GENENV: DIFFICULTY-ALIGNED CO-EVOLUTION BETWEEN LLM AGENTS AND ENVIRONMENT SIMULATORS

Jiacheng Guo[1,*] **Ling Yang**[1,*,†] **Peter Chen**[2,*] **Qixin Xiao**[3,*]
Yinjie Wang[4] Xinzhe Juan[3] Jiahao Qiu[1] Ke Shen[1] Mengdi Wang[1,†]
[1]Princeton University [2]Columbia University [3]University of Michigan [4]University of Chicago
[*]Equal Contribution [†]Corresponding Authors

ABSTRACT

Training capable Large Language Model (LLM) agents is critically bottlenecked by the high cost and static nature of real-world interaction data. We address this by introducing **GENENV**, a framework that establishes a **difficulty-aligned co-evolutionary game** between an agent and a scalable, generative environment simulator. Unlike traditional methods that evolve models on static datasets, GENENV instantiates a **DATA-EVOLVING PARADIGM**: the simulator acts as a dynamic curriculum policy, continuously generating tasks specifically tailored to the agent's "zone of proximal development". This process is guided by a simple but effective **α-Curriculum Reward**, which aligns task difficulty with the agent's current capabilities. We evaluate GENENV on five benchmarks, including API-Bank, ALFWorld, BFCL, Bamboogle, and TravelPlanner. Across these tasks, GENENV improves agent performance by up to **+40.3%** over 7B baselines and matches or exceeds the average performance of larger models. Compared to Gemini 2.5 Pro-based offline data augmentation, GENENV achieves better performance while using **3.3$\times$ less data**. By shifting from static supervision to adaptive simulation, GENENV provides a data-efficient pathway for scaling agent capabilities.

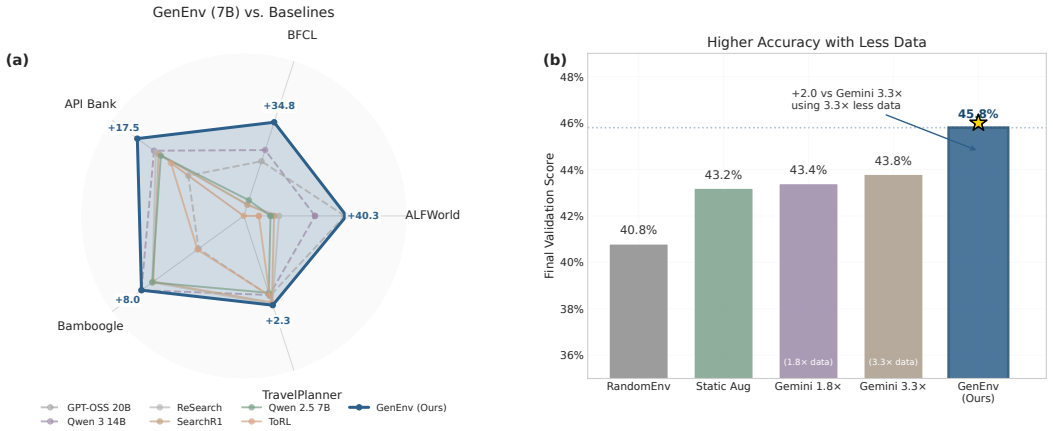


Figure 1: **GenEnv's cross-benchmark gains and data efficiency.** (a) We compare GENENV (7B) against representative baselines (Qwen2.5-7B, ReSearch, SearchR1, ToRL) and larger open models (e.g., Qwen3-14B, GPT-OSS-20B). Blue callouts report the *absolute* improvement of GENENV over Qwen2.5-7B on each benchmark. (b) Validation data-efficiency comparison on BFCL: GENENV surpasses RandomEnv and Static Augmentation, and outperforms Gemini-based offline augmentation even with 3.3$\times$ more synthetic data. In summary, *difficulty-aligned adaptive simulation* can outperform stronger static augmentation baselines under comparable training settings.

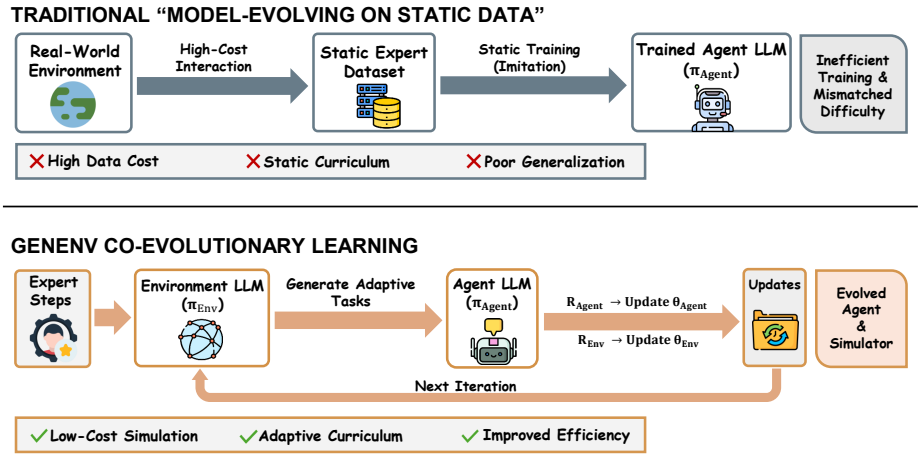


Figure 2: A comparison between the traditional training paradigm and our proposed GENENV framework. The traditional approach (top) relies on high-cost interaction with the real world to create a static dataset, leading to inefficient training and poor generalization. GENENV (bottom) creates a co-evolutionary loop where an Environment LLM generates adaptive tasks for the Agent LLM, enabling low-cost simulation, an adaptive curriculum, and improved efficiency.

1 INTRODUCTION

Training capable Large Language Model (LLM) agents for complex, interactive tasks like web navigation or tool use is constrained by a significant bottleneck: the high cost of data collection through real-world interaction (Yao et al., 2022; Shinn et al., 2023; Ning et al., 2025; Wang et al., 2025c;d). Each step an agent takes in a real-world environment can be slow, expensive, and difficult to parallelize. For instance, a web agent that navigates an e-commerce site may fail when a button's label changes from "Add to Cart" to "Add to Basket" (Gür et al., 2023), but discovering such failure modes requires extensive and costly real-world exploration. This fragility highlights a key limitation in how these agents are commonly trained.

A central driver of this issue is the reliance on static, pre-collected datasets of expert trajectories (Pomerleau, 1991; Levine et al., 2020; Samadi et al., 2024). Such datasets, no matter how large, represent a fixed snapshot of the world and struggle to capture the wide range of variations an agent will encounter in open-ended environments (Levine et al., 2020). Increasing the dataset size alone does not resolve this limitation: the bottleneck often lies not just in data volume, but in the static and costly nature of its collection and its inability to adapt as the agent improves.

The challenge of insufficient and static data has led to significant interest in synthetic data generation. However, despite progress, these methods often produce a large but ultimately static corpus that can fail to adapt to the agent's evolving requirements (Ye et al., 2024; Ding et al., 2024). This can result in inefficient training that still does not effectively target the agent's specific weaknesses. The high cost of interaction and data collection remains a core problem.

To address this, we approach the problem differently by proposing **GENENV**, a framework built on leveraging an LLM as a **scalable environment simulator** to reduce interaction costs. Instead of relying on slow and expensive real-world feedback, our framework trains the agent almost entirely within a simulated environment that can generate diverse and relevant training scenarios at a substantially lower cost. As illustrated in Figure 2, this contrasts with traditional methods that evolve a model on static data. In our approach, a generative environment model is trained to produce an adaptive curriculum of tasks, creating challenges tailored to the agent's performance. This leads to our primary research question: *Can an LLM-based environment simulator provide a scalable, low-cost alternative to real-world interaction for effectively training capable agents?*

In this simulation-centric process, the agent learns to overcome challenges generated by the simulator. The agent's performance provides a natural reward signal that guides the simulator's curriculum

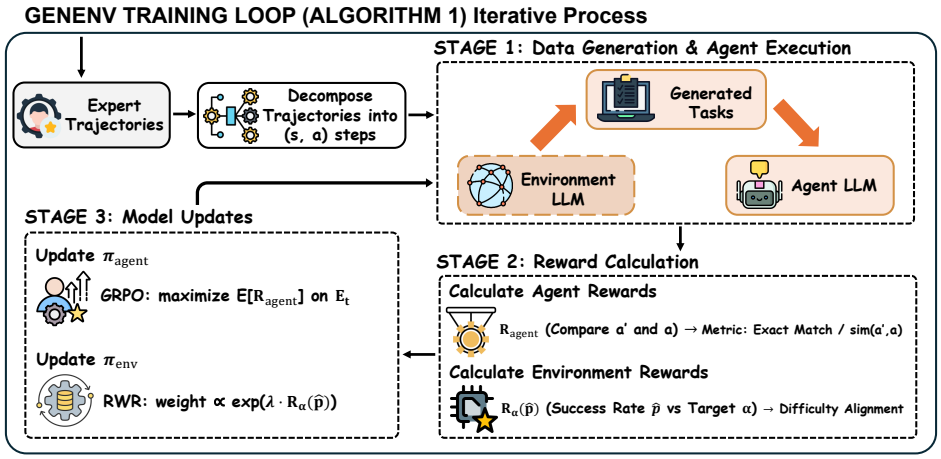


Figure 3: **The GENENV Co-Evolutionary Loop.** (1) The Environment Policy generates tasks. (2) The Agent Policy attempts them. (3) The environment reward (difficulty alignment) updates the simulator, while the agent reward (task success) updates the agent.

generation, allowing both to improve in a self-contained training loop. Throughout the paper, we use "agent" and **Agent Policy** π_{agent} interchangeably, and "environment simulator" and **Environment Policy** π_{env} interchangeably. As previewed in Figure 1, GENENV delivers consistent gains over 7B baselines across five benchmarks and achieves higher accuracy than Gemini-based offline augmentation while using less synthetic data, highlighting the advantage of difficulty-aligned, adaptive simulation over static scaling of data. Our **contributions** are:

- **The Data-Evolving Paradigm:** We propose a co-evolutionary framework where the training data distribution adapts dynamically to the agent's learning progress, breaking the reliance on static corpora.
- **Difficulty-Aligned Simulation:** We introduce the α-*Curriculum Reward*, a mechanism that rewards the simulator for generating tasks within the agent's target success zone (akin to the "zone of proximal development" (Vygotsky, 1978)), ensuring an efficient automated curriculum.
- **Data Efficiency:** On our benchmarks, GENENV matches or surpasses Gemini 2.5 Pro-based static augmentation pipelines while using **3.3× less synthetic data**, suggesting that an adaptive simulator can be more valuable than simply scaling the teacher model.

2 GENENV: DIFFICULTY-ALIGNED CO-EVOLUTION

GENENV views agent training as a *two-player curriculum game* rather than a single-player optimization problem. We maintain two policies: an **Agent Policy** π_{agent} (the agent) and an **Environment Policy** π_{env} (the environment simulator). Unlike standard RL where the environment is fixed, GENENV enables both to co-evolve:

- π_{agent} learns to solve tasks sampled from the current simulator.
- π_{env} is rewarded for generating tasks whose difficulty is *aligned* with the agent's current capability—targeting the "zone of proximal development" where learning is most effective (Vygotsky, 1978).

2.1 DATA-EVOLVING PARADIGM: FROM STATIC CORPORA TO ADAPTIVE SIMULATION

Standard training minimizes a loss $\mathcal{L}(\theta)$ over a static distribution $\mathcal{D}_{\text{static}}$, where θ denotes the parameters of the agent. In contrast, GENENV implements a DATA-EVOLVING PARADIGM. The training data $\mathcal{D}_t$ is generated on-the-fly by π_{env}, conditioned on the agent's historical performance. This creates a feedback loop (Figure 3) where the simulator seeks not to defeat the agent, but to find its "breaking points" to facilitate learning.

2.2 REWARDS: AGENT VS. ENVIRONMENT

Agent Task Reward (R_{agent}). Each environment-generated task induces a target action/trajectory a (e.g., a sequence of tool calls or a final answer), and the agent produces a prediction a'. We distinguish a structured action space $\mathcal{A}_{\text{struct}}$ (e.g., executable API calls) from free-form answers (e.g., natural language). For structured actions we can rely on exact execution; for unstructured ones we use a soft similarity score. We define the agent reward (used to update π_{agent}) as:

$$R_{\text{agent}}(a', a) = \mathbb{I}(a' = a) \cdot \mathbb{I}(a \in \mathcal{A}_{\text{struct}}) + \text{sim}(a', a) \cdot \mathbb{I}(a \notin \mathcal{A}_{\text{struct}}), \tag{1}$$

where $\text{sim}(a', a) \in [0, 1]$ is task-dependent (e.g., normalized token-F1 or embedding similarity). In all benchmarks we scale R_{agent} into $[0, 1]$.

Environment Difficulty-Alignment Reward (R_{env}). The core innovation for π_{env} is a difficulty-aligned reward that targets a *success-rate band* around a desired $\alpha \in (0, 1)$ (we use $\alpha = 0.5$). For each generated *batch* of n task variations, after the agent attempts them we compute the empirical success rate: $\hat{p} = \frac{k}{n}$, where k is the number of successes under R_{agent} (Eq. equation 1). We then assign the environment reward (used to update π_{env}):

$$R_{\text{env}}(\hat{p}) = \exp\left(-\beta(\hat{p} - \alpha)^2\right), \tag{2}$$

where $\beta > 0$ controls sharpness. This bell-shaped reward peaks when the agent's performance matches α, discouraging tasks that are already mastered ($\hat{p} \to 1$) or hopeless ($\hat{p} \to 0$). We additionally apply a *difficulty filter*: task batches with $|\hat{p} - \alpha| > k_{\text{min}}$ (we use $k_{\text{min}} = 0.1$) are excluded from environment updates to prevent overfitting to transient spikes.

2.3 Data Structures and How New Training Data Is Produced

A recurring confusion in co-evolution papers is *where the training data actually comes from*. We therefore make the data flow explicit.

What the environment generates. At epoch t, the environment policy π_{env} generates a task batch $\mathcal{T}_t$. Concretely, $\mathcal{T}_t$ contains n *task instances* (often multiple variations of the same seed), where each instance includes: (i) a task prompt/context (including tool specs, constraints, and goal), (ii) an evaluation specification (e.g., executable checker / exact-match target / reference answer), and optionally (iii) structured "ground truth" target action a when the benchmark provides it (e.g., tool-call arguments).

What the agent produces. The agent interacts with each task instance and yields an interaction trace (rollout) that we denote by an element $e \in \mathcal{E}_t$. Each trace records at minimum:

$$e = (\text{task}, \text{trajectory}, a', r),$$

where `trajectory` can include intermediate reasoning text and tool calls, a' is the final output/action, and $r = R_{\text{agent}}(a', a)$ is computed via Eq. equation 1.

Two growing datasets: one for the agent, one for the environment. We maintain two pools that grow online:

- **Agent training pool $\mathcal{D}_{\text{train}}$:** stores *valid* interaction traces from $\mathcal{E}_t$. A trace is "valid" if it is well-formed and evaluable for the benchmark (e.g., tool calls parse and execute; outputs follow required schema; checker runs without error). We append these valid tuples to $\mathcal{D}_{\text{train}}$ so the agent can (i) learn from fresh on-policy experiences and (ii) retain mastery of earlier curricula by continuing to sample from the accumulated pool.
- **Environment SFT pool $\mathcal{D}_{\text{env}}$:** stores environment generations used to train π_{env} via RWR. Each record is a supervised pair of the form

  $$(\text{env-conditioning context} \to \text{generated task instance}),$$

  where the conditioning context includes the seed prompt plus any summary signals (e.g., recent success statistics) that π_{env} conditions on. We weight each record by a monotone function of the environment reward in Eq. equation 2 (e.g., $\propto \exp(\lambda R_{\text{env}}(\hat{p}))$, where $\lambda = 1.0$ is a temperature hyperparameter).

How this produces a "data-evolving" training set. At every epoch, both $\mathcal{T}_t$ and $\mathcal{E}_t$ are newly generated; thus $\mathcal{D}_{\text{train}}$ is not a fixed offline corpus but an evolving mixture of (a) base data, (b) previously collected valid traces, and (c) newly collected on-policy traces. Meanwhile, $\mathcal{D}_{\text{env}}$ evolves toward generating tasks whose empirical success rate stays near α (Eq. equation 2), which in turn shifts the difficulty distribution of future $\mathcal{T}_{t+1}$. This closes the loop: *new data is produced as a byproduct of interaction*, and is then explicitly *aggregated* into training pools for subsequent updates.

2.4 TWO-PLAYER CURRICULUM RL: OPTIMIZATION LOOP

Player 1 Update (Agent). The Agent Policy π_{agent} is updated to maximize $\mathbb{E}[R_{\text{agent}}]$ (Eq. equation 1). In our experiments, we instantiate this using Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

Player 2 Update (Environment). The Environment Policy π_{env} is updated to maximize $\mathbb{E}[R_{\text{env}}]$ (Eq. equation 2). We implement this via **Reward-Weighted Regression** (RWR): from each environment-generated batch, we compute $\hat{p}$ (Eq. equation 2.2) from agent rollouts and assign $R_{\text{env}}(\hat{p})$. We then construct a weighted SFT set of environment generations and fine-tune π_{env} toward higher-reward generations. For stability, we regularize updates with a KL penalty to the initial simulator and cap per-step updates by a maximum KL threshold.

Algorithm 1 GENENV CO-EVOLUTIONARY LOOP

Initialize: Agent π_{agent}, Environment π_{env}, Agent pool $\mathcal{D}_{\text{train}}$, Env pool $\mathcal{D}_{\text{env}}$.
for epoch $t = 1, \ldots, T$ **do**
  ▷ **Phase 1: `Online Generation & Interaction`**
  Environment generates a task batch $\mathcal{T}_t \sim \pi_{\text{env}}(\cdot)$.
  Agent π_{agent} rolls out on $\mathcal{T}_t$ to obtain traces $\mathcal{E}_t$ and per-trajectory agent rewards R_{agent} via equation 1.
  Compute batch success $\hat{p}$ via equation 2.2 and assign environment reward $R_{\text{env}}(\hat{p})$ via equation 2.
  ▷ **Phase 2: `Dual Update (Two Players, Two Objectives)`**
  Update **agent** π_{agent} via GRPO to maximize $\mathbb{E}[R_{\text{agent}}]$ (equation 1).
  Filter out batches with $|\hat{p} - \alpha| > k_{\text{min}}$ for environment updates.
  Build weighted env SFT set $\tilde{\mathcal{D}}^t_{\text{env}}$ from $\mathcal{T}_t$ with weights $\propto \exp(\lambda R_{\text{env}}(\hat{p}))$ (equation 2).
  Update **environment** π_{env} via RWR on $\tilde{\mathcal{D}}^t_{\text{env}}$ to maximize $\mathbb{E}[R_{\text{env}}]$ (equation 2).
  ▷ **Phase 3: `Aggregation (How New Data Enters Training)`**
  Extract valid traces from $\mathcal{E}_t$ (e.g., parseable/executable/checker-passed) and append to agent pool:
    $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \text{Valid}(\mathcal{E}_t)$.
  Append weighted environment generations to env pool:
    $\mathcal{D}_{\text{env}} \leftarrow \mathcal{D}_{\text{env}} \cup \tilde{\mathcal{D}}^t_{\text{env}}$.
end for

3 THEORETICAL ANALYSIS OF GENENV

In this section, we provide a simple theoretical analysis of the difficulty-aligned co-evolution mechanism in GENENV. Our goal is not to fully characterize the dynamics of large LLMs, but to clarify why (1) tasks whose success rate is close to the target band α carry the strongest learning signal for the Agent Policy π_{agent}, and (2) the α-Curriculum Reward R_{env} provides a statistically consistent signal for the Environment Policy π_{env} to rank task types by how well their difficulty matches the current agent.

3.1 INTERMEDIATE DIFFICULTY MAXIMIZES LEARNING SIGNAL

We first consider a stylized bandit setting in which the Agent Policy π_{agent} interacts with a single environment-generated task type τ (e.g., a family of API-calling problems of similar difficulty). The outcome of each attempt is a scalar reward $r \in \{0, 1\}$, where $r = 1$ denotes success on the task and $r = 0$ denotes failure.[1] For a fixed Agent Policy parameterization θ, let $p(\tau) = \Pr(r = 1 \mid \tau, \theta)$ denote the success probability on task type τ. The Agent Policy is updated with a REINFORCE-style estimator

$$g(\tau, r) = (r - b(\tau))\nabla_\theta \log \pi_\theta(a \mid \tau), \tag{3}$$

where a is the sampled action (e.g., a rollout trajectory of tool-calling sequence) and $b(\tau)$ is a baseline (e.g., an estimate of the expected reward on τ). The quantity $\mathbb{E}[\|g(\tau, r)\|^2]$ can be viewed as measuring how strong the stochastic gradient signal is for this task type.

[1]The analysis extends to $R_{\text{agent}} \in [0, 1]$ by rescaling; we use the binary case for clarity.

Model	ALFWorld	BFCL	API-Bank	Bamboogle	TravelPlanner	Average
Large Scale Models						
Llama 3.1 405B	**65.3**	5.5	**74.4**	**77.6**	16.5	47.9
GPT-OSS 120B	60.4	21.9	53.6	29.6	14.7	36.0
Qwen 2.5 72B	63.5	**35.3**	54.9	69.6	20.5	**48.8**
Llama 3.1 70B	60.1	13.4	64.3	76.8	17.6	46.4
Qwen 3 32B	52.3	33.8	63.8	71.2	**22.5**	48.7
GPT-OSS 20B	53.6	24.4	41.2	33.6	14.9	33.5
Qwen 3 14B	37.8	29.4	66.7	76.0	14.7	44.9
7B Models						
ReSearch	18.7	5.0	65.3	68.0	16.4	34.7
SearchR1	16.1	5.0	63.3	67.2	16.1	33.5
Qwen 2.5 7B	14.2	7.0	61.6	68.0	14.3	33.0
ToRL	8.0	0.0	54.1	34.4	14.8	22.3
GenEnv (Ours)	**54.5**	**41.8**	**79.1**	**76.0**	**16.6**	**53.6**

Table 1: **Main Results.** Comparison on five benchmarks. Models are grouped by size: **Large Scale Models** (> 10B) are sorted by size descending, followed by **7B Models**. Bold numbers indicate the best performance within each group. **GenEnv (7B)** significantly outperforms other 7B baselines and even surpasses the average performance of several 72B/405B models on this suite.

Given the trust-region KL constraint and the gradient-clipping bias in GRPO-style policy updates, it is expected that the squared norm of the score function remains within a trust-region bound. This behavior has been discussed in recent analyses of one-step policy updates (e.g., Chen et al. (2025c, Theorem 3.2)) as well as in the literature on trust-region policy optimization (Schulman et al., 2015). Accordingly, we establish our theoretical analysis under the reasonable assumption that the squared norm of the score function $\nabla_\theta \log \pi_\theta(a \mid \tau)$ does not vary too dramatically when conditioned on the binary outcome r.

Assumption 3.1 (Bounded score variation)**.** For a fixed task type τ, there exist constants $0 < c_{\min} \leq c_{\max} < \infty$ such that $c_{\min} \leq \mathbb{E}\Big[\big\|\nabla_\theta \log \pi_\theta(a \mid \tau)\big\|^2 \Big| r\Big] \leq c_{\max},$ for both $r = 0$ and $r = 1$.

We take the baseline to be the on-task expected reward, $b(\tau) = \mathbb{E}[r \mid \tau] = p(\tau)$, which is the variance minimizer in the standard REINFORCE analysis. Under these conditions we obtain the following result.

Proposition 3.2 (Intermediate difficulty maximizes gradient signal)**.** *Suppose Assumption 3.1 holds and the baseline is chosen as $b(\tau) = p(\tau)$. Then there exist positive constants $C_{\min}$ and $C_{\max}$, independent of $p(\tau)$, such that*

$$C_{\min}\, p(\tau)\big(1 - p(\tau)\big) \leq \mathbb{E}\big[\|g(\tau, r)\|^2\big] \leq C_{\max}\, p(\tau)\big(1 - p(\tau)\big). \tag{4}$$

In particular, up to constant factors, the expected squared gradient norm is proportional to $p(\tau)\big(1 - p(\tau)\big)$, which is maximized when $p(\tau) = 1/2$, i.e., for tasks of intermediate difficulty.

Remark 3.3 ($\frac{1}{2}$-Curriculum reward)**.** Considering the case that $\alpha = \frac{1}{2}$, we have the identity $p(1-p) = \frac{1}{4} - \big(p - \frac{1}{2}\big)^2$. Thus, maximizing the variance term $p(1-p)$ is exactly equivalent to minimizing the squared distance to the target success rate $\alpha = \frac{1}{2}$. In GenEnv, the Environment Policy does not observe the true success probability $p(\tau)$ but only an empirical estimate $\hat{p}(\tau)$ from a finite number of rollouts. The α-Curriculum Reward takes the form

$$R_{\text{env}}(\hat{p}(\tau)) = \exp\big(-\beta(\hat{p}(\tau) - \alpha)^2\big), \tag{5}$$

which is a monotone transformation of $-(\hat{p}(\tau) - \alpha)^2$ and therefore encourages the simulator to propose tasks whose empirical success rate stays close to α. Proposition 3.2 then suggests that, in expectation, this aligns the simulator with task types that provide the strongest learning signal for π_{agent}. In Appendix B.1, we further provide an analysis towards ranking consistency of our proposed reward system.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Backbone Models. Unless otherwise specified, all 7B agents and simulators are initialized from Qwen2.5-7B-Instruct (Bai et al., 2025). For large-scale baselines in Table 1, we include Llama 3.1 models (Grattafiori et al., 2024), GPT-OSS open-weight models (OpenAI, 2025), and Qwen3 models (Yang et al., 2025). These models are evaluated using the same tool-calling interface and prompt templates as our 7B baselines for fairness.

Benchmarks. We evaluate across 5 diverse benchmarks that span tool use, embodied interaction, and real-world planning. **API-Bank** (Li et al., 2023) measures function-calling and tool-augmented reasoning, and **Bamboogle** is a compositional multi-hop QA benchmark built on top of the framework from Press et al. (2023); for both, we follow the evaluation protocols from ToolRL (Qian et al., 2025). **ALFWorld** (Shridhar et al., 2020) aligns textual instructions with embodied environments; we utilize the official validation set, where multi-turn tasks are decomposed into single steps for evaluation. **BFCL** follows the Berkeley Function-Calling Leaderboard setup (Patil et al.); specifically, we evaluate on the long-context subset treating each turn independently. **TravelPlanner** (Xie et al., 2024) captures end-to-end planning and tool use in realistic travel scenarios; we report the average of four metrics: CS Micro (%), CS Macro (%), HD Micro (%), and HD Macro (%). The **Average** column in Table 1 is the unweighted mean of these per-benchmark success metrics.

Baselines & Variants. We compare against standard instructed models (Qwen2.5-7B-Instruct) and specialized search-and-planning agents, including ReSearch, SearchR1, and ToRL (Jin et al., 2025; Li et al., 2025). These methods represent model-evolving pipelines that either improve search policies or alignment rewards on largely static datasets. To strictly evaluate our Data-Evolving contribution, we define:

- **GenEnv-Random:** The simulator generates new tasks every epoch but is *not* trained via R_{env}. This isolates the effect of dynamic data vs. aligned curriculum.
- **GenEnv-Static:** The simulator generates a large batch of synthetic data once before training.
- **Gemini-Offline (2x / 3.3x):** High-quality synthetic data generated offline by Gemini 2.5 Pro (approx. 1.76x and 3.27x the training set size). This represents a strong "teacher-distillation" baseline.

Figure 4: **Emergent curriculum in GENENV.** Across training epochs, the environment simulator gradually increases task complexity (a), reflected by longer task descriptions; the agent correspondingly produces longer reasoning chains (b) as it learns to solve harder tasks; and its success rate (c) remains within a controlled band despite rising difficulty. Together these curves show that GENENV induces an emergent curriculum in which task difficulty and agent capability co-evolve in a stable manner.

Training Configuration. For the Agent Policy (π_{agent}), we train for 10 epochs using GRPO with batch size 64 and maximum sequence length 9,000 tokens (prompt + response). The Environment Policy (π_{env}) is updated at the same epoch frequency via RWR with batch size 64. We use the same optimizer family (AdamW) for both policies, with learning rates detailed in Appendix B.3. All methods are trained on the same base dataset, and for GENENV variants the additional data comes solely from the simulator.

Evaluation Protocol. All models—including large models—are evaluated in a unified tool-calling framework. We use identical system prompts, tool specifications, and decoding settings for all models on a given benchmark. We do not attach additional multi-agent orchestration or human-in-the-loop corrections to any method, to focus the comparison on training and data regimes.

Summary of Results. Across all benchmarks, GENENV improves the 7B base agent significantly, with gains up to **+40.3%** on ALFWorld and **+20.4%** on API-Bank compared to baselines.

4.2 RQ1: DOES GENENV IMPROVE DOWNSTREAM TASK PERFORMANCE?

Table 1 presents the main comparison. GENENV consistently outperforms both general-purpose models and specialized RL agents across all five benchmarks. Notably, on **ALFWorld**, which requires long-horizon planning, GENENV achieves 54.5% accuracy compared to 14.2% for the base model, demonstrating the power of the simulator in generating diverse embodied scenarios that are costly to collect in the real world. On API-Bank and BFCL, GENENV achieves 79.1% and 41.8% success respectively, markedly improving over other 7B baselines that rely on static data or non-adaptive exploration.

Beyond absolute performance, GENENV also closes much of the gap to substantially larger models. The average score of GENENV (53.6) is competitive with—and in many cases exceeds—that of 14B–72B models that do not benefit from a difficulty-aligned simulator. This supports our central claim that *how* data is generated and aligned with the agent matters as much as, or more than, simply scaling model size or collecting larger static datasets. In the remainder of the section, we investigate how the co-evolutionary process shapes the curriculum, data efficiency, and difficulty calibration of the environment.

We also verify that the co-evolutionary training process itself is well behaved. Across all runs, we observe steadily increasing GRPO rewards and batch-level ground-truth accuracy, while validation performance improves monotonically before saturating. Additional training dynamics and learning curves are provided in Appendix C.

4.3 RQ2: DOES GENENV LEARN TO TACKLE HARDER TASKS OVER TIME?

We next investigate whether the environment simulator actually learns a curriculum, rather than simply generating random variations. To this end, we use the average length of the agent's required response as a proxy for reasoning complexity and task difficulty, and track how it evolves throughout training. Figure 4 plots this average response length together with the agent's success rate on simulated tasks across epochs.

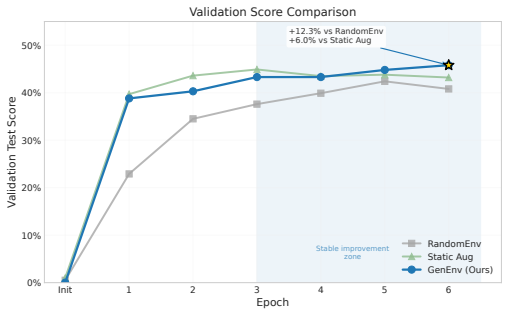
Finding. The required response length increases from 137 to 204 tokens (+49%) by epoch 6. This confirms that π_{env} learns to generate progressively more complex reasoning challenges as π_{agent} becomes more capable, creating an emergent curriculum without manual design. At the same time, the agent's success rate does not collapse; instead, it grows in tandem with task complexity, suggesting that the simulator is adapting difficulty in a controlled way rather than simply making tasks arbitrarily harder.

This pattern is consistent with our design of the α-*Curriculum Reward*: as the agent improves on a given family of tasks, their success rate on that family moves away from the target band around α, reducing its contribution to R_{env} and encouraging the simulator to propose harder variations. The observed increase in response length indicates that these harder tasks require more extensive multi-step reasoning and tool use, rather than superficial changes to surface form. Qualitatively, we observe that later tasks tend to involve more complex compositions of tools and deeper chains of intermediate subgoals.

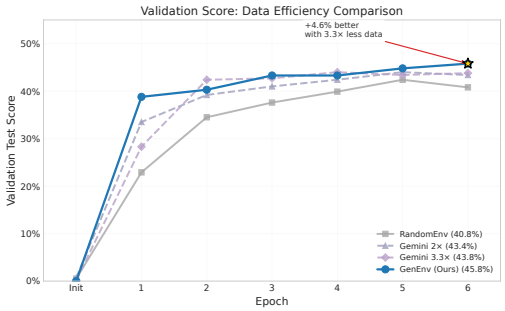
Finally, the fact that curriculum emerges *without any hand-specified difficulty schedule* supports our broader view of GENENV as a data-evolving system: the environment learns where the agent's "breaking points" are and adapts task generation accordingly. This contrasts with conventional curriculum learning, which typically relies on fixed heuristics or manually designed difficulty levels. Here, difficulty is inferred directly from the agent's behaviour via $R_{\text{env}}(\hat{p})$.

4.4 RQ3: IS GENENV MORE DATA-EFFICIENT THAN GEMINI-BASED AUGMENTATION?

A key question is whether GENENV is simply benefiting from "more data" or from *better-targeted* data. We compare GENENV (which generates data on-the-fly) against offline augmentation using Gemini 2.5 Pro, under comparable training budgets. Gemini-Offline (2x / 3.3x) corresponds to large



(a) **Method comparison.** GENENV outperforms both static Gemini-based augmentation and the GenEnv-Random variant, showing that *how* data is generated and aligned with the agent matters more than simply adding more offline synthetic data.



(b) **Data efficiency.** GENENV achieves higher validation performance while using substantially fewer synthetic samples than Gemini-based offline augmentation, indicating that difficulty-aligned, on-policy simulation provides more learning signal per example than untargeted teacher-generated data.

Figure 5: **Static vs. difficulty-aligned simulation.**

static corpora generated before training, while GENENV continuously adapts task generation as the agent evolves.

Finding. As shown in Figure 5b, GENENV (using 1x original data + dynamic simulation) reaches a validation score of 0.458. This outperforms Gemini-Offline (3.3x) (0.438), which uses ≈3.3x more synthetic data generated by a much stronger model. This confirms that *targeted, difficulty-aligned* data generation can be structurally superior to massive but untargeted augmentation. Furthermore, GENENV outperforms GenEnv-Random by 12.3%, indicating that the R_{env} optimization is critical.

These results highlight two distinct effects. First, merely adding more synthetic trajectories from a powerful teacher model quickly encounters diminishing returns: once the static dataset ceases to match the agent's current weaknesses, additional examples provide limited new learning signal. Second, the comparison between GENENV and GenEnv-Random controls for the presence of a simulator: both generate trajectories online, but only GENENV trains the simulator to target the α band. The performance gap between these two variants isolates the benefit of difficulty alignment itself, rather than just the benefit of having a generative environment.

From a practical standpoint, these findings suggest a shift in how we invest computational and annotation budget. Instead of paying for ever larger static datasets created by stronger teachers, it may be more effective to invest in a moderately sized simulator that co-evolves with the student agent. This is especially attractive in domains where collecting real trajectories is expensive or slow, as the simulator can keep generating fresh, on-policy data without requiring repeated human involvement. We further analyze whether the α-Curriculum Reward produces a well-calibrated difficulty distribution, and report additional training-time diagnostics in Appendix C.

5 RELATED WORK

Our work sits at the intersection of **LLM agents**, **trajectory synthesis for agent training**, and **LLM-based environment simulation**. Prior lines of work have improved agents by combining reasoning with tool use and interaction (Yao et al., 2022; Shinn et al., 2023), and by scaling training signals through synthetic trajectories and self-training (Ye et al., 2024; Ding et al., 2024; Song et al., 2024). More closely related are approaches that leverage simulators or generated environments to support exploration and data collection in lieu of costly real-world interaction (Todorov et al., 2012; Wang et al., 2025b; Piao et al., 2025). Compared to these, GENENV frames environment generation itself as a *learnable curriculum policy* and optimizes it with an explicit *difficulty-alignment* objective, yielding a co-evolutionary, data-evolving training loop.

For a more comprehensive discussion and additional related threads, please refer to Appendix A.

6 CONCLUSION

We presented **GENENV**, a framework that shifts agent training from a static, model-evolving process to a dynamic, **data-evolving game**. By establishing a difficulty-aligned co-evolutionary loop between an Agent Policy and an Environment Policy, GENENV achieves superior performance and data efficiency on a diverse suite of agent benchmarks. Our results suggest that future agent training systems should move beyond larger static datasets toward adaptive, self-calibrating simulation environments. Beyond the particular instantiation studied here, we believe that difficulty-aligned simulators can serve as a general recipe for training robust LLM agents in domains where real-world exploration is costly or risky.

IMPACT STATEMENT

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Peter Chen, Xi Chen, Wotao Yin, and Tianyi Lin. Compo: Preference alignment via comparison oracles. *arXiv preprint arXiv:2505.05465*, 2025a.

Peter Chen, Xiaopeng Li, Ziniu Li, Xi Chen, and Tianyi Lin. Stepwise guided policy optimization: Coloring your incorrect reasoning in grpo. 2025b.

Peter Chen, Xiaopeng Li, Ziniu Li, Wotao Yin, Xi Chen, and Tianyi Lin. Exploration vs exploitation: Rethinking rlvr through clipping, entropy, and spurious reward. *arXiv preprint arXiv:2512.16912*, 2025c.

Yihan Chen, Benfeng Xu, Xiaorui Wang, Yongdong Zhang, and Zhendong Mao. Training llm-based agents with synthetic self-reflected trajectories and partial masking. *arXiv preprint arXiv:2505.20023*, 2025d.

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. Data augmentation using large language models: Data perspectives, learning paradigms and challenges. *arXiv preprint arXiv:2403.02990*, 2024.

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

Jiacheng Guo, Zihao Li, Jiahao Qiu, Yue Wu, and Mengdi Wang. On the role of preference variance in preference optimization. *arXiv preprint arXiv:2510.13022*, 2025.

Izzeddin Gür, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2803–2821, 2023.

Thai Quoc Hoang, Kung-Hsiang Huang, Shirley Kokane, Jianguo Zhang, Zuxin Liu, Ming Zhu, Jake Grigsby, Tian Lan, Michael S Ryoo, Chien-Sheng Wu, et al. Lam simulator: Advancing data generation for large action model training via online exploration and trajectory feedback. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12921–12934, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Aleksei Kudrinskii, Saibo Geng, Luca Beurer-Kellner, and Marc Fischer. Faithful simulation of user–agent–environment interactions for scalable llm agent evaluation. In *Workshop on Scaling Environments for Agents*.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, et al. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6140–6150, 2025.

OpenAI. GPT-OSS: Open-weight models for reasoning and agents. `https://openai.com/index/introducing-gpt-oss/`, 2025. OpenAI blog post.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6300–6323, 2025.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, et al. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*, 2025.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Ram Ramrakhya, Andrew Szot, Omar Attia, Yuhao Yang, Anh Nguyen, Bogdan Mazoure, Zhe Gan, Harsh Agrawal, and Alexander Toshev. Scaling synthetic task generation for agents via exploration. *arXiv preprint arXiv:2509.25047*, 2025.

Amir Samadi, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati. Good data is all imitation learning needs. In *European Conference on Computer Vision*, pp. 144–159. Springer, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Agentbank: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories. *arXiv preprint arXiv:2410.07706*, 2024.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. *arXiv preprint arXiv:2506.21605*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Lev S Vygotsky. *Mind in society: The development of higher psychological processes*, volume 86. Harvard university press, 1978.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. Steca: Step-level trajectory calibration for llm agent learning. *arXiv preprint arXiv:2502.14276*, 2025a.

Yiming Wang, Da Yin, Yuedong Cui, Ruichen Zheng, Zhiqian Li, Zongyu Lin, Di Wu, Xueqing Wu, Chenchen Ye, Yu Zhou, et al. Llms as scalable, general-purpose simulators for evolving digital agent training. *arXiv preprint arXiv:2510.14969*, 2025b.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. *arXiv preprint arXiv:2506.03136*, 2025c.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Cure: Co-evolving coders and unit testers via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d.

Yang Xiao, Mohan Jiang, Jie Sun, Keyu Li, Jifan Lin, Yumin Zhuang, Ji Zeng, Shijie Xia, Qishuo Hua, Xuefeng Li, et al. Limi: Less is more for agency. *arXiv preprint arXiv:2509.17567*, 2025.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*, 2024.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *arXiv preprint arXiv:2412.09605*, 2024.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Junjie Ye, Nuo Xu, Yikun Wang, Jie Zhou, Qi Zhang, Tao Gui, and Xuanjing Huang. Llm-da: Data augmentation via large language models for few-shot named entity recognition. *arXiv preprint arXiv:2402.14568*, 2024.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025a.

Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. Deepanalyze: Agentic large language models for autonomous data science. *arXiv preprint arXiv:2510.16872*, 2025b.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. Knowagent: Knowledge-augmented planning for llm-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3709–3732, 2025.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

A FURTHER RELATED WORKS

A.1 LARGE LANGUAGE MODEL AGENTS

Recent advancements have demonstrated the ability of LLMs to function as autonomous agents. Pioneering works like ReAct, Reflexion, and Voyager have shown that combining chain-of-thought reasoning with action generation and memory evaluation enables agents to tackle complex tasks (Yao et al., 2022; Shinn et al., 2023; Wang et al., 2023; Zou et al., 2025; Guo et al., 2025). More recent efforts such as KnowAgent and MemBench further refine planning and memory capabilities in agentic settings (Zhu et al., 2025; Tan et al., 2025). Others, such as Toolformer and WebGPT, have focused on augmenting LLMs with external tools and web-browsing capabilities, expanding their operational scope (Schick et al., 2023; Nakano et al., 2021). While these models showcase strong performance, their training paradigms primarily rely on imitation learning from static, pre-collected datasets of expert trajectories (Nakano et al., 2021; Wang et al., 2023). This dependency on expert data forms a significant bottleneck, limiting the agent's ability to explore and discover strategies beyond the provided demonstrations (Shinn et al., 2023). Our work addresses this limitation by creating a dynamic learning environment that does not solely depend on a fixed dataset.

A.2 TRAJECTORY SYNTHESIS FOR AGENT TRAINING

Recent efforts have focused on **generating synthetic trajectories** to address the limitations of fixed expert data for training LLM agents (Yu et al., 2025). Some methods focus on **offline synthetic data generation**, creating novel trajectories to increase diversity and coverage where expert data is sparse (Ye et al., 2024; Ding et al., 2024). Other approaches leverage LLMs to generate self-reflective trajectories, incorporating reflections and corrections to learn from errors (Chen et al., 2025d), or to provide stepwise guidance from a teacher model toward correcting mistakes (Chen et al., 2025b). For web agents, scalable synthesis pipelines use web tutorials or exploration-driven methods to produce large-scale **synthetic datasets** of multimodal trajectories (Pahuja et al., 2025; Xu et al., 2024). Other works introduce iterative self-training for reflection (Yuan et al., 2025; Wang et al., 2025c), fine-tuning on massive interaction trajectories (Song et al., 2024), step-level calibration (Wang et al., 2025a), and simulators for online exploration to generate high-quality feedback-driven data (Hoang et al., 2025).

Beyond these, several recent frameworks further advance autonomous and adaptive trajectory synthesis. Wang et al. (2025b) propose a scalable LLM-based digital environment that models user-interface transitions as structured trajectories, combining simulation and targeted scaling to expose agents to increasingly complex states. Ramrakhya et al. (2025) build upon exploration-driven task generation, using a two-stage pipeline that first explores application environments and then synthesizes executable trajectories, yielding tens of thousands of realistic multimodal interaction traces. In the data-science domain, Zhang et al. (2025b) integrate curriculum-based agentic training with a data-grounded trajectory synthesis framework to produce high-fidelity analytical workflows, allowing smaller models to outperform larger workflow-based agents. Complementarily, Xiao et al. (2025) challenge the assumption that more data is always better—demonstrating that strategically curated and high-quality trajectories can induce stronger agentic reasoning from only a handful of demonstrations. Chen et al. (2025a) also demonstrates that noisy preference data could be utilized into improve agent alignment. Finally, Zhang et al. (2025a) introduce an implicit-feedback paradigm where agents learn from their own early interactions before formal reinforcement learning, leveraging self-reflection and world-modeling to bootstrap generalization from suboptimal actions.

Taken together, these developments indicate a trend toward **adaptive, self-improving trajectory synthesis**: instead of relying on static or expert-curated data, agents increasingly generate, evaluate, and refine their own experiences—closing the loop between simulation, exploration, and learning. This motivates our approach, which leverages a dynamic environment simulator to generate these adaptive experiences on-demand, directly addressing the limitations of static datasets.

A.3 ENVIRONMENT SIMULATION

Simulators have long been a cornerstone of reinforcement learning, especially in domains such as robotics where interacting with the real world is costly (Todorov et al., 2012). In the context of LLM agents, environment simulation has emerged as a key mechanism for generating training data and evaluating agent capabilities. Wang et al. (2025b) demonstrate how an LLM-powered

digital world simulator can generate structured user-interface states and transitions; its targeted scaling strategy produces diverse, high-impact tasks and yields agents that rival those trained on real UIs. Complementary simulation frameworks move beyond UI tasks: Kudrinskii et al. introduce a scalable closed-loop simulator that samples multi-step tasks from a tool–relationship graph, simulates interactions with configurable user and environment archetypes, and evaluates procedural alignment and success. Experiments reveal that environment reliability and user archetypes are dominant factors in agent performance. Beyond task-centric environments, Piao et al. (2025) integrate LLM-driven agents with a realistic societal environment and a large-scale simulation engine, generating social lives for over ten thousand agents and millions of interactions among agents and their surroundings. These works highlight the importance of faithful and diverse environment simulation in creating high-quality training data and benchmarks.

Our environment LLM diverges from traditional simulators: rather than predicting state transitions or user responses for a fixed task, it generates entire tasks and goals *conditioned on the agent's recent performance*, with an explicit objective to match a target difficulty band. This casts environment design itself as a learnable policy with its own reward signal.

B TECHNICAL DETAILS

B.1 RANKING CONSISTENCY OF THE α-CURRICULUM REWARD

We next show that, despite relying on noisy empirical success rates, the α-Curriculum Reward provides a statistically consistent signal for ranking task types by how well their difficulty matches the target band. The argument is based on standard concentration inequalities.

Consider two task types τ_1 and τ_2. For a fixed Agent Policy π_{agent}, let $p_i = p(\tau_i)$ denote the true success probability on τ_i, and define their distances to the target band α as

$$\Delta_i = |p_i - \alpha|, \quad i \in \{1, 2\}. \tag{6}$$

Without loss of generality, assume $\Delta_1 < \Delta_2$, i.e., τ_1 is closer to the target difficulty than τ_2. For each τ_i we run n_i independent rollouts and compute the empirical success rate $\hat{p}_i = k_i/n_i$, where k_i is the number of successes. The Environment Policy receives the reward

$$R_{\text{env}}(\hat{p}_i) = \exp\big(-\beta(\hat{p}_i - \alpha)^2\big). \tag{7}$$

Since the exponential is monotone, ranking tasks by R_{env} is equivalent to ranking them by their squared distance $(\hat{p}_i - \alpha)^2$ to the target band.

The following theorem shows that the mis-ranking probability decays exponentially in the minimum number of rollouts.

Theorem B.1 (Ranking consistency of R_{env})**.** *Let $n = \min\{n_1, n_2\}$ and $\Delta_1 < \Delta_2$ as above. Define $\delta = (\Delta_2 - \Delta_1)/3 > 0$. Then*

$$\Pr\big(R_{\textit{env}}(\hat{p}_1) \le R_{\textit{env}}(\hat{p}_2)\big) \;\le\; 4\exp\left(-\frac{2}{9}(\Delta_2 - \Delta_1)^2\, n\right). \tag{8}$$

In particular, as $n \to \infty$ the reward ranking is consistent: tasks whose true success probability lies closer to the target band α receive higher α-Curriculum Reward with probability approaching 1 at an exponential rate.

Proof sketch. Because the exponential is monotone, $R_{\text{env}}(\hat{p}_1) \le R_{\text{env}}(\hat{p}_2)$ is equivalent to $|\hat{p}_1 - \alpha| \ge |\hat{p}_2 - \alpha|$. We show that if both empirical estimates $\hat{p}_i$ lie within δ of their true means p_i, then necessarily $|\hat{p}_1 - \alpha| < |\hat{p}_2 - \alpha|$ and hence $R_{\text{env}}(\hat{p}_1) > R_{\text{env}}(\hat{p}_2)$. Thus a mis-ranking can only occur when at least one empirical mean deviates from its expectation by more than δ, which can be bounded using Hoeffding's inequality for Bernoulli random variables. A detailed proof is provided in Appendix B.2. □

Implications for GENENV. Theorem B.1 shows that, even though the Environment Policy only observes noisy empirical success rates $\hat{p}_i$ derived from a finite number of rollouts, the α-Curriculum Reward is a statistically consistent proxy for task difficulty. As we increase the rollout budget per task

type, the environment LLM can more reliably identify and up-weight task families whose difficulty lies in the target zone of proximal development. This provides a formal justification for the empirical convergence behaviour observed in Figure 7, where the agent's success rate on simulated tasks concentrates around a band centered at $\alpha = 0.5$. Together, Proposition 3.2 and Theorem B.1 clarify why aligning the simulator with an intermediate success rate both maximizes learning signal for the agent and yields a stable, difficulty-calibrated curriculum.

B.2 Proofs for Section 3

In this appendix we provide detailed proofs for the theoretical results stated in Section 3.

B.2.1 Proof of Proposition 3.2

Recall that for a fixed task type τ and parameter vector θ, the Agent Policy update uses the REINFORCE-style estimator

$$g(\tau, r) = (r - b(\tau))\nabla_\theta \log \pi_\theta(a \mid \tau),$$

and we choose the baseline to be $b(\tau) = \mathbb{E}[r \mid \tau] = p(\tau)$. Let $S = \nabla_\theta \log \pi_\theta(a \mid \tau)$ for brevity. Conditioned on τ, the reward r is Bernoulli with success probability $p(\tau)$, so $\mathbb{E}[r \mid \tau] = p(\tau)$ and $\mathrm{Var}(r \mid \tau) = p(\tau)(1 - p(\tau))$.

We have

$$\mathbb{E}\big[\|g(\tau, r)\|^2 \mid \tau\big] = \mathbb{E}\big[(r - p(\tau))^2 \|S\|^2 \mid \tau\big] \tag{9}$$

$$= \sum_{r \in \{0,1\}} \Pr(r \mid \tau)\,(r - p(\tau))^2\, \mathbb{E}\big[\|S\|^2 \mid \tau, r\big]. \tag{10}$$

By Assumption 3.1, for both $r = 0$ and $r = 1$ we have

$$c_{\min} \le \mathbb{E}\big[\|S\|^2 \mid \tau, r\big] \le c_{\max}.$$

Therefore

$$\mathbb{E}\big[\|g(\tau, r)\|^2 \mid \tau\big] \ge c_{\min} \sum_{r \in \{0,1\}} \Pr(r \mid \tau)(r - p(\tau))^2 \tag{11}$$

$$= c_{\min}\, \mathbb{E}\big[(r - p(\tau))^2 \mid \tau\big] \tag{12}$$

$$= c_{\min}\, \mathrm{Var}(r \mid \tau) \tag{13}$$

$$= c_{\min}\, p(\tau)\big(1 - p(\tau)\big), \tag{14}$$

and similarly

$$\mathbb{E}\big[\|g(\tau, r)\|^2 \mid \tau\big] \le c_{\max} \sum_{r \in \{0,1\}} \Pr(r \mid \tau)(r - p(\tau))^2 \tag{15}$$

$$= c_{\max}\, p(\tau)\big(1 - p(\tau)\big). \tag{16}$$

Taking $C_{\min} = c_{\min}$ and $C_{\max} = c_{\max}$ yields the bounds in Equation equation 4. Since $p(1 - p)$ is a concave quadratic on $[0, 1]$ with a unique maximum at $p = 1/2$, the expected squared gradient norm is maximized (up to constant factors) for tasks with $p(\tau) = 1/2$, i.e., tasks of intermediate difficulty. This proves Proposition 3.2.

B.2.2 Proof of Theorem B.1

We restate the setting for clarity. For $i \in \{1, 2\}$, the true success probability on task type τ_i is p_i, and we define $\Delta_i = |p_i - \alpha|$ with $\Delta_1 < \Delta_2$. From n_i independent rollouts we obtain the empirical success rate $\hat{p}_i = k_i/n_i$, where k_i is the number of successes. The α-Curriculum Reward is

$$R_{\text{env}}(\hat{p}_i) = \exp\big(-\beta(\hat{p}_i - \alpha)^2\big).$$

Since the exponential function is strictly monotone decreasing in $(\hat{p}_i - \alpha)^2$, we have

$$R_{\text{env}}(\hat{p}_1) \le R_{\text{env}}(\hat{p}_2) \quad \Longleftrightarrow \quad |\hat{p}_1 - \alpha| \ge |\hat{p}_2 - \alpha|.$$

Thus the event that the Environment Policy mis-ranks the two tasks (i.e., gives τ_1 no larger reward than τ_2) is exactly the event

$$\mathcal{E}_{\text{mis}} = \{\, |\hat{p}_1 - \alpha| \geq |\hat{p}_2 - \alpha| \,\}.$$

Let $n = \min\{n_1, n_2\}$ and define

$$\delta = \frac{\Delta_2 - \Delta_1}{3} > 0.$$

Consider the event

$$\mathcal{E}_{\text{good}} = \big\{\, |\hat{p}_1 - p_1| \leq \delta \text{ and } |\hat{p}_2 - p_2| \leq \delta \,\big\}.$$

We claim that on $\mathcal{E}_{\text{good}}$ we must have $|\hat{p}_1 - \alpha| < |\hat{p}_2 - \alpha|$, and hence $R_{\text{env}}(\hat{p}_1) > R_{\text{env}}(\hat{p}_2)$. Indeed, by the triangle inequality,

$$|\hat{p}_1 - \alpha| \leq |p_1 - \alpha| + |\hat{p}_1 - p_1| \leq \Delta_1 + \delta, \tag{17}$$

$$|\hat{p}_2 - \alpha| \geq |p_2 - \alpha| - |\hat{p}_2 - p_2| \geq \Delta_2 - \delta. \tag{18}$$

By the choice of δ, we have

$$\Delta_1 + \delta = \Delta_1 + \frac{\Delta_2 - \Delta_1}{3} = \frac{2\Delta_1 + \Delta_2}{3}, \qquad \Delta_2 - \delta = \Delta_2 - \frac{\Delta_2 - \Delta_1}{3} = \frac{\Delta_1 + 2\Delta_2}{3},$$

and since $\Delta_1 < \Delta_2$, it follows that

$$\Delta_1 + \delta = \frac{2\Delta_1 + \Delta_2}{3} < \frac{\Delta_1 + 2\Delta_2}{3} = \Delta_2 - \delta.$$

Therefore,

$$|\hat{p}_1 - \alpha| \leq \Delta_1 + \delta < \Delta_2 - \delta \leq |\hat{p}_2 - \alpha|$$

on $\mathcal{E}_{\text{good}}$, which implies $R_{\text{env}}(\hat{p}_1) > R_{\text{env}}(\hat{p}_2)$. Consequently, $\mathcal{E}_{\text{mis}}$ can only occur on the complement event $\mathcal{E}_{\text{good}}^c$, and hence

$$\Pr(\mathcal{E}_{\text{mis}}) \leq \Pr(\mathcal{E}_{\text{good}}^c) \leq \Pr\big(|\hat{p}_1 - p_1| > \delta\big) + \Pr\big(|\hat{p}_2 - p_2| > \delta\big),$$

where the last inequality is a union bound.

For each $i \in \{1, 2\}$, $\hat{p}_i$ is the empirical mean of n_i i.i.d. Bernoulli random variables with mean p_i. By Hoeffding's inequality,

$$\Pr\big(|\hat{p}_i - p_i| > \delta\big) \leq 2\exp(-2n_i\delta^2) \leq 2\exp(-2n\delta^2),$$

where $n = \min\{n_1, n_2\}$. Therefore

$$\Pr(\mathcal{E}_{\text{mis}}) \leq 4\exp(-2n\delta^2) \tag{19}$$

$$= 4\exp\left(-2n\left(\frac{\Delta_2 - \Delta_1}{3}\right)^2\right) \tag{20}$$

$$= 4\exp\left(-\frac{2}{9}(\Delta_2 - \Delta_1)^2\, n\right), \tag{21}$$

which establishes the desired exponential bound and completes the proof of Theorem B.1.

B.3 Hyperparameter Details

Table 2 lists the key hyperparameters used in our experiments. Note that for the Agent (π_{agent}), we employ GRPO, while the Environment (π_{env}) uses Reward-Weighted Regression (RWR).

B.4 Environment Baseline Implementation Details

Here we detail the specific implementation of the environment variants used in Section 4:

- **GenEnv-Random:** We use the same base model (Qwen2.5-7B-Instruct) for the environment. The `autoenv.disable_env_training` flag is set to `True`. It generates 4 variations per prompt per epoch dynamically, but the model weights are never updated based on $R_{\text{env}}(\hat{p})$.

Table 2: Hyperparameters for GENENV training.

Parameter	Agent Policy (π_{agent})	Environment Policy (π_{env})
	Optimizer (AdamW)	
Learning Rate	1×10^{-6}	5×10^{-7}
Batch Size	64	64
	Policy Optimization	
Method	GRPO	Reward-Weighted Update (RWR)
Total Epochs	10	10
Target Difficulty α	N/A	0.5
Difficulty Filter k_{min}	N/A	0.1
RWR Temperature λ	N/A	1.0

- **GenEnv-Static:** We use the `generate_static_augmentation.py` script to pre-generate 5 variations for each of the 544 original training samples, resulting in a fixed dataset of 3,264 samples. The agent is trained using standard PPO for 10 epochs.
- **Gemini-Offline Baselines:** We prompted Gemini 2.5 Pro to generate variations of the training data. Due to API constraints and filtering, the **Gemini-2x** setting resulted in 957 samples (≈1.76x) and **Gemini-4x** resulted in 1,777 samples (≈3.27x). These represent high-quality, but static, external data augmentation.

C ADDITIONAL EXPERIMENT RESULTS

Overview. This section provides supporting visualizations for analyses briefly discussed in the main text. In particular, following the discussion on training stability in Section 4, we visualize the corresponding learning dynamics in Figure 6. We then report a focused analysis of **difficulty calibration** induced by the α-Curriculum Reward (Section C.1).

C.1 RQ4: DOES THE ENVIRONMENT REWARD PRODUCE WELL-CALIBRATED DIFFICULTY?

A key design goal of GENENV is that the environment policy π_{env} should *self-calibrate* task difficulty to stay near the target success band centered at α (we use $\alpha = 0.5$). Here we empirically test this calibration behavior during co-evolution, and relate it to the theoretical *ranking consistency* guarantee in Theorem B.1.

Figure 7 tracks the agent's empirical success rate on simulator-generated tasks across training epochs. Starting from 0.138, the success rate moves toward the target band and remains in a relatively narrow range (approximately $[0.4, 0.6]$ for most of training), rather than drifting toward extremes. This indicates that the simulator is not collapsing to trivial tasks ($\hat{p} \to 1$) or saturating on impossible ones ($\hat{p} \to 0$), but instead actively maintains a productive intermediate difficulty regime.

Finding. The agent's success rate on simulator-generated tasks converges toward the target band centered at $\alpha = 0.5$ and stays near it for most of training, consistent with effective difficulty calibration by π_{env}.

This behavior matches the mechanism implied by our analysis: Theorem B.1 shows that, given sufficient rollouts, $R_{\text{env}}(\hat{p})$ provides a statistically consistent signal that favors task types whose true success probabilities lie closest to α. Empirically, early updates shift the task distribution away from being overly easy or overly hard; later, updates become smaller as the system stabilizes around the target band, yielding a self-calibrated curriculum.

Beyond calibration, we also examine *how* problem-solving behavior changes over time. Figure 8 shows that GENENV increases the fraction of solved tasks and reduces unsolved ones more quickly than the RandomEnv variant. This suggests that the calibrated difficulty band coexists with meaningful learning progress: the simulator does not merely keep the agent near 50% success on a fixed distribution; rather, it continuously shifts the task frontier so that "intermediate difficulty" evolves as the agent improves. Taken together with the main-paper downstream gains, these diagnostics support

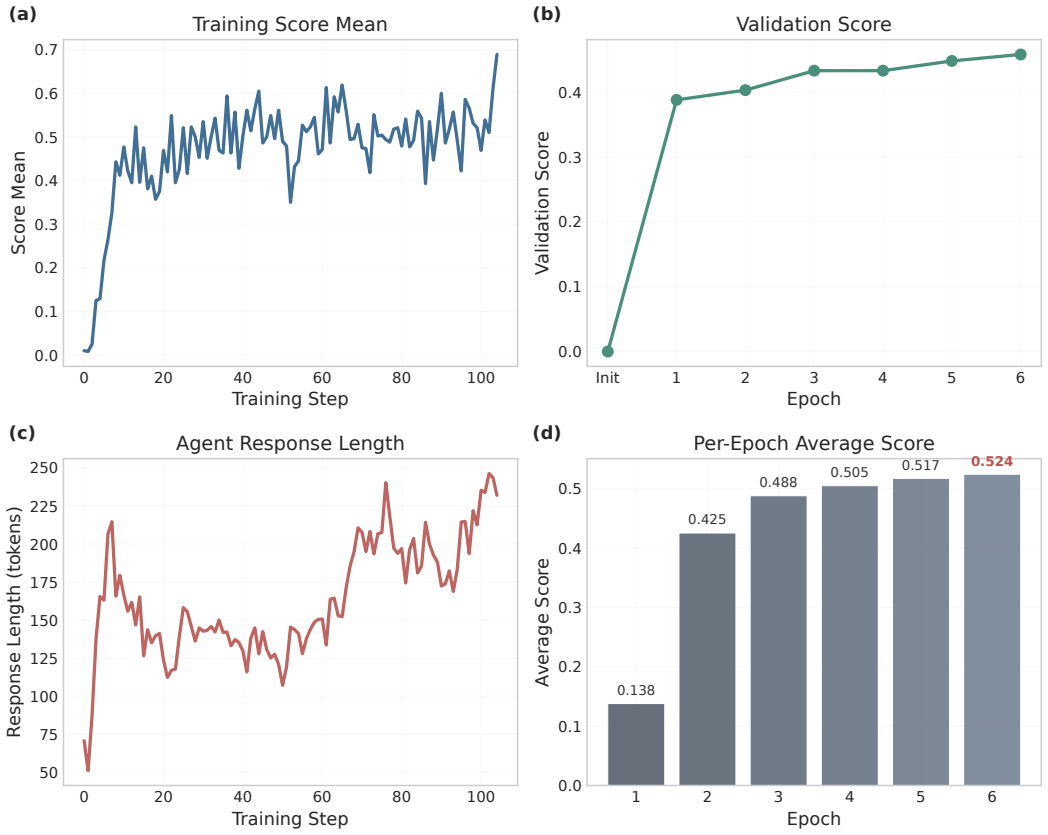


Figure 6: **Training dynamics of GenEnv.** From left to right: (a) training step-wise reward (`critic/score/mean`); (b) validation score across epochs; (c) batch-level ground-truth accuracy; and (d) per-epoch average reward. The curves show that GenEnv trains stably without reward collapse or divergence, with both reward and accuracy improving smoothly over time.

our view of GenEnv as a genuinely co-evolving system, where both the agent and task distribution adapt in lockstep under a stable training process.

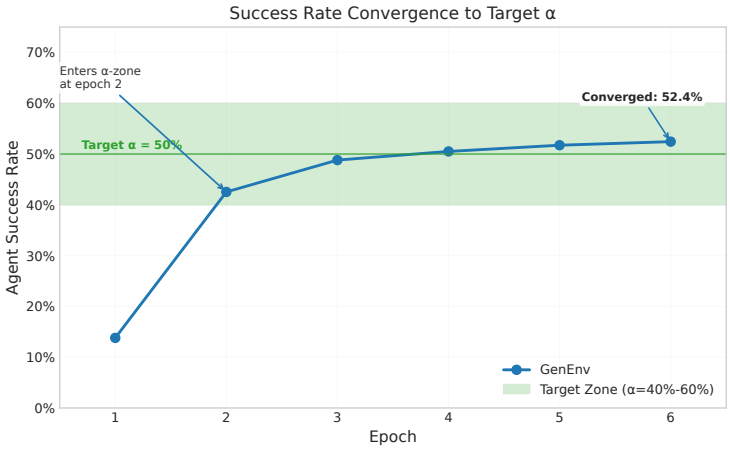


Figure 7: **Difficulty calibration via the α-Curriculum Reward.** As training progresses, the agent's success rate on simulator-generated tasks converges to a band centered at $\alpha = 0.5$, indicating that π_{env} adapts task difficulty to match the agent's current capability. This empirically supports the ranking consistency of R_{env} and suggests that the simulator maintains tasks in the zone of proximal development.

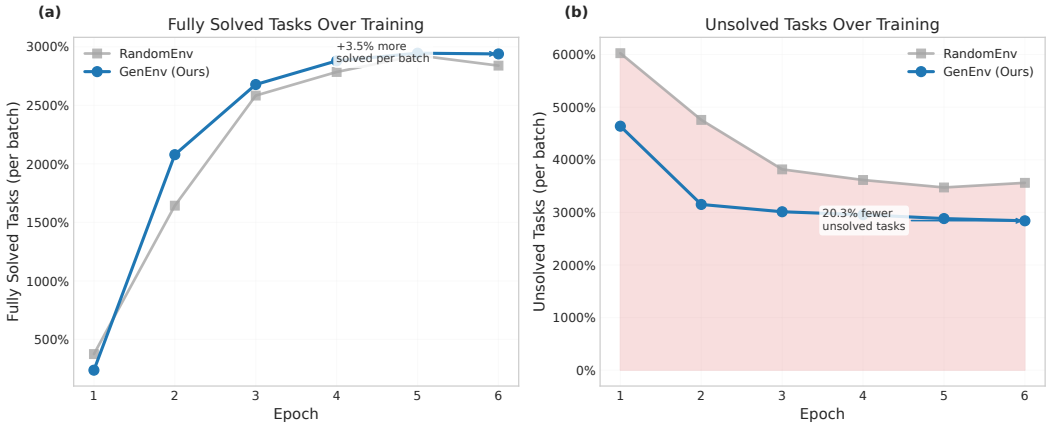


Figure 8: **Problem-solving behavior during training.** (a) GENENV consistently increases the proportion of fully solved tasks per batch, surpassing the RandomEnv variant; (b) the rate of unsolved tasks decreases substantially faster under GENENV. These trends show that difficulty-aligned simulation not only improves average performance but also accelerates the elimination of failure modes compared to unguided task generation.