Vocabulary Dropout for Curriculum Diversity in LLM Co-Evolution

Jacob Dineen, Aswin RRV, Zhikun Xu, Ben Zhou
Arizona State University
{jdineen, aravik13, zhikunxu, xzhou202}@asu.edu

arXiv:2604.03472v1 [cs.CL] 3 Apr 2026

Abstract

Co-evolutionary self-play, where one language model generates problems and another solves them, promises autonomous curriculum learning without human supervision. In practice, the proposer quickly converges to a narrow distribution of problems that satisfy the reward function. This diversity collapse renders the curriculum uninformative for the solver, stalling the co-evolutionary loop. We introduce vocabulary dropout, a random mask applied to the proposer's output logits during both policy training and curriculum generation, as a lightweight mechanism to sustain diversity. The mask is hard and non-stationary, preventing the proposer from locking into fixed token sequences. Training Qwen3-4B and Qwen3-8B on mathematical reasoning via R-Zero, we find that vocabulary dropout sustains proposer diversity across lexical, semantic, and functional metrics throughout training, and yields solver improvements averaging +4.4 points at 8B, with the largest gains on competition-level benchmarks. Our findings suggest that explicit action-space constraints, analogous to the structural role that game rules play in classical self-play, can help sustain productive co-evolution in language. Vocabulary dropout is one simple instantiation of this principle.[1]

1 Introduction

Self-play is a promising training paradigm in which models improve by learning through competition or interaction with copies of themselves. This approach has been particularly successful in games, where the environment provides strong structural constraints on learning. In domains such as Go, StarCraft, and Dota, the rules of the game specify legal actions, fixed dynamics, and explicit objectives, making self-play well-defined (Tesauro, 1995). Beyond these constraints, methods such as population-based training, opponent sampling, and fictitious self-play (Vinyals et al., 2019; Berner et al., 2019; Heinrich & Silver, 2016) expose agents to a broader mixture of opponents and help reduce cycling or collapse to narrow strategies. Together, this combination of game constraints and opponent diversity has been central to strong self-play performance.

Recent co-evolutionary training frameworks apply a similar idea to language by splitting a single model into two roles, a *proposer* that generates reasoning problems and a *solver* that attempts them (Huang et al., 2025). The proposer is trained to produce problems at the edge of the solver's ability, and the solver is trained on the resulting curriculum, creating an iterative loop analogous to two-player self-play training. The core idea is that as the solver improves, the proposer gradually adapts, generating increasingly difficult problems to maintain the training signal. However, in practice, we observe that the proposer converges to a narrow set of question templates that satisfy the reward function, and the curriculum loses its value as a training signal (Liu et al., 2026; Chae et al., 2025). Our experiments show that this stagnation sets in early in self-play training, and coincides with early plateauing of solver capability improvement observed in prior work (Huang et al., 2025), and that standard diversity metrics fail to detect it.

[1]We will release all code and data under an open-source license upon publication.

We hypothesize that this collapse stems from a fundamental difference between games and language. Game environments are symbolically verifiable, whereas natural-language questions are not. In games, the rules define legal actions, state transitions, and success conditions, so constructing a hard position typically requires producing one that is genuinely strategically demanding. This structure makes it difficult for an opponent to remain challenging without actually improving. In language, by contrast, neither the proposer nor the solver is constrained by comparable external rules, and the difficulty of a question is not independently certified by the environment. As a result, the proposer can satisfy the reward function without understanding why a question is hard, by converging on narrow templates that reliably induce solver errors or disagreement. The result is a form of diversity collapse, where the curriculum may retain superficial variation but concentrates on a small set of underlying question templates, ceasing to provide a rich training signal.

Prior work attempts to mitigate this diversity collapse by anchoring the co-evolutionary loop to external signals (Zhao et al., 2025; Huang et al., 2025; Liu et al., 2025; Wilf et al., 2025), and Liu et al. (2026) argue more broadly that self-play stalls when the generated data ceases to provide learnable structure. These approaches improve stability but none directly constrain the proposer's output space.

We take a different approach, motivated by the observation that in game-based self-play, action-space structure is essential for productive training. AlphaZero injects exploration noise scaled to the number of legal moves to prevent degenerate play (Silver et al., 2017), OpenAI Five randomizes game properties to force strategic diversity (Berner et al., 2019), and more broadly, the structure of the action space shapes what policies can learn (Chandak et al., 2019; Farquhar et al., 2020). We introduce **vocabulary dropout** (VD) to play an analogous role in language generation, applying a hard, non-stationary mask over the proposer's output logits that removes a random subset of tokens each batch. Because blocked tokens cannot be generated regardless of the policy's learned distribution, the constraint is not subject to optimization pressure, and because the mask changes every batch, the proposer cannot converge to any fixed set of token sequences (Section 4.1).

We evaluate vocabulary dropout within R-Zero (Huang et al., 2025), a co-evolutionary framework that rewards the proposer based on solver self-consistency without requiring an external verifier, training Qwen3-4B and Qwen3-8B base models via GRPO (Shao et al., 2024). Our contributions are threefold. First, we provide a detailed empirical characterization of proposer diversity dynamics across co-evolution iterations, revealing that functional stagnation sets in early and is not captured by standard lexical metrics. Second, we introduce vocabulary dropout as a simple, hard constraint on the proposer's action space that sustains diversity across lexical, semantic, and functional measures throughout training. Third, we show that this sustained diversity translates into stronger solvers, yielding an average improvement of +4.4 points at 8B, with the largest gains on competition-level benchmarks (AMC, Olympiad, AIME).

2 Related work

Self-play and co-evolution for LLMs. A growing family of methods trains LLMs through self-generated curricula. SPIN distinguishes human from model text to iteratively refine an SFT model (Chen et al., 2024). Absolute Zero, R-Zero, MAE, and Socratic-Zero use executable environments or co-evolving challenger-solver architectures to mine increasingly challenging tasks from minimal seed data (Zhao et al., 2025; Huang et al., 2025; Chen et al., 2025; Wang et al., 2025a). R-Few stabilizes self-evolution with lightweight human supervision (Yu et al., 2025), and SPICE grounds generation in document retrieval (Liu et al., 2025). Mishra (2026) address curriculum collapse by using a semantic coverage signal over an embedding-induced partition, but, like the others, the intervention targets problem selection rather than the proposer's action space. We adopt R-Zero's two-model architecture as our experimental framework because its challenger is directly observable, enabling controlled study of diversity interventions.

Mode collapse in LLM training. Policy-gradient methods concentrate probability on high-reward behaviors, causing entropy collapse even when rewards are verifiable (Zhou et al., 2025; Gai et al., 2025). Cui et al. (2025) establish that downstream performance is bottlenecked by entropy exhaustion. This applies not only to solvers but also to proposers in co-evolutionary settings, where the proposer's policy gradient training drives it toward a narrow set of high-reward problem templates. RL-trained reasoning models exhibit reduced solution diversity (Yue et al., 2025) and may fail to develop reasoning behaviors absent from the base policy (Rrv et al., 2025), while LLM populations grow homogeneous over generations (Jiang et al., 2025; Chae et al., 2025). Proposed mitigations generally target the reward or sampling side, including entropy regularization (Cui et al., 2025), structured reward decomposition (Dineen et al., 2025), diversity-promoting rewards (Li et al., 2025), and population-based training (Jaderberg et al., 2017). Vocabulary dropout complements these approaches, targeting the action space directly rather than the reward or sampling distribution (Section 4.1).

Action-space design in RL. The structure of the action space shapes what policies can learn. In classical RL, action representations influence exploration and generalization (Chandak et al., 2019; Farquhar et al., 2020). In games, rule-imposed constraints on legal moves are what make self-play productive, with exploration noise scaled to the legal action space (Silver et al., 2017) and environment randomization used to force strategic diversity (Berner et al., 2019). For language models, the vocabulary *is* the action space, and its size is typically fixed at the tokenizer's full output. Vocabulary reduction has been studied for efficiency (Nozaki et al., 2025), and RLPT (Pang et al., 2026) masks contextually irrelevant tokens to concentrate the policy on promising outputs. Our vocabulary dropout inverts this logic, masking tokens randomly and non-stationarily to prevent concentration rather than encourage it.

3 Preliminaries: GRPO and R-Zero

R-Zero (Huang et al., 2025) is a co-evolutionary framework for mathematical reasoning (Figure 1a). It trains two models, a proposer π_P that generates problems and a solver π_S that solves them, both initialized from the same base model and updated via GRPO (Shao et al., 2024) in alternating phases. GRPO is a value-free policy gradient method that, for a prompt x, samples a group of G responses $\{o_1, \ldots, o_G\}$ from π_θ, scores each with reward r_i, and computes advantages via group normalization $\hat{A}_i = (r_i - \mu_G)/\sigma_G$. The policy is updated by maximizing a clipped surrogate:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\rho_i \hat{A}_i,\ \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\hat{A}_i\right) - \beta\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right] \tag{1}$$

where $\rho_i = \pi_\theta(o_i|x)/\pi_{\text{old}}(o_i|x)$ is the importance ratio and the KL term regularizes against drift from a reference policy.

The proposer's reward targets problems near the boundary of the solver's current ability. Given a generated problem q, the frozen solver produces M candidate solutions. The proposer is rewarded based on the solver's self-consistency. If responses mostly agree, the problem is too easy, and if they mostly disagree, it may be too hard or ill-posed. The reward peaks a t maximal solver uncertainty:

$$r_P(q) = \begin{cases} \min(\text{acc}(q), 1 - \text{acc}(q)) & \text{if acc}(q) \in [\tau_{\min}, \tau_{\max}] \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\text{acc}(q)$ is the fraction of solver responses agreeing with the majority vote. R-Zero additionally penalizes intra-batch repetition via pairwise BLEU similarity, clustering near-duplicate questions and subtracting a penalty proportional to cluster size from the uncertainty reward. This encourages surface-level diversity within each batch. The solver trains on problems filtered from the proposer's output, receiving binary reward for correctness.

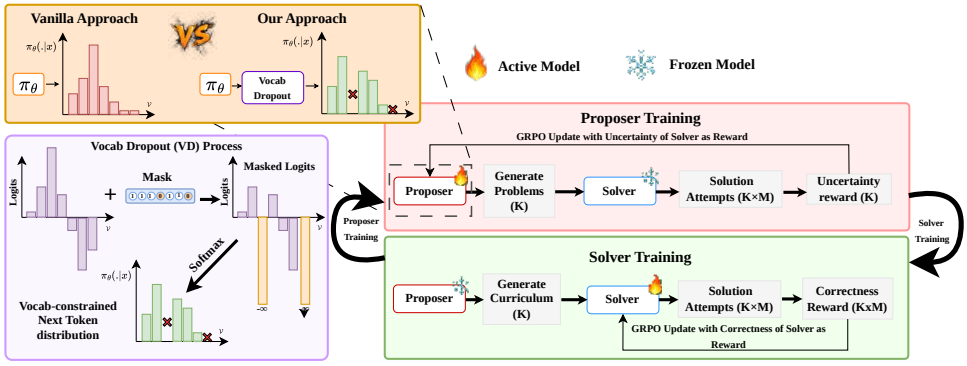

Figure 1: Training pipeline. **Left**: Vocabulary dropout masks a random subset of output logits, constraining the proposer's token distribution. **Right**: The co-evolution loop. In **Phase 1** (proposer training), the proposer generates K problems, the frozen solver attempts each M times, and the proposer is rewarded based on solver uncertainty. In **Phase 2** (solver training), the frozen proposer generates a curriculum of K problems, the solver attempts each M times, and the solver is rewarded for matching the correct answer.

4 Method

We modify the R-Zero pipeline by adding vocabulary dropout to the proposer (Section 4.1).

4.1 Vocabulary dropout

Given the full vocabulary $\mathcal{V}$ with a small protected subset $\mathcal{F} \subset \mathcal{V}$ of format-critical tokens exempt from masking (details in Section B.2), we define a retention probability $\alpha \in (0, 1]$. At each batch b, we sample a mask $\mathbf{m}^{(b)} \in \{0,1\}^{|\mathcal{V}|}$ where each entry is drawn independently:

$$m_v^{(b)} \sim \begin{cases} \text{Bernoulli}(\alpha) & \text{if } v \notin \mathcal{F} \\ 1 & \text{if } v \in \mathcal{F} \end{cases} \tag{3}$$

The masked logits at batch b are then:

$$\tilde{\ell}_v^{(b)} = \begin{cases} \ell_v & \text{if } m_v^{(b)} = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{4}$$

where ℓ_v is the original logit for token v. The resulting token distribution $\tilde{\pi}^{(b)}(\cdot|x) = \text{softmax}(\tilde{\boldsymbol{\ell}}^{(b)})$ varies across batches even for identical inputs, since the surviving token set $\mathcal{V}^{(b)} = \{v : m_v^{(b)} = 1\}$ is resampled every batch.

The mask is applied in both phases of each iteration. During GRPO training, the mask is applied at the sampling stage of each rollout, so the policy can only generate from unmasked tokens. The subsequent policy-gradient update computes log-probabilities over the full vocabulary, allowing gradients to flow through all tokens. Because the available tokens change every batch, the policy cannot concentrate probability on a fixed set of high-reward sequences, slowing entropy collapse (Cui et al., 2025) and preserving exploration across the token distribution (Wang et al., 2025b). During curriculum generation, each batch draws from a different token subset, forcing the proposer to produce varied phrasings and problem structures. The proposer therefore uses a larger effective vocabulary despite fewer tokens per batch, as confirmed in Section 6.2. Implementation details and a minimal code example are in Section B.2.

4.2 Training procedure

Both models are trained with GRPO (Shao et al., 2024) in alternating phases (pseudocode in Algorithm 1, prompt templates in Section F).

Proposer phase. The proposer generates competition-level math problems (Section F) via GRPO under vocabulary dropout with retention probability α. Each rollout is checked for valid format, and valid problems are scored by the frozen solver using the self-consistency reward from Section 3.

Solver phase. After the proposer finishes training, it generates a large question set with vocabulary dropout still active. Problems within the difficulty window are retained, and the solver trains on this filtered set via GRPO with a binary reward for matching the proposer's stated answer. This differs from R-Zero, which uses the solver's majority-vote pseudo-label as the reward target.

5 Experiments

5.1 Setup

Vocabulary dropout. We apply vocabulary dropout in both phases (GRPO training and curriculum generation) and test retention probabilities $\alpha \in \{0.75, 0.85\}$ at each model scale, alongside the baseline ($\alpha = 1.0$) which retains the full vocabulary. These values remove 15% and 25% of the non-protected vocabulary respectively, bracketing the range where the tradeoff between diversity and output coherence is most apparent. To understand where the mask has the most effect, we additionally ablate train-only and gen-only variants, isolating the contribution of each phase. Running all conditions at both scales tests whether the optimal masking strength depends on model capacity.

Models and training. Following R-Zero (Huang et al., 2025), we use Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) as base models, initializing both the proposer and solver from the same checkpoint. All training is conducted on 2 NVIDIA H200 GPUs. We train with GRPO (Shao et al., 2024) using `verl` (Sheng et al., 2025) and `vLLM` (Kwon et al., 2023) for 5 co-evolution iterations, extending beyond R-Zero's 3-iteration protocol (where peak evaluation performance did not always occur at the final iteration). During proposer GRPO, we sample G=4 responses per prompt at temperature 1.0 (global batch size 16). For proposer scoring, the frozen solver is sampled M=10 times per question to compute self-consistency. The solver then trains on the filtered curriculum with global batch size 8. We use AdamW with learning rate 1×10^{-6}, weight decay 10^{-2}, and KL penalty (β=10^{-2}). Full hyperparameters are in Section E.

Diversity metrics. We measure proposer diversity at three granularities. Lexical diversity captures surface repetition via self-BLEU. Semantic diversity captures conceptual variety via the Vendi score (Friedman & Dieng, 2022), which measures the effective number of distinct question types in embedding space, and novelty rate, the fraction of questions whose nearest-neighbour cosine distance to all prior iterations exceeds 0.3. Both metrics use `text-embedding-3-small` (OpenAI, 2024) embeddings. Functional diversity captures curriculum quality via epiplexity, introduced by Finzi et al. (2026) and applied to self-play curricula by Liu et al. (2026), which measures the learnable information content of generated questions through prequential minimum description length. We also report mean difficulty, proposer policy entropy (H_π), and structural metrics including question length and unique token counts (Sections D.5 and 6.2).

Standard benchmarks. We evaluate solver capability on GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AMC (Mathematical Association of America, 2023), OlympiadBench (He et al., 2024), and AIME 2024/2025 (Mathematical Association of America, 2024), spanning grade-school arithmetic through olympiad-level reasoning. Annealing

Table 1: Solver pass@1 accuracy (%) after 5 co-evolution iterations. Vocabulary dropout (VD) is applied to the *proposer* only, and the solver trains on the resulting curriculum. α controls the fraction of the vocabulary retained. Phase ablations decompose α=0.75 into train-only and gen-only. Best per model in **green bold**.

	Pass@1 (%)						
Setting	MATH500	GSM8K	AMC	Olympiad	AIME'24	AIME'25	**Avg.**
Qwen3-4B							
Base (no training)	54.1 ± 2.2	63.5 ± 5.2	29.2 ± 1.8	20.6 ± 1.3	12.2 ± 0.9	1.1 ± 0.9	30.1
Baseline (α=1.0)	64.2 ± 0.8	85.2 ± 0.7	41.7 ± 1.4	23.2 ± 0.3	8.9 ± 0.9	6.7 ± 1.6	38.3
α=0.85	65.7 ± 0.9	81.8 ± 0.4	**50.0** ± 2.4	24.7 ± 0.9	8.9 ± 1.8	4.4 ± 0.9	**39.3**
α=0.75	**66.0** ± 0.7	82.4 ± 1.6	35.8 ± 5.9	**24.8** ± 0.1	5.6 ± 1.8	4.4 ± 1.8	36.5
train-only	64.5 ± 0.4	85.4 ± 1.1	31.7 ± 1.8	23.6 ± 0.2	**11.1** ± 3.3	**7.8** ± 1.8	37.4
gen-only	62.9 ± 1.3	**85.9** ± 0.7	42.5 ± 1.2	24.1 ± 0.8	7.8 ± 0.9	3.3 ± 1.6	37.8
Qwen3-8B							
Base (no training)	70.7 ± 0.1	82.2 ± 0.7	45.8 ± 3.6	18.1 ± 1.3	8.9 ± 2.4	7.8 ± 0.9	38.9
Baseline (α=1.0)	69.9 ± 0.4	86.9 ± 2.5	40.8 ± 7.7	23.1 ± 1.8	12.2 ± 0.9	3.3 ± 1.6	39.4
α=0.85	69.7 ± 2.3	89.5 ± 0.8	45.8 ± 1.8	26.8 ± 0.5	6.7 ± 0.0	7.8 ± 3.3	41.1
α=0.75	68.1 ± 2.3	88.6 ± 1.4	**51.7** ± 3.6	27.7 ± 0.3	**15.6** ± 2.4	**11.1** ± 2.4	**43.8**
train-only	**71.1** ± 1.5	**89.7** ± 0.6	44.2 ± 3.6	27.0 ± 1.1	13.3 ± 0.0	5.6 ± 2.4	41.8
gen-only	71.0 ± 2.6	87.0 ± 1.2	49.2 ± 4.1	**29.2** ± 0.7	11.1 ± 0.9	5.6 ± 2.4	42.2

experiments (Section 7) additionally use Minerva Math (Lewkowycz et al., 2022). We also evaluate the untrained base models under the same protocol as a reference. All evaluation is zero-shot using the model's default chat template (Section F.2) with temperature 0.7 and a maximum context length of 4096 tokens, scored by exact symbolic answer matching via `math_verify` (Hugging Face, 2025). All results are averaged over 3 independent runs, reporting mean ± standard error (Miller, 2024).

6 Results

6.1 Solver performance

Table 1 reports pass@1 accuracy at the final co-evolution iteration. We avoid selecting the best checkpoint or run post hoc to prevent selection bias (Cawley & Talbot, 2010; Dodge et al., 2019).

Vocabulary dropout improves solver performance at both scales. We use VD$\langle\alpha\rangle$ as shorthand for vocabulary dropout at retention probability α (e.g., VD75 means α=0.75). At 8B, VD75 leads all configurations with a +4.4 point average improvement over the baseline, driven by strong gains on competition-level tasks (AMC, AIME). At 4B, VD85 edges out the baseline on average. All configurations, including the baseline, improve over the untrained base models.

On the proposer side, co-evolutionary training does not improve downstream reasoning ability (Table 3). The 8B proposer (VD75: 38.4, baseline: 38.8) slightly degrades relative to the base model (38.9), and 4B shows a similar pattern. At 4B, proposer performance drops modestly under dropout (VD85: 31.9 vs. baseline: 32.8), consistent with the tighter constraint on the smaller model. This is expected, as the proposer is optimized for calibrated problem generation, not problem solving, and these objectives do not appear to transfer.

Ablations. Table 1 also isolates the effects of masking strength and dropout phase. At 4B, VD85 outperforms the baseline while VD75 underperforms, indicating that the optimal masking strength is scale-dependent and that overly aggressive masking can exceed the model's capacity to compensate. The phase decomposition at 4B shows gen-only slightly

edging train-only, but neither recovers the baseline, confirming that the bottleneck at this scale is masking strength rather than which phase the mask is applied in.

At 8B, the pattern differs. Stronger masking consistently improves performance, and both single-phase variants surpass the baseline, suggesting that the 8B model has sufficient redundancy in its token representations to absorb the constraint without degrading output quality. Gen-only slightly edges train-only at both scales, but the combination outperforms either alone at 8B. The phase ranking (gen $\geq$ train) is consistent across scales, and what differs is whether the model has enough capacity to absorb VD75 at all. Embedding-level analysis of these phase ablations (Section D.2, Table 4) confirms that gen-only contributes the larger share of diversity, but the combination sustains the strongest growth across iterations.

6.2 Proposer diversity

Figure 3 tracks six diversity and curriculum quality metrics over co-evolution iterations.

Baseline stagnation is early and hidden. The baseline proposer's functional collapse occurs almost entirely between iterations 1 and 2, then plateaus (Figure 3). Mean difficulty (panel c) jumps from 0.50 to 0.86 at 4B and from 0.71 to 0.92 at 8B in a single iteration, pushing solver accuracy above the $\tau_{\max}$=0.7 threshold. Policy entropy (a), Vendi score (d), and epiplexity (f) all show the same early lock-in. Meanwhile, Self-BLEU and novelty rate (b, e) remain flat throughout, confirming that the proposer recycles similar questions that surface metrics fail to detect. Diversity interventions applied after the first iteration may be too late to recover.

Dropout sustains diversity across all tiers. Under dropout, all functional metrics continue to evolve past iteration 2, and the proposer's distribution remains responsive to the solver's improvement rather than freezing after the first update. Vocabulary dropout improves every diversity metric we measure (Figure 3). At the *lexical* level, Self-BLEU is roughly 2$\times$ lower under dropout at both scales (b). At the *semantic* level, the Vendi score shows a gap of $\sim$15 effective question types at both scales (d), and novelty rate is $\sim$1.5–2$\times$ higher under dropout (e). At the *functional* level, epiplexity is $\sim$35% higher under dropout, indicating substantially more learnable structure in the curriculum (computation details in Sections E.3 and E.4). Cumulative Vendi Scores (Section D.1) show the same pattern, with the baseline plateauing after iteration 2–3 while VD75 continues to grow.

Figure 2 confirms this at the structural level. Dropout questions contain more distinct numeric values and are substantially longer, while baseline runs produce roughly 2$\times$ more valid questions per iteration, consistent with simpler problems that pass format and reward filters at a higher rate. At the token level, VD75 proposers use 36–52% more unique tokens than the baseline despite fewer being available at any given step (Section D.5), confirming that the non-stationary mask forces broader vocabulary utilization rather than concentration.

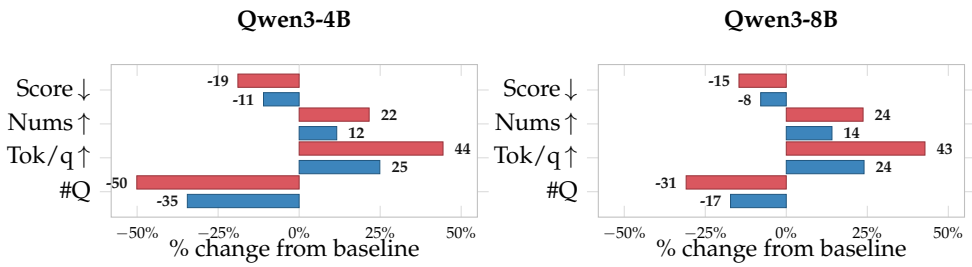

Figure 2: Question profile at iteration 5 (% change from baseline). VD85 VD75.

Diversity is necessary but not sufficient. Comparing diversity metrics to solver performance (Table 1) reveals that higher diversity does not automatically yield a better solver. At 4B, VD75 exceeds the baseline on every diversity metric reported above, yet its solver

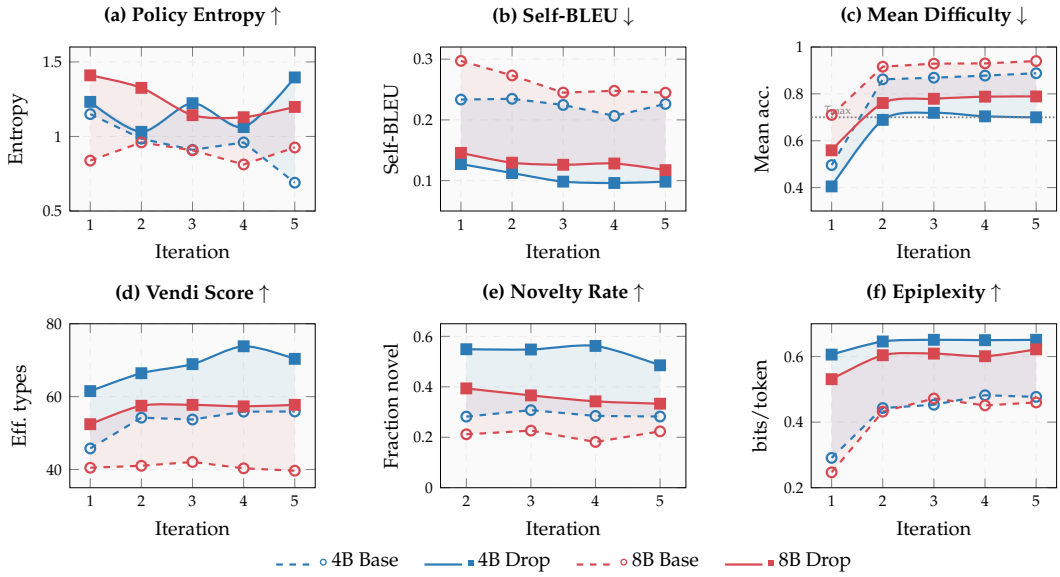

Figure 3: Diversity and curriculum quality over co-evolution iterations (α=0.75, both phases). Top: collapse signals. Bottom: diversity metrics. Dashed = baseline, solid = dropout. Semantic metrics (b, d, e) use `text-embedding-3-small` (OpenAI, 2024).

underperforms (36.5 vs. 38.3 avg). Only VD85, which applies a milder mask, beats the baseline (39.3). The likely explanation is that overly aggressive masking produces diverse but incoherent problems, leaving the curriculum with too many malformed or unsolvable items for the solver to extract a consistent training signal. This is consistent with the phase ablations, where single-phase VD75 variants at 4B also fall short of the baseline, while at 8B the same masking strength produces both higher diversity and higher solver accuracy. Masking strength must be calibrated so that the diversity gain does not exceed the model's capacity to maintain problem coherence.

7 Discussion

Action-space constraints for language self-play. Our results support the hypothesis that co-evolutionary self-play in language benefits from action-space constraints analogous to those that game rules provide. Without such constraints, the baseline proposer converges to a narrow problem distribution within the first two iterations, despite R-Zero's BLEU-based repetition penalty in the reward function. This confirms that reward-side diversity pressure alone is insufficient to prevent functional stagnation. Vocabulary dropout addresses this by operating directly on the proposer's output space rather than through the reward. The non-stationary mask forces the proposer into different regions of token space, much as game rules force players into different strategic positions each match. The diversity gains we observe at the lexical, semantic, and functional levels (Section 6.2) are consistent with this mechanism, and the resulting solver improvements (Table 1) confirm that the sustained diversity translates to a more informative curriculum.

Curriculum quality vs. curriculum diversity. Proposer correctness degrades across all conditions as training progresses (Table 2), reflecting a limitation of self-consistency verification shared by both the baseline and vocabulary dropout. It cannot distinguish genuinely hard problems from ill-posed ones, since both produce solver disagreement. However, vocabulary dropout addresses the orthogonal problem of diversity collapse. While the baseline proposer generates a narrow, repetitive curriculum of declining quality, vocabulary dropout maintains a diverse curriculum in which the well-posed subset is varied enough to compensate for the noise. This is consistent with Shao et al. (2025), who show that GRPO

can amplify pre-training priors even under spurious rewards, and Setlur et al. (2024), who demonstrate that training on incorrect synthetic data can still yield gains when negative signals are properly structured. Pairing vocabulary dropout with stronger verification (e.g., code execution or symbolic solvers) could capture the diversity benefit while filtering the noise (Sections C and C.2).

Non-stationarity and scale. Our phase ablations (Table 1) show that the value of vocabulary dropout is scale-dependent. One explanation is that larger models have more redundant token representations and can reroute around masked tokens without degrading coherence. Stronger non-stationary pressure then produces diversity gains without degrading coherence, though we do not directly test this mechanism. This points toward adaptive masking schedules that scale α with model size or anneal it over training.

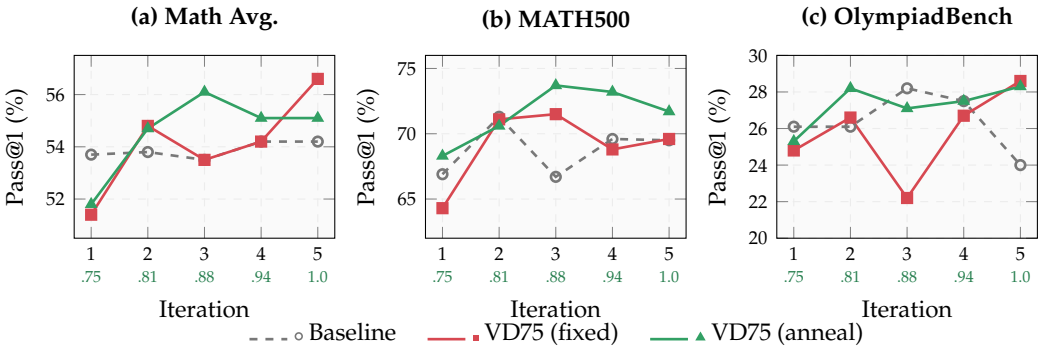

Figure 4: Qwen3-8B solver accuracy across iterations under fixed vs. annealed (0.75→1.0) vocabulary dropout. Green α values below each tick show the anneal schedule. (a) Mean of MATH500, GSM8K, OlympiadBench, and Minerva Math.

We test a linear schedule increasing α from 0.75 to 1.0 over the 5 iterations (Figure 4). The anneal peaks two iterations earlier than fixed VD75 (iteration 3 vs 5), wins 3 of 5 iterations on math-average, and achieves the highest cumulative performance across all iterations. It also produces a more stable trajectory, with lower iteration-to-iteration volatility and an OlympiadBench range of 3.0 points vs 6.4 for fixed (Figure 4c). Frontloading diversity pressure when stagnation risk is highest and relaxing it as the solver matures merits further investigation.

Scope and limitations. When the proposer already has more capacity than the solver (8B→4B cross-scale, Section D.3), adding dropout hurts rather than helps (−1.4 avg), because the difficulty calibration is already misaligned and diversity pressure amplifies the mismatch. Similarly, co-evolutionary training with or without dropout does not consistently benefit instruction-tuned models at the scales we tested (Section D.4). The intervention is most effective when proposer and solver are capacity-matched and trained from base models.

8 Conclusion

We showed that a simple, non-stationary constraint on the proposer's output vocabulary is sufficient to sustain curriculum diversity and improve solver performance in co-evolutionary self-play. Just as dropout on hidden units prevents co-adaptation of neurons, vocabulary dropout softens co-adaptation of token sequences, requiring no auxiliary models, or reward modifications. The random mask forces the proposer to maintain a diverse, informative curriculum rather than collapsing to trivially easy problems that stop driving solver improvement. Our results suggest that action-space structure, rather than reward design alone, is a productive lever for controlling co-evolutionary dynamics in language. Vocabulary dropout is one instantiation of this idea. Combining it with stronger verification and extending it to non-mathematical domains are the most immediate extensions toward LLM self-play systems that sustain diverse, informative curricula as both models improve.

Ethics Statement

This work studies training dynamics in co-evolutionary LLM systems using mathematical reasoning as a testbed. The models, data, and methods are standard in the field. We do not foresee specific negative societal consequences beyond those common to language model research.

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Gavin C. Cawley and Nicola L.C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.*, 11:2079–2107, August 2010. ISSN 1532-4435.

Justin Yang Chae, Md Tanvirul Alam, and Nidhi Rastogi. Towards understanding self-play for llm reasoning. *arXiv preprint arXiv:2510.27072*, 2025.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pp. 941–950. PMLR, 2019.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Jacob Dineen, Aswin Rrv, Qin Liu, Zhikun Xu, Xiao Ye, Ming Shen, Zhaonan Li, Shijie Lu, Chitta Baral, Muhao Chen, et al. Qa-lign: Aligning llms through constitutionally decomposed qa. *arXiv preprint arXiv:2506.08123*, 2025.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. Show your work: Improved reporting of experimental results. 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. 2185–2194, 2019.

Gregory Farquhar, Laura Gustafson, Zeming Lin, Shimon Whiteson, Nicolas Usunier, and Gabriel Synnaeve. Growing action spaces. In *International Conference on Machine Learning*, pp. 3040–3051. PMLR, 2020.

Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J Zico Kolter, and Andrew Gordon Wilson. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *arXiv preprint arXiv:2601.03220*, 2026.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410*, 2022.

Jingchu Gai, Guanning Zeng, Huaqing Zhang, and Aditi Raghunathan. Differential smoothing mitigates sharpening and improves llm reasoning. *arXiv preprint arXiv:2511.19942*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Hugging Face. Math-verify: Robust mathematical expression evaluator. https://github.com/huggingface/Math-Verify, 2025. GitHub repository; accessed 2025-09-09.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, Alon Albalak, and Yejin Choi. Artificial hivemind: The open-ended homogeneity of language models (and beyond). *arXiv preprint arXiv:2510.22954*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Wei Liu, Siya Qi, Yali Du, and Yulan He. Self-play only evolves when self-synthetic pipeline ensures learnable information gain. *arXiv preprint arXiv:2603.02218*, 2026.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

Mathematical Association of America. American mathematics competitions (AMC) 2023. https://artofproblemsolving.com/wiki/index.php/AMC, 2023.

Mathematical Association of America. American invitational mathematics examination (AIME) 2024. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I, 2024.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Vaibhav Mishra. Preventing curriculum collapse in self-evolving reasoning systems. *arXiv preprint arXiv:2603.13309*, 2026.

Yuta Nozaki, Dai Nakashima, Ryo Sato, Naoki Asaba, and Shintaro Kawamura. Efficient vocabulary reduction for small language models. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, Steven Schockaert, Kareem Darwish, and Apoorv Agarwal (eds.), *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 771–783, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-industry.64/.

OpenAI. New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2024. Accessed: 2025-06-01.

OpenAI. GPT-4.1 and GPT-4.1 mini. https://openai.com/index/gpt-4-1/, 2025. Accessed: 2025-06-01.

Jing-Cheng Pang, Liang Lu, Xian Tang, Kun Jiang, Sijie Wu, Kai Zhang, and Xubin Li. Reinforcement learning with promising tokens for large language models. *arXiv preprint arXiv:2602.03195*, 2026.

Aswin Rrv, Jacob Dineen, Divij Handa, Md Nayem Uddin, Mihir Parmar, Chitta Baral, and Ben Zhou. Thinktuning: Instilling cognitive reflections without distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31236–31250, 2025.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *Advances in Neural Information Processing Systems*, 37:43000–43031, 2024.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017. URL http://dx.doi.org/10.1038/nature24270.

Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. 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. URL https://api.semanticscholar.org/CorpusID:204972004.

Shaobo Wang, Zhengbo Jiao, Zifan Zhang, Yilang Peng, Xu Ze, Boyu Yang, Wei Wang, Hu Wei, and Linfeng Zhang. Socratic-zero: Bootstrapping reasoning via data-free agent co-evolution. *arXiv preprint arXiv:2509.24726*, 2025a.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025b.

Alex Wilf, Pranjal Aggarwal, Bryan Parno, Daniel Fried, Louis-Philippe Morency, Paul Pu Liang, and Sean Welleck. Propose, solve, verify: Self-play through formal verification. *arXiv preprint arXiv:2512.18160*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*, 2025.

A Algorithm

Algorithm 1 gives the full co-evolutionary training loop.

Algorithm 1: Vocabulary Dropout Co-Evolutionary Self-Play

Input : Base model θ_0;
retention prob. α;
difficulty window $[\tau_{\min}, \tau_{\max}]$;
iterations T;
proposer GRPO group size G;
solver self-consistency samples M

Output: Trained proposer $\pi_P^{(T)}$ and solver $\pi_S^{(T)}$

$\pi_P^{(0)}, \pi_S^{(0)} \leftarrow \theta_0$
$\mathcal{F} \leftarrow$ protected format-critical tokens `// always survive`
for $t = 1, \ldots, T$ **do**
  `/* Phase 1: Train proposer (solver frozen) */`
  for *each GRPO batch* **do**
    $\mathcal{M}^{(b)} \leftarrow \{v \in \mathcal{V} : m_v \sim \text{Bern}(\alpha)\} \cup \mathcal{F}$ `// fresh mask per batch`
    Sample $\{o_i\}_{i=1}^{G}$ from π_P, setting logits to $-\infty$ for tokens $\notin \mathcal{M}^{(b)}$
    for *each rollout* o_i **do**
      Parse q_i, a_i from o_i `// question, boxed answer`
      if *parse fails* **then** $r_i^P \leftarrow 0$;
      continue
      Query frozen $\pi_S^{(t-1)}$ with q_i;
      collect M responses
      $\text{acc}_i \leftarrow$ fraction matching majority vote
      $r_i^P \leftarrow \min(\text{acc}_i,\ 1-\text{acc}_i) \cdot \mathbf{1}[\text{acc}_i \in [\tau_{\min}, \tau_{\max}]]$
    end
    Update π_P via GRPO on $\{(o_i, r_i^P)\}$
  end
  $\pi_P^{(t)} \leftarrow \pi_P$
  `/* Phase 2: Generate curriculum, then train solver */`
  $\mathcal{Q} \leftarrow$ sample problems from $\pi_P^{(t)}$ with per-batch $\mathcal{M}^{(b)}$
  $\mathcal{Q}_{\text{train}} \leftarrow \{(q, a) \in \mathcal{Q} : \text{acc}(q) \in [\tau_{\min}, \tau_{\max}]\}$ `// a is the proposer's stated answer`
  for *each GRPO batch* **do**
    Sample solver rollouts for $(q, a) \in \mathcal{Q}_{\text{train}}$
    $r^S \leftarrow \mathbf{1}[\text{solver answer} = a]$
    Update π_S via GRPO on $\{(q, r^S)\}$
  end
  $\pi_S^{(t)} \leftarrow \pi_S$
end
return $\pi_P^{(T)}, \pi_S^{(T)}$

B Implementation details

B.1 Verification via solver self-consistency

Following R-Zero (Huang et al., 2025), we verify proposer outputs using the solver itself rather than an external reward model. Each proposer output is checked for valid format

(<question> tags and a \boxed{} answer). Valid questions are sent to the frozen solver, which attempts each problem 10 times. The proposer reward is $r = \min(\text{acc}, 1-\text{acc})$, where acc is the fraction of solver responses matching the majority vote. This peaks at 50% solve rate, incentivizing problems at the solver's decision boundary.

For solver training, generated questions are filtered to the difficulty window $[\tau_{\min}, \tau_{\max}] = [0.3, 0.7]$, following R-Zero.

B.2 Vocabulary dropout implementation

Vocabulary dropout is implemented via vLLM's `allowed_token_ids` parameter in `SamplingParams`, which sets logits to $-\infty$ for all tokens outside the allowed set before sampling. At the start of each batch, we draw a fresh Bernoulli mask over the full vocabulary and take the union with a protected set $\mathcal{F}$ of format-critical tokens needed for structural validity and answer formatting. The resulting token ID list is passed to vLLM, and no modifications to the model weights or sampling kernel are required. Figure 5 shows the minimal code change relative to the standard R-Zero rollout.

```
def rollout(model, prompts,
-  sampling_params):
+  sampling_params,
+  tokenizer, alpha=0.75):
+ V = tokenizer.vocab_size
+ mask = torch.bernoulli(
+     torch.full((V,), alpha)
+ ).bool()
+ # Protect format-critical tokens
+ for t in get_protected_token_ids(
+     tokenizer):
+   mask[t] = True
+ sampling_params.allowed_token_ids = (
+     mask.nonzero(as_tuple=True)[0]
+     .tolist())
   completions = model.generate(
       prompts, sampling_params
   )
   return completions
```

Figure 5: Vocabulary dropout as a unified diff. The only change is sampling a Bernoulli mask and passing the surviving token IDs to vLLM's `allowed_token_ids` before generation. The mask is resampled every batch.

B.3 Compute requirements

All experiments use 2 NVIDIA H200 GPUs per run. During the proposer phase, one GPU handles GRPO training while the other runs a vLLM inference server for self-consistency evaluation. Curriculum generation and question evaluation are parallelized across both GPUs. During the solver phase, both GPUs are used for GRPO training. We run 5 iterations per experiment and 3 independent seeds per configuration. Benchmark evaluations are conducted using vLLM on 4–8 NVIDIA H100 GPUs.

Vocabulary dropout requires no model weight or architecture changes. The only addition is sampling a Bernoulli mask and passing the resulting token ID list to vLLM's `allowed_token_ids` parameter, which is enforced at the logit level within the existing sampling kernel.

C Qualitative examples

We present representative problems generated by the baseline and vocabulary dropout proposers at the same co-evolution iteration, as well as examples of ill-posed problems that vocabulary dropout can produce.

C.1 Baseline vs. vocabulary dropout

The following pairs are drawn from iteration-5 question sets of the 8B proposer. For each pair, the baseline and dropout examples share the same broad semantic theme. Dropout examples were filtered to those with solver accuracy ≥ 0.7.

Baseline 8B – Baking

Sarah is baking cookies for her class. She has 5 trays, and each tray can hold 12 cookies. If she wants to bake 60 cookies in total, how many batches does she need to bake if each batch uses all 5 trays?

VD75 8B – Baking

Sarah is baking cookies to sell at the bake sale to raise funds for the kindergarten program at her local elementary school. She plans to bake three types: sugar cookies, oat cakes, and granola bars. She baked 120 cookies overall. If half are sugar cookies and four-fifths of the remaining are oat cakes, how many granola bars did she bake?

Baseline 8B – Farming

A farmer has a rectangular field that is 30 meters long and 20 meters wide. He wants to plant apple trees in rows spaced 5 meters apart and plants 3 trees in each row. How many apple trees can the farmer plant in his field?

VD75 8B – Farming

At the local farmer stand, apples are sold at $2 per kilo and bananas at $3 per kilo. A customer bought 1 kilo of apples and 1 kilo of bananas on Monday, and 2 kilos of apples and 2 kilos of bananas on Tuesday. What was the customer's total expenditure over the two days?

C.2 Ill-posed problems

Self-consistency verification rewards solver disagreement, but cannot distinguish genuinely hard problems from ill-defined ones. Both the baseline and vocabulary dropout proposers produce problems where the proposer's stated ground truth is incorrect. In each case below, all 10 solvers answered correctly but were scored as wrong (score = 1.0) because they disagreed with the proposer's erroneous answer.

Baseline 8B

Sophie bought 126 seeds. She planted 34 on Saturday and 58 on Sunday. How many left? **Prop:** 92 **Solvers:** 34 ✓ (10/10)

Subtracted only Saturday; forgot Sunday. Correct: 126−34−58=34.

VD75 8B

Alice has 48 marigolds and 36 roses, bundles them equally (largest possible). How many marigolds per bundle? **Prop:** 48 **Solvers:** 12 ✓ (10/10)

Wrote total count (48) instead of bundle size (GCD = 12).

Baseline 8B

It takes 30 seconds to fill a 6-liter jug. How many liters in 2 minutes? **Prop:** 4 **Solvers:** 24 ✓ (10/10)

Computed fills $(120 \div 30 = 4)$ *but forgot jug capacity. Correct:* $4 \times 6 = 24$.

VD75 8B

Mike paints a fence in 2 hr, Tom in 3 hr. How many minutes together? **Prop:** 1 min . . . **Solvers:** 72 ✓ (10/10)

Confused hourly with per-minute rates; answer truncated. Correct: 72 min.

Baseline 4B

John had 50 dollars. He bought a 10 dollar ice cream. How many dollars left? **Prop:** 50 **Solvers:** 40 ✓ (10/10)

Echoed starting amount without subtracting. Correct: 50−10=40.

VD75 4B

Simon had 30 cars at 12, half of age-15 count. Simon had half of Lou's at 15. Lou's total? **Prop:** 60 **Solvers:** 120 ✓ (10/10)

Stopped one step early; attributed Simon's count to Lou. Correct: $60 \times 2 = 120$.

Baseline 4B

180 pages in a book. Emma reads $\frac{1}{5}$ *in week 1 and* $\frac{2}{5}$ *in week 2. How many in week 2?* **Prop:** 36 **Solvers:** 72 ✓ (10/10)

Computed week-1 fraction instead of week-2. Correct: $\frac{2}{5} \times 180 = 72$.

VD75 4B

John bought 4 fish, returned half. How many left? **Prop:** 4 **Solvers:** 2 ✓ (10/10)

Echoed starting count without applying return. Correct: 4−2=2.

These failures arise from a fundamental limitation of self-consistency verification, which equates solver disagreement with problem difficulty regardless of answer correctness. The proposer receives maximum reward for these problems precisely because it provided a wrong answer that no solver reproduces. Both the baseline and dropout settings exhibit this failure mode, which is a property of the verification scheme rather than vocabulary dropout.

C.3 Proposer answer correctness

To estimate the fraction of generated problems where the proposer's stated answer is correct, we use GPT-4.1-mini (OpenAI, 2025) as an independent verifier. For each problem at every iteration, the verifier solves the problem independently and compares its answer to the proposer's. Table 2 reports correctness trajectories across all conditions and phase ablations.

Table 2: Proposer answer correctness (%) over co-evolution iterations, judged by GPT-4.1-mini solving each problem independently.

Model	**Setting**	It. 1	It. 2	It. 3	It. 4	It. 5
Qwen3-8B	Baseline	81.4	45.5	42.2	39.1	38.0
	Train-only	79.0	43.2	42.0	40.4	39.0
	Gen-only	62.0	38.3	33.7	33.4	31.9
	VD75	61.8	34.9	33.1	36.7	32.4
Qwen3-4B	Baseline	75.6	38.1	38.0	42.2	37.7
	Train-only	76.0	37.2	44.2	38.0	38.2
	Gen-only	56.0	34.5	32.1	29.5	32.4
	VD75	61.2	31.2	30.5	30.7	31.9

Table 3: Proposer reasoning ability (pass@1 %) on standard benchmarks. VD is applied to the proposer during co-evolution. Co-evolutionary training does not improve the proposer's own problem-solving ability.

	Pass@1 (%)						
Setting	MATH500	GSM8K	AMC	Olympiad	AIME'24	AIME'25	**Avg.**
Qwen3-4B							
Base (no training)	54.1 ± 2.2	63.5 ± 5.2	29.2 ± 1.8	20.6 ± 1.3	12.2 ± 0.9	1.1 ± 0.9	30.1
Baseline (α=1.0)	56.0 ± 1.1	72.7 ± 3.4	33.3 ± 3.0	20.3 ± 0.9	10.0 ± 1.6	4.4 ± 0.9	32.8
VD85 (α=0.85)	58.4 ± 0.8	60.5 ± 4.1	37.5 ± 4.1	19.3 ± 1.1	11.1 ± 0.9	4.4 ± 2.4	31.9
Qwen3-8B							
Base (no training)	70.7 ± 0.1	82.2 ± 0.7	45.8 ± 3.6	18.1 ± 1.3	8.9 ± 2.4	7.8 ± 0.9	38.9
Baseline (α=1.0)	65.1 ± 2.1	87.5 ± 1.2	40.8 ± 3.8	21.8 ± 0.7	7.8 ± 0.9	10.0 ± 1.6	38.8
VD75 (α=0.75)	61.7 ± 1.7	85.2 ± 1.4	40.0 ± 3.1	23.3 ± 1.6	11.1 ± 0.9	8.9 ± 1.8	38.4

All conditions show a sharp correctness drop from iteration 1 to 2, then stabilize. This mirrors the stagnation pattern observed in Section 6.2, where the proposer's distribution shifts rapidly in the first iteration as it learns to target the solver's decision boundary, then locks in. Two clusters emerge at convergence. Conditions without generation-phase dropout (baseline, train-only) stabilize at ∼38–42% correctness, while conditions with generation-phase dropout (gen-only, VD75) settle at ∼30–34%. The ∼7% gap reflects the cost of diversity, as the non-stationary mask produces more varied but less reliably correct problems. Despite this, VD75 produces the strongest solver improvements at 8B (Table 1), suggesting that the diversity of the well-posed subset more than compensates for the increased noise. This reinforces the case for composing vocabulary dropout with stronger verification, as discussed in Section 7.

Table 4: Cumulative question diversity (Vendi Score) over co-evolution iterations via `text-embedding-3-small` embeddings. *Growth* is the gain from iteration 1 to iterations 1–5 pooled.

		Cumulative Vendi Score ↑					
Model	**Setting**	It. 1	1–2	1–3	1–4	1–5	**Growth**
Qwen3-4B	Baseline	44.6	46.1	48.9	49.5	52.1	+7.5
	Train-only	46.1	51.0	52.0	53.7	55.7	+9.6
	Gen-only	59.7	62.9	63.9	63.9	64.3	+4.6
	Both	61.5	63.9	68.6	69.7	**70.2**	**+8.7**
Qwen3-8B	Baseline	40.0	41.5	42.1	43.2	42.2	+2.2
	Train-only	41.9	42.6	43.5	44.2	46.1	+4.2
	Gen-only	52.9	54.1	55.2	55.3	55.5	+2.6
	Both	51.1	53.7	56.0	57.0	**57.2**	**+6.1**

D Additional results

D.1 Cumulative embedding diversity

Figure 3 reports per-iteration Vendi scores and novelty rates. As a complementary view, Figure 6 tracks the cumulative Vendi Score, where all questions from iterations 1 through N are pooled (subsampled to 2,000), embedded with `text-embedding-3-small` (OpenAI, 2024), and scored. VD75 maintains ∼35% higher cumulative diversity than the baseline at both scales, and the baseline plateaus after iteration 2–3 while VD75 continues to grow.

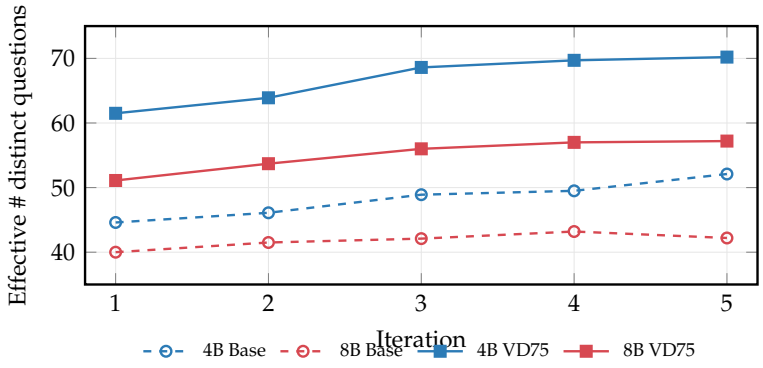

Figure 6: Cumulative Vendi Score (questions pooled across iterations 1–N, subsampled to 2,000). VD75 maintains higher diversity at both scales.

D.2 Phase ablation embedding analysis

Table 4 tracks cumulative Vendi Scores as questions are pooled across iterations, and Figure 7 shows per-iteration trends for all three embedding metrics. Gen-only dropout starts with high diversity but plateaus quickly (+4.6/+2.6 growth at 4B/8B), suggesting that without a regularized solver the proposer's distribution converges. Train-only starts lower but sustains steady growth (+9.6/+4.2), indicating that the regularized solver feeds back different reward signals that prevent proposer collapse. Both phases combined yield the highest cumulative diversity and the strongest growth (+8.7/+6.1), consistent with the solver accuracy results in Table 1.

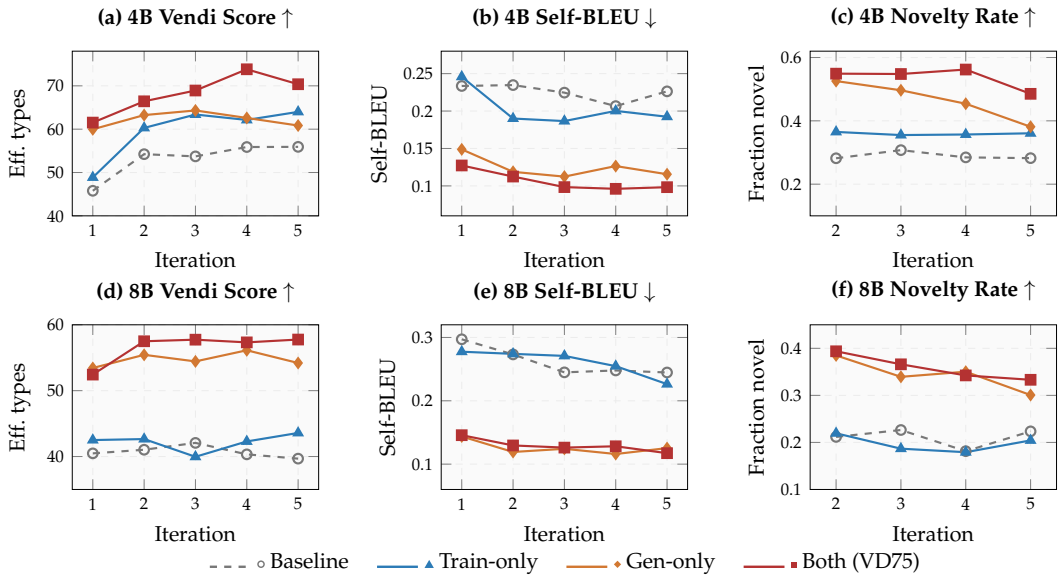

Figure 7: Embedding diversity by dropout phase (α=0.75). Both phases combined achieves the highest diversity. All metrics use `text-embedding-3-small`.

D.3 Cross-scale co-evolution

All experiments in Section 6 pair each model with itself (4B→4B, 8B→8B). A natural question is whether a stronger proposer generates a better curriculum for a weaker solver. We test this by pairing a Qwen3-8B proposer with a Qwen3-4B solver, with and without vocabulary dropout (α=0.85, gen-only) on the 8B proposer.

Table 5: Cross-scale co-evolution: 8B proposer → 4B solver, compared with symmetric 4B→4B from Table 1. Best per group in **green bold**.

		Pass@1 (%)						
Proposer	**Setting**	MATH500	GSM8K	AMC	Olympiad	AIME'24	AIME'25	**Avg.**
4B	Baseline	64.2 ± 0.8	**85.2** ± 0.7	41.7 ± 1.4	23.2 ± 0.3	8.9 ± 0.9	**6.7** ± 1.6	38.3
4B	+ VD85	**65.7** ± 0.9	81.8 ± 0.4	**50.0** ± 2.4	**24.7** ± 0.9	**8.9** ± 1.8	4.4 ± 0.9	**39.3**
8B	Baseline	63.5 ± 2.7	82.7 ± 2.5	**44.2** ± 2.4	**24.4** ± 0.1	11.1 ± 1.6	**6.7** ± 5.4	**38.8**
8B	+ VD85 (gen-only)	**66.1** ± 1.1	**83.4** ± 2.7	35.8 ± 4.3	23.4 ± 0.7	**13.3** ± 2.7	2.2 ± 1.6	37.4

The cross-scale baseline (8B→4B, no dropout) performs comparably to the symmetric baseline (4B→4B): 38.8 vs. 38.3 avg. A stronger proposer does not automatically produce a better curriculum, likely because the self-consistency reward calibrates difficulty to the solver's frontier regardless of proposer capacity.

Adding vocabulary dropout to the 8B proposer hurts rather than helps (37.4 vs. 38.8 avg), with a particularly large drop on AMC (−8.4) and AIME'25 (−4.5). This contrasts with the symmetric setting, where VD85 improves the 4B solver by +1.0 avg. This is consistent with the scale-dependent findings in Section 6: vocabulary dropout addresses proposer stagnation during iterative co-adaptation, but when the proposer already has substantially more capacity than the solver, the difficulty calibration is misaligned and the additional diversity pressure exacerbates the mismatch rather than helping. This negative result supports the view that vocabulary dropout is not a generic regularizer but a targeted intervention for co-evolutionary dynamics between capacity-matched models.

D.4 Instruction-tuned models

All main paper experiments use base (pre-trained) models, consistent with R-Zero (Huang et al., 2025) and General-Reasoner (Ma et al., 2025). We additionally run the pipeline on Qwen2.5-1.5B-Instruct (Table 6). No configuration consistently improves over the untrained base on either benchmark. At the tested scale, co-evolutionary training, with or without vocabulary dropout, does not benefit instruction-tuned models in our experiments.

Table 6: Co-evolution on Qwen2.5-1.5B-Instruct (5 iterations).

Model	MATH500	AMC
Base (no training)	47.7	25.3
Baseline solver	47.3	**26.0**
VD75 solver	47.4	25.2

D.5 Vocabulary utilization

Table 7 reports the number of unique tokens used in generated questions at iteration 5. Despite having fewer tokens available per batch, VD75 proposers use substantially more of the vocabulary than the baseline, confirming that the non-stationary mask forces the proposer to explore a wider range of token compositions rather than concentrating on a fixed subset.

Table 7: Unique tokens in iteration-5 generated questions.

	Qwen3-4B		**Qwen3-8B**	
Setting	Unique tok.	# Q	Unique tok.	# Q
Baseline	2,717	6,771	1,881	7,044
VD75	**3,697**	3,377	**2,868**	4,864
Δ	+36%		+52%	

E Hyperparameters

E.1 Evaluation benchmarks

Table 8: Evaluation benchmarks.

Benchmark	**# Examples**	**Domain**
GSM8K (Cobbe et al., 2021)	1,319	Grade-school math
MATH500 (Hendrycks et al., 2021)	500	Competition math
AMC (Mathematical Association of America, 2023)	40	Competition math
OlympiadBench (He et al., 2024)	910	Olympiad math
AIME 2024 (Mathematical Association of America, 2024)	30	Olympiad math
AIME 2025 (Mathematical Association of America, 2024)	30	Olympiad math
Additional benchmarks (annealing experiments)		
Minerva Math (Lewkowycz et al., 2022)	272	STEM math

E.2 Training hyperparameters

Parameter	Proposer	Solver
Base model	Qwen3-4B / Qwen3-8B	
Optimizer	AdamW	
Learning rate	1×10^{-6}	
Weight decay	1×10^{-2}	
LR warmup	0	
Max grad norm	1.0	
KL penalty	low-variance KL, $\beta = 10^{-2}$	
Gradient checkpointing	enabled	
Precision	bfloat16	
Global batch size	16	8
Micro-batch (update)	2	2
Micro-batch (experience)	4	4
Rollout batch size	16	8
GRPO samples per prompt (G)	4	–
Self-consistency samples (M)	10	–
Temperature	1.0	1.0
Top-p	0.99	0.99
Max response length	2048	2048
Max steps per phase	6	20
Epochs per phase	10	10
Co-evolution iterations		5
Difficulty window $[\tau_{\min}, \tau_{\max}]$		$[0.3, 0.7]$
Vocab dropout α	0.75	–
BLEU rep. penalty τ_{BLEU}	0.5	–

Table 9: GRPO training hyperparameters for proposer and solver.

E.3 Vendi score computation

We compute the Vendi score (Friedman & Dieng, 2022) for each (experiment, iteration) pair by embedding the proposer's generated questions and computing the eigenspectrum of the cosine similarity kernel. Given n questions with L2-normalized embeddings $E \in \mathbb{R}^{n \times d}$, the similarity matrix is $K = EE^\top$. We compute $\text{VS} = \exp(-\sum_i \hat{\lambda}_i \log \hat{\lambda}_i)$ where $\hat{\lambda}_i = \lambda_i / \sum_j \lambda_j$ are the normalized eigenvalues of K. This yields the effective number of distinct question types.

Parameter	Value
Embedding model	`text-embedding-3-small`
Embedding dimension	1536
Normalization	L2 (cosine kernel)
Max samples (eigendecomp.)	5,000
Subsampling seed	42
Eigenvalue threshold	10^{-12}

Table 10: Vendi score hyperparameters.

E.4 Epiplexity computation

We compute epiplexity following Finzi et al. (2026) and the prequential MDL procedure of Liu et al. (2026). For each (experiment, iteration) pair, a fresh LoRA observer is fine-tuned on the proposer's generated questions. Epiplexity measures how much structure the observer can extract, defined as the difference between the online loss (before any training on each example) and the converged training loss, converted to bits per token. Higher epiplexity indicates more learnable structure in the curriculum.

The observer is initialized from the same base model used in the co-evolutionary loop, with LoRA adapters on all attention and MLP projections. Training uses an online-then-converge protocol where epoch 1 records the loss on each batch before updating (prequential code length), then training continues for up to 20 epochs with early stopping on a held-out validation set.

Parameter	Value
Observer model	(same as base)
LoRA rank (r)	16
LoRA alpha	32
LoRA dropout	0.05
Target modules	q/k/v/o_proj, gate/up/down_proj
Max sequence length	512
Batch size	64
Learning rate	10^{-4}
Optimizer	AdamW (weight decay 0.01)
Max epochs	20
Early stopping patience	5 epochs
Validation split	10%
Min question length	20 characters
Precision	bfloat16

Table 11: Epiplexity computation hyperparameters.

Epiplexity per token is computed as $(\mathcal{L}_{\text{online}} - \mathcal{L}_{\text{train}})/\ln 2$, where $\mathcal{L}_{\text{online}}$ is the total cross-entropy loss from the prequential (epoch 1) pass and $\mathcal{L}_{\text{train}}$ is the total loss at the best MDL epoch. The MDL criterion selects the epoch that minimizes epiplexity$/N_{\text{train}} + \mathcal{L}_{\text{val}}/(\ln 2 \cdot N_{\text{val}})$.

F Prompt templates

F.1 Original R-Zero questioner prompt

The following is the out-of-the-box R-Zero questioner prompt, which targets various fields of mathematics at competition level. All results in Section 6 use this prompt.

R-Zero Questioner (System Message)

```
You are an expert competition-math problem setter.
FIRST, in your private scratch-pad, think step-by-step to design a brand-new,
non-trivial problem. The problem could come from any field of mathematics, including
but not limited to algebra, geometry, number theory, combinatorics, prealgebra,
probability, statistics, and calculus. Aim for a difficulty such that fewer than 30%
of advanced high-school students could solve it. Avoid re-using textbook clichés or
famous contest problems.

THEN, without revealing any of your private thoughts, output exactly the following
two blocks:

<question>
{The full problem statement on one or more lines}
</question>

\boxed{final_answer}

Do NOT output anything else: no explanations, no extra markup.
```

R-Zero Questioner (User Message)

```
Generate one new, challenging reasoning question now. Remember to format the output
exactly as instructed.
```

F.2 Evaluation prompt

All benchmarks are evaluated zero-shot using the following user message, with no system prompt:

Evaluation (User Message)

```
{question}
Let's think step by step and provide your final answer inside \boxed{} notation.
```

The question text is passed verbatim from each dataset's problem field. The prompt is wrapped in the model's native chat template (e.g., `<|im_start|>user ... <|im_end|>` for Qwen3 models). No system message is set.