Beyond Simply Environment Scaling: Designing Effective Environment Distributions for Multimodal Agent Learning

Anonymous ACL submission

Abstract

Recent works train agents by constructing large-scale multimodal environment pools. However, we find that simply increasing the number of multimodal environments does not always benefit. We further analyze the limitations in current multimodal environment distributions through a series of experiments. Based on these findings, we study how to build more effective training environment distributions from two dimensions: **diversity** and **difficulty structure**. For diversity, we propose **Ability-aware Environment Selection (AES)** to obtain diverse environment set. For difficulty structure, we propose **Hierarchical Difficulty Curriculum (HDC)**, which organizes curriculum learning through two difficulty levels: harness weakening and state-scale progression. Experiments show that AES and HDC effectively improve the training performance of multimodal agents.

1 Introduction

Recent advances in multimodal large language models (MLLMs) and agents are shifting the focus of model training from static datasets to dynamic environments (Shi et al., 2026). To improve different abilities of agents, a growing body of work constructs large-scale multimodal environments and trains agents jointly on them (Meng et al., 2026). Existing efforts on scaling multimodal environments mainly emphasize the quality of each individual environments (Wang et al., 2026b; Zhang et al., 2025b). They typically verify whether an environment can be executed (Wu et al., 2026) or provide reliable reward (He et al., 2025). We refer to this perspective as **sample-level quality**. Such checks are necessary for ensuring that each environment is usable. However, for agent training, the quality of individual environments is not sufficient. What an agent actually learns, and how well it generalizes, is also directly depended on the effectiveness of training environment **distribution**.

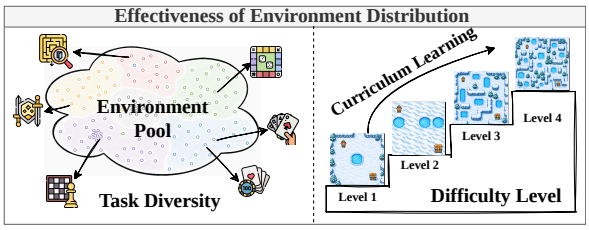

Figure 1: Environment distribution needs to be evaluated from two dimensions: diversity and difficulty level.

To analyze the limitations of existing environment distributions, we build a pool of 200 environments based on prior works (Wang et al., 2026b; Meng et al., 2026) and conduct a series of preliminary experiments in Section 2. First, the results in Section 2.1 show that as the number of training environment types increases, model performance may fluctuate substantially or even degrade. This suggests that more multimodal environment types are not necessarily better. Second, in Section 2.2, we find that this mixed-training failure is more severe in multimodal environments. For the same environment distribution, we construct both text-symbolic and multimodal versions for comparison. The results show that the multimodal version exhibits stronger negative transfer and gradient conflicts across environments, and is more likely to suffer performance degradation under mixed training. These preliminary findings indicate that training multimodal agents requires a deeper analysis of the effectiveness of training environment distributions, going beyond simple environment scaling.

Thus our research question is: **how can we evaluate and design effective training environment distributions for multimodal agents?** We argue that the effectiveness of environment distribution should be evaluated along two dimensions: *diversity* and *difficulty structure*, as shown in Figure 1. Diversity requires the environment set to cover broad abilities while avoiding redundancy and conflicts. Difficulty structure refers to whether the

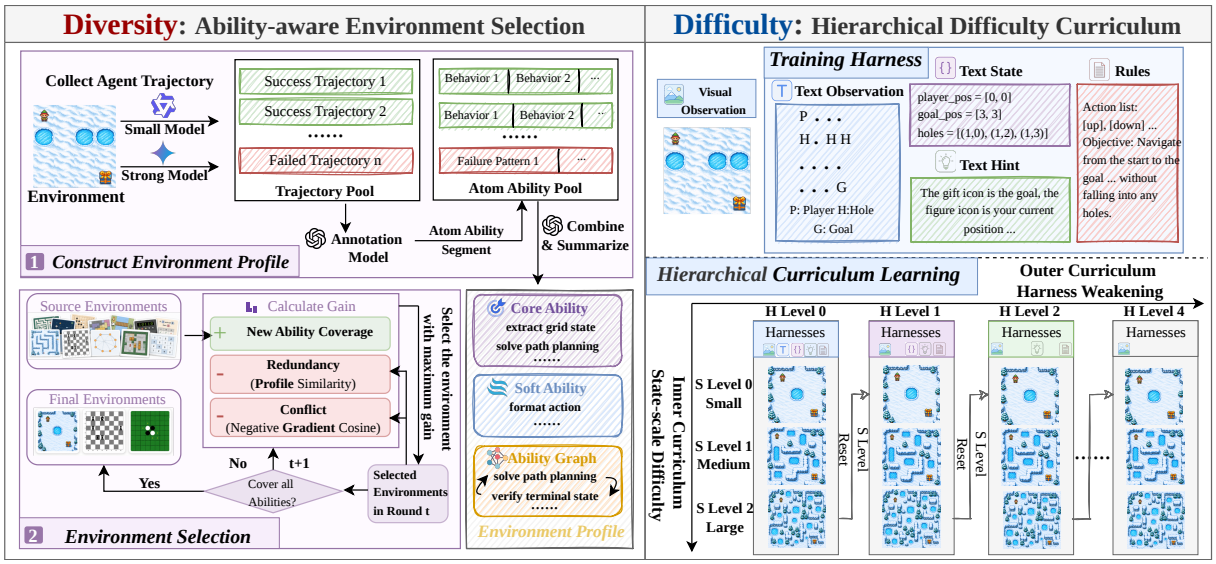

Figure 2: An overview of our methods for evaluating and designing effective environment distribution.

environment distribution provides a meaningful difficulty progression, so that the agent can improve continuously and stably. Based on this perspective, we analyze these two dimensions separately and propose corresponding methods. The overall framework of our method is illustrated in Figure 2.

(1) For diversity, our goal is to obtain an environment distribution with high coverage, low redundancy and optimization conflict. As discussed in Section 3, instead of measuring diversity through surface-level signals, such as task-description representations, as in prior works (Zhang et al., 2025b), we argue that the underlying diversity of environments should be analyzed from the perspective of agent behaviors and the abilities learned during training. We therefore analyze agent trajectories in different environments, decompose them into reusable atomic abilities, and construct a meta-ability profile for each environment. Based on these profiles, we propose **Ability-aware Environment Selection (AES)**. AES selects environments by maximizing the coverage of meta-abilities. Meanwhile, it reduces redundancy through profile similarity analysis and mitigates potential optimization conflicts through mechanistic gradient analysis.

(2) For difficulty structure, we further study multimodal-specific difficulty structure in Section 4. We first show that two major ability bottlenecks of multimodal agents are visual state extraction and world modeling, which is consistent with prior multimodal agent studies (Wang et al., 2026b; Meng et al., 2026). However, existing works often follow the same framework as text environments and mainly define difficulty structure by state scale, such as grid size. This does not directly target the unique bottlenecks of multimodal agents. To this end, we propose **Hierarchical Difficulty Curriculum (HDC)**, a multimodal difficulty curriculum. Our two-level difficulty structure consists of harness weakening and state-scale progression. Here, a harness refers to a type of training scaffold that provides additional textual auxiliary information to help the agent extract visual states and understand world dynamics. As shown on the right side of Figure 2, we design multiple harnesses and gradually remove these scaffolds to help the agent overcome the two major ability bottlenecks. Within each harness stage, we further apply conventional state-scale curriculum learning as the inner curriculum to improve training stability and generalization.

We evaluate our framework on multiple settings. The experiments show that the high-quality environment subset selected by **AES** can outperform training on the full environment pool while using fewer environments. In addition, **HDC** leads to more stable training and better performance than both direct training and conventional scale-only curriculum learning. Together, **AES** and **HDC** achieve a 143.2% average relative gain across different settings. These results highlight that designing effective environment distributions is an important and promising direction for multimodal agent research.

2 Preliminaries

We first collect and unify 200 multimodal environments based on prior works (Wang et al., 2026b; Meng et al., 2026). Then we conduct a series of controlled experiments on them to analyze the key

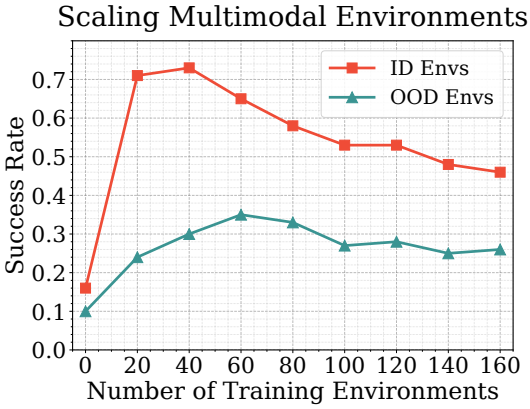

Figure 3: Effect of simply environment scaling.

Setting	Avg. Single	Avg. Mixed	Drop
Text-symbolic	70.4	69.5	-1.3%
Multimodal	48.4	43.2	-10.7%

Table 1: Results of mixed-environment training. Avg. Single denotes the average performance of separately trained single-environment models, while Avg. Mixed denotes the average performance of one model trained on the mixed environment set.

problems in current multimodal environments. Detailed settings are provided in Appendix B.

2.1 Naive Scaling Multimodal Environments Not Always Benefits

To train agents with diverse abilities, recent works have constructed large-scale multimodal environment collections. However, does simply scaling the number of environment types always benefits? To study this question, we train agents on our 200-environment pool. We gradually increase the number of environment types used for training and keep the total compute budget fixed. As shown in Figure 3, model performance does not monotonically improve as the number of environment types increases. Under some environment scales, the performance may even decrease. This indicates that simply scaling the number of multimodal environments is not always stable. It may introduce redundancy or conflicts. Therefore, we need to further analyze the distribution of environment set.

2.2 Multimodal Environments Are More Prone to Negative Transfer

To determine whether the mixed-training degradation simply comes from logic conflicts among environments, we conduct a controlled comparison. Specifically, we construct two versions of the same environment set: text-symbolic and multimodal. In the text-symbolic version, visual observations are converted into text observations, while other information remains unchanged. For each version, we train separate models on individual environments and also train one model on mixed-environment.

We first compare how mixed-environment training affects the two versions. As shown in Table 1, the text-symbolic version is more stable under mixed training. Compared with single-environment training, mixed training causes only a 1.3% drop in the text-symbolic setting, but a much larger 10.7% drop in the multimodal setting, indicating stronger negative transfer. Appendix E.1 provides the full transfer matrices, including cross-environment evaluation results of single-environment models. These results further show that multimodal environments are more prone to negative transfer.

We further analyze training conflicts from the gradient perspective. Given two environments, we compute their training gradients g_i and g_j, and use cosine similarity to measure the consistency of their update directions. A large $|\cos(g_i, g_j)|$ indicates strong coupling between the update directions, and negative values suggest conflict. As shown in Figure 4. Compared with the text-symbolic version, the multimodal version exhibits more polarized gradient similarities, including several strongly negative correlations. This indicates that even when the underlying task logic remains unchanged, the multimodal environments can amplify optimization conflicts between environments. Overall, these results suggest that mixed-environment training degradation is more pronounced in multimodal environments. This motivates a closer study of how to construct effective training environment distributions for multimodal agents.

2.3 Main Bottlenecks of Multimodal Agents

Finally, we analyze the main bottlenecks of models in multimodal environments. We examine 200 failure trajectories of Qwen3-VL-4B and conduct manual error analysis. As shown in Figure 5, we find that model failures mainly stem from two factors. The first is *visual state extraction*. Models often fail to reliably extract the environment state from visual observations. The second is *world modeling*. The model may fail to correctly understand environment rules and how actions affect the environment. These findings are consistent with recent studies (Wang et al., 2026b). However, these two bottlenecks cannot be solved simply by increasing the number of environments. Instead, it requires a difficult schedule tailored to the unique challenges

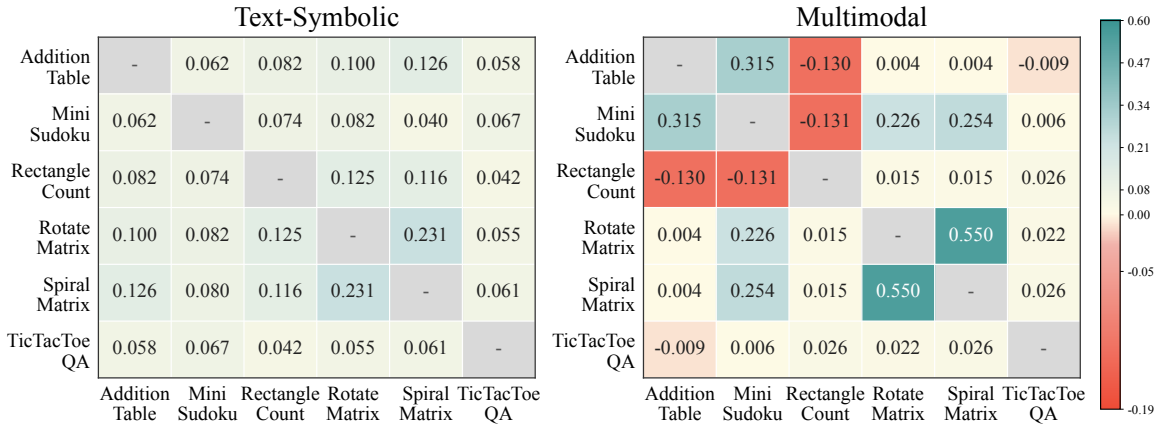

Figure 4: Gradient cosine similarity analysis between environments under different versions.

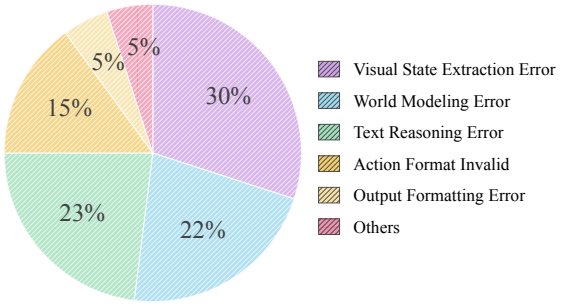

Figure 5: Error analysis of multimodal agents.

of multimodal agents to help the model overcome these limitations (Meng et al., 2026).

2.4 From Simply Environment Scaling to Effective Environment Distribution

The preliminary experiments show that the key to multimodal environments is not merely to construct **more "usable" environments**, but to construct a more **effective training environment distribution**. Therefore, we focus on how to design effective environment distributions for multimodal agents. We argue that the effectiveness of an environment distribution should be evaluated along at least two core dimensions: **diversity** and **difficulty structure**. In the following sections, we will study these two dimensions and introduce corresponding methods.

3 Diversity: Environmental Selection

The first key dimension is **diversity**. Existing works often measure environment diversity using surface representations such as task descriptions or environment code (Song et al., 2026; Wang et al., 2026a). However, these representations do not necessarily reflect the actual abilities that agents need to learn in the environments. Therefore, we characterize environment diversity from the perspective of *agent ability diversity*. The core idea is that the underlying diversity of environments should be analyzed through agent behaviors and the abilities required to solve the tasks. Based on this idea, we first decompose agent trajectories into a set of reusable *atomic abilities*, and then build a *meta-ability profile* for each environment. We further propose **Ability-aware Environment Selection (AES)**, which selects a compact and complementary environment subset based on meta-ability profiles and gradient-based conflict analysis. We describe this method in detail below. We show the complete pipeline on the left side of Figure 2.

3.1 Construct Meta-Ability Profiles

As shown in the upper-left panel of Figure 2, for each environment, we collect 40 agent trajectories: half from Qwen3-VL-4B and half from a stronger model, Gemini-3-Flash (Google DeepMind, 2025). Using models with different capability levels allows the trajectory pool to contain both successful strategies and failure patterns, which reveal the underlying ability requirements from complementary perspectives. We then use a strong annotation model, GPT-5 (OpenAI, 2025), to perform **atomic ability segmentation**. Given the environment information and model responses, GPT-5 decomposes each complete trajectory into a sequence of atomic abilities. Here, *atomic* does not refer to the lowest-level tokens, but to interpretable behavior units that are meaningful for solving the task, such as "identifying the target position". To control the segmentation granularity, we design a unified prompt with few-shot examples and manually inspect the outputs to filter low-quality segmentations, such as overly coarse labels like "perception". Detailed prompts are provided in Appendix F.

After obtaining atomic abilities from trajectories, we aggregate them to build a meta-ability

profile P_e for each environment e. We first use GPT-5 to merge semantically equivalent atomic abilities. Then, we compute the frequency of each meta-ability and the transition frequency between different meta-abilities, forming an ability graph. Low-frequency and unstable abilities are filtered out. Finally, each environment is represented by a **meta-ability profile**, which contains meta-abilities, transition edges, and their frequencies. We further divide meta-abilities into *core and soft abilities*. Core abilities appear stably in an environment and are directly related to task completion, while soft abilities appear occasionally or are only weakly related to solving the task. Through this process, we construct meta-ability profiles for all 200 environments and summarize 72 core meta-abilities.

3.2 Methodology: Ability-aware Environment Selection (AES)

As shown in the lower-left panel of Figure 2, after obtaining the meta-ability profile of each environment, we propose **Ability-aware Environment Selection (AES)**. We point out that the goal of diversity should be **broader core capability coverage, with less redundancy and conflict**.

For each candidate environment e, we define its coverage set $C(e)$, which is constructed from the core and soft abilities in its profile, where core abilities are assigned higher weights. Given the current selected environment set S, the new coverage brought by e is defined as:

$$\text{NewCoverage}(e, S) = \sum_{a \in C(e) \setminus C(S)} w_a,$$

where w_a denotes the weight of ability a, and $C(S)$ denotes the union of abilities already covered by the selected set S.

We also consider two factors that may influence joint training: **redundancy** and **conflict**. Redundancy is measured by the similarity between meta-ability profiles. If an environment is highly similar to the selected environments, it may provide limited new training information. We define:

$$\text{Redundancy}(e, S) = \max_{e' \in S} \text{sim}(P_e, P_{e'}),$$

where $\text{sim}(\cdot, \cdot)$ is the weighted profile similarity.

Conflict is estimated by the gradient between environments. As discussed in Section 2.2, when two environments have negative gradient cosine similarity, they are more likely to introduce optimization interference during joint training. Therefore, we define the conflict score as:

$$\text{Conflict}(e, S) = \max_{e' \in S} \max(0, -\cos(g_e, g_{e'})).$$

Finally, in each selection step, the gain of a candidate environment e is defined as:

$$\begin{aligned}\text{Gain}(e \mid S) = \lambda_1 &\text{NewCoverage}(e, S) \\ &- \lambda_2 \text{Redundancy}(e, S) \\ &- \lambda_3 \text{Conflict}(e, S).\end{aligned}$$

Here, $\lambda_1, 2, 3$ are hyperparameters that control the relative weights. The algorithm starts from an empty set. At each step, it selects the environment with the highest gain and adds it to S. This process is repeated until all core behaviors are covered.

Based on AES, we select a diverse subset of 30 environments from the original environment pool. Figure 6 shows the coverage curve as environments are added sequentially. The coverage of core meta-abilities increases steadily and eventually reaches full coverage, suggesting that AES can preserve broad ability coverage with only a small number of environments. In Appendix E.2, we further visualize the selected environments in profile space.

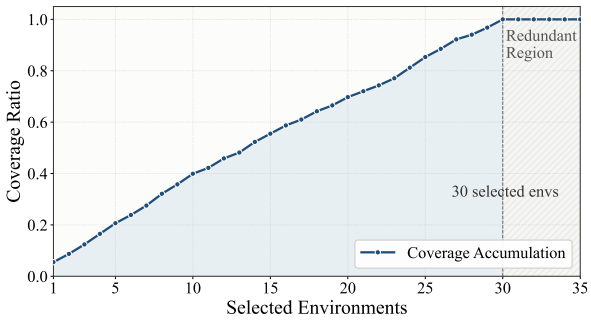

Figure 6: Core ability coverage curve when selecting environments using AES.

4 Difficulty: Curriculum Learning

The experiments in Section 2.3 show that the bottleneck of multimodal agents comes more from the insufficient ability to extract visual states and model environmental rules. Thus when only raw visual inputs are provided at the early training stage, the model often struggles to solve the tasks, leading to sparse RL reward signals and unstable training.

Based on this, we design several textual harnesses to help models overcome the bottlenecks. In this work, a *harness* refers to a type of *training scaffold*: auxiliary information provided by the environment to help the model extract states

Harness	Selected Envs.		Held-Out Envs.		Avg.	
	ST	MT	ST	MT	ST	MT
None	13.1	11.9	13.8	9.5	13.5	10.7
Text Obs.	28.2	18.7	30.6	17.1	29.4	17.9
Text State	**30.5**	**20.3**	**32.9**	**21.4**	**31.7**	**20.9**
Text Hint	20.7	15.1	24.6	13.6	22.7	14.4
Rule	18.8	16.5	20.8	15.7	19.8	16.1

Table 2: The effects of different textual harness information. ST and MT denote single-turn and multi-turn.

from visual observations and understand rules. We consider four types of harnesses: textual observations, text states, text hints, and rule descriptions. They provide, respectively, text-symbolic descriptions of visual observations, key environment states, hints about important visual content, and explicit task rules. Examples of different harness types are shown in Figure 2. We evaluate the effect of these harnesses in Table 2. With harness assistance, the model can solve more environments, which can provide more reliable training signals for RL.

We further use harnesses to design a two-dimensional difficulty schedule tailored to the bottlenecks of multimodal agents. Specifically, we propose **Hierarchical Difficulty Curriculum (HDC)**, which organizes training along two difficulty axes: **harness weakening** and **state-scale progression**. Harness weakening gradually removes auxiliary scaffolds, forcing the model to rely more on raw visual observations. *State-scale difficulty* refers to the scale complexity of an environment instance and is controlled by environment-specific state parameters such as grid size. Our curriculum follows a hierarchical structure. State-scale progression serves as the inner curriculum, gradually increasing instance complexity within each harness stage and Harness weakening serves as the outer curriculum. Next, we will introduce them separately.

4.1 Harness Weakening as Outer Curriculum

We first define outer curriculum axis based on harness and divide it into five difficulty levels, denoted as H_0 to H_4. H_0 corresponds to full harnesses. As the level increases, harness is gradually removed. Finally, H_4 retains only the basic information, including raw visual observations and task descriptions. Table 7 in Appendix summarizes the harnesses available at each level.

In practice, for each environment e, we maintain a current harness frontier: $r_e \in \{0, 1, 2, 3, 4\}$. During training, samples are not always generated from r_e. Instead, earlier harness levels can also be sampled, which avoids abrupt distribution shifts and mitigates forgetting. For each training instance, we sample its harness level from an environment-specific distribution D_e:

$$h \sim D_e(h \mid r_e).$$

This distribution assigns a probability p_{cur} to the current frontier harness r_e. The remaining probability mass $1 - p_{\text{cur}}$ is distributed over earlier harness levels $h < r_e$ according to an exponential decay:

$$w_h = \exp\big(-\alpha(r_e - h)\big),$$

$$D_e(h \mid r_e) = (1 - p_{\text{cur}})\frac{w_h}{\sum_{j<r_e} w_j}, \quad h < r_e.$$

4.2 State-Scale Difficulty as Inner Curriculum

Even under a strong harness, the agent may fail to obtain rewards if it is exposed too early to complex state (e.g. complex grids). Therefore we introduce a second curriculum axis, *state-scale difficulty*.

For each environment e, we define a set of state-scale difficulty levels: $s \in \{0, 1, \ldots, S_e\}$. The meaning of s is environment-specific (e.g. grid size). And a state-scale level corresponds to a range of environment parameters. Formally, for each environment e and scale level s, we define an environment-specific parameter distribution:

$$\theta \sim Q_e(\theta \mid s),$$

where θ denotes the parameters used to generate an environment instance.

During training, each environment independently maintains a current state-scale frontier u_e. Similar to harness sampling, we avoid training only on a single difficulty level. Instead, we sample s from a sliding window of size Δd:

$$s \sim \text{Uniform}\big(\{\ell_e, \ell_e + 1, \ldots, u_e\}\big),$$

where the lower bound of the sampling window is defined as $\ell_e = \max(0, u_e - \Delta d)$.

4.3 Methodology: Hierarchical Difficulty Curriculum (HDC)

We now describe the overall algorithm. For each environment e, we independently maintain its curriculum state as a tuple (r_e, ℓ_e, u_e), where r_e denotes the current harness frontier and $[\ell_e, u_e]$ denotes the current state-scale sampling window.

Each training sample is generated as follows. We first sample an environment, $e \sim \mathcal{E}_{\text{train}}$. Given

Model	Method	ID-Split	ID Envs.			OOD Envs.			General Benchmarks			
			ST	MT	Rel.	ST	MT	Rel.	MathVision	MMMU	MMStar	Rel.
4B	Base Model	Random	15.7	13.1	0.0	13.8	9.5	0.0	52.3	66.0	62.7	0.0
	Base Model	AES	13.1	11.9	0.0	13.8	9.5	0.0	52.3	66.0	62.7	0.0
	All Envs.	Full	25.4	19.8	80.1	17.6	8.3	7.5	51.6	66.1	**63.6**	0.1
	Random-K	Random	32.2	21.3	83.8	17.9	7.1	2.2	52.6	66.7	62.0	0.2
	Random-K + HDC	Random	37.3	29.4	131.0	18.7	9.0	15.1	52.0	65.6	63.9	0.2
	AES	AES	37.7	25.4	150.6	21.0	12.2	40.3	53.0	67.9	62.3	1.2
	AES + HDC	AES	**45.0**	**36.2**	**223.9**	**22.1**	**16.8**	**68.5**	**53.3**	**67.2**	63.1	**1.5**
8B	Base Model	Random	17.6	14.8	0.0	16.3	11.2	0.0	54.6	69.1	64.0	0.0
	Base Model	AES	16.0	13.3	0.0	16.3	11.2	0.0	54.6	69.1	64.0	0.0
	All Envs.	Full	29.7	20.5	69.9	19.6	12.5	15.9	**55.3**	68.6	**65.3**	**0.9**
	Random-K	Random	34.7	25.9	86.1	20.8	9.5	6.2	54.3	69.9	64.4	0.4
	Random-K + HDC	Random	43.0	33.2	143.4	22.3	10.3	14.4	55.6	68.8	64.7	0.8
	AES	AES	41.0	30.9	144.3	23.2	17.0	47.1	54.0	69.2	65.0	0.2
	AES + HDC	AES	**49.9**	**40.1**	**206.7**	**26.5**	**20.7**	**73.7**	54.9	**70.1**	63.5	0.4

Table 3: Main results with $K = 30$. ST denotes single-turn success rate, and MT denotes normalized return in multi-turn environments. ID-Split specifies the evaluation split for the Random and AES subsets. Rel. is the relative gain over the corresponding base model. Gray rows denote **our complete method** and show the best performance.

the current harness frontier r_e, we sample a harness level from $h \sim D_e(h \mid r_e)$, as described in Section 4.1. We then sample a state-scale level from the current window, $s \sim \mathrm{Uniform}(\{\ell_e, \ell_e + 1, \ldots, u_e\})$. Finally, we generate a concrete training instance according to the tuple (e, h, s) and optimize the policy. Importantly, environments do not need to share the same difficulty level; each environment maintains its own curriculum progress.

As shown in Figure 2, the curriculum update is also hierarchical. When the model reaches the scale advancement threshold τ_s, we advance the inner curriculum, $u_e \leftarrow u_e + 1$. When u_e reaches the target state-scale level target_e and the model satisfies the harness advancement threshold τ_h, we advance the outer curriculum, $r_e \leftarrow r_e + 1$, and then reset the state-scale. Therefore, **HDC** can be summarized as follows: within each harness level, we increase state-scale; after the model achieves sufficient competence, we weaken the harness and restart scale. Appendix D.1 and E.4 provides the pseudocode and training curve of HDC.

5 Experiments

We conducted extensive experiments to evaluate the effectiveness of our methods: AES and HDC.

5.1 Experimental Setup

We conduct the main experiments on Qwen3-VL-4B/8B-Instruct. As a control baseline, we construct **Random-K** by sampling 30 environments from the original 200-environment pool, with environments evenly sampled across human-annotated categories inherited from the original works (Wang et al., 2026b; Meng et al., 2026). We further evaluate the trained models on general multimodal benchmarks from different categories. Detailed experimental settings are provided in Appendix B.

5.2 Main Results

Effect of Diversity-Aware Environment Selection (AES). Training on the AES-selected environments achieves stronger environment performance than both Random-K and All Envs. Averaged over the ID/OOD environment groups and two model scales, AES obtains a 95.6% relative gain over the base model, compared with 43.4% for All Envs. This suggests that using more environments is not always better, since large environment pools may contain redundant or conflicting environments that reduce the efficiency of joint training. AES also outperforms Random-K, which only achieves a 44.6% relative gain. This indicates that human-defined categories is less effective than AES which selects environments from the model's ability perspective. Meanwhile, AES shows good generalization and robustness. Although the selected environments are constructed using trajectory and gradient information from Qwen3-VL-4B, they remain effective for Qwen3-VL-8B, where AES achieves a 144.3% relative gain on ID environments and a 47.1% relative gain on OOD environments. Appendix E.3 further illustrates the comparison between AES and existing diversity methods.

Method	#Env	OOD-ST	OOD-MT	Rel.
Base Model	–	13.8	9.5	0.0
Random-K	30	17.9	7.1	2.2
AES w/o Redund.	41	18.6	11.0	25.3
AES w/o Conflict	34	18.5	6.8	2.8
AES	**30**	**21.0**	**12.2**	**40.3**

Table 4: Results of ablation experiments on AES.

Method	ID Ens.		OOD Ens.		Rel.
	ST	MT	ST	MT	
Base Model	13.1	11.9	13.8	9.5	–
AES	37.7	25.4	21.0	12.2	0.0
AES + Scale-only	40.7	29.8	21.6	14.1	11.5
AES + Harness-only	41.9	33.1	21.2	15.3	18.1
AES + HDC	**45.0**	**36.2**	**22.1**	**16.8**	**27.7**

Table 5: Results of ablation experiments on HDC.

Effect of Difficulty-Aware Curriculum Learning (HDC). HDC further improves performance across different environment sets. The strongest result is achieved by combining HDC with AES, where AES + HDC obtains an average relative gain of 143.2% over the base model across the ID/OOD environment groups and two model scales. This shows that diversity and difficulty are not only two important dimensions of environment distribution, but can also complement each other to further improve training effectiveness. HDC also improves Random-K, increasing its average relative gain from 44.6% to 73.7%. This suggests that appropriate difficulty scheduling can partially mitigate the instability caused by heterogeneous multimodal environments. By starting from easier instances with stronger harnesses, the model can obtain more reliable learning signals, thereby alleviating imbalance among environments to some extent.

5.3 Ablation Study

Ablation on AES. AES consists of three key factors: meta-ability coverage, redundancy and conflict control. To examine the effects of redundancy and conflict control, we ablate these two components separately. All variants are trained without HDC. As shown in Table 4, the full AES achieves the largest relative gain 40.3% over the base model on OOD environments. Removing redundancy control reduces the relative gain from 40.3% to 25.3%, while removing conflict control reduces it to only 2.8%. These results show that both components are important for our AES method.

Ablation on HDC. We ablate the two axes of HDC: harness weakening and state-scale progression. All variants use the same AES-selected environment set. As shown in Table 5, both single-axis curricula benefit. The scale-only curriculum brings an average relative gain of 11.5%, while the harness-only curriculum brings a larger gain of 18.1%. Combining both axes achieves the best performance, with an average relative gain of 27.7%. These results show that harness weakening and state-scale progression are complementary and jointly improve multimodal environment training.

6 Related Works

We summarize the research most relevant to our core ideas, and provide more discussions in Appendix A. Recently, multimodal environmental and agent learning have become cutting-edge research areas (Shi et al., 2026). Among them, we mainly focused on general multimodal environments (Meng et al., 2026; Wang et al., 2026b). Another important line of work studies environment quality verification (Jain et al., 2025). They focused on whether environments are executable (Pan et al., 2025), whether state transitions are correct and so on (Hu et al., 2025; Wu et al., 2026). These approaches mainly address sample-level quality. However, how to assess the effectiveness of environment distribution remains less systematically studied (Huang et al., 2025). In particular, for multimodal environments, the diversity and difficulty structure remain insufficiently explored. Current diversity control methods often rely on task-description embeddings (Zhang et al., 2025b) or tool categories (Li et al., 2023). Meanwhile, difficulty control in multimodal environments largely follows the framework of text-based environments (Zeng et al., 2025; Verma et al., 2025; Ying et al., 2026). To bridge this gap, we analyze diversity and difficulty separately and propose corresponding methods.

7 Conclusion

In this paper, we show that simply scaling multimodal environments is not always effective for agent training. Our analyses reveal that the effectiveness of an environment distribution depends on both diversity and difficulty structure. Based on this insight, we propose Ability-aware Environment Selection (AES) to construct diverse environment set, and Hierarchical Difficulty Curriculum (HDC) specific to multimodal agents. Experiments demonstrate that AES and HDC consistently improve agent training.

Limitations

Although we have conducted extensive experiments and analyses, there are still some aspect that need improvement. First, due to cost constraints, our environment pool is mainly built upon existing multimodal environment works, and we do not further study large-scale environment synthesis. This remains an important direction for future exploration. Second, due to limited computational resources, we mainly compare different methods under a unified compute budget. Since multimodal environment training, especially multi-turn training, is computationally expensive, some environments may not be fully trained. Future work can further validate our findings under larger training budgets. Finally, AES uses gradient information to characterize optimization conflicts between environments, which introduces additional offline computation compared with traditional diversity measures. Future work may explore more efficient conflict estimation methods or alternative metrics.

Ethics and Artifact Use Statement

Potential risks. This work studies how to design effective training environment distributions for multimodal agent learning. A potential risk is that environment-based training may improve agents' ability of decision making in interactive tasks. If such models are deployed in open real-world systems without sufficient safeguards, they may produce incorrect actions or other unintended effects. Therefore, the models trained in this work are evaluated only in controlled and verifiable multimodal research environments, and should not be directly deployed in open real-world systems without proper safety treatment.

Artifacts, licenses, and intended use. Our environment pool is mainly built upon existing multimodal environment works and is used only for research purposes. We properly cite all assets and adhere to their licenses and terms of use.

Data privacy and content safety. The environments used in this work are virtual multimodal interactive environments and do not involve collecting personal data from real users.

Use of AI assistants. First, in our method, LLMs were used for atomic ability segmentation and environment profile construction with manual inspection by the authors. Second, during writing, LLMs were used to check grammar and polish text, and we carefully verified that their use did not alter the original meaning. In addition, AI assistants were used to improve the presentation of some tables and figures, while all experimental results were checked by the authors and remained unchanged.

References

Pranjal Aggarwal, Graham Neubig, and Sean Welleck. 2026. Gym-anything: Turn any software into an agent environment. *CoRR*, abs/2604.06126.

Muhammad Awais Ahmad, Tauqir Ahmed, Muhammad Aslam, Amjad Rehman, Faten S. Alamri, Saeed Ali Bahaj, and Tanzila Saba. 2025. Mathvision: An accessible intelligent agent for visually impaired people to understand mathematical equations. *IEEE Access*, 13:6155–6165.

Shihao Cai, Runnan Fang, Jialong Wu, Baixuan Li, Xinyu Wang, Yong Jiang, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zhen Zhang, Fuli Feng, Pengjun Xie, and Xiaobin Wang. 2025. Autoforge: Automated environment synthesis for agentic reinforcement learning. *CoRR*, abs/2512.22857.

Yilin Cao, Yufeng Zhong, Zhixiong Zeng, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, Wenji Mao, and Wan Guanglu. 2026. Mobiledreamer: Generative sketch world model for gui agent. *arXiv preprint arXiv:2601.04035*.

Delong Chen, Théo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. 2025. Planning with reasoning using vision language world model. *CoRR*, abs/2509.02722.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. 2024b. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

ARC Foundation. 2026. Arc-agi-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*.

Google DeepMind. 2025. Gemini 3.1 pro. Accessed: 2026-03-16.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nat.*, 645(8081):633–638.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, and 1 others. 2025. Matrix-game 2.0: An open-source real-time and streaming interactive world model. *arXiv preprint arXiv:2508.13009*.

Mengkang Hu, Bowei Xia, Yuran Wu, Ailing Yu, Yude Zou, Qiguang Chen, Shijian Wang, Jiarui Jin, Kexin Li, Wenxiang Jiao, Yuan Lu, and Ping Luo. 2025. Agent2world: Learning to generate symbolic world models via adaptive multi-agent feedback. *CoRR*, abs/2512.22336.

Yuchen Huang, Sijia Li, Wei Liu, Zhiyuan Fan, Yi R Fung, and 1 others. 2025. Scaling environments for llm agents in the era of learning from interaction: A survey. In *Workshop on Scaling Environments for Agents*.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. 2025. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *CoRR*, abs/2504.07164.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, and 1 others. 2025. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*.

Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. 2025. Osworld-mcp: Benchmarking MCP tool invocation in computer-use agents. *CoRR*, abs/2510.24563.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. 2025. Vcrl: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*.

Dinging Li, Yingxiu Zhao, Xinrui Cheng, Kangheng Lin, Hongbo Peng, Hongxing Li, Zixuan Wang, Yuhong Dai, Haodong Li, Jia Wang, and 1 others. 2026a. Spatialevo: Self-evolving spatial intelligence via deterministic geometric environments. *arXiv preprint arXiv:2604.14144*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3102–3116.

Renda Li, Hailang Huang, Fei Wei, Feng Xiong, Yong Wang, and Xiangxiang Chu. 2026b. Adacurl: Adaptive curriculum reinforcement learning with invalid sample mitigation and historical revisiting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23123–23131.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. 2025. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704*.

Fanqing Meng, Lingxiao Du, Jiawei Gu, Jiaqi Liao, Linjie Li, Zijian Wu, Xiangyan Liu, Ziqi Zhao, Mengkang Hu, Yue Zhang, Zichen Liu, Jiaheng Zhang, and Michael Qizhe Shieh. 2026. Gym-v: A unified vision environment system for agentic vision research. *CoRR*, abs/2603.15432.

OpenAI. 2025. Introducing gpt-5.4. Accessed: 2026-03-16.

Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. 2026. Gameworld: Towards standardized and verifiable evaluation of multimodal game agents. *CoRR*, abs/2604.07429.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. 2025. Training software engineering agents and verifiers with swe-gym. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and 1 others. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, and 1 others. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pages 79791–79821.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.

Chengshuai Shi, Wenzhe Li, Xinran Liang, Yizhou Lu, Wenjia Yang, Ruirong Feng, Seth Karten, Ziran Yang, Zihan Ding, Gabriel Sarch, and 1 others.

2026. Odysseus: Scaling vlms to 100+ turn decision-making in games via reinforcement learning. *arXiv preprint arXiv:2605.00347*.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2026. Envscaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis. *CoRR*, abs/2601.05808.

Shuang Sun, Huatong Song, Lisheng Huang, Jinhao Jiang, Ran Le, Zhihao Lv, Zongchao Chen, Yiwen Hu, Wenyang Luo, Wayne Xin Zhao, Yang Song, Hongteng Xu, Tao Zhang, and Ji-Rong Wen. 2026. Swe-world: Building software engineering agents in docker-free environments. *CoRR*, abs/2602.03419.

Qwen Team. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Vivek Verma, David Huang, William Chen, Daniel Klein, and Nicholas Tomlin. 2025. Measuring general intelligence with generated games. *CoRR*, abs/2505.07215.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. 2025. VAGEN: reinforcing world model reasoning for multi-turn VLM agents. *CoRR*, abs/2510.16907.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. 2026a. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *CoRR*, abs/2602.10090.

Zirui Wang, Junyi Zhang, Jiaxin Ge, Long Lian, Letian Fu, Lisa Dunlap, Ken Goldberg, Xudong Wang, Ion Stoica, David M. Chan, Sewon Min, and Joseph E. Gonzalez. 2026b. Visgym: Diverse, customizable, scalable environments for multimodal agents. *CoRR*, abs/2601.16973.

Yifan Wu, Yiran Peng, Yiyu Chen, Jianhao Ruan, Zijie Zhuang, Cheng Yang, Jiayi Zhang, Man Chen, Yenchi Tseng, Zhaoyang Yu, Liang Chen, Yuyao Zhai, Bang Liu, Chenglin Wu, and Yuyu Luo. 2026. Autowebworld: Synthesizing infinite verifiable web environments via finite state machines. *CoRR*, abs/2602.14296.

John Yang, Kilian Leret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. 2025. Swe-smith: Scaling data for software engineering agents. *CoRR*, abs/2504.21798.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854.

Lance Ying, Ryan Truong, Prafull Sharma, Kaiya Ivy Zhao, Nathan Cloos, Kelsey R. Allen, Thomas L. Griffiths, Katherine M. Collins, José Hernández-Orallo, Phillip Isola, Samuel J. Gershman, and Joshua B. Tenenbaum. 2026. Ai gamestore: Scalable, open-ended evaluation of machine general intelligence with human games. *Preprint*, arXiv:2602.17594.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, and 3 others. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 9556–9567. IEEE.

Yucheng Zeng, Weipeng Lu, Linyun Liu, Shupeng Li, Zitian Qu, Chenghao Zhu, Shaofei Li, Zhengdong Tan, Mengyue Liu, Haotian Zhao, and 1 others. 2026. Logigen: Logic-driven generation of verifiable agentic tasks. *arXiv preprint arXiv:2603.00540*.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, Chenyang Zhao, Yulia Tsvetkov, Simon Shaolei Du, Natasha Jaques, Hao Peng, Pang Wei Koh, and Hannaneh Hajishirzi. 2025. RLVE: scaling up reinforcement learning for language models with adaptive verifiable environments. *CoRR*, abs/2511.07317.

Dazhi Zhan, Xin Liu, Wei Bai, Wei Li, Shize Guo, and Zhisong Pan. 2025. GAME-RL: generating adversarial malware examples against API call based detection via reinforcement learning. *IEEE Trans. Dependable Secur. Comput.*, 22(5):5431–5447.

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Lu Qianchun. 2025a. Learning like humans: Advancing llm reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6630–6644.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2024. Lmms-eval: Reality check on the evaluation of large multimodal models. *Preprint*, arXiv:2407.12772.

Ruizhi Zhang, Ye Huang, Yuangang Pan, Chuanfu Shen, Zhilin Liu, Ting Xie, Wen Li, and Lixin Duan. 2026a. Pokegym: A visually-driven long-horizon benchmark for vision-language models. *CoRR*, abs/2604.08340.

Wei Zhang, Jack Yang, Renshuai Tao, Lingzheng Chai, Shawn Guo, Jiajun Wu, Xiaoming Chen, Ganqu Cui, Ning Ding, Xander Xu, Hu Wei, and Bowen Zhou.

2025b. V-gamegym: Visual game generation for code large language models. *CoRR*, abs/2509.20136.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. 2026b. Infiniteweb: Scalable web environment synthesis for GUI agent training. *CoRR*, abs/2601.04126.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

A More Related Works

We discuss the works most closely related to our main thread in Section 6. In this appendix, we provide a detailed review of several related areas.

A.1 Multimodal Environments

Recent research on multimodal agents has gradually shifted from static datasets to dynamic interactive environments (Wu et al., 2026; Li et al., 2026a). Existing studies extend multimodal environments from different perspectives. Some works focus on scalable construction of Web and GUI environments (Zhang et al., 2026b; Aggarwal et al., 2026). Other works further use game environments for reinforcement learning of VLM agents, emphasizing verifiable feedback, long-horizon interaction, and world-model reasoning (Zhan et al., 2025; Wang et al., 2025). Meanwhile, some works mainly focus on evaluation. Several studies build standardized benchmarks for multimodal game agents (Ouyang et al., 2026; Zhang et al., 2026a; Foundation, 2026; Shi et al., 2026). Most related to our work are more general multimodal environment systems such as Gym-V (Meng et al., 2026) and VisGym (Wang et al., 2026b), which unify diverse visual tasks into gym-style or multi-turn interaction frameworks and provide a foundation for studying multimodal environment scaling.

A.2 Sample-Level Environment Quality

Another line of work examines the quality of constructed environments, typically focusing on whether each individual environment is executable, has valid state transitions, and provides reliable reward signals (Yang et al., 2025; Jain et al., 2025). Common practices include using sandbox execution and unit tests to verify whether a task can run correctly (Pan et al., 2025; Zhang et al., 2025b). Many works further validate the correctness of state transitions and task solvability with golden trajectories or expert tool sequences (Hu et al., 2025; Cai et al., 2025). Similarly, for multimodal environments where images or videos are difficult to test directly with unit tests, metric such as interaction consistency is often used as a correctness check (Jang et al., 2025; He et al., 2025). GUI and Web environments also commonly combine execution-based evaluators with expert review (Wu et al., 2026; Yang et al., 2023). These methods provide important guarantees for the executability, solvability, and evaluation reliability of individual environments. However, they mainly answer whether a single environment is valid. In contrast, our work further studies the distribution-level quality of an environment pool when it is used as a training distribution.

A.3 Environment Diversity

A key aspect of distribution-level quality is diversity (Sun et al., 2026). Existing works usually control diversity in two ways. The first line removes semantically similar tasks using information such as embedding similarity (Wang et al., 2026a). For example some works reduce repeated samples through embedding-based deduplication, topic dispersion analysis, or clustering of game seeds (Wang et al., 2026a; Song et al., 2026; Zhang et al., 2025b). The second line controls diversity through structured construction. Some works analyze diversity from the perspectives of tool categories and tool combinations (Li et al., 2023; Cai et al., 2025), while Web and GUI environments often judge by the coverage of application scenarios (Deng et al., 2023; Zhou et al., 2024; Luo et al., 2025). However, these methods mostly measure surface-level diversity, such as task descriptions or domain sources, and may not reflect the diversity of abilities that the model actually learns during training. In contrast, our work studies the diversity of underlying logic and dynamics of the environment through agent behavior and gradient-level mechanism analysis.

A.4 Difficulty Grading and Curriculum Learning

Difficulty grading and scheduling within an environment set are also crucial to training-set quality. Existing works often characterize task complexity using structural parameters, such as the number of tools, state entities, or code scale (Jia et al., 2025; Zeng et al., 2026). Some studies calibrate difficulty

with strong models or human experts (Verma et al., 2025; Ying et al., 2026). For long-horizon planning tasks, difficulty is also adjusted by planning horizon or the number of search iterations (Cao et al., 2026; Chen et al., 2025).

Several works further use difficulty schedules for curriculum learning. Early curriculum learning methods often rely on predefined stages or static easy-to-hard ordering (Parashar et al., 2025). Recent RL methods for LLMs further emphasize the dynamic nature of difficulty. RLVE (Zeng et al., 2025) adaptively adjusts the difficulty of problems in verifiable environments based on model performance. ADCL (Zhang et al., 2025a) periodically re-estimates sample difficulty to mitigate difficulty shift during training. VCRL (Jiang et al., 2025) selects samples with strong learning signals for the current model using group reward variance for rollouts. AdaCuRL (Li et al., 2026b) performs curriculum learning with coarse-to-fine difficulty estimation. WebRL (Qi et al., 2025) constructs an online self-evolving curriculum by generating tasks from failed trajectories.

Existing difficulty grading for multimodal environments is also often based on environment scale (Wang et al., 2026b; Meng et al., 2026). However, the difficulty of multimodal environments does not only come from task scale, but also from multimodal-specific ability bottlenecks. Prior works provide limited systematic analysis of difficulty grading and curriculum learning tailored to these multimodal-specific factors.

B Experiment Details

This section describes the experimental settings in detail, including training and evaluation. We first introduce the settings shared by all experiments, and then describe the specific settings for each group of experiments.

B.1 General Settings

Training. We use Qwen3-VL-4B-Instruct (Team, 2025) as the base model for training. In the main experiments, we further include Qwen3-VL-8B-Instruct to evaluate whether our method remains effective with a larger model. To ensure a fair comparison, we follow the setting of RLVE (Zeng et al., 2025) and keep the compute budget the same across different training runs. Specifically, all runs use the same total number of training samples, which is set to 7,680 in our experiments. We use GRPO as the training algorithm (Guo et al., 2025), and set the rollout group size for each prompt to 16. Our reinforcement learning framework is modified from VeRL (Sheng et al., 2025) and follow its default implementation. We thank the authors for their excellent open source work. All training experiments are conducted on 4 A100 GPUs (80G).

Evaluation. The main evaluation in this paper is conducted on multimodal environments, with the goal of measuring both model performance and generalization. We consider two evaluation settings: in-distribution (ID) and out-of-distribution (OOD). ID evaluation uses unseen instances generated from environment types that appear during training. The specific set of ID environments may vary across experiments. OOD evaluation uses 30 held-out environments randomly selected from our pool of 200 environments. These 30 environments are never used in any training experiment in this paper. Therefore, all OOD results refer to evaluation on this same held-out set. For each environment type, we generate 100 instances for evaluation. Our evaluation scripts for multimodal environments are based on the implementation of Gym-V (Meng et al., 2026). We thank the authors for their excellent open source work.

B.2 Settings for Preliminary Experiments

For the environment scaling experiment in Section 2.1, we conduct all experiments on Qwen3-VL-4B-Instruct and keep the compute budget fixed. The environment sets are nested across different scales, meaning that a larger set always contains the smaller set as a subset. In this experiment, the ID evaluation set is defined as the smallest subset of 20 environments, so that all trained models are evaluated on the same group of ID environments.

For the gradient conflict analysis in Section 2.2, we sample 10 batches from each environment. Each batch contains 10 samples, and each prompt uses 16 rollouts. We then compute the gradient direction for each batch.

For the error analysis in Section 2.3, two human annotators independently label the model errors. The annotations are then cross checked, and disagreements are resolved to obtain the final labels.

B.3 Settings for Main Experiments

Training. In the main experiments, we further include Qwen3-VL-8B-Instruct to test whether our method is still effective at a larger scale. We use the same compute budget for all methods. Note

that the All Envs setting trains on the pool of 170 environments, after excluding the 30 held-out environments. This setting is used to analyze whether naive scaling of the environment pool is effective. It should also be noted that among the 30 environments in the AES-selected set, there are 22 single-turn and 8 multi-turn environments. Among the 30 OOD environments in the held-out set, there are 20 single-turn and 10 multi-turn environments.

Evaluation. For the main environment evaluation, ID has slightly different meanings for different methods. For all settings except Random-K, ID refers to the 30 environments selected by our environment selection method in Section 3.2. For Random-K, ID refers to the 30 randomly selected training environments. In addition, we evaluate the trained models on common general multimodal benchmarks to examine whether training on interactive environments harms their general perception and reasoning ability. We use three benchmarks of different types: MathVision (Ahmad et al., 2025), MMMU (Yue et al., 2024) with the validation split, and MMStar (Chen et al., 2024b). The evaluation scripts for these general multimodal benchmarks are based on lmms-eval (Zhang et al., 2024). We thank the authors for their excellent open source implementation.

C Details of Ability-aware Environment Selection (AES)

C.1 Atomic Ability Segmentation

We first use Qwen3-VL-4B-Instruct and Gemini-3-Flash to generate 20 trajectories for each environment. In practice, we sample 5 instances for each environment type, and generate 4 trajectories for each instance with each model.

For atomic ability segmentation, we use GPT-5 as the segmentation model. The prompt is shown in Appendix F. The goal is to decompose a complete agent trajectory into reusable and interpretable behavior units. When defining the segmentation granularity, we first rely on the model's understanding, and then manually inspect the results to slightly adjust the granularity.

C.2 Construction of Environment Meta Ability Profiles

After obtaining atomic ability segmentations from 40 trajectories for each environment, we merge these atomic abilities into environment level meta abilities. The resulting meta ability profile is used to represent the underlying logic of environment.

During merging, we first use GPT-5 to combine abilities with the same meaning but slightly different wording. We then filter out atomic abilities whose frequency is lower than 10%, treating them as noise. For the remaining abilities, we use GPT-5 to classify them into core abilities and soft abilities. Core abilities refer to abilities that are essential for solving the task, such as path planning, while soft abilities refer to auxiliary abilities, such as answer formatting. These two types of abilities are assigned different weights in later diversity based selection 3.2. We also record transition edges between abilities as ability graph information. Finally, we aggregate core abilities, soft abilities, ability graph edges, and their frequencies into the meta ability profile of each environment.

C.3 Settings of the AES Method

Our selection method computes the utility of each candidate environment based on three factors: covering as many core abilities as possible, reducing redundancy, and reducing conflict. We set the weights of these three terms, denoted by λ, to 1 in our current experiments.

For redundancy, we use the maximum profile similarity between the candidate environment and the already selected environments as the redundancy penalty. To compute cosine similarity, we embed core abilities, soft abilities, and ability graph edges. These three components are weighted differently. The weight of a core ability vector is its frequency, while the weights of soft ability vectors and edge vectors are 0.3 times their frequencies. This is because soft abilities should not dominate the redundancy estimate. For example, two completely different environments may both require the ability of answer formatting. Finally, we measure conflict using gradients, following the same implementation as described in Appendix B.2.

Table 6 lists the 30 high quality environments selected by AES. These environments are selected to achieve broad coverage, low redundancy, and low conflict. The table also provides a brief task description for each environment.

Environment	Task Description
DoorKey	In a first-person partial-view room, find the key, unlock the door, pass through, and reach the green goal.
Tents-QA	Visual QA on Tents and Trees: place tents next to trees while satisfying row/column counts and non-adjacency constraints.
PegJump	Jump one peg over another to remove it; goal is to finish with exactly one peg remaining.
Arc1D	Infer the 1D input-output transformation rule from examples and predict the output for the test input.
LavaGap	Cross a lava gap and reach the goal in first-person view without stepping on lava.
Minimum Spanning Tree Counting	Count the number of distinct minimum spanning trees in a weighted undirected graph, modulo a given value.
Othello	Two-player Othello: place discs to flip opponent pieces; win by having more pieces when the game ends.
Frozen Lake	Navigate from start to goal on a slippery frozen-lake grid without falling into holes.
Multi Room	Explore multiple connected rooms in first-person view; use keys and doors to reach the goal.
Mini Sudoku	Fill a Mini Sudoku so each row, column, and block contains each digit exactly once.
Tree Even Partitioning	Partition a tree with $N \times K$ vertices into N connected groups of K vertices each.
Sum Triangle Area	Compute the sum of areas of all triangles formed by given 2D points, where collinear triples count as 0.
Matchstick Equation	Fix a broken matchstick equation by moving one match per action until the equation is mathematically correct.
Knight Swap	Determine whether white and black knights can swap positions via legal L-moves; output the move sequence or "impossible".
Texas Holdem	Two-player limit Texas Hold'em: form the best 5-card hand from hole and community cards; win via call, raise, fold, or check.
Hue-QA	Visual QA on a color-gradient board about cell colors, gradient patterns, and color matching.
Circuit Logic	Given input assignments and a logic-circuit diagram, compute the circuit output, either 0 or 1.
Weighted Binarytree	Build a binary tree with fixed inorder traversal $0, \ldots, N-1$ that maximizes the given recursive score function.
Nine Puzzle	Transform the start grid into the target grid using bounded cyclic row/column shifts.
Math Path	Find a path from S to G on a number grid whose visited cells sum to the target value.
Minimum Chromatic Number	Find the minimum number of colors needed to properly color the vertices of an undirected graph.
Tree To Traversal	Read a binary tree visualization and output preorder, inorder, and postorder traversals.
Chess Ranger-QA	Visual QA on Chess Ranger: capture-only chess puzzles aiming to leave exactly one piece on the board.
Klo Blocks	Redistribute values via adjacent ± 1 operations to maximize the longest contiguous subarray with all values at least K.

Continued on next page

Table 6 continued from previous page.

Environment	**Task Description**
Rush Hour	Slide cars to free the red car [X] and drive it out through the right edge.
Largest Island	Find the maximum area of a 4-connected land island in a binary grid; return 0 if none exists.
Grid Local Minimum Counting	Count valid numberings of 1 to $N \times M$ such that marked cells are exactly the 8-neighbor local minima.
Binary Matrix	For each cell in a binary matrix, compute the Manhattan distance to the nearest 0.
Ultra TicTacToe-QA	Visual QA on Ultimate Tic-Tac-Toe, consisting of nine nested 3×3 boards, about rules, moves, or scoring.
Ska Rock Garden	Choose coordinate swaps on points to minimize the axis-aligned bounding-box perimeter, with total swap cost used as a tie-breaker.

Table 6: Selected environments and task descriptions.

Algorithm 1 Hierarchical Difficulty Curriculum

Require: Training environments $\mathcal{E}_{\text{train}}$
Require: Harness sample distribution D_e and scale range $[\ell_e, u_e^\star]$, $\ell_e = 0$
Require: Thresholds τ_{scale} and τ_{harness}

```
Initialize r_e = 0 and u_e = ℓ_e for each environment e ∈ E_train
for each training step do
    Sample e ~ E_train
    Sample h ~ D_e(h | r_e)
    Sample s ~ Uniform({ℓ_e, ..., u_e})
    Generate an instance with (e, h, s)
    Update the policy with RL
    Estimate recent performance p_e on scale u_e
    if p_e > τ_scale then
        u_e ← min(u_e + 1, u_e^⋆)
    end if
    if u_e = u_e^⋆ and p_e > τ_harness then
        r_e ← r_e + 1; reset u_e ← ℓ_e
    end if
end for
```

D Details of Hierarchical Difficulty Curriculum (HDC)

D.1 Pseudocode of HDC

This section provides the pseudocode of the Nested Harness-Annealed Curriculum in Algorithm 1 to facilitate understanding of our curriculum design. The core idea is that each environment maintains its own curriculum state, including the current harness frontier r_e and the state-scale difficulty frontier u_e. During training, we first sample an environment, then sample a harness level and a scale level, and finally generate a concrete training instance.

As described in Section 4.3, this curriculum is nested. Within each harness level, the model first progresses along the state-scale difficulty axis. Once the target scale is reached, we weaken the harness, advance the outer curriculum frontier, and restart the growth of state-scale difficulty.

D.2 Harness Level

This section presents detailed summarization of the harness level discussed in Section 4.1. Here, H_0 represents the strongest harness, where the model can simultaneously access textual observations, text state, text hints, and complete rule descriptions. As the level increases, auxiliary information is gradually removed. Finally, H_4 retains only basic information such as visual observations and task descriptions.

Level	Text Obs.	Text State	Text Hint	Rule	Basic Info.
H_0	✓	✓	✓	✓	✓
H_1	–	✓	✓	✓	✓
H_2	–	–	✓	✓	✓
H_3	–	–	–	✓	✓
H_4	–	–	–	–	✓

Table 7: Harness levels used in the outer curriculum.

D.3 State-Scale Implementation Details

This section describes the implementation details of the inner curriculum. In our method, the inner curriculum progresses according to state-scale difficulty levels, such as graph size, grid size, obstacle density, or the number of entities. Since different environments support different state-scale parameters, Table 8 summarizes the adjustable state-scale factors for the 30 selected environments. The progression of the inner curriculum follows the implementation strategy of RLVE (Zeng et al., 2025).

Environment	Supported	Scale Parameters
DoorKey	Yes	`size`: map size; e.g., `size=5/6/8/16`.
Tents-QA	Yes	`grid_size`; `num_trees`; e.g., `grid_size=[8,8]`, `num_trees=12`.
PegJump	No	–
Arc1D	Yes	`min_size/max_size`: sequence length; `num_train`: number of examples; e.g., `min_size=10`, `max_size=30`, `num_train=3`.
LavaGap	Yes	`size`: map size; e.g., `size=5/6/7`.
Minimum Spanning Tree Counting	Yes	`max_n`: number of graph nodes; `edge_ratio`: edge density; e.g., `max_n=14`, `edge_ratio=2.5`.
Othello	Yes	`board_size`: board side length; e.g., `board_size=4/6/8/10`.
Frozen Lake	Yes	`size`: grid size; `num_holes`; e.g., `size=4`, `num_holes=5`.
MultiRoom	Yes	`min_num_rooms/max_num_rooms`: room-count range; `max_room_size`: maximum room size; e.g., `min_num_rooms=2`, `max_num_rooms=6`, `max_room_size=10`.
Mini Sudoku	Yes	`min_empty/max_empty`: number of blank cells in a fixed 4×4 grid; e.g., `min_empty=8`, `max_empty=12`.
Tree Even Partitioning	Yes	`max_n`: number of tree groups; `max_k`: group size; e.g., `max_n=10`, `max_k=6`.
Sum Triangle Area	Yes	`max_n`: number of 2D points; e.g., `max_n=9`.
Matchstick Equation	Yes	`break_moves`: number of perturbation moves; `enforce_min_distance`: minimum repair-distance constraint; e.g., `break_moves=2`, `enforce_min_distance=True`.
Knight Swap	Yes	`min_nodes/max_nodes`: board graph size; `min_steps/max_steps`: solution depth; e.g., `min_nodes=9`, `max_nodes=10`, `min_steps=8`, `max_steps=20`.
Texas Holdem	No	–
Hue-QA	Yes	`board_size`: color-board size; `num_lines`: number of gradient lines; e.g., `board_size=8`, `num_lines=6`.
Circuit Logic	Yes	`min_terms/max_terms`: number of logic terms; `min_inputs/max_inputs`: number of inputs; e.g., `min_inputs=4`, `max_inputs=6`, `min_terms=4`, `max_terms=8`.
Weighted Binarytree	Yes	`max_n`: number of tree nodes; `max_score`: weight range; e.g., `max_n=12`, `max_score=20`.
Nine Puzzle	Yes	`max_n_m`: grid size; `steps`: number of scrambling steps; e.g., `max_n_m=4`, `steps=12`.
Math Path	Yes	`size`: grid size; e.g., `size=4`.
Minimum Chromatic Number	Yes	`max_n`: number of graph nodes; `edge_density`: graph density; e.g., `max_n=14`, `edge_density=0.6`.
Tree To Traversal	Yes	`min_nodes/max_nodes`: binary-tree node-count range; e.g., `min_nodes=12`, `max_nodes=16`.
Chess Ranger-QA	Yes	`num_pieces`: number of chess pieces on the board; e.g., `num_pieces=8`.
Klo Blocks	Yes	`N`: array or board length; e.g., `N=7`.
Rush Hour	Yes	`difficulty`: predefined difficulty level; e.g., `difficulty=easy/medium/hard`.

Continued on next page

Table 8 continued from previous page.

Environment	Supported	Scale Parameters
Largest Island	Yes	`min_num_islands/max_num_islands`: number of islands; `max_island_size`: maximum island size; e.g., `min_num_islands=4`, `max_num_islands=8`, `max_island_size=20`.
Grid Local Minimum Counting	Yes	`max_n_m`: maximum grid side length; e.g., `max_n_m=5`.
Binary Matrix	Yes	`min_n/max_n`: matrix size; e.g., `min_n=5`, `max_n=8`.
Ultra TicTacToe-QA	No	–
SkaRock Garden	Yes	`max_n`: number of points; e.g., `max_n=7`.

Table 8: Scale controllability of selected environments.

Method	OOD-ST	OOD-MT	Rel.
Base Model	13.8	9.5	0.0
Description Embedding	16.3	8.3	2.8
Code Embedding	18.6	9.9	19.5
AES	**21.0**	**12.2**	**40.3**

Table 9: Comparison with representation-based environment selection. Description Embedding selects environments based on embeddings of textual environment descriptions, while Code Embedding selects environments based on embeddings of key environment code.

E More Experimental Results

E.1 Performance Transfer across Modalities

In the preliminary experiments 2.2, we analyze performance transfer across environments under both text-only and multimodal settings. Due to space limitations, the main text only reports a summarized table. Here, we provide the complete performance transfer matrices in Table 10 and Table 11.

E.2 More Results on AES

This section provides additional results for Section 3.2. Figure 7 shows the distribution of the selected environments in the t-SNE space of all environment profiles. The selected environments are well dispersed and cover different regions of the environment space. Our 200-environment pool is built from existing multimodal environment construction works. Therefore, the colors in the figure denote the human-defined categories from the original sources. We observe that some regions contain points of different colors, suggesting that human-defined categories do not always align with environment similarity from the perspective of agent behavior and training dynamics.

E.3 Comparison with Existing Diversity Selection Method

In Section 3, we mention that prior works often analyze environment diversity through surface-level information, such as representations of environment descriptions (Zhang et al., 2025b). In contrast, we argue that environment diversity should be analyzed from the perspective of the model's meta-abilities, which better reflect the underlying mechanisms required to solve the environments.

In this section, we compare AES with representation-based environment selection methods. Specifically, we consider two settings: one based on embeddings of textual environment descriptions, such as task descriptions and prompts, and the other based on embeddings of key environment code. For both settings, we use BAAI/bge-m3 (Chen et al., 2024a) as the embedding model and apply a cosine-distance-based greedy k-center selection strategy over the candidate environment pool. We start from the environment closest to the global center, and iteratively add the environment that is farthest from the current selected set. Finally, each method selects 30 environments, which are compared with the 30 environments selected by AES. We train Qwen3-VL-4B on each environment subset. As shown in Table 9, description-embedding selection brings only limited improvement, achieving a relative gain of 2.8%. Code-embedding selection performs better, with a relative gain of 19.5%. However, both methods are clearly worse than AES, which achieves a relative gain of 40.3%. These results suggest that surface representations are insufficient to capture the abilities required by multimodal agents. In contrast, AES selects environments from the model's meta-ability perspective and constructs a more effective training environment distribution.

E.4 Training Curves of HDC

This section further illustrates the effectiveness of our bi-level curriculum. We train a model on the Frozen Lake environment alone and visualize the recent success rate as training progresses in Figure 8. The shaded regions represent different harness levels, corresponding to different stages of the outer curriculum. The curve shows an adaptive learning process. At a fixed difficulty level, the model performance gradually improves. Once the reward reaches the threshold, the curriculum advances to the next state-scale level. After the target scale is completed, the harness level is reset to a harder setting, and the state-scale progression starts again.

E.5 Error Correction Analysis after HDC

To further analyze how HDC changes the model's ability, we evaluate the trained model on the 200 samples used in our previous error analysis. We focus on whether the model can correct the previously observed failure modes after training. As shown in Figure 9, HDC substantially reduces errors related to visual state extraction and world modeling, which are exactly the two major bottlenecks targeted by our curriculum design. These results provide additional evidence that HDC does not merely improve the final success rate, but also directly mitigates the key multimodal reasoning

Train env.	Text-symbolic evaluation environment Acc. (%)					
	addition_table	mini_sudoku	rectangle_count	rotate_matrix	spiral_matrix	tictactoe_qa
Base Model	16.5	55.2	11.0	72.5	76.2	47.6
addition_table	**37.5** (+21.0)	59.3 (+4.1)	18.5 (+7.5)	79.5 (+7.0)	83.2 (+7.0)	48.1 (+0.5)
mini_sudoku	24.6 (+8.1)	**82.1** (+26.9)	19.3 (+8.3)	81.3 (+8.8)	80.1 (+3.9)	45.3 (-2.3)
rectangle_count	17.6 (+1.1)	55.0 (-0.2)	**61.0** (+50.0)	77.1 (+4.6)	79.6 (+3.4)	42.1 (-5.5)
rotate_matrix	21.5 (+5.0)	56.0 (+0.8)	21.0 (+10.0)	**92.0** (+19.5)	93.0 (+16.8)	43.5 (-4.1)
spiral_matrix	22.1 (+5.6)	56.7 (+1.5)	19.8 (+8.8)	92.8 (+20.3)	**93.7** (+17.5)	42.1 (-5.5)
tictactoe_qa	18.8 (+2.3)	54.5 (-0.7)	13.5 (+2.5)	75.7 (+3.2)	82.2 (+6.0)	**55.8** (+8.2)
mixed6	36.1 (+19.6)	80.2 (+25.0)	60.0 (+49.0)	93.3 (+20.8)	94.9 (+18.7)	52.6 (+5.0)
Δ	-1.4	-1.9	-1.0	+1.3	+1.2	-3.2

Table 10: Transfer matrix in the text-symbolic setting. Each row denotes the training environment, and each column denotes the evaluation environment. All values are reported as accuracy in percentage. Colored values in parentheses denote the absolute accuracy change relative to the Base Model on the same evaluation environment. Diagonal entries correspond to single-environment training performance. The last row reports the change from mixed training to the corresponding single-environment result, i.e., $\Delta_j = M_j - T_{j,j}$.

Train env.	Multimodal evaluation environment Acc. (%)					
	addition_table	mini_sudoku	rectangle_count	rotate_matrix	spiral_matrix	tictactoe_qa
Base Model	7.8	41.7	6.5	25.0	18.8	32.8
addition_table	**27.8** (+20.0)	49.2 (+7.5)	17.0 (+10.5)	26.0 (+1.0)	20.0 (+1.2)	34.0 (+1.2)
mini_sudoku	12.5 (+4.7)	**72.9** (+31.2)	16.7 (+10.2)	27.1 (+2.1)	21.1 (+2.3)	35.1 (+2.3)
rectangle_count	6.5 (-1.3)	40.0 (-1.7)	**56.0** (+49.5)	24.5 (-0.5)	19.2 (+0.4)	33.2 (+0.4)
rotate_matrix	9.3 (+1.5)	43.5 (+1.8)	19.5 (+13.0)	**45.0** (+20.0)	45.8 (+27.0)	31.7 (-1.1)
spiral_matrix	9.5 (+1.7)	43.0 (+1.3)	20.5 (+14.0)	47.3 (+22.3)	**43.5** (+24.7)	31.0 (-1.8)
tictactoe_qa	8.1 (+0.3)	41.0 (-0.7)	5.3 (-1.2)	22.1 (-2.9)	17.0 (-1.8)	**45.1** (+12.3)
mixed6	26.1 (+18.3)	59.1 (+17.4)	51.5 (+45.0)	42.5 (+17.5)	40.8 (+22.0)	39.3 (+6.5)
Δ	-1.7	-13.8	-4.5	-2.5	-2.7	-5.8

Table 11: Transfer matrix in the multimodal setting.

failures identified in our analysis.

Clarification on Human Annotation. The analysis in this section and Section 2.3 was conducted manually by human annotators. The annotators were members of the research team or internal collaborators. Before annotation, they received the necessary disclosures and provided consent, and the work was supported through institutional funding. We provided clear guidelines, including the definition of each error category and representative examples. For ambiguous cases, the final labels were determined through internal discussion.

F Prompt for Atomic Ability Annotation

This appendix presents the prompt used to annotate model rollout trajectories into atomic behavioral steps and atomic abilities. The annotations are used to construct environment-level ability profiles and ability graphs.

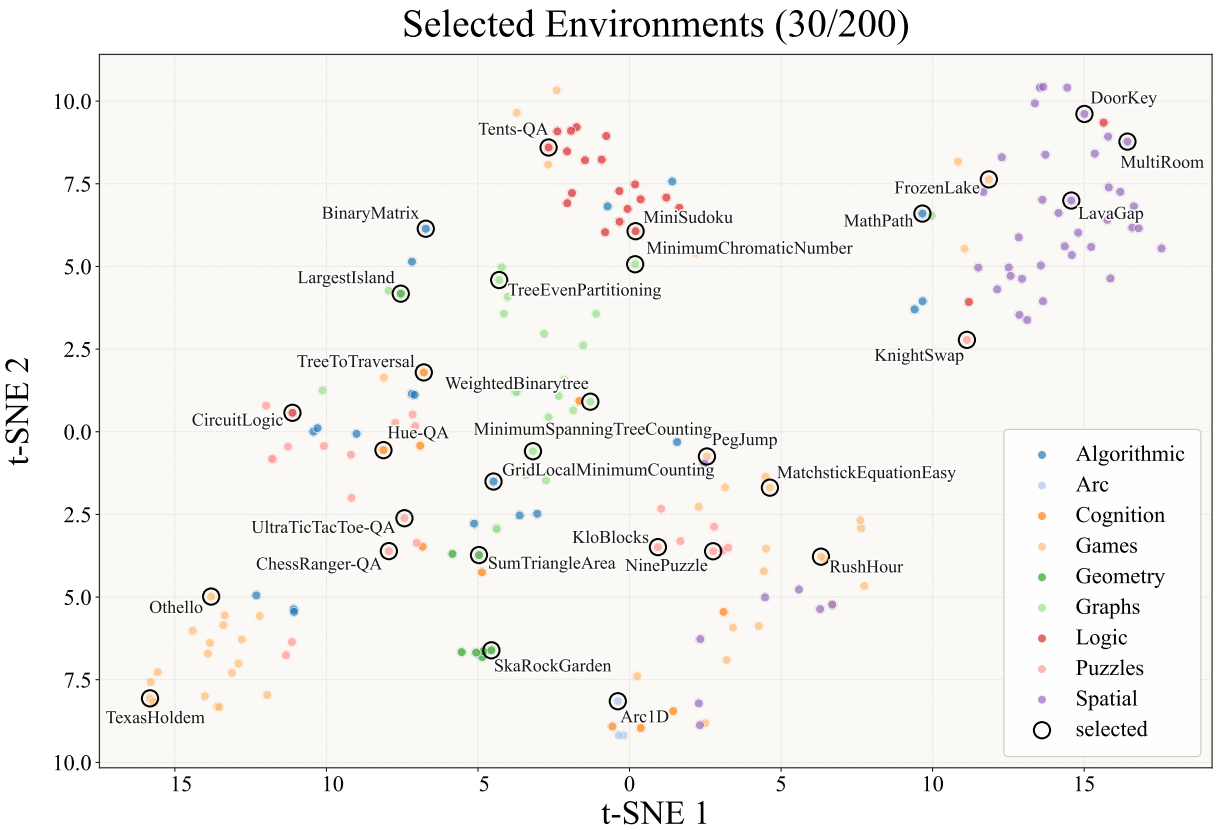

Figure 7: The distribution of the 30 environments selected by AES across the entire profile space of 200 environment pools.

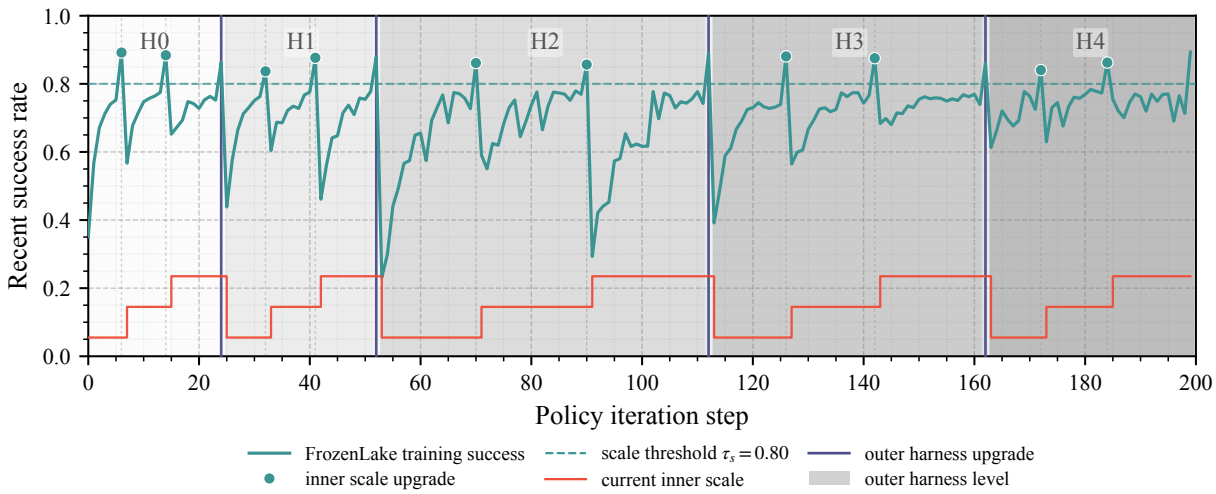

Figure 8: The training curve of HDC on Frozen Lake.

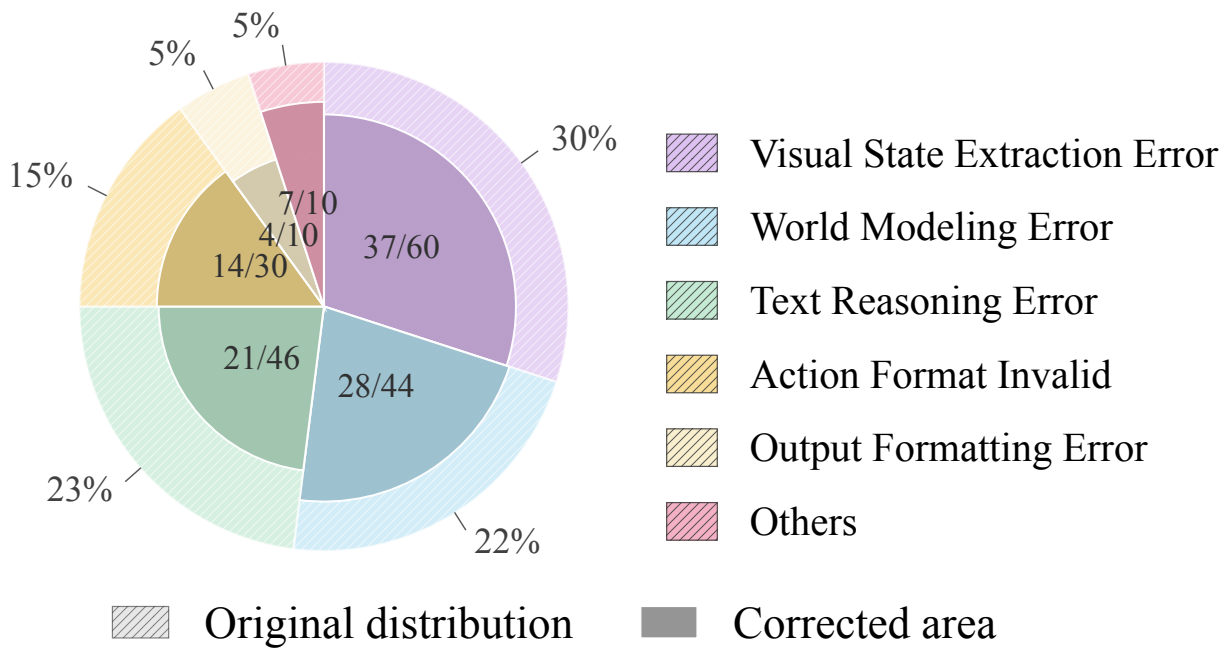

Figure 9: Corrected error analysis when learning algorithm using HDC.

Table 12: System prompt for atomic behavioral step and atomic ability annotation.

Prompt for Atomic Ability Annotation

SYSTEM PROMPT:
You are an expert analyst for multimodal agent environment rollouts.

Your task is to segment a model rollout episode into reusable atomic behavioral steps and infer solver-pattern-level atomic abilities. The annotations will be used to:

1. cluster common atomic steps across environments;
2. build an environment-level skill graph;
3. measure ability overlap and redundancy between environments;
4. support diverse environment selection.

A rollout step is one interaction with the environment. However, an atomic behavioral step is not necessarily identical to one environment interaction. One atomic behavioral step may span multiple environment interactions, and one environment interaction may also involve multiple atomic behavioral steps. Segment the trajectory according to the model's functional problem-solving process, rather than the simulator's step granularity.

Use solver-pattern-level labels. Each label should be abstract enough to transfer across environments, but concrete enough to distinguish different solution mechanisms.

Good labels include: `explore_unseen_area`, `extract_grid_or_matrix_state`, `extract_graph_or_relation_state`, `solve_index_mapping_or_traversal`, `manipulate_object`, `solve_constraint_satisfaction`, `solve_grid_propagation`, `simulate_rule_based_dynamics`, `generate_legal_actions`, `update_belief_from_feedback`, `infer_visual_transformation_rule`, `solve_combinatorial_counting`, `solve_graph_matching`, `solve_state_transformation`, and `verify_goal_or_terminal_state`.

Do not use labels that are too coarse, such as `perception`, `reasoning`, `planning`, `understanding`, `decision_making`, or `memory`.

Do not use labels that are too fine-grained or environment-specific, such as `compute_first_player_value`, `infer_cellular_automaton_rule`, `read_arrow_at_current_cell`, `identify_sudoku_empty_cell`, `map_cell_coordinates_under_rotation`, `maintain_spiral_boundary_indices`, `count_live_neighbors_in_game_of_life`, `pick_Alice_move`, or `move_to_red_box_in_TMaze`.

Instead, map environment-specific behaviors to reusable solver-pattern labels. For example, `compute_first_player_value` should be mapped to `solve_adversarial_search`; `infer_cellular_automaton_rule` should be mapped to `simulate_rule_based_dynamics`; `read_arrow_at_current_cell` should be mapped to `model_transition_or_operation_rule`; `identify_sudoku_empty_cell` should be mapped to `solve_constraint_satisfaction`; `map_cell_coordinates_under_rotation` and `maintain_spiral_boundary_indices` should be mapped to `solve_index_mapping_or_traversal`; and `move_to_red_box_in_TMaze` should be mapped to `solve_path_planning` or `explore_unseen_area`.

For Sudoku-like constraint puzzles, use `verify_solution_constraints` for both partial legality checks and final solution validation. Do not emit a separate `detect_inconsistency` label for Sudoku.

An atomic behavioral step is a locally coherent functional segment in the rollout, such as extracting or updating structured state, building or updating a spatial map, modeling a rule, constraint, transition, or legal action set, choosing or executing a path-planning subgoal, satisfying an action precondition, manipulating an object, verifying success, failure, legality, or terminal state, or recovering from a wrong path or failed assumption.

Follow these segmentation rules:

1. merge consecutive environment interactions if they serve the same immediate function;
2. split when the functional role changes;
3. split at key state-changing or informative events, such as pickup, drop, toggle, unlock, collision, death, goal reached, invalid action, feedback received, backtrack, or retry;
4. for failed trajectories, still segment by attempted functions;
5. prefer stable reusable labels over environment-specific labels;
6. use the same label for similar solver patterns across categories.

Return exactly one valid JSON object only. Do not wrap the JSON in markdown fences. Each atomic step must contain `start_index` and `end_index` referring to rollout indices. Each atomic step must include evidence from the rollout. Include failure modes for failed or mixed rollouts. Include an `environment_skill_graph` with nodes and directed edges. Also include `cross_category_skill_tags` for downstream clustering. Do not include any explanations outside the JSON object.