

# SPICE : Self-Play In Corpus Environments Improves Reasoning

Bo Liu<sup>1,2,\*</sup>, Chuanyang Jin<sup>1,†</sup>, Seungone Kim<sup>1,‡</sup>, Weizhe Yuan<sup>1</sup>, Wenting Zhao<sup>1</sup>, Ilia Kulikov<sup>1</sup>, Xian Li<sup>1</sup>, Sainbayar Sukhbaatar<sup>1</sup>, Jack Lanchantin<sup>1,∇</sup>, Jason Weston<sup>1,∇</sup>

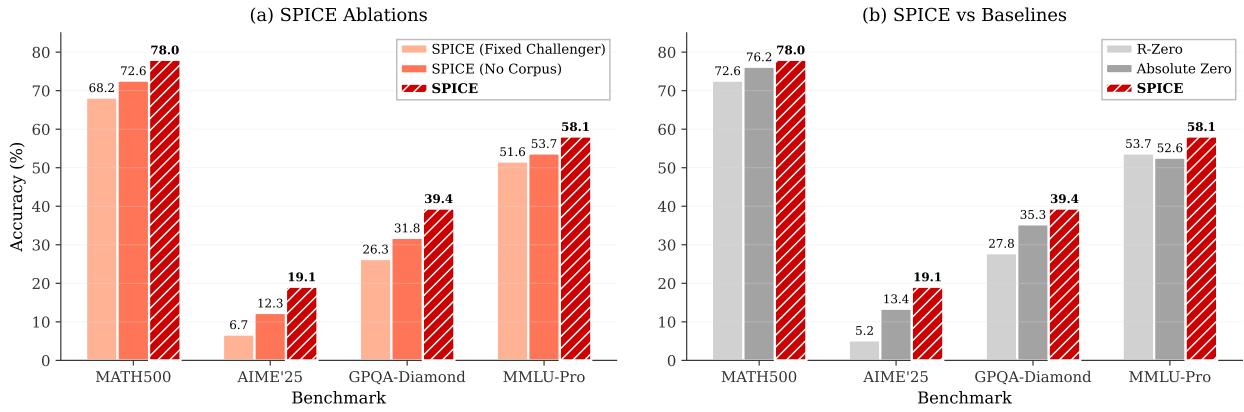
<sup>1</sup>FAIR at Meta, <sup>2</sup>National University of Singapore

\*Work done at Meta, <sup>†</sup>Joint second author, <sup>∇</sup>Joint last author

Self-improving systems require environmental interaction for continuous adaptation. We introduce SPICE (Self-Play In Corpus Environments), a reinforcement learning framework where a single model acts in two roles: a **Challenger** that mines documents from a large corpus to generate diverse reasoning tasks, and a **Reasoner** that solves them. Through adversarial dynamics, the **Challenger** creates an automatic curriculum at the frontier of the **Reasoner**'s capability, while corpus grounding provides the rich, near-inexhaustible external signal necessary for sustained improvement. Unlike existing ungrounded self-play methods that offer more limited benefits, SPICE achieves consistent gains across mathematical (+8.9%) and general reasoning (+9.8%) benchmarks on multiple model families. Our analysis reveals how document grounding is a key ingredient in SPICE to continuously generate its own increasingly challenging goals and achieve them, enabling sustained self-improvement.

**Date:** October 29, 2025

**Correspondence:** [benjaminliu.eecs@gmail.com](mailto:benjaminliu.eecs@gmail.com), [{jacklanchantin, jase}@meta.com](mailto:{jacklanchantin, jase}@meta.com)



**Figure 1** SPICE (Self-Play In Corpus Environments) outperforms state-of-the-art self-play methods for LLMs on Qwen3-4B-Base (right). Training the model in SPICE to be both a **Challenger** and **Reasoner**, creating and solving challenging *corpus-grounded* tasks for itself via self-play RL is crucial to this success (left), see ablations for details.

## 1 Introduction

Self-improving artificial intelligence (Schmidhuber, 2007; Clune, 2019) has long been envisioned as a path toward artificial general intelligence, where systems autonomously enhance their capabilities through environmental interaction and continuous adaptation. This vision is becoming tangible with large language models (LLMs; Kaplan et al. (2020); Achiam et al. (2023); Dubey et al. (2024)), which demonstrate remarkable reasoning abilities across diverse domains (Wei et al., 2022; Kojima et al., 2022). Recent breakthroughs show that reinforcement learning can unlock more sophisticated reasoning. Models like OpenAI o1 (OpenAI, 2024) and DeepSeek-R1 (DeepSeek Team, 2024) achieve expert-level performance on mathematical and coding tasks

through reinforcement learning with verifiable rewards (RLVR). To scale these capabilities without human supervision, self-play offers a promising paradigm (Silver et al., 2017; Sukhbaatar et al., 2017), where models improve by competing against themselves and generating automatic feedback through competition.

However, most existing self-play methods for language models achieve initial improvements but quickly face fundamental barriers. Without external grounding, models inevitably plateau or collapse (Huang et al., 2025; Chen et al., 2025b; Kuba et al., 2025) due to two critical issues: (1) hallucination amplification, where factual errors in both generated questions and answers compound as models train on their own unverifiable synthetic data, and (2) information symmetry, where both the problem generator and solver share the same knowledge base, preventing genuine challenge and leading to simpler, more repetitive patterns. Even approaches maintaining diversity through variational synthesis (Liang et al., 2025) ultimately remain bounded by their initial coverage, which is merely a compressed representation of the original pretraining data (Morris et al., 2025). These systematic empirical failures indicate that self-improvement requires interaction with an external source providing diverse, verifiable feedback, rather than closed-loop pure introspection.

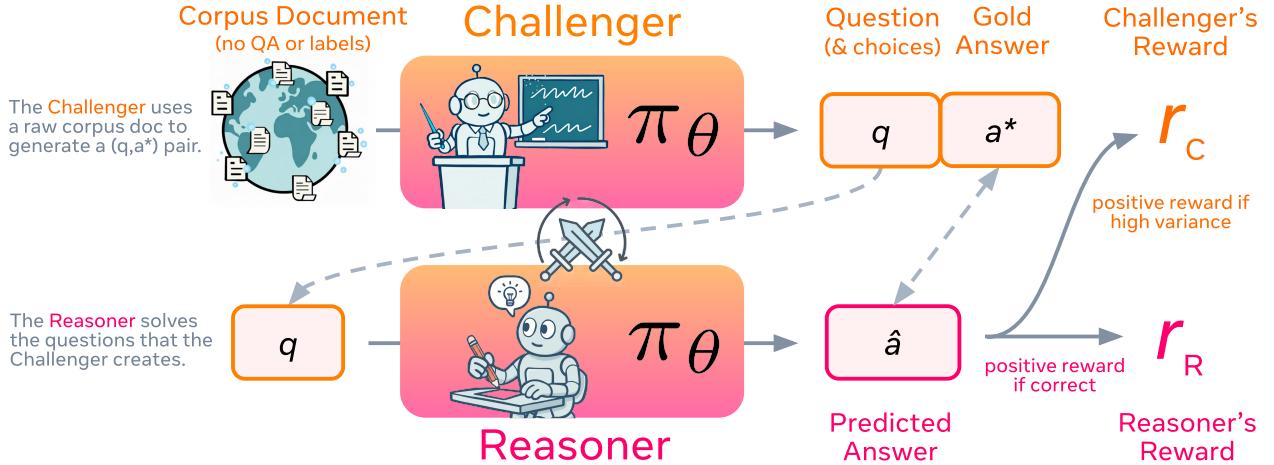
We introduce **SPICE** (Self-Play In Corpus Environments), a self-play reinforcement learning framework that mines contexts from a large document corpus to generate diverse reasoning tasks with document-grounded answers. A single model acts in two roles: a **Challenger** that constructs a curriculum of such challenging document-grounded tasks, and a **Reasoner** that develops robust reasoning capabilities by solving the tasks without document access. A key component is information asymmetry: the **Challenger** grounds questions and gold answers in retrieved documents unseen by the **Reasoner**, creating genuine challenge. The vast diversity of documents ensures continual novelty beyond the model’s internalized knowledge. Simultaneously, corpus grounding prevents hallucination by anchoring both questions and gold answers in real-world content rather than model-generated fantasies, ensuring factual accuracy throughout the self-play loop.

During self-play, the **Challenger** is rewarded for generating problems at the frontier of the **Reasoner**’s capability (maximizing variance in success rates), while the **Reasoner** is rewarded for correct answers. This adversarial yet symbiotic interaction with corpus grounding enables the system to continuously discover new challenges grounded in real knowledge and overcome them. Importantly, our approach relies on raw documents without predefined questions or labels. Tasks are generated with diverse formats (multiple-choice questions and free-form questions with integer/expression/string answers) which serve as universal verifiers, enabling self-play across any language domain without requiring specialized executors or rule-based validators. This breaks the verification bottleneck that has confined prior work to narrow domains like mathematics and code, while document-grounded answers ensure verification remains factually anchored.

Training with corpus-grounded self-play produces reasoning capabilities that transfer broadly across diverse model choices. On Qwen3-4B-Base, SPICE achieves 44.9% average performance versus 35.8% baseline performance (+9.1% absolute gain), while Qwen3-8B-Base improves from 43.0% to 48.7% (+5.7%). OctoThinker-3B-Hybrid-Base shows the largest gains from 14.7% to 25.2% (+10.5%), and OctoThinker-8B-Hybrid-Base improves from 20.5% to 32.4% (+11.9%), where SPICE surpasses both standard RLVR and pure (ungrounded) self-play baselines in all cases. These gains span both mathematical reasoning (average +8.9%) and general reasoning tasks (+9.8% across MMLU-Pro, GPQA-Diamond, SuperGPQA, and BBEH), demonstrating that corpus grounding develops broadly applicable capabilities. The adversarial dynamics between **Challenger** and **Reasoner** create an automatic curriculum: the fixed **Reasoner**’s pass rate decreases from 55% to 35% as it learns to generate progressively harder problems, while the fixed **Challenger**’s pass rate increases from 55% to 85%, indicating successful co-evolution of both roles. Corpus grounding proves critical for sustained improvement: without it, models show limited improvement and fail to maintain the adaptive challenge necessary for sustained learning. With corpus grounding, the system maintains stable advancement throughout training by continuously mining new document contexts for novel challenges.

Overall, our work makes the following contributions:

1. We show that treating a large document corpus as an external knowledge source enables sustained self-improvement, outperforming ungrounded methods that rely solely on the model’s intrinsic knowledge.
2. We propose corpus-grounded self-play where a model acting in two roles (**Challenger** and **Reasoner**) generates tasks with document-extracted answers and solves them, using diverse task formats that enable verification across all domains without specialized tools.



**Figure 2** SPICE is a self-play framework where a single LLM,  $\pi_\theta$ , acts in two roles: a **Challenger** (role =  $C$ ), which poses difficult questions, and a **Reasoner** (role =  $R$ ), which tries to correctly answer such questions. The Challenger uses a raw document (which does not contain existing questions or labels) from a corpus to generate a  $(q, a^*)$  pair. Depending on the contents of the document, the Challenger can generate either a closed-form multiple choice question or a free-form question with different answer types (integer, float, expression). The Challenger gets rewarded for questions with high variance in Reasoner’s correctness, and the Reasoner gets rewarded for answering questions correctly.

3. We empirically validate that corpus-grounded self-play achieves consistent improvements across mathematical and general reasoning tasks, overcoming the domain-specific limitations of existing approaches.

SPICE presents a paradigm shift in self-improving reasoning methods: from closed-loop self-play that often stagnates due to hallucination drift, to open-ended improvement through interaction with the vast, verifiable knowledge embedded in web document corpora.

## 2 SPICE: Self-Play In Corpus Environments

SPICE is an end-to-end framework featuring a single model that acts in two roles through self-play: a **Challenger** ( $C$ ) and **Reasoner** ( $R$ ). When playing the role of **Challenger**, the model grounds itself in web documents to propose questions that challenge the **Reasoner**. Subsequently, the model switches roles to the **Reasoner** to answer the questions. This iterative process, governed by adversarial dynamics, allows both roles to co-evolve, leading to a progressively more capable model, with the corpus providing the necessary external signal for sustained improvement. The entire framework is self-supervised, requiring no human intervention, only a large unstructured corpus. We overview SPICE in [Figure 2](#), and detail the training method in [Algorithm 1](#). In the following subsections, we outline the different roles during self-play.

### 2.1 Notations and Preliminaries

We formulate corpus-grounded self-play as a game where a single model  $\pi_\theta$  acts in two roles: generating questions from documents (role =  $C$ ) or answering questions (role =  $R$ ). Let  $\mathcal{D}$  denote a corpus of documents,  $\mathcal{Q}$  the space of questions, and  $\mathcal{A}$  the space of answers. When role =  $C$ , the model has access to document  $d \sim \mathcal{D}$ ; when role =  $R$ , it only sees question  $q \in \mathcal{Q}$ , creating information asymmetry between the roles.

### 2.2 Challenger: Document-Grounded Task Generation

When acting as **Challenger** ( $C$ ), the model  $\pi_\theta$  learns to generate problems that maximally challenge the **Reasoner** while remaining solvable. Given a corpus  $\mathcal{D}$ , the model in role =  $C$  produces diverse tasks as follows.

**Document Sampling.** We uniformly sample passages from a large document corpus, extracting segments as documents  $d \in \mathcal{D}$ . Each document provides context for generating questions with verifiable answers extracted directly from the text.

**Multi-Format Task Generation.** Given document  $d$ , the **Challenger** takes multiple attempts to create valid question-answer pairs. The model generates question  $q$  and extracts gold answer  $a^*$  directly from the document:

$$(q, a^*) \sim \pi_\theta(\cdot | d, \text{role} = C)$$

The **Challenger** selects between two formats based on document evaluation (see Appendix E): (i) multiple-choice questions (MCQ) with four options and a document-grounded correct answer, or (ii) free-form questions with typed answers (integer, expression, string) extracted from the document. A generated question  $q$  is considered *valid* if it is formatted correctly and parseable from the generation. Our prompt (details given in Appendix E) guides the **Challenger** through multi-step complex information extraction, difficulty enhancement, and self-testing to ensure questions are challenging yet solvable without the source document.

**Variance-Based Curriculum Reward.** For each valid question  $(q, a^*)$ , we sample  $K$  responses  $\hat{a}_i$  from the **Reasoner** and compute the **Challenger**'s reward using a scaled variance of the responses:

$$r_C(q, a^*) = \begin{cases} \exp\left(-\frac{(\text{Var}(\{l_1, \dots, l_K\}) - 0.25)^2}{2 \cdot 0.01}\right) & \text{if } q \text{ is valid} \\ \rho & \text{otherwise (penalty)} \end{cases} \quad (1)$$

where  $l_i = \mathbb{1}[\hat{a}_i = a^*]$ . This Gaussian-shaped reward function is scaled to  $[0, 1]$ , maximizing at 1.0 when variance equals 0.25 (50% pass rate), indicating optimal task difficulty. Tasks that are too easy or too hard receive exponentially lower rewards; as the **Reasoner** improves, the **Challenger** is rewarded for increasing task difficulty, creating an automatic curriculum.

### 2.3 Reasoner: Solving Tasks Without Document Access

When acting as **Reasoner** (R), the model  $\pi_\theta$  learns to solve the **Challenger**'s tasks without document access.

**Answer Generation.** Given valid question  $q$  alone, the model generates answer  $\hat{a} \sim \pi_\theta(\cdot | q, \text{role} = R)$ . The model is prompted to reason step by step and place its final answer within \boxed{\} tags, thus forcing it to rely solely on internalized knowledge.

**Binary Correctness Reward.** The **Reasoner** receives a correctness reward  $r_R(\hat{a}, a^*) = \mathbb{1}[\hat{a} = a^*]$  conditioned on a rule-based verifier that checks answer equivalence with the document-extracted gold answer.

### 2.4 Training with Role-Specific Advantages

We optimize both roles jointly (shared weights) by maximizing expected rewards:

$$J(\theta) = \mathbb{E}_{d \sim \mathcal{D}} [\mathbb{E}_{(q, a^*) \sim \pi_\theta(\cdot | d, \text{role} = C)} [r_C(q, a^*)] + \mathbb{E}_{\hat{a} \sim \pi_\theta(\cdot | q, \text{role} = R)} [r_R(\hat{a}, a^*)]] \quad (2)$$

We use DrGRPO (Liu et al., 2025c) with a separate advantage computation for each role. Given **Challenger** trajectories with variance rewards and **Reasoner** trajectories with correctness rewards, we compute role-specific advantages:

$$\hat{A}_C^i = r_C^i - \text{mean}(\{r_C^j\}_j) \quad (3)$$

$$\hat{A}_R^i = r_R^i - \text{mean}(\{r_R^j\}_j) \quad (4)$$

By centering returns around role-specific expectations without standard deviation normalization, we ensure gradient updates reflect genuine learning signals rather than difficulty-induced noise.

### 2.5 Implementation

To implement SPICE, we develop a self-play reinforcement learning system for finetuning LLMs. Our training framework builds on Oat (Liu et al., 2024), which provides interfaces for a distributed actor-learner architecture (Espeholt et al., 2018). We instantiate actors to execute the self-play loop, using vLLM (Kwon et al., 2023) for efficient model inference during both **Challenger** and **Reasoner** generation phases.

---

**Algorithm 1** SPICE : Self-Play In Corpus Environments

---

**Require:** Pretrained LLM  $\pi_\theta$ ; corpus  $\mathcal{D}$ ; batch size  $B$ ; group size  $G$ ; iterations  $T$ ; penalty  $\rho$

```

1: for  $t \leftarrow 1$  to  $T$  do
2:   Challenger Role: Generate challenging problems with document access
3:   for  $b \leftarrow 1$  to  $B$  do
4:     Sample document  $d \sim \mathcal{D}$ 
5:     Generate multiple attempts:  $\{(q_i, a_i^*)\}_{i=1}^N \leftarrow \pi_\theta(d, \text{role} = C)$             $\triangleright$  MCQ or free-form
6:      $\mathcal{T}_C \leftarrow$  Subsample  $G$  trajectories preserving valid:invalid ratio
7:     if  $q_i \in \mathcal{T}_C$  is valid then
8:        $\{\hat{a}_k\}_{k=1}^G \leftarrow \pi_\theta(q_i, \text{role} = R)$                                       $\triangleright$  No document access
9:        $r_C(q_i, a_i^*) \leftarrow$  Compute reward using Eq. (1)                                 $\triangleright$  Gaussian variance reward
10:    else
11:       $r_C(q_i, a_i^*) \leftarrow \rho$                                                $\triangleright$  Invalid task penalty
12:    end if
13:   end for
14:   Reasoner Role: Solve generated problems without document access
15:   Select a random valid task  $(q, a^*)$  from Challenger phase
16:    $\mathcal{T}_R \leftarrow \{\hat{a}_i\}_{i=1}^G \sim \pi_\theta(q, \text{role} = R)$                        $\triangleright G$  responses for training
17:   for  $i \leftarrow 1$  to  $G$  do
18:      $r_R(\hat{a}_i, a_i^*) \leftarrow \mathbf{1}[\hat{a}_i = a^*]$                                       $\triangleright$  Binary correctness reward
19:   end for
20:   Update Phase: Optimize  $\pi_\theta$  with role-specific advantages
21:   for trajectory  $i$  in  $\mathcal{T}_C$  do
22:      $\hat{A}_C^i \leftarrow r_C^i - \text{mean}(\{r_C^j : j \in \mathcal{T}_C\})$                           $\triangleright$  DrGRPO
23:   end for
24:   for trajectory  $i$  in  $\mathcal{T}_R$  do
25:      $\hat{A}_R^i \leftarrow r_R^i - \text{mean}(\{r_R^j : j \in \mathcal{T}_R\})$                           $\triangleright$  DrGRPO
26:   end for
27:   Update  $\pi_\theta$  via policy gradient with advantages  $\{\hat{A}_C^i\}$  and  $\{\hat{A}_R^i\}$ 
28: end for
29: return Trained model  $\pi_\theta$ 

```

---

The actors generate experiences by alternating between roles: first generating questions with document access (as **Challenger**), then answering them without document access (as **Reasoner**). The resulting trajectories are sent to the collocated learner, which updates the LLM via DrGRPO with role-specific advantages. The responses sampled from the **Reasoner** serve dual purposes: computing variance for the **Challenger**'s reward and forming the trajectory group for the **Reasoner**'s update. We implement answer verification using Math-Verify<sup>1</sup>, which handles equivalence checking for mathematical expressions and exact matching for other answer types. The complete training procedure is detailed in Algorithm 1.

### 3 Experimental Results

#### 3.1 Setup

**Training Configuration.** Following Algorithm 1, we train on corpus  $\mathcal{D}$  with  $|\mathcal{D}| = 20,000$  documents from high-quality, freely available sources. For mathematics, we use Nemotron-CC-Math (Mahabadi et al., 2025); for general reasoning, we use documents from NaturalReasoning (Yuan et al., 2025), a subset of DCLM (Li et al., 2024). We extract document segments of up to 5,992 tokens to fit within the model's context window. Each iteration attempts to generate at least one valid task,  $(q, a^*)$ , per document with up to  $N = 1024$  attempts. While multiple valid questions may be generated, we randomly select one per document to balance the training data size between **Challenger** and **Reasoner** roles. We use temperature 1.0 for both roles, with

<sup>1</sup><https://github.com/huggingface/Math-Verify>

group size  $G = 8$  responses per question serving dual purposes: computing variance for the **Challenger**'s reward and forming the group for DrGRPO advantage computation. The penalty for invalid questions is set to  $\rho = -0.1$ . We train for a fixed  $T = 640$  iterations with batch size  $B = 128$ . Detailed training configurations for all methods are provided in Appendix B.

**Models and Baselines.** We evaluate SPICE on four base models: Qwen3-4B-Base, Qwen3-8B-Base (Yang et al., 2025a), OctoThinker-3B-Hybrid-Base, and OctoThinker-8B-Hybrid-Base (Wang et al., 2025). We compare against several baselines: (1) **Base Model**, the pretrained model without any post-training, establishing the starting performance; (2) **Strong Challenger**, which uses a fixed stronger model (Qwen3-32B-Instruct) as the **Challenger** to generate questions with our model training only as **Reasoner**, testing whether a stronger question generator improves learning; (3) **R-Zero** (Huang et al., 2025), pure self-play without corpus grounding where the model generates its own questions from scratch without document access, representing ungrounded self-play; and (4) **Absolute Zero** (Zhao et al., 2025), self-play restricted to code generation tasks with Python execution as verification, representing domain-specific grounded self-play. We run all baseline methods ourselves using their publicly available code to ensure fair comparison with identical training infrastructure, unified evaluation protocols, and model-specific prompt templates detailed in Appendix E, with training configurations provided in Appendix B.

**Evaluation Benchmarks.** We conduct a comprehensive evaluation across mathematical and general reasoning tasks. For mathematical reasoning, we evaluate on MATH-500 (Hendrycks et al., 2021), a curated subset of challenging competition problems; OlympiadBench (He et al., 2024), featuring olympiad-level problems; Minerva Math (Lewkowycz et al., 2022), covering STEM topics; GSM8K (Cobbe et al., 2021), grade-school word problems; and AMC (MAA, b), AIME'24, AIME'25 (MAA, a) competition problems from the American Mathematics Competitions. For general reasoning, we use SuperGPQA (Du et al., 2025), a large-scale benchmark targeting graduate-level reasoning across 285 disciplines with Google-search-resistant questions; GPQA-Diamond (Rein et al., 2023), graduate-level questions resistant to pattern-matching; MMLU-Pro (Wang et al., 2024), a challenging multi-task understanding dataset; and BBEH (Kazemi et al., 2025), extending BIG-Bench Hard with more complex reasoning tasks. We use the simple-evals<sup>2</sup> framework with GPT-4o (OpenAI, 2024) for answer equivalence checking. Most evaluations use greedy decoding with training-consistent prompts following Ma et al. (2025), except AIME'24 and AIME'25 which are averaged over 32 sampling runs following Zeng et al. (2025). Detailed evaluation settings are provided in Appendix A.

## 3.2 Quantitative Analysis

Table 1 shows the main results on the four different base models we use. SPICE consistently outperforms the baselines across all four model families, delivering the largest overall improvements versus the base models: +9.1 (Qwen3-4B-Base), +5.7 (Qwen3-8B-Base), +10.5 (OctoThinker-3B-Hybrid-Base), and +11.9 (OctoThinker-8B-Hybrid-Base). We observe improvements across *both* math and general reasoning tasks, highlighting the benefits of using a diverse corpus of documents.

**Adversarial Learning Dynamics.** Figure 3 illustrates the co-evolution of **Challenger** and **Reasoner** capabilities. To analyze the contribution of each role, we take checkpoints from full self-play training (steps 200-640) and evaluate them against a fixed step-200 checkpoint on a pool of 128 documents (with 128 generation attempts each). When evaluating different **Challenger** checkpoints against a fixed step-200 **Reasoner** (subplot a), we observe the **Reasoner**'s pass rate decreases from 55% to 35% as later **Challenger** checkpoints generate increasingly challenging questions. Conversely, when evaluating different **Reasoner** checkpoints against a fixed step-200 **Challenger** (subplot b), the **Reasoner**'s pass rate increases from 55% to 85% as later checkpoints become better at solving questions. In full SPICE training where both roles evolve simultaneously, this adversarial dynamic drives mutual improvement beyond what either role achieves in isolation.

**Challenger Learning Impact.** An important component of SPICE is learning the **Challenger** simultaneously with the **Reasoner**, as opposed to using a fixed **Challenger**. Figure 4(a) demonstrates that co-training the **Challenger** alongside the **Reasoner** is essential for maximizing gains, validating our co-evolutionary approach. Without training the **Challenger**, the **Reasoner** isn't challenged enough and improves more slowly.

**Corpus Grounding Impact.** Finally, we examine the critical role of corpus grounding in SPICE. Figure 4(b)

---

<sup>2</sup><https://github.com/openai/simple-evals>

**Table 1** Comprehensive evaluation across mathematical and general reasoning benchmarks comparing SPICE against state-of-the-art self-play methods for LLMs. All methods use self-play except “Strong Challenger”, which uses the fixed Qwen-32B-Instruct model for question generation. Best results per base model are in **bold**. SPICE consistently outperforms all baselines across all four model types.

Method	Overall	Mathematical Reasoning							General Reasoning					$\Delta$ vs Base
		MATH			AIME		AIME		Super-	GPQA-	MMLU-	BBEH		
		AMC	Minerva	500	GSM8K	Olymp.	24	25	GPQA	Diamond	Pro			
<i>Qwen3-4B-Base</i>														
Base Model	35.8	47.5	42.3	68.2	72.6	34.8	10.3	6.7	25.4	26.3	51.6	8.1	–	
+ Strong Challenger	43.0	57.5	44.9	78.0	91.6	41.5	12.7	12.9	27.4	37.9	56.1	12.3	+7.2	
+ R-Zero	39.5	45.0	52.2	72.6	90.5	39.7	10.1	5.2	27.8	27.8	53.7	10.4	+3.7	
+ Absolute Zero	40.7	50.0	41.9	76.2	89.3	41.5	12.2	13.4	27.1	35.3	52.6	8.3	+4.9	
<b>+ SPICE (ours)</b>	<b>44.9</b>	<b>57.5</b>	<b>51.9</b>	<b>78.0</b>	<b>92.7</b>	<b>42.7</b>	<b>12.2</b>	<b>19.1</b>	<b>30.2</b>	<b>39.4</b>	<b>58.1</b>	<b>12.3</b>	<b>+9.1</b>	
<i>Qwen3-8B-Base</i>														
Base Model	43.0	57.5	49.3	74.4	91.2	40.4	15.3	12.1	31.0	33.3	58.1	10.5	–	
+ Strong Challenger	45.6	60.0	52.5	78.2	92.4	41.6	15.3	12.1	35.0	35.8	64.8	14.4	+2.6	
+ R-Zero	46.3	70.0	59.2	78.0	91.8	41.3	11.7	14.2	32.8	36.4	61.7	12.0	+3.3	
+ Absolute Zero	46.5	62.5	52.9	76.6	92.0	47.8	18.4	18.2	33.5	36.8	62.5	10.8	+3.5	
<b>+ SPICE (ours)</b>	<b>48.7</b>	<b>70.0</b>	<b>59.2</b>	<b>79.4</b>	<b>92.7</b>	<b>42.5</b>	<b>18.4</b>	<b>18.2</b>	<b>35.7</b>	<b>39.4</b>	<b>65.0</b>	<b>14.9</b>	<b>+5.7</b>	
<i>OctoThinker-3B-Hybrid-Base</i>														
Base Model	14.7	15.0	14.3	38.2	44.3	10.8	3.4	3.5	12.8	3.5	14.9	1.0	–	
+ Strong Challenger	21.0	15.0	15.1	39.7	73.7	14.4	0.3	0.1	15.7	25.3	24.2	7.0	+6.3	
+ R-Zero	20.3	20.4	22.1	48.0	74.5	15.4	0.2	0.4	12.6	6.6	18.7	4.0	+5.6	
+ Absolute Zero	21.7	17.5	18.7	43.2	75.8	13.2	2.7	0.5	17.7	19.2	24.3	6.1	+7.0	
<b>+ SPICE (ours)</b>	<b>25.2</b>	<b>22.5</b>	<b>22.1</b>	<b>48.0</b>	<b>78.8</b>	<b>15.4</b>	<b>3.4</b>	<b>3.5</b>	<b>18.2</b>	<b>26.8</b>	<b>28.9</b>	<b>9.3</b>	<b>+10.5</b>	
<i>OctoThinker-8B-Hybrid-Base</i>														
Base Model	20.5	27.5	22.1	44.2	68.6	16.7	2.7	0.8	11.4	15.7	14.7	0.6	–	
+ Strong Challenger	28.2	25.0	27.2	54.4	86.4	20.0	6.6	3.3	17.6	27.8	32.9	9.5	+7.7	
+ R-Zero	29.9	32.5	33.1	58.4	85.2	22.6	9.9	1.9	17.9	21.7	37.4	7.8	+9.4	
+ Absolute Zero	29.4	32.5	34.9	56.8	87.0	25.6	0.1	3.3	18.8	27.8	31.4	5.0	+8.9	
<b>+ SPICE (ours)</b>	<b>32.4</b>	<b>35.0</b>	<b>34.9</b>	<b>58.4</b>	<b>87.3</b>	<b>25.6</b>	<b>9.9</b>	<b>7.4</b>	<b>18.8</b>	<b>31.8</b>	<b>37.4</b>	<b>9.5</b>	<b>+11.9</b>	

illustrates the results. With corpus grounding, performance reaches 43.9% through continuous access to diverse document contexts that provide near-inexhaustible question material. Without access to external documents, the model performance is lower at 40.7%, indicating the importance of corpus grounding.

### 3.3 Qualitative Analysis

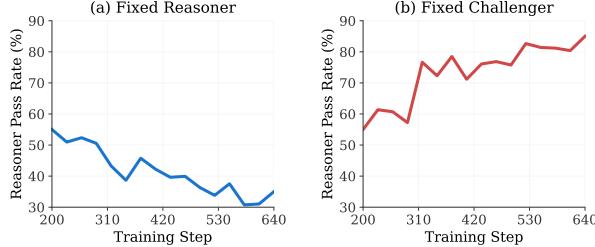
To better understand SPICE’s learning dynamics, we analyze the evolution of tasks and reasoning patterns produced by the **Challenger** and **Reasoner**, respectively, throughout training.

**Challenger Task Development.** Figure 5 shows how task complexity evolves when the **Challenger** is given the same document at different training steps. Early tasks focus on surface-level information, while later tasks require deep comprehension and multi-step reasoning. At earlier steps, the **Challenger** cannot propose tasks that are too difficult, as the **Reasoner** will not be able to learn from them. As training progresses, the **Challenger** learns to generate more difficult tasks to adapt to the improved **Reasoner**’s capabilities.

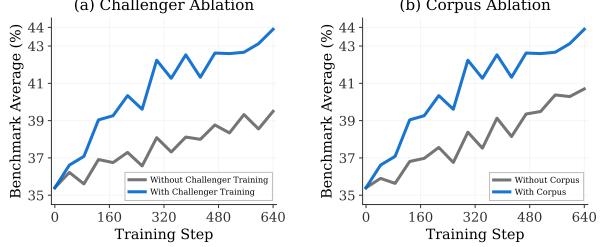
**Reasoner Pattern Development.** Figure 6 illustrates the evolution of the **Reasoner**’s structured problem-solving approach given the same question at different training steps. As training progresses, increasingly sophisticated reasoning patterns emerge, reflecting the **Reasoner**’s adaptation to the progressively more challenging tasks generated by the **Challenger**. This progression reveals authentic reasoning capabilities rather than mere memorization, as the model learns to systematically decompose problems, validate intermediate steps, and self-correct when inconsistencies arise.

## 4 Ablations

To understand the key components driving SPICE’s performance, we conduct comprehensive ablation studies on Qwen3-4B-Base. We examine three critical design choices: corpus composition, task type distribution, and



**Figure 3** Reasoner pass rates when evaluating SPICE checkpoints at steps 200-640 against a fixed step-200 checkpoint on 128 documents. **(a) Fixed Reasoner:** Pass rate decreases from 55% to 35% as later Challenger checkpoints generate harder questions. **(b) Fixed Challenger:** Pass rate increases from 55% to 85% as later Reasoner checkpoints improve at solving questions.



**Figure 4** Training dynamics comparing key components of SPICE. **(a) Challenger Ablation:** Without Challenger training (fixed Challenger), performance improves less than with full adversarial training. **(b) Corpus Ablation:** With corpus grounding, the model achieves steady improvement to 43.9%, while without it performance is lower at 40.7%, showing the importance of external grounding.

Challenger reward strategies. All experiments use the same training configuration as our main experiments.

#### 4.1 Corpus Distribution

One of the main ingredients of SPICE is the external corpora of documents. Compared to no corpora (pure ungrounded self-play), SPICE achieves substantial gains on both math and general reasoning tasks. However, we include two large document datasets: NaturalReasoning and Nemotron-CC-Math. In order to understand the effects of the document types, we ablate each document dataset separately. We show the results in Table 2. As expected, the NaturalReasoning dataset leads to the largest gains in general reasoning tasks as it contains documents targeted toward such tasks. Similarly, Nemotron-CC-Math leads to the biggest improvements in the math tasks. However, both datasets combined lead to the best overall performance.

**Table 2** Effect of corpus composition, used by the Challenger, on reasoning improvements. We show average performance across mathematical reasoning benchmarks (Math), general reasoning benchmarks (General), and their overall average. Best results shown in **bold**, second best in underline. Full benchmark results are given in Appendix Table 6.

Corpus Type	Math Avg	General Avg	Overall Avg
NaturalReasoning	44.4	<b>37.0</b>	41.7
Nemotron-CC-Math	<b>53.4</b>	29.8	43.2
NaturalReasoning + Nemotron-CC-Math	<u>50.6</u>	<u>35.0</u>	<b>44.9</b>

#### 4.2 Task Type

A second key feature of SPICE is the ability to generate different task types dependent on the current document. SPICE uses two different task types: multiple-choice (MCQ) and free-form. Table 3 shows the results of only using MCQ or free-form versus using both. While using free-form questions only leads to the highest math gains, combining MCQ and free-form (mixing tasks) achieves the best overall performance. We attribute this to MCQs providing reliable verification while free-form questions encourage flexible reasoning.

**Table 3** Effect of the task type, proposed by the Challenger, on reasoning improvements. MCQ provides structured answer choices while free-form requires open-ended generation. Combining both leads to the best overall performance. Full results are provided in Appendix Table 7.

Task Type	Math Avg	General Avg	Overall Avg
MCQ only	46.9	<b>35.7</b>	42.0
Free-form only	<b>52.5</b>	31.8	43.7
MCQ + Free-form	<u>50.6</u>	<u>35.0</u>	<b>44.9</b>

### Early Training (Step 50): Surface-level Question

**Document:** [Passage about solar system scale] "The diameter of the Sun is 1,391,000 km. The diameter of the Moon is 3,475 km. The distance from Earth to the Sun is 149,600,000 km. For a solar eclipse to occur, the Moon must appear the same angular size as the Sun from Earth. In a scale model, a circle with diameter 2 cm represents Earth. When Earth is scaled to 0.3 mm (like a period), the entire solar system model changes dramatically..."

**Generated MCQ:** What is the diameter of the Moon?

- A) 1,391 km
- B) 3,475 km
- C) 34,750 km
- D) 347,500 km

**Answer:** B

### Late Training (Step 480): Multi-step Reasoning

**Document:** [Passage about solar system scale] "The diameter of the Sun is 1,391,000 km. The diameter of the Moon is 3,475 km. The distance from Earth to the Sun is 149,600,000 km. For a solar eclipse to occur, the Moon must appear the same angular size as the Sun from Earth. In a scale model, a circle with diameter 2 cm represents Earth. When Earth is scaled to 0.3 mm (like a period), the entire solar system model changes dramatically..."

**Generated MCQ:** An alien civilization has a moon with diameter 3,475 km that creates perfect solar eclipses. Their star has the same diameter as our Sun. If their moon orbits at 374,000 km (maintaining the same angular size ratio as Earth's eclipse geometry), approximately what would be the distance from their planet to their star in kilometers?

- A) 374,000,000 km
- B) 149,600,000 km
- C) 1,391,000,000 km
- D) 74,800,000 km

**Answer:** B

**Figure 5** Evolution of the **Challenger**'s task complexity on the same document. The **Challenger** progresses from extracting explicit facts to generating questions requiring understanding of angular size relationships and proportional reasoning to arrive at document-stated values.

### 4.3 Challenger's Reward

Lastly, we find that the **Challenger**'s reward function critically influences task difficulty calibration. We compare four reward strategies that shape how a **Challenger** can select problems during self-play with a **Reasoner** (see Appendix D, Figure 7 for reward visualizations). Absolute Zero rewards the **Challenger** with one minus the **Reasoner**'s average success rate: tasks where the **Reasoner** fails more often receive higher reward. While intuitive, this task conflates difficulty with learning value. A threshold reward over number of correct rollouts uses a binary signal: reward 1 for relatively solvable tasks, 0 for tasks with 0% or 100% pass rate (impossible or trivial). The R-Zero reward explicitly targets maximum uncertainty: it rewards tasks where the **Reasoner**'s multiple responses split evenly between different answers, peaking when exactly half agree with the most common answer. Our variance approach uses a related principle, but captures the full spread of the answer distribution across multiple **Reasoner** samples rather than measuring agreement with a single mode, creating a curriculum calibrated at the frontier of the model's capability. Results, given in Table 4, show SPICE's variance-based reward achieves the best performance across all metrics. A critical insight is that optimal learning occurs when the **Reasoner** has a balanced success rate.

**Table 4** Effect of the **Challenger** reward strategies on reasoning performance. Each method targets different aspects of task difficulty for curriculum generation. Our Variance reward outperforms previously proposed rewards.

Challenger Reward	Math	General	Overall
Absolute Zero	48.2	30.8	40.7
Threshold	48.6	31.6	41.4
R-Zero	50.0	33.9	43.6
Variance (SPICE)	<b>50.6</b>	<b>35.0</b>	<b>44.9</b>

## Early Training: Unstructured Reasoning

**Question:** An alien civilization has a moon with diameter 3,475 km that creates perfect solar eclipses. Their star has the same diameter as our Sun. If their moon orbits at 374,000 km (maintaining the same angular size ratio as Earth's eclipse geometry), approximately what would be the distance from their planet to their star in kilometers?  
A) 374,000,000 km B) 149,600,000 km C) 1,391,000,000 km D) 74,800,000 km

**Reasoner Output:** "Perfect eclipses mean moon and star look same size. Moon is at 374,000 km. Stars are way farther than moons, so maybe 1000 times farther? That gives 374,000,000 km. The answer is A)374,000,000km."

## Late Training: Structured Multi-step Reasoning

**Question:** An alien civilization has a moon with diameter 3,475 km that creates perfect solar eclipses. Their star has the same diameter as our Sun. If their moon orbits at 374,000 km (maintaining the same angular size ratio as Earth's eclipse geometry), approximately what would be the distance from their planet to their star in kilometers?  
A) 374,000,000 km B) 149,600,000 km C) 1,391,000,000 km D) 74,800,000 km

**Reasoner Output:** "Let me analyze this step-by-step.

## Step 1: Identify given information

- Alien moon diameter = 3,475 km
- Alien star diameter = same as Sun = 1,391,000 km
- Moon orbital distance = 374,000 km
- Need: star distance for perfect eclipses

## Step 2: Understand perfect eclipse condition

For perfect eclipses, angular sizes must be equal:  
 $\frac{\text{Moon diameter}}{\text{Moon distance}} = \frac{\text{Star diameter}}{\text{Star distance}}$

## Step 3: Set up the equation  
$$\frac{3,475}{374,000} = \frac{1,391,000}{d}$$

## Step 4: Solve for star distance d

Cross multiply:

$$3,475 \times d = 1,391,000 \times 374,000$$
$$d = \frac{1,391,000 \times 374,000}{3,475}$$

## Step 5: Calculate

$$d = \frac{520,234,000,000}{3,475} = 149,708,489 \text{ km}$$

## Step 6: Match to closest option  
 $149,708,489 \text{ km} \approx 149,600,000 \text{ km}$

## Step 7: Verify

$$\text{Moon angular size: } \frac{3,475}{374,000} = 0.00929$$

$$\text{Star angular size: } \frac{1,391,000}{149,600,000} = 0.00930$$

These match (small rounding difference), confirming our solution.

The answer is B)149,600,000km."

**Figure 6** Evolution of Reasoner's reasoning patterns. The Reasoner progresses from intuitive guessing to systematic application of angular size principles, proper ratio setup, algebraic manipulation, and solution verification.

## 5 Related Work

**Reinforcement Learning for LLM Reasoning** Reinforcement learning has evolved from alignment tasks using RLHF (Jaques et al., 2019; Ouyang et al., 2022; Bai et al., 2022) to directly improving reasoning capabilities. Recent models like OpenAI o1 (OpenAI, 2024) and DeepSeek-R1 (DeepSeek Team, 2024) demonstrate that RL with verifiable rewards (RLVR) can unlock chain-of-thought reasoning using rule-based rewards (Lightman et al., 2023; Uesato et al., 2022). However, these approaches depend on human-curated problem sets and domain-specific reward engineering, limiting their scalability. The Era of Experience (Silver and Sutton, 2025) argues that future AI systems must learn from environmental interaction rather than static datasets (Jin et al., 2025). Yet, current RLVR methods remain bound to pre-collected problems. SPICE aims to address this by generating problems dynamically from corpus interactions, enabling learning without human curation.

**Multi-Agent RL for Language Models** Implementing multi-agent RL for full-scale LLMs presents significant technical challenges (Wan et al., 2025; Liu et al., 2025b). Prior work has made various compromises: Sarkar et al. (2025) uses RNNs instead of transformers; Jacob et al. (2022) focuses on simplified environments that don't require full autoregressive generation; and Liao et al. (2024) shows the effectiveness of self-play

with SFT on proprietary models. SPIRAL (Liu et al., 2025a) demonstrates that zero-sum games between agents can teach transferable reasoning through multi-turn interactions, but requires carefully designed game environments. SPICE builds on multi-agent frameworks but grounds the adversarial dynamics in corpus contexts, enabling automatic curriculum emergence without engineered environments.

**Self-Play for Autonomous Improvement** Self-play has revolutionized game-playing AI, including TD-Gammon’s backgammon mastery (Tesauro, 1995), AlphaGo’s superhuman Go performance (Silver et al., 2017), and CICERO’s comprehension of cooperative strategies (FAIR et al., 2022). Self-play can even create automatic curricula through intrinsic motivation (Sukhbaatar et al., 2017). In language models, self-play began with alignment methods like SPIN (Chen et al., 2024) and Self-Rewarding Language Models (Yuan et al., 2024). Recent work explores capability improvement: SPAG (Cheng et al., 2024) applies self-play to Adversarial Taboo but uses offline updates and remains confined to a single word game; SPC (Chen et al., 2025a) and Genius (Xu et al., 2025) require human task distributions as seeds; SPELL (Yang et al., 2025b) targets long-context evolution through self-play; Language Self-Play (Kuba et al., 2025) explores data-free training paradigms. Pure self-play methods consistently demonstrate empirical limitations: R-Zero (Huang et al., 2025) achieves initial gains but degrades after 3-4 iterations with pseudo-label accuracy dropping from 79% to 63%; Absolute Zero (Zhao et al., 2025) and SQLM (Chen et al., 2025b) generate coding tasks and hence remain limited in domain.

**Synthetic Question Generation from Corpora** Synthetic question generation methods either bootstrap from existing datasets or mine from corpora. Bootstrapping approaches like STaR (Zelikman et al., 2022), MetaMath (Yu et al., 2023), and SvS (Liang et al., 2025) remain bounded by initial dataset coverage. Self-Instruct (Wang et al., 2022) and reasoning-based CoT-self-instruct (Yu et al., 2025) generate high-quality synthetic prompts from few-shot examples, but create static datasets rather than adaptive curricula. Corpus-mining methods achieve broader coverage: WebInstruct (Yue et al., 2024) harvests QA pairs but requires rule-based filters; General-Reasoner (Ma et al., 2025) and NaturalReasoning (Yuan et al., 2025) generate millions of questions from web content but create static offline datasets. SPICE differs fundamentally through online adversarial generation where the Challenger continuously mines corpus contexts to generate tasks calibrated to the Reasoner’s current capability, preventing both coverage limitations and quality degradation.

## 6 Conclusion

We introduced SPICE, a self-play RL framework that aims to overcome the issues of hallucination and information symmetry, outperforming pure (ungrounded) self-play. By treating a large document corpus as a near-inexhaustible external environment, SPICE presents a new paradigm for self-improvement, allowing LLMs to interact with a continually changing world via web documents. Through adversarial dynamics between a Challenger that generates tasks and a Reasoner that solves them, our system creates its own ever more challenging goals and strives to achieve them. SPICE achieves strong improvements across mathematical and general reasoning benchmarks compared to state-of-the-art self-play methods. By mining training signals from the vast knowledge embedded in corpora, SPICE develops reasoning capabilities that transfer broadly across domains. We believe this shift from closed-loop self-play to corpus-grounded adversarial learning opens new avenues for self-improvement without explicit human supervision.

## References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- Jiaqi Chen, Bang Zhang, Ruotian Ma, Peisong Wang, Xiaodan Liang, Zhaopeng Tu, Xiaolong Li, and Kwan-Yee K Wong. Spc: Evolving self-play critic via adversarial games for llm reasoning. *arXiv preprint arXiv:2504.19162*, 2025a.
- Lili Chen, Mihi Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025b.
- Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *ICML*, 2024.
- Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Yong Dai, Lei Han, Xiaolong Li, et al. Self-playing adversarial language game enhances llm reasoning. *Advances in Neural Information Processing Systems*, 37:126515–126543, 2024.
- Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.
- 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, 2021. <https://arxiv.org/abs/2110.14168>.
- DeepSeek Team. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2401.00000*, 2024.
- Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.
- FAIR, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.
- 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. *NeurIPS*, 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.
- Athul Paul Jacob, Abhishek Gupta, and Jacob Andreas. Emergent linguistic phenomena in multi-agent communication games. *arXiv preprint arXiv:2205.05984*, 2022.
- Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Chuanyang Jin, Jing Xu, Bo Liu, Leitian Tao, Olga Golovneva, Tianmin Shu, Wenting Zhao, Xian Li, and Jason Weston. The era of real-world human interaction: RL from user conversations. *arXiv preprint arXiv:2509.25137*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. Big-bench extra hard, 2025. <https://arxiv.org/abs/2502.19187>.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Jakub Grudzien Kuba, Mengting Gu, Qi Ma, Yuandong Tian, and Vijai Mohan. Language self-play for data-free training. *arXiv preprint arXiv:2509.07414*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 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.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlrv. *arXiv preprint arXiv:2508.14029*, 2025.

Austen Liao, Nicholas Tomlin, and Dan Klein. Efficacy of language model self-play in non-zero-sum games, 2024. <https://arxiv.org/abs/2406.18872>.

Hunter Lightman, Vineet Kosaraju, Yuri 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.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025a.

Mickel Liu, Liwei Jiang, Yancheng Liang, Simon Shaolei Du, Yejin Choi, Tim Althoff, and Natasha Jaques. Chasing moving targets with online self-play reinforcement learning for safer language models, 2025b. <https://arxiv.org/abs/2506.07468>.

Zichen Liu, Changyu Chen, Xinyi Wan, Chao Du, Wee Sun Lee, and Min Lin. Oat: A research-friendly framework for llm online alignment. <https://github.com/sail-sg/oat>, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

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

MAA. American invitational mathematics examination (AIME). Mathematics Competition Series, n.d.a. <https://maa.org/math-competitions/aime>.

MAA. American mathematics competitions (AMC 10/12). Mathematics Competition Series, n.d.b. <https://maa.org/math-competitions/amc>.

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc-math: A 133 billion-token-scale high quality math pretraining dataset. *arXiv preprint arXiv:2508.15096*, 2025.

John X Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much do language models memorize? *arXiv preprint arXiv:2505.24832*, 2025.

OpenAI. GPT-4o system card, 2024. <https://arxiv.org/abs/2410.21276>.

OpenAI. Learning to reason with llms. *OpenAI Blog*, 2024. <https://openai.com/o1>.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark, 2023. <https://arxiv.org/abs/2311.12022>.

Bidipta Sarkar, Warren Xia, C Karen Liu, and Dorsa Sadigh. Training language models for social deduction with multi-agent reinforcement learning. *arXiv preprint arXiv:2502.06060*, 2025.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial general intelligence*, pages 199–226. Springer, 2007.

David Silver and Richard S. Sutton. Welcome to the era of experience. In *Designing an Intelligence*. MIT Press, 2025. Preprint.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play, 2017. <https://arxiv.org/abs/1703.05407>.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, et al. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

Fangzhi Xu et al. Genius: A generalizable and purely unsupervised self-training framework for advanced reasoning, 2025.

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

Ziyi Yang, Weizhou Shen, Ruijun Chen, Chenliang Li, Fanqi Wan, Ming Yan, Xiaojun Quan, and Fei Huang. Spell: Self-play reinforcement learning for evolving long-context language models. *arXiv preprint arXiv:2509.23863*, 2025b.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengu Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Ping Yu, Jack Lanchantin, Tianlu Wang, Weizhe Yuan, Olga Golovneva, Ilia Kulikov, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. Cot-self-instruct: Building high-quality synthetic prompts for reasoning and non-reasoning tasks. *arXiv preprint arXiv:2507.23751*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Ilia Kulikov, Kyunghyun Cho, Dong Wang, Yuandong Tian, Jason E Weston, et al. Naturalreasoning: Reasoning in the wild with 2.8 m challenging questions. *arXiv preprint arXiv:2502.13124*, 2025.

Xiang Yue, Tianyu Zheng, Ge Zhang, and Wenhui Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. <https://arxiv.org/abs/2503.18892>.

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.

# Appendix

<a href="#">A Evaluation Settings</a>	<b>17</b>
<a href="#">B Training Configuration Details</a>	<b>17</b>
<a href="#">C Additional Results and Analysis</a>	<b>18</b>
<a href="#">C.1 Comprehensive Benchmark Results</a>	18
<a href="#">D Challenger Reward Functions</a>	<b>19</b>
<a href="#">E Prompt Templates</a>	<b>20</b>

## A Evaluation Settings

**Evaluation Protocol.** All models are evaluated in a zero-shot setting to assess whether the reasoning capabilities developed through corpus-grounded self-play transfer to standard benchmarks without task-specific adaptation. We use greedy decoding (temperature 0) for most evaluations to ensure reproducibility, following the evaluation protocols from Ma et al. (2025). For AIME’24 and AIME’25, we follow SimpleRL (Zeng et al., 2025) and report the average accuracy over 32 sampling runs with temperature 0.6 to better assess model performance on these challenging competition problems. For other mathematical reasoning benchmarks, we report pass@1 accuracy with greedy decoding for MATH-500 (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), Minerva Math (Lewkowycz et al., 2022), GSM8K (Cobbe et al., 2021), and AMC, evaluating exact match after answer extraction and normalization. All mathematical evaluations use GPT-4o (OpenAI, 2024) verification through the simple-evals framework, which handles equivalence checking for various mathematical formats including fractions, decimals, and algebraic expressions.

**General Reasoning Metrics.** For general reasoning benchmarks, we evaluate on GPQA-Diamond (Rein et al., 2023), the most challenging subset of graduate-level science questions designed to resist memorization; SuperGPQA (Du et al., 2025), covering 285 disciplines with Google-search-resistant questions; MMLU-Pro (Wang et al., 2024), an enhanced version of MMLU with more rigorous multiple-choice questions requiring deeper understanding; and BBEH (Kazemi et al., 2025), extending BIG-Bench Hard with additional complex reasoning tasks. All general reasoning evaluations use greedy decoding and exact match on the extracted answer choice (A/B/C/D) for multiple-choice questions. We maintain consistent prompting across all models, using the same system prompt and answer extraction format as during training to ensure fair comparison. The evaluation prompts instruct models to provide step-by-step reasoning before producing a final answer in a boxed format for mathematical tasks or selecting a letter choice for multiple-choice questions. All evaluation code and prompts will be released for reproducibility.

## B Training Configuration Details

**Table 5** Training configurations for all compared methods. All experiments use 640 iterations with batch size 128. <sup>†</sup>R-Zero shows performance degradation after 5 iterations, so we report its best performance. \*Absolute Zero applies -0.5 for incorrect but well-formatted responses and -1 for formatting errors.

Configuration	SPICE	Strong Challenger	R-Zero	Absolute Zero
<i>Data Source</i>				
Corpus documents	20,000	20,000	–	–
Question source	Document-grounded	Document-grounded	Self-generated	Self-generated
External grounding	✓	✓	✗	Python executor
<i>Training Details</i>				
Challenger training	✓	✗ (fixed)	✓	✓
Challenger sampling	8	–	4	1
Reasoner training	✓	✓	✓	✓
Reasoner sampling	8	8	5	1
Temperature	1.0	1.0	1.0	1.0
Optimizer	DrGRPO	DrGRPO	GRPO	REINFORCE++
<i>Reward Design</i>				
Challenger reward	Gaussian Variance (max 1.0)	N/A	$1 - 2 p - 0.5 $	$1 - p$
Reasoner reward	Binary correctness	Binary correctness	Binary correctness	Binary correctness
Invalid penalty	-0.1	N/A	-1 (Challenger)	-0.5/-1*
<i>Performance</i>				
Training iterations	640	640	5 <sup>†</sup>	640

**Implementation Infrastructure.** All experiments utilize a distributed actor-learner architecture implemented in Oat (Liu et al., 2024), with vLLM (Kwon et al., 2023) for efficient inference during both question generation and answer sampling phases. We employ Math-Verify for answer equivalence checking, which handles various

mathematical formats including fractions, decimals, and algebraic expressions. Training is conducted on 8 H200 GPUs per experiment with a learning rate of 1e-6 using constant learning rate schedule, gradient checkpointing, flash attention, and ZeRO Stage 2 optimization for memory efficiency. We use DrGRPO without KL regularization ( $\beta = 0$ ) to focus purely on reward optimization. The system processes 128 trajectories per gradient update with rollout batch size of 128 distributed across devices.

**Baseline Implementation Details.** R-Zero uses 5 rollouts for Reasoner training and 4 for Challenger training, with the Challenger computing uncertainty rewards  $r_{\text{uncertainty}} = 1 - 2|p - 0.5|$  that peak when the Reasoner achieves 50% accuracy, maximizing learning potential at the frontier of model capability (Huang et al., 2025). Absolute Zero uses 1 rollout during training but estimates learnability from 8 Reasoner attempts, with rewards  $r_{\text{propose}} = 1 - p$  (for  $0 < p < 1$ ) that linearly decrease with solve rate and assign zero reward to trivial or impossible tasks. Both baselines use binary correctness rewards for their reasoners: R-Zero validates against majority-voted pseudo-labels using Math-Verify, while Absolute Zero checks Python value equality with ground-truth outputs. For invalid responses, R-Zero’s Challenger assigns -1 penalty while its Reasoner uses weighted format+accuracy scoring; Absolute Zero applies -0.5 for incorrect but well-formatted responses and -1 for format errors. Our DrGRPO approach removes standard deviation normalization to avoid difficulty bias, computing  $\hat{A}_i = R_i - \text{mean}(\{R_j\})$  to ensure gradient updates reflect genuine learning signal rather than question difficulty artifacts.

**Corpus Composition.** The 20,000 document corpus combines Nemotron-CC-Math (Mahabadi et al., 2025) (50%) for mathematical content and NaturalReasoning (Yuan et al., 2025) (50%) for general reasoning, providing diverse contexts across STEM, humanities, and social sciences. Documents are sampled uniformly during training, with each document used approximately 2-3 times over 640 iterations. This diversity prevents overfitting to specific domains while ensuring broad coverage of reasoning patterns.

## C Additional Results and Analysis

This appendix provides comprehensive benchmark-level results for all ablation studies presented in Section 4. These detailed breakdowns reveal how different design choices affect specific reasoning capabilities.

### C.1 Comprehensive Benchmark Results

We present complete results across all 11 evaluation benchmarks for each ablation configuration. Tables 6, 7, and 8 show the full performance breakdown, revealing nuanced patterns not visible in the averaged results.

**Table 6** Comprehensive ablation study on corpus composition effects (Qwen3-4B-Base). Best results per column in **bold**, second best underlined.

Corpus Type	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH500	GSM8K	Olympiad	AIME24	AIME25	SuperGPQA	GPQA-Dmd	MMLU-Pro	BBEH	
DCLM (NaturalReasoning)	47.5	46.8	71.6	86.3	37.2	9.8	11.9	<u>31.8</u>	<u>42.4</u>	<u>60.3</u>	<u>13.5</u>	41.7
Math (Nemotron-CC-Math)	<b>62.5</b>	<b>56.8</b>	<b>81.2</b>	91.8	<b>46.3</b>	<b>14.8</b>	<b>22.1</b>	26.2	30.8	52.7	9.6	43.2
DCLM + Math (SPICE)	57.5	51.9	78.0	<b>92.7</b>	<u>42.7</u>	<u>12.2</u>	<u>19.1</u>	30.2	39.4	<u>58.1</u>	<u>12.3</u>	<b>44.9</b>

Several key insights emerge from the detailed corpus ablation (Table 6):

- Mathematical benchmarks strongly favor Nemotron-CC-Math, with gains of up to +10.2 points on AIME25
- General reasoning benchmarks uniformly prefer NaturalReasoning, particularly GPQA-Diamond (+11.9) and BIG-Bench Extra Hard (+3.9)
- The combined approach achieves the highest GSM8K score (92.7%), suggesting synergistic effects for certain problem types

The task type analysis (Table 7) reveals complementary strengths:

- MCQs excel at AMC (61.0%) where answer selection is the natural format

**Table 7** Comprehensive ablation study on response format effects (Qwen3-4B-Base). Best results per column in **bold**, second best underlined.

Response Format	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH500	GSM8K	Olympiad	AIME24	AIME25	SuperGPQA	GPQA-Dmd	MMLU-Pro	BBEH	
MCQ	<b>61.0</b>	46.2	70.8	87.6	37.8	10.4	14.8	<u>30.6</u>	<b>41.8</b>	<b>59.4</b>	12.1	42.0
Free-form	54.5	<b>55.7</b>	<b>82.0</b>	92.1	<b>45.8</b>	<b>15.2</b>	<b>23.6</b>	27.8	34.2	54.3	10.8	43.7
MCQ + Free-form (SPICE)	<b>57.5</b>	<u>51.9</u>	<u>78.0</u>	<b>92.7</b>	<u>42.7</u>	<u>12.2</u>	<u>19.1</u>	<b>30.2</b>	<u>39.4</u>	<u>58.1</u>	<b>12.3</b>	<b>44.9</b>

- Free-form dominates on complex mathematical problems requiring detailed solutions (MATH500: 82.0%, AIME25: 23.6%)
- The mixed approach achieves optimal GSM8K performance, suggesting that task diversity aids in word problem comprehension

The task type analysis (Table 7) reveals complementary strengths:

- MCQs excel at AMC (61.0%) where answer selection is the natural format
- Free-form dominates on complex mathematical problems requiring detailed solutions (MATH500: 82.0%, AIME25: 23.6%)
- The mixed approach achieves optimal GSM8K performance, suggesting that task diversity aids in word problem comprehension

**Table 8** Comprehensive ablation study on challenger reward strategies (Qwen3-4B-Base). Best results per column in **bold**, second best underlined.

Challenger Reward	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH500	GSM8K	Olympiad	AIME24	AIME25	SuperGPQA	GPQA-Dmd	MMLU-Pro	BBEH	
Absolute Zero Reward	50.0	41.9	76.2	89.3	41.5	12.2	13.4	27.1	35.3	52.6	8.3	40.7
Threshold	52.5	<u>43.7</u>	<u>75.8</u>	<u>89.7</u>	40.8	11.8	14.2	27.8	36.2	53.4	9.1	41.4
R-Zero Reward	<b>55.0</b>	<u>50.2</u>	<u>77.4</u>	<u>91.6</u>	<b>41.8</b>	<u>11.9</u>	17.3	<u>29.4</u>	<u>37.8</u>	<u>56.3</u>	<u>11.2</u>	<u>43.6</u>
Variance (SPICE)	<b>57.5</b>	<u>51.9</u>	<u>78.0</u>	<u>92.7</u>	<b>42.7</b>	<u>12.2</u>	<u>19.1</u>	<b>30.2</b>	<u>39.4</u>	<u>58.1</u>	<u>12.3</u>	<b>44.9</b>

The reward strategy comparison (Table 8) demonstrates the superiority of variance-based rewards:

- Consistent improvements across all benchmarks compared to absolute zero baseline
- Largest gains on challenging benchmarks: BIG-Bench Extra Hard (+4.0), AIME25 (+5.7), and MMLU-Pro (+5.5)
- The variance-based approach appears particularly effective for problems requiring multi-step reasoning

These detailed results confirm that SPICE’s design choices work synergistically to produce a well-rounded reasoning system, with each component contributing unique strengths that combine to achieve state-of-the-art performance.

## D Challenger Reward Functions

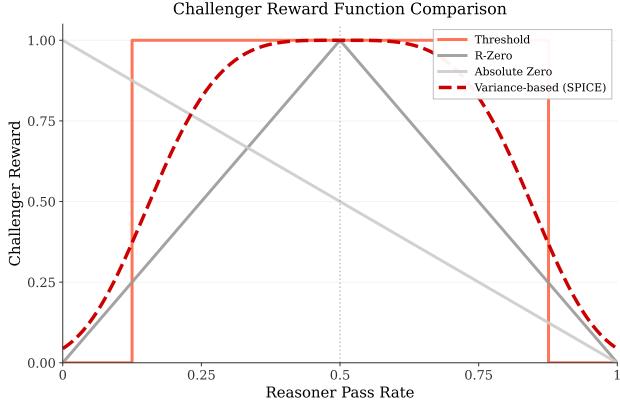
Figure 7 visualizes the four challenger reward strategies compared in our ablation study. Each function represents a different philosophy for task selection during self-play:

The variance-based reward function used by SPICE is formulated as:

$$r_c = \exp\left(-\frac{(\sigma^2 - \sigma_{\text{opt}}^2)^2}{2\tau}\right) \quad (5)$$

where  $\sigma^2 = p(1-p)$  represents the variance of the reasoner’s success rate,  $\sigma_{\text{opt}}^2 = 0.25$  is the optimal variance (achieved at  $p = 0.5$ ), and  $\tau = 0.01$  is a temperature parameter controlling the sharpness of the reward peak.

This formulation ensures that:



**Figure 7** Comparison of challenger reward functions as a function of reasoner pass rate  $p$ . The **threshold reward** (rollout size 8) assigns 0 for trivial (8/8) and impossible (0/8) tasks, 1 otherwise. **Absolute Zero** ( $1 - p$ ) linearly decreases with pass rate. **R-Zero** ( $1 - 2|p - 0.5|$ ) peaks at maximum uncertainty. **SPICE’s variance-based reward** (dashed) uses  $\exp\left(-\frac{(\sigma^2 - \sigma_{\text{opt}}^2)^2}{2 \cdot 0.01}\right)$  where  $\sigma^2 = p(1 - p)$  and  $\sigma_{\text{opt}}^2 = 0.25$ , creating a smooth curriculum centered at the frontier of model capability.

- Maximum reward occurs when the reasoner has a 50% success rate, indicating problems at the edge of its capability.
- The reward smoothly decreases for both easier (high  $p$ ) and harder (low  $p$ ) problems.
- Unlike the threshold function, there are no discontinuities that could destabilize training.
- Unlike R-Zero, the Gaussian form provides better gradient signals near the optimum.

The empirical results in Table 4 demonstrate that this variance-based approach yields superior performance across all benchmarks, validating our hypothesis that optimal learning occurs when training focuses on problems with balanced difficulty.

## E Prompt Templates

**Question Format Selection.** Before generating questions, the **Challenger** evaluates each document to determine the optimal question format using a structured prompt (Figure 9). This evaluation assesses whether the document contains complex relationships suitable for MCQs with plausible distractors, or precise information better suited for free-form questions with deterministic answers. The prompt returns a JSON response specifying the recommended format (MCQ or free-form) and, for free-form questions, the appropriate answer type (integer, expression, or string).

**Question Generation.** Following format selection, the **Challenger** uses specialized prompts to generate challenging questions (Figure 10). The MCQ prompt guides generation through structured steps: identifying complex multi-concept relationships, creating balanced distractors, and self-testing difficulty—ensuring questions require synthesis rather than simple retrieval. The free-form prompt similarly emphasizes multi-step reasoning but focuses on generating questions with unambiguous, extractable answers. Both prompts enforce that questions must be self-contained and solvable without the source document, preventing reference-dependent questions that would be unanswerable for the **Reasoner**.

**Answer Generation.** The **Reasoner** uses model-specific templates optimized for each architecture. Qwen3 models use chat-style formatting with explicit instructions to reason step-by-step before providing boxed answers, while OctoThinker models employ a conversational format emphasizing internal reasoning processes. These templates ensure consistent answer extraction across different model families while maintaining their optimal performance characteristics.

Prompt Templates for Different Model Families

**Qwen3 Family Template:**

```
<|im_start|>system
You are a helpful assistant.
<|im_end|>
<|im_start|>user
{question}

Please reason step by step, and put your final answer within \boxed{}.
<|im_end|>
<|im_start|>assistant
```

**OctoThinker Family Template:**

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.

User: {question}

You must put your answer inside \boxed{}.

Assistant:

**Figure 8** Model-specific prompt templates for the Reasoner role. The Challenger role uses similar formatting with task-specific instructions for question generation from documents. The {question} placeholder is replaced with the actual question during inference.

### Task Type Selection Prompt

Analyze this document and decide whether it's better suited for a CHALLENGING multiple-choice question (MCQ) or a free-form question.

Document:  
{document}

Consider the prompts that will be used:

For MCQ:

- Needs complex relationships and multi-step reasoning paths
- Should allow creating 3 plausible but wrong distractors
- Requires synthesis of multiple concepts
- Can test understanding through carefully crafted wrong answers

For Free-form:

- Best for questions requiring specific calculations (Integer answers)
- Good for deriving formulas or expressions (Expression answers)
- Suitable for conceptual answers requiring precise terminology (String answers)
- Should have a single clear correct answer

Based on the document content, choose EXACTLY ONE type that would produce the highest quality CHALLENGING question.

You MUST respond with ONLY a valid JSON object (no markdown, no explanation before or after):

```
{{
  "suitable_for_mcq": <true or false>,
  "suitable_for_free_form": <true or false>,
  "best_answer_type": <"Integer" or "Expression" or "String" or null>,
  "reason": "<brief explanation without special characters>"
}}
```

**CRITICAL RULES:**

1. Return ONLY the JSON object, no other text
2. Exactly ONE of suitable\_for\_mcq or suitable\_for\_free\_form must be true
3. Do NOT use backticks or markdown formatting
4. Do NOT include LaTeX or special characters in the reason field
5. Keep reason under 100 characters

**Figure 9** Task type selection prompt used by the Challenger to determine optimal question format. The prompt analyzes document content characteristics and returns a structured JSON response specifying whether the document better supports MCQ generation (with plausible distractors) or free-form questions (with deterministic answers). This format selection ensures questions leverage the document's content effectively.

## MCQ Challenger Prompt

Your task is to create CHALLENGING exam questions from a document by identifying complex relationships and multi-step reasoning paths.

## Text [BEGINNING OF THE DOCUMENT] {text} [END OF THE DOCUMENT]

## Instructions

### Step 1: Complex Information Extraction \*\*PRIORITY:\*\* Focus on information that requires synthesis and reasoning\*\*  
Scan the text and identify information that requires connecting multiple concepts:  
\* Relationships between multiple variables or concepts that span different sections  
\* Multi-step calculations or procedures where each step depends on previous ones  
\* Formulas or principles that require understanding interactions between components  
\* Implicit conclusions that can be derived by combining stated facts  
\* Comparative analyses or trade-offs between different approaches  
\* Conditional relationships (if X then Y, but if Z then W)  
\* Systems where changing one parameter affects multiple others

\*\*AVOID\*\*: \* Single, directly stated facts (these create Easy questions)  
\* Simple definitions that stand alone  
\* Values or numbers mentioned in isolation  
\* Information that requires no synthesis

### Step 2: Difficulty Enhancement Process \*\*EXPLICITLY STATE YOUR HARDENING PROCESS\*\* Before generating the question, describe your strategy to make it harder:  
1. What simple version would you avoid?  
2. What complexity layers will you add?  
3. Which concepts will you force students to connect?  
4. What common shortcuts will you block?  
5. How will you ensure multi-step reasoning is required?

Document this in the output field "hardening\_process".

### Step 3: Advanced Question Generation For each complex relationship identified, create a question that:  
\* Requires applying multiple concepts from different parts of the document  
\* Tests understanding of relationships, not just recall of facts  
\* Forces reasoning through multiple steps to reach the answer  
\* May require comparing or contrasting different scenarios  
\* Could involve "what if" scenarios based on principles in the text  
\* Tests ability to apply concepts to slightly modified situations

\*\*CRITICAL - Self-Contained Requirements\*\*: \* Questions must be 100% self-contained and standalone  
\* NEVER use: "according to the text", "in the document", "as mentioned", "the passage states", "based on the analysis", etc.  
\* Write as if for a formal exam with no reference material  
\* Include all necessary context within the question itself  
\* Define any specialized terms if needed for clarity

### Step 4: Difficulty-Driven Design \*\*TARGET\*\*: Generate HARD/EXTRA HARD questions by design\*\*  
\* HARD: Synthesis 4+ concepts; multi-step problem solving; pattern recognition  
\* EXTRA HARD: Complex system analysis; counter-intuitive applications; edge cases

Design questions that CANNOT be answered by:  
\* Looking up a single fact  
\* Finding one sentence with the answer  
\* Simple keyword matching

### Step 5: Knowledge Integration Requirements Document the reasoning path that shows why this is a difficult question:  
\* List 3+ distinct pieces of information needed from different parts  
\* Show the logical connections required between these pieces  
\* Explain why simple lookup won't work  
\* Include intermediate reasoning steps

### Step 6: Multiple Choice Design Guidelines Create a multiple choice question with 4 options following these STRICT rules:

\*\*Length Balance\*\*: All options must be approximately equal length ( $\pm 20\%$  Unit Consistency)  
\*\*Tone Neutrality\*\*: Avoid overly certain language ("definitely", "always", "never") unless justified  
\*\*Plausibility\*\*: All distractors must be genuinely plausible based on partial understanding

Format: Question: [Complete, self-contained question with all necessary context] A) [Balanced length option] B) [Balanced length option] C) [Balanced length option] D) [Balanced length option]  
Correct: [Letter]

\*\*Distractor Design\*\*: \* Common calculation errors from the multi-step process  
\* Results from applying only partial reasoning  
\* Mixing up related concepts from the document  
\* Reasonable approximations that miss key factors

### Step 7: Self-Testing Filter (AFTER MCQ Creation) \*\*SOLVE YOUR OWN MCQ AS A STUDENT WOULD\*\* Now test the complete multiple choice question:  
1. What's the quickest path a student might try with these options?  
2. Can you eliminate 2+ options without full understanding?  
3. Does seeing the options make the answer obvious?  
If yes, improve distractors  
4. Count the reasoning steps required even with options visible - if less than 3, REJECT  
5. Time estimate: Would this MCQ take <30 seconds?  
If yes, make it harder  
6. Could a student guess correctly by pattern matching the options?  
If yes, rebalance

Document your solving process in "self\_test\_solution".

### Step 8: Final Complexity Verification Before finalizing, verify your question is NOT Easy by checking:  
\* Can it be answered by finding one sentence? If yes, redesign  
\* Does it require connecting multiple document sections? If no, add complexity  
\* Would someone need to understand relationships, not just facts? If no, refocus  
\* Are all MCQ options balanced and using consistent formatting?  
If no, revise  
\* Did your self-test of the MCQ take more than 1 minute? If no, increase difficulty

## Output Format FIRST, think step-by-step about your question design (this is your private thinking).

THEN, provide your complete analysis in a JSON object with these fields. CRITICAL: Output ONLY valid JSON without any markdown formatting or code blocks. DO NOT wrap your JSON in "json" or any other markers. Start directly with and end with

Example CORRECT format (copy this structure): `{ {"identified_answer": "your answer", "answer_quote": ["quote1", "quote2"], "hardening_process": "strategy" } }`

Example WRONG format (DO NOT do this): `"json "identified_answer": "your answer" "`

- `"identified_answer"`: The complex relationship or multi-step conclusion derived from synthesizing document content
- `"answer_quote"`: Multiple relevant quotes showing the different pieces needed (not just one quote)
- `"hardening_process"`: Your explicit strategy for making this question difficult (from Step 2)
- `"exam_question"`: A challenging, self-contained question requiring synthesis. Return empty string if document lacks sufficient complexity.
- `"correct_answer"`: Complete answer showing the reasoning chain using document content. Return empty string if not derivable from document.
- `"multiple_choice_question"`: Self-contained MC question with balanced options. Return empty string if no question generated.
- `"multiple_choice_correct"`: The correct option letter (A, B, C, or D). Return empty string if no MC question.
- `"self_test_solution"`: Your step-by-step solution of the MCQ showing the difficulty (from Step 7)
- `"knowledge_and_reasoning_steps"`: Detailed reasoning path showing why this is Hard/Extra Hard difficulty.
- `"question_difficulty"`: Target difficulty (Hard/Extra Hard). Avoid "Easy" unless document truly lacks complexity.

## Free-form Question Challenger Prompt

Your task is to create CHALLENGING free-form questions from a document that require deep understanding and complex reasoning.

## Text [BEGINNING OF THE DOCUMENT] {text} [END OF THE DOCUMENT]

## Answer Type You must generate a question with answer type: answer\_type

## Instructions

### Step 1: Complex Information Extraction for answer\_type \*\*PRIORITY:\*\* Focus on information that requires synthesis and multi-step reasoning\*\* Based on the answer type answer\_type, scan the text and identify:

\*\*For Integer/Float answers:\*\* \* Multi-variable calculations spanning different sections \* Sequential computations where each step depends on previous results \* Counting problems requiring careful categorization \* Rate/ratio/percentage problems with multiple components \* Optimization problems with constraints \* Statistical calculations requiring data aggregation

\*\*For Expression answers:\*\* \* Relationships between multiple variables that form equations \* Patterns that can be generalized into formulas \* Systems of equations from different constraints \* Derivative relationships or functional dependencies \* Algebraic expressions combining multiple principles \* Recursive or iterative formulas

\*\*For String answers (MUST BE CONCISE):\*\* \* Single words or short phrases (1-3 words maximum) \* Technical terms, names, or identifiers \* Categories or classifications (single term only) \* Named entities (person, place, concept, method name) \* Units, symbols, or abbreviated forms \* AVOID: Long descriptions, sentences, or explanations \* Examples: "Newton", "TCP/IP", "gradient descent", "Paris", "O(n log n)"

\*\*For Boolean answers:\*\* \* Complex logical conditions with multiple clauses \* Statements requiring verification across multiple facts \* Comparative claims needing multi-point analysis \* Existence or uniqueness proofs \* Conditional truths depending on context \* Negations requiring comprehensive checking

\*\*AVOID\*\*: \* Direct lookups or single-fact answers \* Simple arithmetic or basic calculations \* Definitions stated verbatim in text \* Trivial yes/no questions

### Step 2: Difficulty Enhancement Strategy \*\*EXPLICITLY STATE YOUR HARDENING PROCESS\*\* Before generating the question, document your strategy: 1. What simple version would be too easy? 2. What complexity layers will you add? 3. Which document sections must be synthesized? 4. What intermediate steps are required? 5. How will you prevent shortcut solutions? 6. What makes this require deep understanding?

Document this in "hardening\_process".

### Step 3: Advanced Question Generation for answer\_type Create a question that: \* Requires connecting 3+ concepts from different parts \* Cannot be answered by simple lookup or keyword matching \* Forces multi-step reasoning to reach the answer \* Tests understanding of relationships, not memorization \* May involve applying principles to modified scenarios \* Requires precise interpretation for the specific answer type

\*\*CRITICAL - Self-Contained Requirements:\*\* \* Questions must be 100% NEVER use: "according to the text", "in the document", "as mentioned", etc. \* Write as if for a formal exam with no reference material \* Include all necessary context and definitions \* Specify units, formats, or constraints clearly

### Step 4: Answer Precision for answer\_type \*\*CRITICAL - Answer Format Requirements\*\*:

\*\*Integer answers:\*\* \* Must be whole numbers only \* Specify units if applicable (e.g., "in meters", "number of items") \* No decimals, fractions, or ranges \* CORRECT JSON: "answer": 42 or "answer": "42" \* WRONG JSON: "answer": [42] or "answer": "value": 42

\*\*Float answers:\*\* \* Specify precision required (e.g., "to 2 decimal places") \* Include units if applicable \* Use decimal notation, not fractions \* Example: 3.14, not " $\pi$ " or "22/7"

\*\*Expression answers:\*\* \* Use standard mathematical notation \* Variables must be clearly defined \* Simplify to canonical form \* CORRECT JSON: "answer": "2\*x^2 + 3\*x - 5" \* WRONG JSON: "answer": ["2\*x^2 + 3\*x - 5"] or "answer": "expr": "2\*x^2"

\*\*String answers:\*\* \* Specify exact format expected \* Case sensitivity requirements \* No extra punctuation or quotes \* CORRECT JSON: "answer": "Newton's Third Law" \* WRONG JSON: "answer": ["Newton's Third Law"] or "answer": "text": "Newton"

\*\*List answers:\*\* \* Specify ordering (alphabetical, chronological, by magnitude) \* Delimiter format (comma-separated, JSON array) \* Whether duplicates are allowed \* Example: ["apple", "banana", "cherry"] for JSON format

\*\*Boolean answers:\*\* \* Must be exactly "true" or "false" (lowercase) \* No "yes/no", "T/F", or other variations \* Clear truth conditions \* Example: true, not "True" or "yes"

### Step 5: Solution Verification Process \*\*SOLVE YOUR OWN QUESTION STEP-BY-STEP\*\* Work through the complete solution: 1. Identify all required information pieces 2. Show each calculation or reasoning step 3. Handle any edge cases or special conditions 4. Arrive at the final answer in the correct format 5. Verify the answer matches the specified type exactly 6. Estimate solving time (should be >1 minute for hard questions)

Document your solution in "step\_by\_step\_solution".

### Step 6: Difficulty Calibration Rate your question's difficulty and justify:

\*\*MEDIUM (2-3 steps, 1-2 minutes):\*\* \* Requires combining 2-3 document sections \* Clear path once relationships identified \* Some calculation or reasoning required

\*\*HARD (4-5 steps, 2-3 minutes):\*\* \* Synthesizes 4+ concepts \* Multiple valid approaches possible \* Requires careful analysis to avoid errors

\*\*EXTRA HARD (6+ steps, 3+ minutes):\*\* \* Complex system with many interactions \* Counter-intuitive results possible \* Requires deep understanding of principles

Document reasoning in "difficulty\_justification". ...

## Free-form Question Challenger Prompt (continued)

### Step 7: Alternative Interpretations Check \*\*ENSURE UNAMBIGUOUS ANSWER\*\* Verify your question has exactly ONE correct answer: 1. Could the question be interpreted differently? 2. Are all constraints clearly specified? 3. Is the answer format unambiguous? 4. Would different valid approaches yield the same answer? 5. Are edge cases properly handled?

If ambiguous, revise the question for clarity.

### Step 8: Final Complexity Verification Before finalizing, verify: \* Cannot be answered by simple text search \* Requires understanding, not just extraction \* Answer type matches answer\_type exactly \* Solution requires multiple reasoning steps \* Question is self-contained and clear \* Difficulty matches your target level

### ## CRITICAL ANSWER FORMAT RULES - MUST FOLLOW EXACTLY

The "answer" field MUST be a simple value, NOT nested in lists or dicts. The "question" field MUST be a single string, NOT a list of questions.

CORRECT formats: - question: "What is 2+2?" (NOT ["What is 2+2?"] or ["Q1", "Q2"]) - Integer answer: 42 or "42" (NOT [42] or "value": 42) - Expression answer: "2\*x + 5" (NOT ["2\*x + 5"] or "expr": "2\*x + 5") - String answer: "Paris" or "TCP/IP" (1-3 words max, NOT ["Paris"]) or "answer": "Paris"

WRONG formats that will FAIL: - question: ["What is 2+2?"] - Don't return list of questions! - question: ["Q1: ...", "Q2: ..."] - Return ONLY ONE question! - answer: [42] - Don't wrap in list! - answer: "value": 42 - Don't wrap in dict! - answer: "answer": "Paris" - Don't nest the answer!

## Output Format FIRST, think step-by-step about your question design (this is your private thinking).

THEN, provide your complete analysis in a JSON object with these fields. CRITICAL: Output ONLY valid JSON without any markdown formatting or code blocks. DO NOT wrap your JSON in "json" or any other markers. Start directly with and end with

Example CORRECT format (copy this structure): "identified\_information": ["info1", "info2"], "question": "your question", "answer": 42, "answer\_type": "Integer"

Example WRONG format (DO NOT do this): "json "question": "your question", "answer": 42 "

- "identified\_information": List the 3+ key pieces of information from different document sections needed to solve - "relevant\_quotes": Include multiple verbatim quotes from the document showing the different pieces needed - "hardening\_process": Describe your explicit 4-step strategy for making this question difficult - "question": EXACTLY ONE complete, challenging, self-contained question as a single string (NOT a list, NOT multiple questions). Return empty string if document lacks complexity for answer\_type questions. - "answer": The precise answer in correct answer\_type format. Return empty string if not derivable from document. - "answer\_type": Must be exactly "answer\_type" - "step\_by\_step\_solution": List each step of the complete solution showing the reasoning chain - "intermediate\_results": Dictionary of intermediate calculations or conclusions from each step - "difficulty\_level": Either "Hard" or "Extra Hard" (no Medium for this complexity level) - "difficultyJustification": Explain why this specific difficulty rating based on steps and concepts required - "solving\_time\_estimate": Realistic estimate in minutes for a student to solve - "required\_concepts": List the specific concepts from the document that must be understood - "potential\_errors": Common mistakes or edge cases students might encounter

\*\*CRITICAL RULES\*\*: 1. If document lacks complexity for answer\_type, return "question": "", "answer": "", "answer\_type": "answer\_type" 2. Answer field must EXACTLY match answer\_type format requirements 3. Never reference "the document" or "the text" in the question 4. Ensure answer is derivable from provided document content 5. Question must be solvable with document information alone 6. For String answers: MAXIMUM 3 words - prefer single terms, names, or short identifiers

**Figure 10** Challenger Prompts. Depending on the document, the Challenger uses either the MCQ or Free-form prompt to generate tasks for the Reasoner.