            break
        end if
    end for
end if

normalization from the direct path to the retry path at an epoch boundary τ, while preserving retry allocation:

$$(\lambda_{\text{direct}}(t), \lambda_{\text{retry}}(t)) = \begin{cases} (1, 0), & t < \tau, \\ (0, 1), & t \geq \tau. \end{cases} \tag{6}$$

The first stage suppresses incorrect modes on direct groups while preserving base-only positive updates for newly discovered retry successes. The second stage removes direct-path normalization and applies Catfish normalization to successful retry groups. We also study $(1, 0) \to (0, 0)$ and $(1, 1) \to (0, 1)$, which remove direct-path normalization while keeping retry placement fixed. Such switches are not assumed to dominate persistent placements: the experiments treat scheduling as a task- and model-dependent trade-off.

Algorithm 1 summarizes the procedure.

4 Experiments

4.1 Experimental Setup

Models and tasks. We evaluate Catfish GRPO using Qwen2.5-3B/7B-Instruct (Yang et al., 2024), Llama-3.2-3B-Instruct (Meta AI, 2024), and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). The mathematical reasoning study trains on 4,096 GSM8K (Cobbe et al., 2021) prompts for two epochs and evaluates greedily on GSM8K and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023). The question-answering study trains Qwen2.5-7B-Instruct separately on DROP (Dua et al., 2019) and 2WikiMultiHopQA (Ho et al., 2020), with Quoref (Dasigi et al., 2019) and HotpotQA (Yang et al., 2018) as the corresponding out-of-domain evaluations. Real competitors are frozen Qwen2.5-7B- or 14B-Instruct models whose prompt-level binary outcomes are cached before policy training. Virtual competitors instead draw a binary outcome from two conditional probabilities, $p_1 = \Pr(c = 1 \mid \max_i r_i > 0)$ and $p_2 = \Pr(c = 1 \mid \max_i r_i = 0)$.

Training and evaluation. Unless stated otherwise, the student samples $G = 4$ responses at temperature 0.8 and top-$p = 0.95$; Catfish GRPO permits at most $J = 3$ retry groups. Methods within each model–task block use the same prompt order, reward function, generation parameters, and base-prompt exposure. We report exact-match accuracy on the in-domain/out-of-domain evaluation pair. Each Catfish result in the main comparison uses the task-specific calibrated reference regime:

Table 1: Final GSM8K/MATH-500 exact-match accuracy (%). Each cell reports GSM8K/MATH-500. $\mathrm{Catfish}_{0*}$ averages the 00 and 01 placements, $\mathrm{Catfish}_{1*}$ averages 10 and 11, and $\mathrm{Catfish}_{\mathrm{sw}}$ denotes $10 \rightarrow 01$. Qwen2.5-7B uses the real 14B competitor; Qwen2.5-3B and Llama use calibrated virtual competitors. Bold components are the best observed result within each student model. † is the last checkpoint before a reproduced RL-ZVP divergence.

	Baselines					Ours		
Student	Vanilla	DAPO	RL-ZVP	LENS	AR3PO	$\mathrm{Catfish}_{0*}$	$\mathrm{Catfish}_{1*}$	$\mathrm{Catfish}_{\mathrm{sw}}$
Qwen2.5-3B	69.8/43.7	71.0/45.3	69.2/42.0	69.4/42.3	70.0/44.2	72.7/45.4	**72.8**/44.6	72.7/**46.0**
Qwen2.5-7B	80.6/57.2	84.0/56.6	77.1/45.8	80.6/55.8	80.8/**57.4**	83.2/57.1	78.8/50.0	**84.1**/**57.4**
Llama-3B	70.7/40.7	**75.1**/41.0	69.3/37.5	70.1/42.0	68.8/42.6	73.9/43.3	72.3/40.3	73.6/**45.8**
Llama-8B	81.3/43.1	81.3/42.9	14.0/1.3†	79.8/43.4	80.7/45.6	81.5/**45.8**	80.8/45.1	**83.8**/43.9

Qwen2.5-7B uses the real 14B competitor for both reasoning and QA, while the Qwen2.5-3B and Llama studies use calibrated virtual competitors.

Compared methods. We compare Vanilla GRPO with DAPO, RL-ZVP, LENS, and AR3PO (Shao et al., 2024; Yu et al., 2025; Feng et al., 2025; Le et al., 2026; Zhang et al., 2025) under the same $G = 4$ protocol. For Catfish GRPO, the two-bit code ab specifies whether Catfish participates in advantage normalization on the direct branch (a) and retry branch (b). Catfish-guided retry allocation remains active for all four codes. We use two family summaries in the broad comparison: $\mathrm{Catfish}_{0*}$ averages $\mathrm{Catfish}_{00}$ and $\mathrm{Catfish}_{01}$, while $\mathrm{Catfish}_{1*}$ averages $\mathrm{Catfish}_{10}$ and $\mathrm{Catfish}_{11}$. The stage-wise method $\mathrm{Catfish}_{\mathrm{sw}}$ follows $10 \rightarrow 01$.

4.2 Main Results

4.2.1 Overall Performance

Does heterogeneous competition improve group-relative learning across model families and scales? Table 1 compares the two Catfish configuration families and the stage-wise schedule with five strong baselines on GSM8K and MATH-500. A Catfish variant attains the best observed result on at least one evaluation for every student: $\mathrm{Catfish}_{1*}$ leads Qwen2.5-3B GSM8K, the stage-wise variant leads Qwen2.5-3B MATH and Qwen2.5-7B/Llama-8B GSM8K, and $\mathrm{Catfish}_{0*}$ leads Llama-8B MATH. DAPO remains strongest on Llama-3B GSM8K, while AR3PO ties the stage-wise result on Qwen2.5-7B MATH. The preferred placement therefore varies with both model and evaluation; the result is not a claim that one Catfish configuration dominates universally.

The broad comparison is corroborated by the independent clean-SDPA three-seed control: with a Qwen2.5-3B student and real 7B competitor, $\mathrm{Catfish}_{10}$ reaches 74.45 ± 0.20 on GSM8K and 45.46 ± 3.51 on MATH-500, compared with 72.38 ± 0.43 and 42.41 ± 1.79 for Vanilla GRPO. Taken together, the results show that the comparison population is a useful optimization axis, but they also suggest that its best use depends on where and when the competitor enters the update.

4.2.2 Real versus Virtual Competitors

Does a fixed real reference and a calibrated virtual reference behave similarly as the student improves? Table 2 compares the two under the same Qwen2.5-7B two-epoch setting. The virtual competitor fixes its conditional relation to the current student group at $(p_1, p_2) = (0.95, 0.70)$, whereas the real competitor uses cached outcomes from Qwen2.5-14B-Instruct.

The contrast exposes a stability difference that is hidden by final accuracy alone. With the calibrated virtual competitor, 10 improves from $82.0/55.5$ to $83.5/58.0$ and 11 from $82.2/56.6$ to $84.3/59.3$. With the real competitor, the same direct-on placements instead fall by $2.5/5.2$ points for 10 and $4.0/6.0$ points for 11 from Epoch 1 to Epoch 2. This pattern is consistent with the hypothesis that stable continued shaping requires the competitor to retain an informative advantage relative to the evolving student. The virtual construction preserves its conditional relation by design; a fixed real competitor need not remain equally informative after the student has fitted its early signal. This is not evidence that a virtual reference is universally preferable. Real outcomes provide prompt-specific

Table 2: Real and virtual competitor dynamics for a Qwen2.5-7B student. Cells report GSM8K/MATH-500 exact-match accuracy (%). The virtual reference uses the calibrated $(p_1, p_2) = (0.95, 0.70)$ operating point; the real reference uses cached Qwen2.5-14B outcomes. Bold components are the best within each reference/epoch row.

Reference	Epoch	Vanilla	Catfish_{00}	Catfish_{01}	Catfish_{10}	Catfish_{11}
Virtual	E1	79.8/55.3	**82.9/56.6**	82.0/56.3	82.0/55.5	82.2/**56.6**
	E2	81.0/54.4	84.0/53.9	83.4/56.1	83.5/58.0	**84.3/59.3**
Real	E1	80.9/57.1	81.7/**58.9**	82.3/57.6	81.4/54.4	**82.7**/56.7
	E2	80.6/**57.2**	**83.4**/57.1	83.0/57.1	78.9/49.2	78.7/50.7

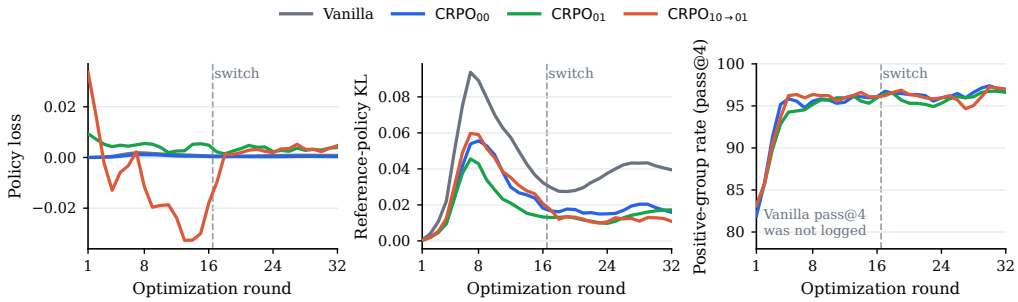


Figure 3: Llama-8B training dynamics on GSM8K. Here we refer to Catfish GRPO as CRPO. Curves show a three-round moving average; the dashed line marks the epoch boundary. The stage-wise trajectory uses Catfish_{10} in Epoch 1 and switches to Catfish_{01} in Epoch 2. Groupwise pass@4 was not emitted by the homogeneous Vanilla logger and is therefore not estimated.

information about which failures are plausibly recoverable, whereas the virtual reference supplies only calibrated statistical structure. The practical trade-off is therefore early selection precision versus a stable late-stage capability gap; the stage-wise results below suggest that changing placement can balance the two.

4.2.3 Learning Dynamics

How does the gain emerge over training? Figure 3 shows the Llama-8B trajectories for Vanilla, fixed 00, fixed 01, and the $10 \to 01$ schedule. Early in training, Catfish primarily expands useful exploration: the positive-group rate (pass@4) rises from roughly 80–82% to 94–97% during the first epoch. At fixed rollout count, this means that more prompts contain at least one successful on-policy trajectory; matched-cohort analyses likewise place the early gain on prompts that initially produced all-wrong groups. These trajectories can then be reinforced, so the larger evaluation-accuracy differences appear after the coverage gain rather than simultaneously with it.

By the epoch boundary, however, the Catfish pass@4 trajectories are already close to saturation and their differences have narrowed. Continued direct shaping can therefore keep moving the policy after its marginal discovery benefit has become small. In this example, fixed 10 falls from 81.3 to 80.7 on GSM8K between epochs, whereas switching to 01 reaches 83.8 while following a lower-KL late trajectory. The associated MATH result does not improve in the same way, so the curves do not establish a universal causal rule. They support a narrower temporal interpretation: Catfish first changes which successful trajectories the student can discover, and stage-wise placement can reduce late policy drift once that exploration advantage saturates.

4.3 Ablation Study

4.3.1 Signal Placement and Stage-Wise Scheduling

Where should the competitor affect the update, and should that placement stay fixed? Table 3 separates Catfish-guided allocation from advantage shaping. 00 retains only competitor-guided retry

Table 3: Stage-wise placement ablation. Cells report GSM8K/MATH accuracy (%). $\mathrm{Catfish}_{\mathrm{sw}}$ is $10 \rightarrow 01$ and therefore coincides with its 10 branch in Epoch 1. Bold numbers are the best result for each evaluation within a model/epoch row.

Model	Epoch	Vanilla	$\mathrm{Catfish}_{00}$	$\mathrm{Catfish}_{01}$	$\mathrm{Catfish}_{10}$	$\mathrm{Catfish}_{11}$	$\mathrm{Catfish}_{\mathrm{sw}}$
Qwen2.5-7B	E1	80.9/57.1	81.7/**58.9**	82.3/57.6	81.4/54.4	**82.7**/56.7	81.4/54.4
	E2	80.6/57.2	83.4/57.1	83.0/57.1	78.9/49.2	78.7/50.7	**84.1**/**57.4**
Llama-8B	E1	75.5/**45.0**	**82.4**/44.7	80.4/40.7	81.3/43.7	81.5/41.0	81.2/41.5
	E2	81.3/43.1	81.0/**45.8**	82.0/**45.8**	80.7/45.6	80.9/44.7	**83.8**/43.9

Table 4: Competitor-quality ablation with a Qwen2.5-3B student. Cells report GSM8K/MATH accuracy (%). Here p_1 and p_2 are the competitor success probabilities conditional on a student positive group and an all-wrong group, respectively. Bold components are the best within each row. † The corrupted-cache row is a backend-induced failure case from a historical LoRA run; it is informative within the row but is not an intentional weak-teacher comparison to the virtual rows.

Reference regime	(p_1, p_2)	$\mathrm{Catfish}_{00}$	$\mathrm{Catfish}_{01}$	$\mathrm{Catfish}_{10}$	$\mathrm{Catfish}_{11}$
Corrupted real cache†	$(0.37, 0.17)$	71.80/**42.86**	**72.71**/41.24	52.84/32.88	57.09/32.88
Weak virtual	$(0.60, 0.10)$	70.81/42.86	71.65/**45.82**	72.71/44.74	**73.09**/43.40
Intermediate virtual	$(0.85, 0.38)$	71.27/43.94	**74.22**/**46.90**	73.31/44.74	72.25/44.47
Matched/strong virtual	$(0.95, 0.65)$	**73.46**/**46.09**	71.87/42.32	71.49/41.51	71.11/45.82

allocation; 01 adds shaping on retry groups; 10 shapes only direct groups; and 11 shapes both branches. This decomposition reveals a stage-dependent trade-off. For Qwen2.5-7B, 11 gives the strongest first-epoch GSM8K endpoint, but the direct-on configurations degrade sharply during the second epoch. In contrast, 00 and 01 continue improving, and the stage-wise continuation reaches 84.1/57.4. Llama-8B shows the same qualitative late-stage split on GSM8K: fixed 10 and 11 decline after the first epoch, whereas moving to 01 produces the strongest final endpoint.

The $10 \rightarrow 01$ continuation directly tests the resulting schedule: it preserves the direct-shaped first epoch, then moves competitor pressure to the selective retry branch. Relative to fixed 10, switching raises the Qwen2.5-7B endpoint by 5.2 points on GSM8K and 8.2 on MATH, and raises Llama-8B GSM8K by 3.1 points, although the latter trades off MATH accuracy. Thus, the evidence favors stage-dependent placement rather than a universally fixed code, while the optimal switching criterion remains task dependent.

4.3.2 Competitor Quality and Reference Regime

What makes a useful competitor? Table 4 varies the conditional success rates of a virtual reference and includes a corrupted real cache as a failure case. Competitor utility is not monotonic in nominal strength. In the weak and intermediate virtual regimes, 01 gives the most balanced two-domain endpoint; at the matched/strong point, 00 is strongest. More importantly, when the real cache becomes unreliable, enabling direct shaping (10 or 11) produces a large degradation, whereas the direct-off configurations remain near the normal range.

These results make the relevant notion of competitor quality relational rather than absolute. A useful reference must still succeed on some prompts for which the student produces an all-wrong group, but it need not be maximally strong on every prompt; once the relation changes, the same placement can become counterproductive. Virtual Catfish further shows that the effect does not require teacher trajectories or knowledge transfer: calibrated scalar outcomes recover much of the benefit without competitor inference. The trade-off is selection precision. A real competitor supplies prompt-specific rescue information, whereas a virtual competitor maintains a stable predefined student–competitor relation as the student evolves. Catfish GRPO therefore depends on maintaining an informative competitive relationship, not simply on choosing the largest available reference model. ....

5 Related Work

Group-Relative Reinforcement Learning and External Guidance. Group Relative Policy Optimization (GRPO) estimates advantages from relative rewards within groups of sampled responses (Shao et al., 2024). Recent work improves group-relative learning through more efficient sampling and rollout allocation (Yu et al., 2025; Nguyen et al., 2026; Zhang et al., 2025), or by recovering learning signals from zero-variance groups through reward and advantage shaping (Feng et al., 2025; Le et al., 2026; Niu et al., 2026). Other approaches encourage exploration by controlling entropy or reasoning-path diversity (Wang et al., 2026a; Chen et al., 2026b). A complementary line introduces external information to guide difficult rollouts (Guo et al., 2026; Wang et al., 2026b; Chen et al., 2026a; Lee et al., 2026). Despite their differences, these approaches primarily intervene through the learner's sampling process, learning signal, or available supervision. How the composition of the comparison population itself affects group-relative learning remains comparatively underexplored.

Population Composition and Heterogeneous Competition. Population-based and multi-agent reinforcement learning has long studied how the composition of interacting agents shapes learning dynamics. Population-based training jointly adapts models and their training configurations (Jaderberg et al., 2017), while game-theoretic and open-ended learning methods construct or expand opponent populations to support continued learning (Lanctot et al., 2017; Balduzzi et al., 2019; Vinyals et al., 2019). Related work further shows that population and behavioral diversity can affect exploration, generalization, and collective performance in multi-agent reinforcement learning (McKee et al., 2022; Bettini et al., 2024a;b). These studies predominantly consider interacting, co-adapting, or jointly optimized populations. It remains unclear how introducing a fixed heterogeneous competitor into the comparison population affects the learning dynamics of a single group-relative learner.

6 Conclusion

We introduced Catfish GRPO, a group-relative reinforcement-learning framework in which a heterogeneous competitor reshapes advantages and retry allocation without providing demonstrations or entering the policy loss. Across Qwen and Llama students and three task pairs, the results show that competitor augmentation is useful but not placement invariant: direct shaping is often most valuable while the student is still expanding its set of successful trajectories, whereas later training can benefit from concentrating competitor influence on selective retry. The resulting $10 \rightarrow 01$ schedule gives the strongest Qwen2.5-7B and Llama-8B GSM8K endpoints in our comparison and also improves Qwen2.5-3B transfer performance. Its mixed Llama transfer result nevertheless makes clear that a single temporal rule does not dominate all tasks.

The reference-quality ablation further shows that competitor utility is relational rather than monotonic in model strength. Calibrated virtual competitors reproduce much of the effect without teacher trajectories or competitor inference, while real competitors retain the advantage of prompt-specific rescue information. On Qwen2.5-7B, fixed direct shaping grows through the second epoch under a calibrated virtual relation but declines under the fixed real reference; this is consistent with, but does not by itself prove, the need to maintain an informative capability gap as the student evolves. Conversely, unreliable reference outcomes can make direct shaping sharply harmful. These findings identify the evolving student–competitor relationship—where the signal is applied, when it is applied, and whether the reference remains informative—as a central design axis for group-relative reinforcement learning.

AI Use Statement

We used generative AI tools to assist with literature search and summarization, software development and debugging, analysis workflow development, and language editing and drafting of the manuscript. AI tools were also used to provide feedback during the development and refinement of experimental analyses. All AI-assisted code, experimental procedures, analyses, citations, and written content were manually reviewed and verified by the authors against the original sources and experimental outputs. All experiments were executed by the authors, and all reported results were obtained from the resulting experimental runs. The authors determined the final methodology, interpretation of results, and conclusions, and take full responsibility for the content of this work.

Ethics Statement

This work studies reinforcement learning methods for language models using publicly available mathematical reasoning datasets and does not involve human subjects or private data. As with other outcome-based reinforcement learning methods, the trained models may still produce incorrect or misleading reasoning despite obtaining improved task performance. We encourage further evaluation of such methods before deployment in safety-critical applications.

Reproducibility Statement

We provide the algorithmic details of Catfish GRPO in Section 3 and report the training configurations, evaluation protocols, model settings, and ablations in Section 4 and the Appendix. Additional implementation details, hyperparameters, and experimental results are included in the supplementary material. We will release the code and scripts required to reproduce the main experiments.

References

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 434–443. PMLR, 2019. URL https://proceedings.mlr.press/v97/balduzzi19a.html.

Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. Controlling behavioral diversity in multi-agent reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3611–3636. PMLR, 2024a. URL https://proceedings.mlr.press/v235/bettini24a.html.

Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. The impact of behavioral diversity in multi-agent reinforcement learning. *arXiv preprint arXiv:2412.16244*, 2024b. URL https://arxiv.org/abs/2412.16244.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=hfNnQHkTtv.

Xiwen Chen, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Hao Wang, Haiyu Wu, Huayu Li, Aris Sotiras, Yalin Wang, and Abolfazl Razi. DRA-GRPO: Your GRPO needs to know diverse reasoning paths for mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13995–14019. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.findings-acl.685. URL https://aclanthology.org/2026.findings-acl.685/.

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. URL https://arxiv.org/abs/2110.14168.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5925–5932. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1606. URL https://aclanthology.org/D19-1606/.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp.

2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Yunzhen Feng, Parag Jain, Anthony Hartshorn, Yaqi Duan, and Julia Kempe. Don't waste mistakes: Leveraging negative RL-groups via confidence reweighting. *arXiv preprint arXiv:2510.08696*, 2025. URL https://arxiv.org/abs/2510.08696.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Yongxin Guo, Wenbo Deng, Zhenglin Cheng, and Xiaoying Tang. G^2RPO-a: Guided group relative policy optimization with adaptive guidance. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4525–4539. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.206. URL https://aclanthology.org/2026.acl-long.206/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. URL https://arxiv.org/abs/1711.09846.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4190–4203, 2017.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kiXFIESZKv.

Byung-Kwan Lee, Ximing Lu, Shizhe Diao, Minki Kang, Saurav Muralidharan, Karan Sapra, Andrew Tao, Yu-Chiang Frank Wang, and Ryo Hachiuma. Zone of proximal policy optimization: Teacher in prompts, not gradients. *arXiv preprint arXiv:2606.18216*, 2026. URL https://arxiv.org/abs/2606.18216.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Kevin R. McKee, Joel Z. Leibo, Charlie Beattie, and Richard Everett. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):21, 2022. doi: 10.1007/s10458-022-09548-8.

Meta AI. Llama 3.2-3B-Instruct model card. Hugging Face model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Z5sWYACAop.

Wenzhe Niu, Wei He, Zongxia Xie, Jinpeng Ou, Huichuan Fan, Yuchen Ge, Yanru Sun, Ziyin Wang, Yizhao Sun, Chengshun Shi, Jiuchong Gao, Jinghua Hao, and Renqing He. From absolute to relative: Rethinking reward shaping in group-based reinforcement learning. *arXiv preprint arXiv:2601.23058*, 2026. URL https://arxiv.org/abs/2601.23058.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Chen Wang, Lai Wei, Yanzhi Zhang, Chenyang Shao, Zedong Dan, Weiran Huang, Ge Lan, and Yue Wang. Targeted exploration via unified entropy control for reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16789–16802. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.findings-acl.828. URL https://aclanthology.org/2026.findings-acl.828/.

Xinyi Wang, Jinyi Han, Zishang Jiang, Tingyun Li, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. Don't tell the answer, truly guide the reasoning during RL rollouts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3437–3455. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.findings-acl.169. URL https://aclanthology.org/2026.findings-acl.169/.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. Improving sampling efficiency in RLVR through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025. URL https://arxiv.org/abs/2509.25808.

Table 5: Datasets and evaluation protocol used in the main experiments. All methods within a model–task block use the same training prompts, ordering, reward function, and evaluation split.

Training task	Train prompts	In-domain eval.	OOD eval.	Base model
GSM8K	4,096 × 2 epochs	GSM8K	MATH-500 (numeric)	Qwen-3B/7B, Llama-3B/8B
DROP	4,096 × 2 epochs	DROP (1,024)	Quoref (500)	Qwen-7B
2WikiMultiHopQA	4,096 × 2 epochs	2Wiki (1,024)	HotpotQA (500)	Qwen-7B

A Additional Experimental Details

This section provides the implementation and evaluation details omitted from the main text. The main experiments are designed around a common comparison protocol so that differences across methods primarily reflect the learning rule rather than changes in data exposure, rollout settings, or evaluation.

A.1 Models, tasks, and evaluation splits

We evaluate two model families and multiple model scales. The Qwen experiments use Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct as base models. The Llama experiments use Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct. Catfish models are always frozen and are never optimized jointly with the base model.

For mathematical reasoning, training uses 4,096 prompts sampled from the GSM8K training set and exposes every method to the same ordered prompt sequence for two epochs. Evaluation is greedy and uses the official GSM8K test set together with the numeric-answer subset of MATH-500. For question answering, Qwen2.5-7B-Instruct is trained separately on 4,096 prompts from DROP and 4,096 prompts from 2WikiMultiHopQA. The corresponding in-domain evaluations use 1,024 held-out examples, while Quoref and HotpotQA are used as out-of-domain evaluations with 500 examples each. DROP and 2WikiMultiHopQA use multi-reference exact-match rewards: an answer receives reward one if it matches any accepted reference answer and zero otherwise.

A.2 Training protocol

The broad model comparison uses a common GRPO-style training protocol based on the same implementation stack for Vanilla GRPO, Catfish GRPO, and the compared baselines. Unless stated otherwise, each base prompt produces $G = 4$ on-policy responses with temperature 0.8 and top-$p = 0.95$. Training uses 32 optimization rounds, each consuming 256 base-prompt groups, which corresponds to two complete passes over the 4,096-prompt training set. All methods therefore match *base-prompt exposure*. Methods that allocate retries may consume additional rollouts; these additional generations are part of the method rather than being deducted from the common base-prompt budget.

The shared optimizer settings use full-parameter training with learning rate 10^{-6}, clipping range $[0.8, 1.28]$ (equivalently clip-low 0.2 and clip-high 0.28), and a reference-policy KL coefficient of 0.02. The rollout group size, prompt order, generation parameters, reward implementation, and evaluation procedure are held fixed across methods in each comparison block. The main comparison uses tensor parallelism of two and the same Megatron/SGLang training stack for all methods.

For Catfish GRPO, an all-wrong initial group may trigger up to $J = 3$ retry groups when the Catfish scalar is one. Retry rounds use progressively larger sampling temperatures 0.9, 1.0, and 1.1. The first retry group containing at least one correct response is used for optimization; if all retry groups remain all wrong, the prompt produces no policy-gradient update.

A.3 Real Catfish construction

A real Catfish is a frozen heterogeneous model evaluated on the same prompt as the base model. Only its binary task outcome is retained. Its generated text, token probabilities, hidden states, and parameters are never exposed to the base-model loss. In the Qwen2.5-7B main experiments, the real

Table 6: Shared hyperparameters for the broad comparison. Task-specific generation-length limits are kept identical across methods within each task.

Hyperparameter	Setting
Base prompts	4,096
Prompt epochs	2
Optimization rounds	32
Base groups per round	256
Rollouts per base group G	4
Base rollout temperature	0.8
Top-p	0.95
Maximum retry groups J	3
Retry temperatures	0.9 / 1.0 / 1.1
Learning rate	1×10^{-6}
Clip-low / clip-high	0.20 / 0.28
Reference-policy KL coefficient	0.02
Optimization	full-parameter
Tensor parallelism	2
Comparison budget	matched base-prompt exposure

Table 7: Catfish regime used in the broad comparison. A real Catfish uses a fixed prompt-level outcome cache; a virtual Catfish samples only from calibrated conditional outcome statistics.

Base model / task	Catfish type	Reference regime
Qwen2.5-3B / GSM8K	Virtual	$(p_1, p_2) = (0.85, 0.38)$
Qwen2.5-7B / GSM8K	Real	Qwen2.5-14B-Instruct
Qwen2.5-7B / DROP	Real	Qwen2.5-14B-Instruct
Qwen2.5-7B / 2Wiki	Real	Qwen2.5-14B-Instruct
Llama-3B / GSM8K	Virtual	$(p_1, p_2) = (0.85, 0.30)$
Llama-8B / GSM8K	Virtual	$(p_1, p_2) = (0.85, 0.30)$

Catfish is Qwen2.5-14B-Instruct. Catfish outcomes are generated before policy training and stored as a prompt-indexed binary cache. The cache is therefore fixed during training, which separates changes in the student policy from changes in the competitor itself.

Competitor generation uses a larger response budget than the student when needed to avoid truncating the stronger model. In particular, the 14B cache is generated with a 512-token response limit. This choice is important empirically: a shorter 128-token limit substantially underestimates 14B correctness because many otherwise-correct responses are truncated before the final answer. All reported 14B Catfish experiments use the regenerated 512-token cache.

A.4 Virtual Catfish calibration

The virtual Catfish replaces prompt-level competitor inference by a conditional Bernoulli variable. For a base group $\mathbf{r} = \{r_1, \ldots, r_G\}$,

$$\Pr(\tilde{c} = 1 \mid \mathbf{r}) = \begin{cases} p_1, & \max_i r_i > 0, \\ p_2, & \max_i r_i = 0. \end{cases} \tag{7}$$

The two probabilities are calibrated from paired student–competitor interactions. p_1 measures Catfish correctness conditional on the base group already containing at least one correct response, while p_2 measures Catfish correctness conditional on an all-wrong base group. Thus, p_1 mainly controls the expected shift of the direct-branch normalization anchor, whereas p_2 controls how often the all-wrong pool is selected for additional exploration.

For the Qwen2.5-3B experiments, the calibrated virtual regime is $(p_1, p_2) = (0.85, 0.38)$, estimated from clean Qwen2.5-7B Catfish interactions. The broader virtual-Catfish study also evaluates weaker and stronger regimes to test sensitivity to the relative capability gap. Both Llama models use $(p_1, p_2) = (0.85, 0.30)$, as specified in Section 3.2.

A.5 BASELINE IMPLEMENTATIONS

We compare against Vanilla GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), RL-ZVP (Le et al., 2026), LENS (Feng et al., 2025), and AR3PO (Zhang et al., 2025). For comparability, all methods use the common $G = 4$ protocol and the same prompt budget, rollout parameters, optimizer, KL coefficient, and evaluation procedure described above. When a published method assumes a materially different systems configuration, we preserve its defining mechanism while keeping the common experimental protocol fixed. The resulting implementation differences are summarized below.

Vanilla GRPO. Vanilla GRPO uses the standard within-group normalized outcome advantage. Zero-variance groups have zero advantage and therefore contribute no policy-gradient signal.

DAPO. Our DAPO implementation preserves dynamic sampling semantics, asymmetric clipping, and token-level policy-gradient aggregation from Yu et al. (2025). Zero-variance groups are removed from both policy-gradient and KL computation. Two deviations are required by the matched protocol. First, we do not use the paper's overlong-response filtering because the substantially shorter response budget in our setting causes truncation to be common and filtering these samples removes an important negative signal. Second, we do not refill a minibatch after removing degenerate groups; keeping the original 256 base prompts preserves the common prompt-exposure accounting.

RL-ZVP. RL-ZVP retains standard GRPO on non-zero-variance groups and injects a confidence-weighted signal into zero-variance groups following Le et al. (2026). Our implementation uses rollout $-\log p$ as the token-level confidence proxy because actor-side entropy is unavailable through the training hook used in our stack. The direction of the intervention is unchanged—low-confidence tokens receive larger correction—but the reported RL-ZVP numbers should therefore be interpreted as a log-probability-confidence variant rather than an exact entropy implementation.

LENS. Our LENS implementation follows the confidence-reweighting construction of Feng et al. (2025). Response confidence is computed from the geometric mean of rollout token probabilities. Mixed groups use standard GRPO, while all-wrong groups use the calibrated negative-group construction with weight $\alpha = 0.25$. To maintain the shared comparison protocol, we use $G = 4$, temperature 0.8, and KL coefficient 0.02 rather than the original paper's larger group and no-KL setting. The negative-group weight scales the policy-gradient term; the separately implemented KL term is not multiplied by α.

AR3PO. Our AR3PO comparison isolates adaptive rollout allocation from Zhang et al. (2025). Each prompt begins with four responses and receives another group of four responses when needed, up to 16 responses in total. Sampling stops once a correct response is observed, and GRPO normalization is performed across all responses collected for that prompt. We do not implement response reuse or the associated off-policy correction; consequently, this row is an *adaptive-rollout-only* AR3PO ablation rather than a complete reproduction of the original method.

A.6 INDEPENDENT THREE-SEED REAL-CATFISH CONTROL

In addition to the broad common-stack comparison, we retain an independent three-seed experiment that uses a Qwen2.5-3B-Instruct student and a real Qwen2.5-7B-Instruct Catfish. This control uses the same 4,096 unique GSM8K prompts for two prompt epochs and evaluates on all 1,319 GSM8K test examples and the 371 numeric-answer MATH-500 examples. Student sampling uses $G = 4$, temperature 0.8, top-$p = 0.95$, and a 128-token training limit; MATH evaluation uses up to 256 new tokens.

This implementation trains LoRA parameters on the student only, with rank 8, alpha 16, and dropout 0.05. AdamW uses learning rate 5×10^{-6}, gradient clipping at 1.0, and a sampled-action KL anchor to the frozen base model with coefficient 0.02. Four effective prompt groups are accumulated per optimizer step. Catfish generations are cached in advance, and retries are additional student rollouts rather than replacements for base-prompt exposure. This experiment is reported separately because its optimization stack differs from the full-parameter broad comparison; it serves as an independent confirmation rather than a row to be pooled with the single-seed common-stack results.

Table 8: Exact semantics of the four Catfish GRPO placement codes.

Code	Direct normalization	Retry normalization	Catfish-guided retries
00	base only	base only	yes
01	base only	Catfish-augmented	yes
10	Catfish-augmented	base only	yes
11	Catfish-augmented	Catfish-augmented	yes

B Catfish GRPO Routing and Update Details

This section makes the branch routing explicit. The Catfish contributes only a scalar outcome and never an optimized trajectory. The two effects of the Catfish are therefore conceptually separate: it can determine *where* additional student rollouts are allocated, and it can change *how* student advantages are normalized.

B.1 Direct branch

Given an initial base-model group $\mathbf{r} = \{r_1, \ldots, r_G\}$, the direct branch is active whenever $\max_i r_i > 0$. We compute both the base-only advantage

$$\hat{A}_i = \frac{r_i - \mu(\mathbf{r})}{\sigma(\mathbf{r})} \tag{8}$$

and the Catfish-augmented advantage

$$\hat{A}_i^c = \frac{r_i - \mu(\mathbf{r} \cup \{c\})}{\sigma(\mathbf{r} \cup \{c\})}. \tag{9}$$

The optimized advantage is

$$A_i^{\text{direct}} = (1 - \lambda_{\text{direct}})\hat{A}_i + \lambda_{\text{direct}}\hat{A}_i^c. \tag{10}$$

Flat reward sets are assigned zero advantage. Importantly, only the G base-model trajectories are optimized, even when the Catfish outcome participates in the normalization statistics.

B.2 Retry branch

If the initial group is all wrong, the base-only GRPO advantage is identically zero. Catfish GRPO does not directly optimize the negative-only group. Instead, the Catfish scalar determines whether the prompt receives additional exploration. If $c = 0$, the prompt is skipped. If $c = 1$, the base model is sampled again for at most $J = 3$ retry rounds. Let $j \in \{1, 2, 3\}$ denote the retry round. The first retry group containing at least one correct response is optimized using

$$A_{i,j}^{\text{retry}} = (1 - \lambda_{\text{retry}})\hat{A}_{i,j} + \lambda_{\text{retry}}\hat{A}_{i,j}^c. \tag{11}$$

The update is multiplied by the common difficulty weight

$$w_j = \min\left(1.5,\ 1 + 0.5\log(1 + j)\right). \tag{12}$$

Once a positive-containing retry group is found, later retry rounds are not generated. If all J retry groups are all wrong, the prompt produces no policy-gradient update.

B.3 Placement codes

The two-bit code ab denotes $(\lambda_{\text{direct}}, \lambda_{\text{retry}}) = (a, b)$. Retry *allocation* remains Catfish-guided for all four codes; the bits control only whether Catfish participates in advantage normalization after a usable student group has been found.

The codes therefore implement a structured trade-off rather than four unrelated algorithms. In particular, 00 isolates Catfish-guided rollout allocation from Catfish-based advantage shaping; 10 isolates direct-branch shaping while preserving a larger positive update for a newly discovered retry success; and 11 applies the Catfish normalization pressure on both branches.

Table 9: Additional Catfish GRPO placement results. $\mathrm{Catfish}_{\mathrm{sw}}$ denotes the stage-wise $10 \rightarrow 01$ schedule. A dash denotes a schedule that was not evaluated.

Model / task	$\mathrm{Catfish}_{00}$	$\mathrm{Catfish}_{01}$	$\mathrm{Catfish}_{10}$	$\mathrm{Catfish}_{11}$	$\mathrm{Catfish}_{\mathrm{sw}}$
Qwen2.5-3B, GSM/MATH	71.27/43.94	74.22/46.90	73.31/44.74	72.25/44.47	72.7/46.0
Qwen2.5-7B, GSM/MATH	83.4/57.1	83.0/57.1	78.9/49.2	78.7/50.7	84.1/57.4
Qwen2.5-7B, DROP/Quoref	77.54/76.80	78.12/75.40	78.12/75.60	77.15/73.40	–
Qwen2.5-7B, 2Wiki/HotpotQA	67.19/58.40	66.21/57.80	65.04/58.60	65.14/57.20	64.9/58.4
Llama-8B, GSM/MATH	81.0/45.8	82.0/45.8	80.7/45.6	80.9/44.7	83.8/43.9

Table 10: Stage-wise placement ablation. Cells report GSM8K/MATH accuracy (%). The switched trajectory uses placement 10 during the first epoch and placement 01 during the second epoch.

Model	Epoch	Vanilla	00	01	10	11	$10 \rightarrow 01$
Qwen2.5-7B	E1	80.9/57.1	81.7/58.9	82.3/57.6	81.4/54.4	82.7/56.7	81.4/54.4
	E2	80.6/57.2	83.4/57.1	83.0/57.1	78.9/49.2	78.7/50.7	84.1/57.4
Llama-8B	E1	75.5/45.0	82.4/44.7	80.4/40.7	81.3/43.7	81.5/41.0	81.2/41.5
	E2	81.3/43.1	81.0/45.8	82.0/45.8	80.7/45.6	80.9/44.7	83.8/43.9

B.4 Stage-dependent placement

The stage-wise configuration used in the broad comparison changes the placement at the epoch boundary. Denoting the boundary by τ,

$$(\lambda_{\mathrm{direct}}(t), \lambda_{\mathrm{retry}}(t)) = \begin{cases} (1,0), & t < \tau, \\ (0,1), & t \geq \tau. \end{cases} \tag{13}$$

Thus the first stage uses direct-branch Catfish shaping while keeping successful retries base-normalized; the second stage removes direct-branch shaping and moves Catfish normalization to the selective retry branch. Retry allocation remains Catfish-guided in both stages.

C Additional Placement Results

Table 1 summarizes the four fixed placements into two families, $\mathrm{Catfish}_{0*}$ and $\mathrm{Catfish}_{1*}$. Table 9 reports individual placements for the model–task pairs listed below. All cells report in-domain/out-of-domain exact-match accuracy in percent.

Independent three-seed control. The common-stack table above is complemented by the independent real-Catfish control described in Appendix A.6. At the two-epoch boundary, Vanilla GRPO reaches 72.38 ± 0.43 on GSM8K and 42.41 ± 1.79 on numeric MATH-500, while the direct-only Catfish placement $(1, 0)$ reaches 74.45 ± 0.20 and 45.46 ± 3.51, respectively. These numbers are not pooled with the broad table because they use a different optimization stack.

C.1 Stage-wise endpoints by epoch

C.2 Competitor-quality sweep

For completeness, Table 11 reports the Qwen2.5-3B virtual-competitor regimes used to study sensitivity to the conditional capability gap. The corrupted-cache row is a historical backend failure case and should not be interpreted as an intentionally weak competitor.

D Definitions of Training-Dynamics Metrics

We distinguish trajectory-level correctness, group-level discovery, retry allocation, and optimization statistics. This distinction is important because Catfish GRPO changes the number and difficulty of sampled student trajectories; the mean reward over generated trajectories is therefore not interchangeable with fixed-set evaluation accuracy.

Table 11: Competitor-quality ablation with Qwen2.5-3B. Cells report GSM8K/MATH accuracy (%).

Reference regime	(p_1, p_2)	00	01	10	11
Corrupted real cache	(0.37, 0.17)	71.80/42.86	72.71/41.24	52.84/32.88	57.09/32.88
Weak virtual	(0.60, 0.10)	70.81/42.86	71.65/45.82	72.71/44.74	73.09/43.40
Intermediate virtual	(0.85, 0.38)	71.27/43.94	74.22/46.90	73.31/44.74	72.25/44.47
Matched/strong virtual	(0.95, 0.65)	73.46/46.09	71.87/42.32	71.49/41.51	71.11/45.82

Student rollout accuracy. For a set of generated student trajectories, rollout accuracy is the fraction receiving task reward one:

$$\mathrm{RolloutAcc} = \frac{1}{N}\sum_{n=1}^{N} r_n. \tag{14}$$

This is an on-policy training statistic and should not be interpreted as greedy evaluation accuracy.

Positive-group rate (pass@4). For $G = 4$, the positive-group rate is the fraction of base groups containing at least one correct student trajectory:

$$\mathrm{Pass@4} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}\Big[\max_i r_{m,i} > 0\Big]. \tag{15}$$

It measures how often the current policy discovers at least one usable on-policy trajectory for a prompt under the training sampler.

All-wrong rate. The all-wrong rate is

$$\mathrm{AllWrong} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}\Big[\max_i r_{m,i} = 0\Big], \tag{16}$$

which equals $1 - \mathrm{Pass@4}$ for binary rewards and a fixed group size.

Retry-trigger rate. A retry is triggered only for an all-wrong base group with Catfish scalar one. We report either the number of triggered prompts or the conditional trigger rate within the all-wrong pool:

$$\mathrm{TriggerRate} = \Pr(c = 1 \mid \max_i r_i = 0). \tag{17}$$

For a virtual Catfish this quantity should track the configured p_2 up to sampling noise. For a real Catfish it can drift over training because the composition of the surviving all-wrong pool changes as the base model improves.

Retry success rate. Among prompts that trigger retries, retry success is the fraction for which at least one retry round contains a correct student response:

$$\mathrm{RetrySuccess} = \frac{\#\{\text{triggered prompts with a successful retry}\}}{\#\{\text{triggered prompts}\}}. \tag{18}$$

A successful prompt may require one, two, or three retry rounds; only the first positive-containing retry group is optimized.

Effective-group rate. An effective group is a base prompt that produces a nonzero student update, either through a positive-containing direct group or through a successful retry. The effective-group rate is the fraction of base prompts producing such an update. It is useful for separating increased optimization density from the semantic placement of the Catfish signal.

Reference-policy KL. The KL statistic is the sampled-action divergence between the current policy and the frozen reference policy used by the optimizer. We use it as an optimization-stability diagnostic rather than as a task-performance metric. In the stability analyses, a sharp KL increase precedes reward and output-format degradation.

Policy loss. Policy loss is the policy-gradient component emitted by the common training stack. Because Catfish GRPO changes both the set of effective groups and their advantages, its magnitude is interpreted only as a trajectory diagnostic and is not compared as an objective value across methods.

Evaluation accuracy. All reported task endpoints use greedy generation and task-specific exact-match scoring. For multi-reference QA tasks, any accepted reference answer counts as correct. Endpoint accuracy is computed on a fixed evaluation set and is therefore distinct from training rollout accuracy, which is measured on adaptively generated trajectories.