GLYPHBENCH: A PLAYGROUND FOR LANGUAGE-MODEL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **GlyphBench**, an environment suite for reinforcement learning (RL) post-training of language-model agents, with over 360 tasks spanning diverse games. GlyphBench renders spatial observations as two-dimensional Unicode grids and connects training, evaluation, and trajectory replay through a unified interface designed to support efficient and reproducible research. We use GlyphBench to study how observation interfaces, reasoning effort, and agent harnesses affect performance, and how RL configurations shape learning dynamics. Our results show that glyph observations outperform native text and pixels in our Craftax experiments, with further gains on several BALROG environments. RL on 100 GlyphBench tasks improves Qwen3.5-4B on held-out Reasoning Gym problems, reaching 63.48% accuracy and outperforming the base model, a math-trained baseline, and a code-trained baseline. These experiments provide empirical evidence that reasoning gains from gameplay can yield stronger transfer than math or code. Together, these results highlight GlyphBench's value as a testbed for systematic research on how language-model agents learn, interact, and generalize.

1 INTRODUCTION

Reinforcement learning (RL) has improved the performance of language models on problems with verifiable rewards, such as mathematics and code (Guo et al., 2025). Tasks that are interactive by nature are particularly challenging: models must choose actions that shape subsequent observations, act with incomplete information, and account for consequences that may only become apparent later. For example, a software-engineering agent edits code, runs tests, and uses the resulting feedback to guide further decisions (Golubev et al., 2025). To improve agents' reasoning, we need to understand how they use the information available at each decision, how they learn from environmental feedback, and whether the capabilities they acquire extend to new problems. Studying these questions requires diverse interactive tasks that researchers can use for training and evaluate under controlled conditions.

Games have been a central testbed for deep RL over the past decade, exposing different learning challenges through repeatable interactions. Benchmarks such as the ALE (Bellemare et al., 2013) helped establish deep RL (Mnih et al., 2015), while Procgen (Cobbe et al., 2020), MiniHack (Samvelyan et al., 2021), and Craftax (Matthews et al., 2024) expose distinct challenges in generalization, exploration, and planning. Games have also driven landmark results in Go and StarCraft II (Silver et al., 2017; Vinyals et al., 2019), which have sparked further advances in the field. Collectively, these benchmarks support both targeted studies of individual capabilities and experiments in which several capabilities must work together.

For language-model post-training, however, differences in observation and action interfaces complicate comparisons across environments: performance depends on how information is presented as well as on the decisions a task requires (Paglieri et al., 2025). Recent training studies use settings as different as BabyAI-Text (Carta et al., 2023), symbolic multi-turn games (Wang et al., 2025), and visual arcade games (Xie et al., 2026). BALROG and lmgame-Bench bring several games together through shared interfaces (Paglieri et al., 2025; Hu et al., 2025), but game collections, observation formats, and training setups still differ across studies. Making these well-studied environments accessible through a common text interface that preserves spatial layout offers a practical way to study learning and generalization across games within a shared experimental framework.

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

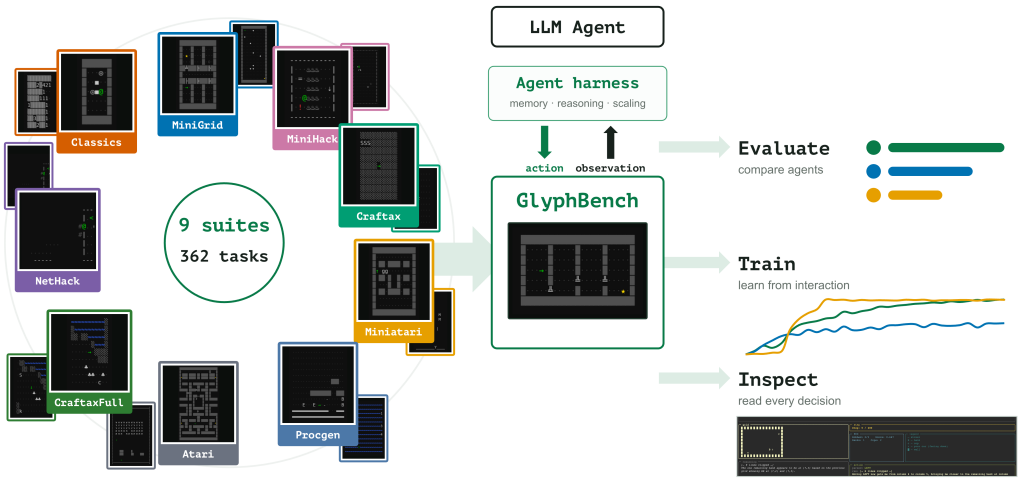

Figure 1: GlyphBench connects nine game families through spatial text observations and named actions. A shared task interface supports policy evaluation, RL training, and trajectory replay. Six families form the standard track; three provide extended tasks.

We introduce **GlyphBench** to study these questions across a diverse collection of interactive environments. GlyphBench brings together puzzles, arcade games, and established RL benchmarks through a common text interface: agents observe two-dimensional Unicode grids and interact through named actions. These grids preserve the relative positions of objects in a form that language models can consume directly and researchers can inspect. Researchers can train and evaluate policies across these environments, replay trajectories, and test changes to training or the agent harness (Figure 1). GlyphBench supports this cycle of measuring performance, diagnosing behavior, and testing improvements across diverse tasks.

We first examine how models act before additional training, comparing task performance, episode length, and token use across models of different sizes and capabilities. Our experiments in Craftax (Matthews et al., 2024) show the benefit of spatial glyph observations: they achieve higher mean reward than native text and pixels for all three models we evaluate. We separately examine how harness design affects the use of additional reasoning. Single-task policies improve across all six standard suites, revealing a range of learning challenges alongside differences in existing model capabilities.

To illustrate how researchers can investigate training dynamics, we vary learning rate and rollout group size in Snake under a fixed interaction budget. Smaller rollout groups allow more updates, but the resulting gains depend strongly on learning rate. This experiment illustrates GlyphBench's use as a tool for studying how training choices interact.

Finally, we study joint post-training and transfer. A single policy trained on 100 GlyphBench tasks improves on held-out environment seeds across the six suites. The trained policy also improves on held-out Reasoning Gym (Stojanovski et al., 2025) problems, outperforming a math-trained baseline and a code-trained baseline. These results provide empirical evidence that reasoning gains from gameplay can transfer more effectively to this benchmark than those from math or code RL. Together, our evaluations demonstrate GlyphBench's usefulness as a benchmark, a research tool, and a post-training suite, and provide broader insight into the value of games for understanding and improving language models.

2 Related Work

Language-model agents in games. Applying language models to game environments introduces a fundamental design question: how should pretrained knowledge be translated into effective actions? SPRING addresses this through structured reasoning over knowledge extracted from the environment (Wu et al., 2023), while Voyager generates, refines, and reuses executable skills in Minecraft (Wang

et al., 2024). Both illustrate how an agent's reasoning procedure and supporting components can improve gameplay without updating the underlying model's parameters. Broader benchmarks examine these design choices across environments. BALROG evaluates language-model and vision-language agents on established RL tasks and identifies limitations in handling complex environments and visual observations (Paglieri et al., 2025). lmgame-Bench studies how perception, memory, and prompt design affect the reliability of game-based evaluation (Hu et al., 2025). Agentick extends comparisons to RL, language-model, and vision-language agents through common tasks with multiple observation modalities (Creus Castanyer et al., 2026). These studies show why comparisons must account for the interaction interface alongside the underlying model, motivating our experiments on harness design and inference-time reasoning.

Learning through interaction and transfer. Learning from interaction raises two further questions: how to train policies effectively, and whether the resulting improvements extend beyond the training tasks. GLAM investigates online RL for grounding pretrained language-model policies in textual environments with spatial and navigation tasks (Carta et al., 2023). RAGEN studies training instability and the effects of rollout diversity and freshness in multi-turn RL (Wang et al., 2025), while AgentGym-RL develops a framework for training across interactive environments and a strategy that progressively increases interaction horizons (Xi et al., 2026). These works make the collection and optimization of interaction trajectories central to the study of agent learning. Evidence for transfer from games also highlights the importance of the evaluation domain. The initial lmgame-Bench study reports that RL on Sokoban or Tetris improves performance on other games, planning problems, and WebShop (Hu et al., 2025). ViGaL reports that RL on visual arcade games improves multimodal reasoning on benchmarks including MathVista and MMMU (Xie et al., 2026). These findings motivate investigating which capabilities transfer under different training conditions. We use GlyphBench to study optimization under a fixed interaction budget and multitask training across diverse spatial text games, evaluating both performance on held-out game seeds and transfer to held-out Reasoning Gym problems (Stojanovski et al., 2025). Appendix A expands these connections.

3 GLYPHBENCH

GlyphBench provides over 360 tasks across nine game families through a common framework for language-model training, evaluation, and trajectory replay. Researchers can study learning on individual tasks, train policies across task mixtures, and inspect the decisions behind aggregate performance. Glyph observations and named, task-specific actions provide a consistent interaction protocol across the suite. Standard tasks bound episode lengths and returns to support repeated RL experiments, while extended tasks support longer episodes of exploration and progression.

3.1 TASKS, OBSERVATIONS, AND AGENT HARNESSES

Each task e is an episodic, partially observed decision process with transition rule P_e, observation rule O_e, and rewards r_t. A language model with parameters θ and a harness η induce a policy $\pi_{\theta,\eta}(a_t \mid h_t)$ over actions given interaction history h_t. The harness constructs prompts, retains history or memory, and parses model outputs. For task distribution $\mathcal{D}$, training seeks to maximize

$$J_{\mathcal{D}}(\theta;\eta) = \mathbb{E}_{e\sim\mathcal{D},\,\tau\sim(\pi_{\theta,\eta},e)}\big[R_e(\tau)\big], \qquad R_e(\tau) = \sum_{t=0}^{T-1} r_t, \tag{1}$$

where τ is an episode of length T. This notation distinguishes the variables studied below: the observation rule and harness determine the agent's inputs, RL updates θ, and the training mixture determines $\mathcal{D}$.

We place spatial information in a two-dimensional Unicode grid, identify symbols with a legend, and use a heads-up display (HUD) for quantities such as health, inventory, or velocity. Actions use a task-specific discrete vocabulary. This representation makes adjacency and layout explicit in text (Figure 2); Appendix B shows the full interface in replay.

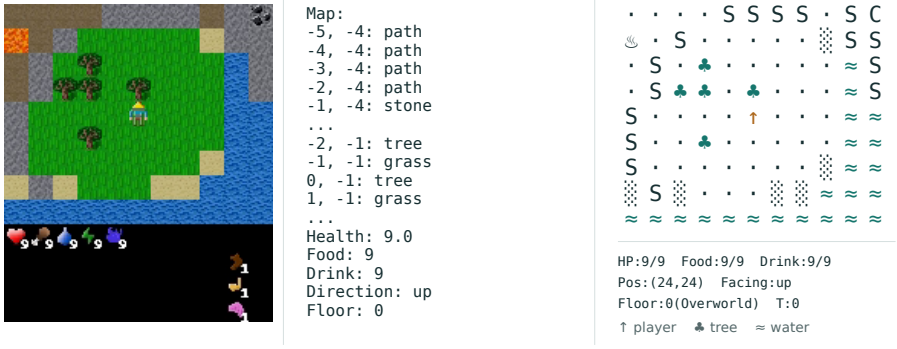

Figure 2: The same Craftax state in pixels, Craftax's original text renderer (native text), and GlyphBench's glyph format. Native text is a comparison interface, not part of GlyphBench. We show excerpts here and both complete text observations in Figure 11. Glyph colors aid visualization only.

3.2 Standard and Extended Tasks

We select game families with an established role in RL research on control, procedural generalization, navigation and memory, and survival with staged progression (Appendix Table 1). Using these familiar challenges gives researchers a basis for interpreting language-model behavior and choosing tasks that test a particular hypothesis.

We reimplement their mechanics from scratch in 303 standard tasks and bound episode lengths and cumulative returns by design: $T \leq 512$ and $R_e(\tau) \in [-1, 1]$. The episode cap limits the cost of collecting each training example, while the shared return range makes it easier to combine tasks and compare progress across a mixture. The 59 extended tasks support longer episodes through arcade reimplementations and wrappers around the existing full Craftax and NetHack engines. We use standard tasks for RL training and the full Craftax game to study agents that must explore, gather resources, and prepare for challenges across nine floors. Appendix C illustrates these floors, and Appendix B describes the training interfaces and trajectory replay tools.

4 Experimental Setup

We evaluate GlyphBench as a benchmark for agents, a tool for research, and a suite for post-training. We first compare existing models and train individual task policies to examine the capabilities and learning challenges captured by the benchmark. We then use Snake to illustrate how researchers can study the interaction between optimization choices. Finally, we train a policy on a task mixture and evaluate it on external reasoning problems to test whether learning through gameplay transfers beyond the training environments. Training uses our Prime-RL integration (Prime Intellect, 2025b); Appendix E specifies the objectives and configurations.

Benchmarking agents. We evaluate Qwen3.5-4B and Qwen3.8-27B on all 303 standard tasks with 25 matched seeds per task (7575 episodes per model; Figure 3). Returns are averaged within tasks and tasks receive equal weight. Episode length and prompt-plus-completion tokens complement return by measuring how much interaction and generation each model uses.

We use the full nine-floor Craftax game to study how agents interpret observations and use information from earlier decisions. We compare two agent harnesses, with examples in Appendix C.2. The *Basic* harness presents the game instructions and conversation history at each turn. The *Guided Memory* harness instead organizes information into a compact memory of plans, landmarks, and recent actions, and supplies guidance for the current floor. It updates this memory through separate model calls, giving the agent a way to revisit its plans as the game changes. Both harnesses select one native game action per turn and cannot execute scripts.

In Figure 4, we evaluate GPT-5.6-{Luna, Terra, Sol}. We first compare native text, glyphs, and pixels with Guided Memory at high reasoning effort. We then compare the two harnesses using glyphs and vary Sol's reasoning effort to examine how harness design affects the use of additional computation. We report five-episode means with 95% t intervals, scoring reward as $100R/226$ (Craftax contributors, 2026). We include one exploratory GPT-6-Astra episode with Guided Memory at maximum effort to illustrate how far a stronger model can progress. Appendix C describes the Craftax interfaces and harnesses. We also compare original and glyph interfaces for five models in BabyAI, Baba Is AI, Crafter, and MiniHack from BALROG (Paglieri et al., 2025) (Figure 12, Appendix D).

We train 25 separate Qwen3.5-4B policies across all six standard suites to check whether the tasks supply useful reward signals and the common training interface supports improvement. Figure 5 shows selected learning curves; Appendix Figure 13 shows all runs.

Studying learning dynamics. To illustrate how GlyphBench supports studies of learning dynamics, we use Snake to examine how learning rate and rollout group size interact during RL. Across its three difficulty levels, we compare frequent updates from small groups with fewer updates from large groups under the same interaction budget (Figure 6). We cross five group sizes (32–512) with four learning rates (10^{-6}–10^{-5}), yielding 60 conditions. Each update uses one group, so 6,144 rollouts permit $6{,}144/G$ updates at group size G. In Appendices J and K, we examine higher learning rates and longer rollout budgets.

Post-training and transfer. We then study learning across a mixture of tasks over 100 gradient steps with one Qwen3.5-4B policy trained on 100 GlyphBench tasks. We use group size 32, learning rate 5×10^{-6}, and 32 groups per update (1,024 rollouts), with Prime-RL's IPO-style loss. We evaluate on held-out environment seeds every 50 updates (Figure 7).

To test whether learning through interaction improves reasoning outside the training environments, we compare the base model and policies obtained after GlyphBench, Math, and Code RL on identical Reasoning Gym problems (Stojanovski et al., 2025) (Figure 8). Appendix I describes the baseline datasets and evaluations; Appendix H reports additional evaluations on MathArena and SWE-bench (Balunović et al., 2025; Jimenez et al., 2024; Deng et al., 2025).

5 Results

Our experiments establish three uses of GlyphBench. As a benchmark, it distinguishes agent capabilities and exposes varied learning challenges (Section 5.1). As a research tool, it reveals interactions between optimization choices (Section 5.2). As a post-training suite, it supports joint learning across games and stronger Reasoning Gym transfer than math or code RL in our main experiment (Section 5.3).

5.1 GlyphBench as a benchmark for agents

We first characterize performance before additional RL training on GlyphBench. Qwen3.8-27B improves on 270 of 303 tasks, with a mean paired return gain of 0.436. Return distributions shift upward in every suite, while scores remain spread across tasks within each family (Figure 3). The collection therefore distinguishes the two models while retaining tasks on which the stronger model has room to improve, establishing a reference point for the training experiments that follow.

Higher performance also comes with lower interaction cost: mean episode length falls from 72.8 to 23.9 turns and mean total tokens from 2.15M to 596k. This seemingly counterintuitive result is consistent with a stronger policy reaching goals directly, while a weaker one spends longer repeating unproductive actions or reasoning. Researchers can use GlyphBench's replay interface to examine these behaviors and connect differences in return and token use to individual decisions (Appendix B).

Observation format, harness support, and reasoning effort. Our results show that glyph observations give the highest mean reward in full Craftax for Luna, Terra, and Sol with Guided Memory at high reasoning effort (Figure 4(a)). For Sol, the mean reaches 11.4% of maximum reward, compared with 3.7% for pixels and 0.3% for native text. Glyphs preserve adjacency and spatial layout while

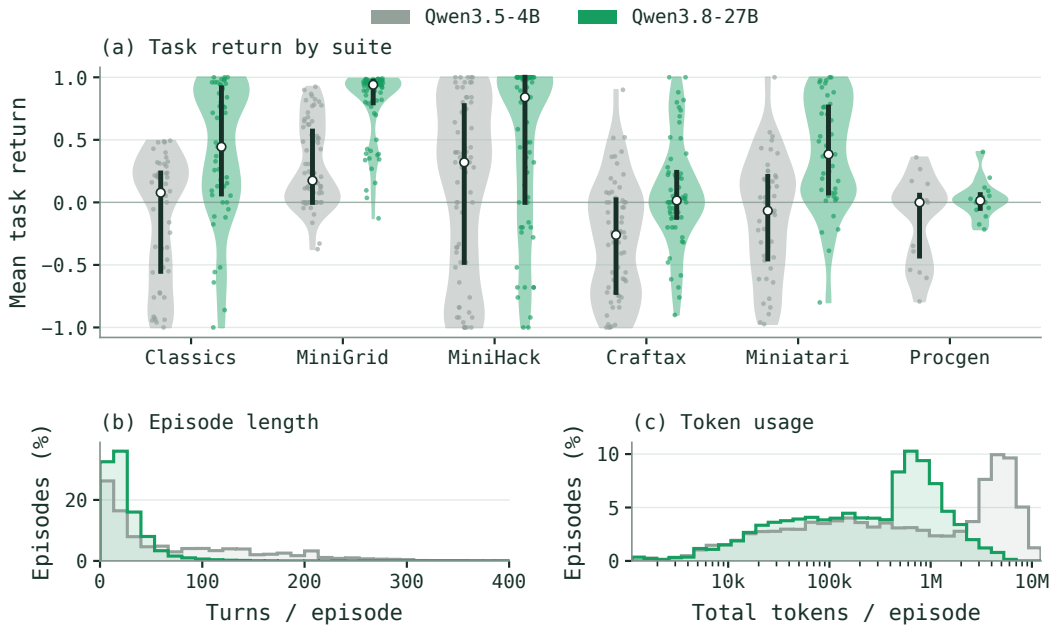

Figure 3: Zero-shot performance on 303 tasks with 25 matched seeds each. Panel (a) shows task-mean returns, with interquartile ranges and median dots; (b–c) show episode length and token use over 7575 episodes per model. Tokens include prompts and completions under each model's tokenizer.

naming entities through a legend. This reduces the need to reconstruct a map from coordinate descriptions or identify objects visually, offering a plausible explanation for the advantage. The result also parallels Agentick's findings on spatial text observations (Creus Castanyer et al., 2026).

Richer harness support becomes most useful when paired with sufficient reasoning. At high effort, Guided Memory improves Terra and slightly improves Sol, while Luna achieves a higher mean with Basic (panel (b)). Sol's effort sweep makes the interaction clearer (panel (c)): performance with Basic levels off near 11%, whereas Guided Memory continues improving to 15.8% at maximum effort. The steeper improvement with Guided Memory suggests that additional reasoning helps Sol use the richer memory and guidance. Harness design therefore affects how much an agent gains from additional inference computation, motivating studies that vary both together (Snell et al., 2025).

Astra reaches 51.4% in the exploratory episode with Guided Memory at maximum reasoning effort, the highest individual return in our Craftax evaluation. This run illustrates how far an agent can progress when a stronger model combines the glyph interface with memory and guidance.

Additionally, we use BALROG (Paglieri et al., 2025) to examine how glyph rendering affects performance outside the GlyphBench task collection (Figure 12, Appendix D).

Learning challenges across tasks. We next train separate policies to test whether the shared interface and task rewards support learning across all six standard suites. Figure 5 shows two tasks from each suite, with all 25 runs in Appendix F. Training returns improve across the suites, although the pace and stability of learning differ between tasks. The different learning trajectories show that the suite offers both learnable reward signals and tasks that remain challenging after training. Researchers can therefore study differences in learning speed and stability across tasks through a common interface.

5.2 GlyphBench as a tool for research

We use the following experiment to illustrate how GlyphBench supports research on learning dynamics and hyperparameter choices in post-training. Specifically, we examine how a fixed amount of interaction is used for policy improvement. In GlyphBench's Snake environment, we vary rollout group size and learning rate across three difficulty levels while holding the total number of sampled

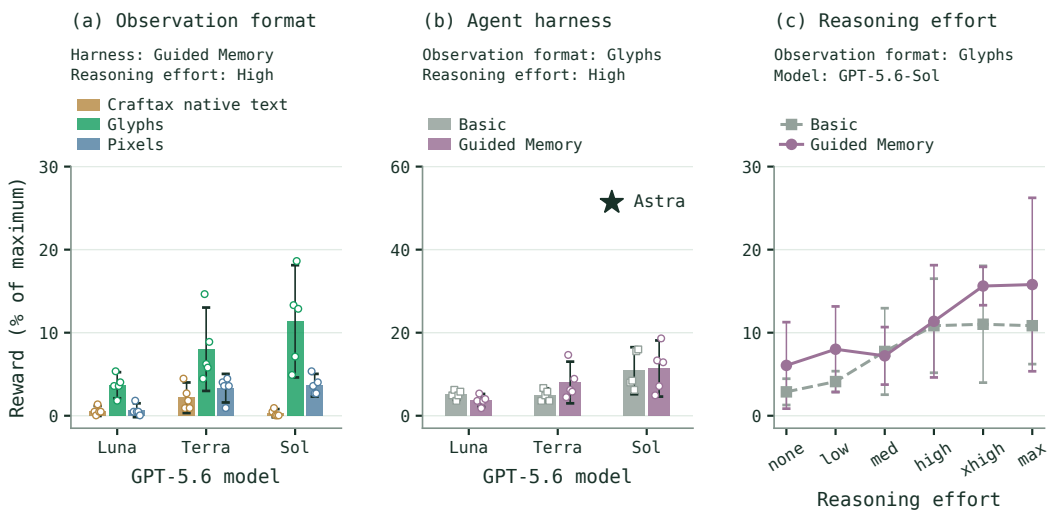

Figure 4: Zero-shot Craftax performance across observation formats (a), harnesses (b), and reasoning efforts (c), as a percentage of maximum reward ($100R/226$). Each panel specifies the settings held fixed. Whiskers show 95% t intervals around five-episode means; open markers show individual episodes in (a–b). The star marks one exploratory Astra episode with Guided Memory at maximum effort.

episodes fixed (Figure 6). These choices jointly control the updates made from that experience: smaller groups permit more frequent updates, while learning rate scales each update. A 6,144-rollout budget gives 192 updates at group size 32, compared with 12 at group size 512.

The benefit of frequent updates depends on how much each update changes the policy. At 10^{-6}, progress is modest and group sizes remain relatively close. At intermediate rates, smaller groups pull ahead: raising the rate to 5×10^{-6} increases group 32's final return on easy Snake from 0.587 to 0.900, with analogous gains on medium and hard. The precise optimum varies with difficulty, but the common pattern is that learning rate and update frequency must be chosen together.

Larger rates eventually erase these gains: 10^{-5} and 10^{-4} collapse toward zero (Appendix J). Conversely, extending the budget to 25,600 rollouts at 10^{-6} preserves group 32's advantage over group 512 on all three difficulties (Appendix K). Across these Snake tasks, smaller rollout groups are most effective when paired with a suitable learning rate. The experiment illustrates how GlyphBench can

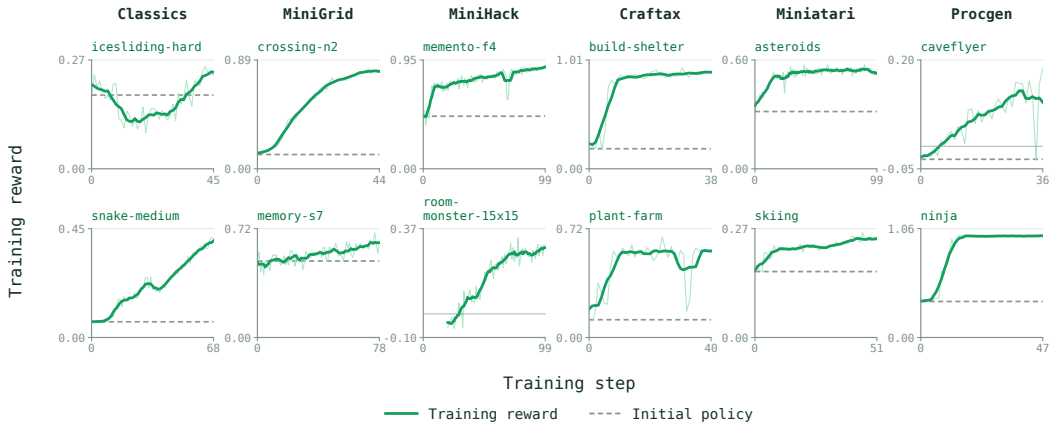

Figure 5: Policies learn on individual tasks across all six suites. Pale traces show training reward; dark traces show centered seven-point means. Dashed lines show mean initial-policy returns. Axes vary by task. Appendix F explains the selection and shows all 25 learning curves.

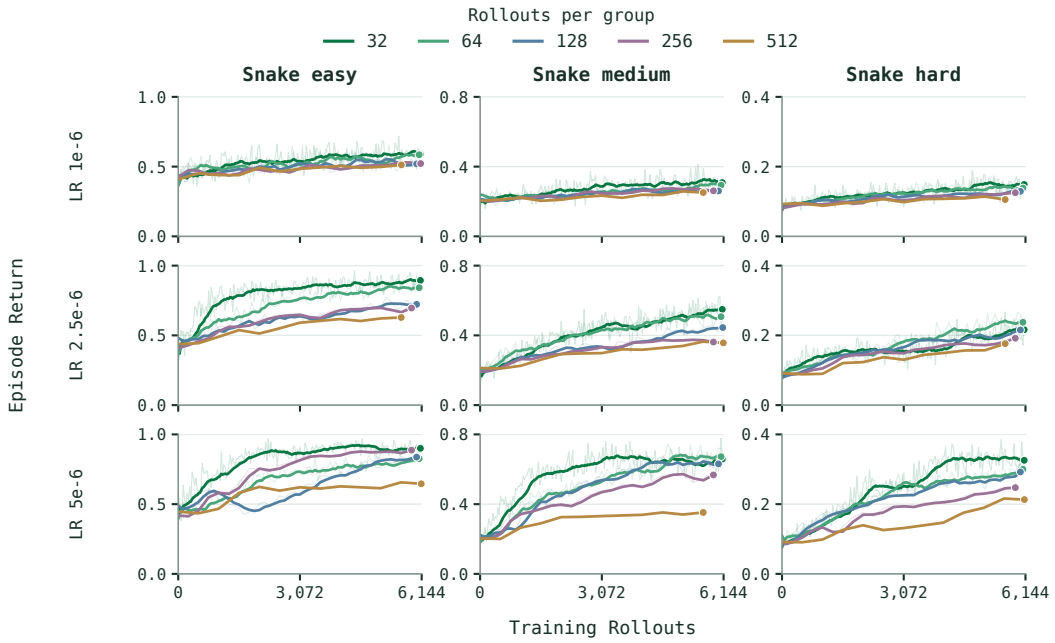

Figure 6: Learning in Snake under a fixed 6,144-rollout budget. The 45 conditions vary learning rate (rows), difficulty (columns), and group size (colors). Pale traces show batch returns; solid curves show trailing 512-rollout means; dots mark final observations. Return scales match within columns. We examine higher learning rates in Appendix J.

be used to study the interaction between optimization choices and task difficulty under a common interaction budget.

5.3 GlyphBench as a post-training suite

Learning across a task mixture. The single-task experiments train a separate policy for each task. We now test whether one policy can improve across a mixture of 100 tasks with different rules and reward structures. Mean return on new instances of these tasks rises from 0.012 to 0.319 within 100 gradient steps (Figure 7). These gains show that one policy can learn from rewards across games with different rules, making the mixture a useful source of interactive experience for post-training. Evaluation on new instances tests learning beyond the particular episodes seen during training; the external reasoning evaluation below tests transfer beyond these task types. Appendix G provides task-level learning curves and evaluations by suite.

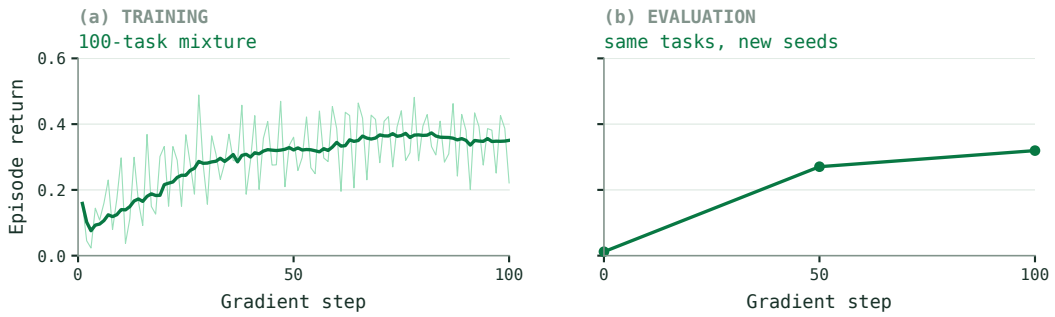

Figure 7: One policy learns across 100 tasks over 100 gradient steps. Panel (a) shows training returns with a trailing nine-step mean; (b) evaluates the same tasks on new environment seeds.

To relate these gains to other common domains for language-model RL, we apply the same optimization recipe to a set of 100 math problems and a set of 100 code problems. In Appendix I, we show

the learning curves for these runs. We next evaluate the policies obtained after training on held-out reasoning problems to compare how their gains transfer beyond their respective training domains.

Transfer beyond gameplay. We now test whether the improvements acquired through gameplay extend to reasoning problems outside the training environments. We evaluate the base model and policies obtained after GlyphBench, Math, and Code RL on the same 3,100 Reasoning Gym problems with a 32K output-token budget (Figure 8). GlyphBench RL reaches 63.48% accuracy, compared with 62.29% for Math RL, 58.98% for Code RL, and 56.04% for the base model. The gain over Math RL is 1.19 percentage points, with a paired 95% interval of $[0.38, 2.02]$.

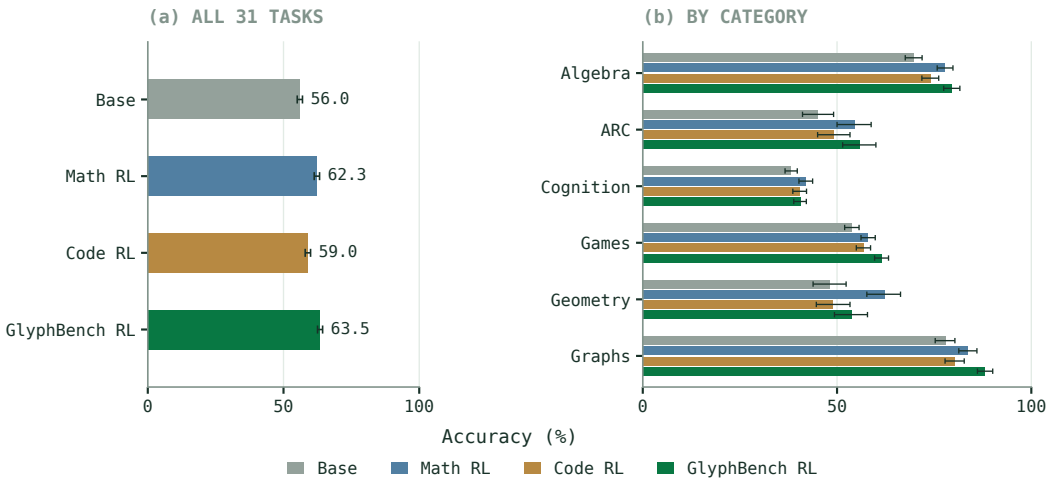

Figure 8: Reasoning Gym accuracy at a 32K output-token budget for the base model and policies trained on games, math, or code. Each policy answers the same 3,100 problems, with three attempts per problem. Panel (a) averages accuracy across 31 tasks; (b) separates the results by category. Whiskers show 95% problem-bootstrap intervals, resampling within tasks and keeping each problem's three attempts together.

The gains extend across several forms of reasoning. GlyphBench RL leads on algebra, ARC, games, and graphs, while Math RL leads on cognition and geometry. Training through gameplay can therefore improve performance on problems that require a direct answer, even though the policy learned from sequences of actions and environmental feedback. These results provide empirical evidence that gameplay can yield stronger reasoning transfer than math or code RL in this setting.

In Appendices H and I, we report evaluations at additional output budgets, category results, and generation diagnostics. We also evaluate on MathArena and SWE-bench in Appendix H, where the changes are inconclusive.

6 Conclusion

Our experiments establish GlyphBench as a common setting for evaluating agents, studying learning dynamics, and post-training models whose gains can extend beyond gameplay. Our Craftax experiments show that glyph observations improve performance, while the benefit of memory and guidance depends on the model and its reasoning effort. Policies learn on individual tasks and across a task mixture, and the Snake experiments show how learning rate affects the benefit of frequent updates from smaller rollout groups. Gameplay training also improves performance on Reasoning Gym, outperforming math- and code-trained baselines in the main experiment.

The gains on Reasoning Gym suggest that learning through gameplay can contribute to capabilities that are useful beyond the games themselves. A next step is to understand which kinds of interaction support this transfer and how training can encourage it. GlyphBench makes it possible to study these questions across different tasks, connecting changes in performance to the observations and decisions made during learning.

AI USE STATEMENT

We used LLMs to polish the writing and correct grammar. We also used LLMs and coding agents to assist with project development, including implementing, reviewing, and debugging the GlyphBench environments, running experiments, and preparing the figures in this paper.

REFERENCES

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0679. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1d27c01ebd3e3aebe226b44fc970d803-Abstract-Datasets_and_Benchmarks_Track.html.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. DeepMind Lab. *arXiv preprint arXiv:1612.03801*, 2016. URL https://arxiv.org/abs/1612.03801.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/jair.3912. URL https://jair.org/index.php/jair/article/view/10819.

Richard Bornemann, Pierluigi Vito Amadori, and Antoine Cully. CODE-SHARP: Continuous open-ended discovery and evolution of skills as hierarchical reward programs. *arXiv preprint arXiv:2602.10085*, 2026. URL https://arxiv.org/abs/2602.10085.

William Brown. Verifiers: Environments for LLM reinforcement learning. https://github.com/PrimeIntellect-ai/verifiers, 2025.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pp. 3676–3713, 2023. URL https://proceedings.mlr.press/v202/carta23a.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. MiniGrid & MiniWorld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 73383–73394, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/e8916198466e8ef218a2185a491b49fa-Abstract-Datasets_and_Benchmarks.html.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*, pp. 2048–2056, 2020. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018. URL https://arxiv.org/abs/1806.11532.

Craftax contributors. Craftax: Scoreboard. https://github.com/MichaelTMatthews/Craftax, 2026. Accessed September 21, 2026. Rewards are reported as a percentage of the maximum (226).

Roger Creus Castanyer, Pablo Samuel Castro, and Glen Berseth. Agentick: A unified benchmark for general sequential decision-making agents. *arXiv preprint arXiv:2605.06869*, 2026. URL https://arxiv.org/abs/2605.06869.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, et al. SWE-bench pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025. URL https://arxiv.org/abs/2509.16941.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pp. 8657–8677, 2023. URL https://proceedings.mlr.press/v202/du23f.html.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18343–18362, 2022. doi: 10.52202/068431-1333. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

Alexander Golubev, Maria Trofimova, Sergei Polezhaev, Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Sergey Abramov, Andrei Andriushchenko, Filipp Fisin, Sergei Skvortsov, and Boris Yangel. Training long-context, multi-turn software engineering agents with reinforcement learning. *arXiv preprint arXiv:2508.03501*, 2025. URL https://arxiv.org/abs/2508.03501.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 2442–2448, 2019. doi: 10.24963/ijcai.2019/339. URL https://www.ijcai.org/proceedings/2019/339.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7903–7910, 2020. doi: 10.1609/aaai.v34i05.6297. URL https://ojs.aaai.org/index.php/AAAI/article/view/6297.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/df7b97ccf54c528516b62aa9a012d02a-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-Bench: How good are LLMs at playing games? *arXiv preprint arXiv:2505.15146v1*, 2025. URL https://arxiv.org/abs/2505.15146v1.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In *International Joint Conference on Artificial Intelligence*, pp. 2684–2691, 2019. doi: 10.24963/ijcai.2019/373. URL https://www.ijcai.org/proceedings/2019/373.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 341–348, 2016. doi: 10.1109/CIG.2016.7860433. URL https://arxiv.org/abs/1605.02097.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack learning environment. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7671–7684, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/569ff987c643b4bedf504efda8f786c2-Abstract.html.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning*, pp. 35104–35137, 2024. URL https://proceedings.mlr.press/v235/matthews24a.html.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard S. Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygf-kSYwH.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fp6t3F669F.

Prime Intellect. INTELLECT-3-RL. Hugging Face dataset, code configuration, 2025a. URL https://huggingface.co/datasets/PrimeIntellect/INTELLECT-3-RL. Accessed September 21, 2026.

Prime Intellect. PRIME-RL, 2025b. URL https://github.com/PrimeIntellect-ai/prime-rl/tree/v0.9.0. Software, version 0.9.0. Accessed September 21, 2026.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 2186–2188, 2019. URL https://www.ifaamas.org/Proceedings/aamas2019/pdfs/p2186.pdf.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. MiniHack the planet: A sandbox for open-ended reinforcement learning research. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract-round1.html.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning Gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1913. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/528d56195a2c77c808494c86fa7c77ad-Abstract-Datasets_and_Benchmarks_Track.html.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024. URL https://arxiv.org/abs/2407.17032.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z. URL https://www.nature.com/articles/s41586-019-1724-z.

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. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

Yue Wu, Shrimai Prabhumoye, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. SPRING: Studying the paper and reasoning to play games. In *Advances in Neural Information Processing Systems*, volume 36, pp. 22383–22687, 2023. doi: 10.52202/075280-0983. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/46c2a9a6f2b2be68682013eb1173c801-Abstract-Conference.html.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *International Conference on Learning Representations*,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1571ce1ff735be4551db50887043726e-Abstract-Conference.html.

Yunfei Xie, Yinsong Ma, Shiyi Lan, Alan Yuille, Junfei Xiao, and Chen Wei. Play to generalize: Learning to reason through game play. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1cdbce34681a3aaa0154b300706743d8-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

A Extended Related Work

Game benchmarks as experimental infrastructure. Over the past decade, the RL community has widely used game environments to develop and compare learning algorithms. The Arcade Learning Environment (ALE) helped establish this approach by bringing games with diverse control problems under a shared interface (Bellemare et al., 2013). Deep Q-learning subsequently demonstrated the value of this common testbed for studying a single learning algorithm across tasks (Mnih et al., 2015). A complementary approach is to construct smaller diagnostic problems. The Behaviour Suite separates properties such as exploration, memory, and robustness so that a result can be connected to a specific algorithmic strength or failure (Osband et al., 2020). GlyphBench follows both traditions: its collection spans familiar game families, while focused tasks and replay help researchers investigate the behavior underlying aggregate performance. This breadth lets researchers ask where a method helps: for example, whether improved exploration in one game also benefits tasks that require memory or careful resource use.

Partial observability, generalization, and coordination. DeepMind Lab and ViZDoom provide configurable first-person environments for studying navigation, perception, and decisions from incomplete observations (Beattie et al., 2016; Kempka et al., 2016). MiniGrid offers compact, configurable goal-directed tasks, and MiniHack uses the NetHack engine to vary navigation and the composition of game skills (Chevalier-Boisvert et al., 2023; Samvelyan et al., 2021). Procgen explicitly separates training levels from unseen levels, making generalization part of evaluation (Cobbe et al., 2020); Obstacle Tower combines procedural variation with visual control and planning (Juliani et al., 2019). SMAC isolates a different challenge: decentralized cooperation when each unit acts from local observations (Samvelyan et al., 2019). Together, these environments have made it possible to study capabilities that a single game would leave entangled or untested. An agent that excels at reactive control may still struggle to remember a location, generalize to an unseen level, or coordinate with another agent. GlyphBench brings several of these established single-agent task families into a common language-model interface, so researchers can examine how an intervention affects different demands on the policy.

Exploration and dependencies between goals. Crafter evaluates progress through achievements that require resource collection, survival, and crafting, connecting short decisions to longer sequences of prerequisites (Hafner, 2022). Craftax expands the mechanics and progression while using a fast JAX implementation to support large RL experiments (Matthews et al., 2024). NetHack adds a much larger space of interactions and difficult exploration problems (Küttler et al., 2020). In Minecraft, MineRL pairs human demonstrations with a simulator to study sample-efficient learning, while MineDojo connects open-ended tasks to large collections of external knowledge (Guss et al., 2019; Fan et al., 2022). The cost of studying these dependencies grows with the length and complexity of an episode. GlyphBench addresses this tradeoff with focused tasks for repeated optimization experiments at bounded interaction budgets, alongside extended environments in which agents pursue goals across longer sequences of decisions.

Text as an interface to a world. TextWorld allows researchers to generate text games with controlled difficulty and language, including variations in partial observability and reward sparsity (Côté et al., 2018). Jericho exposes human-authored interactive fiction, where agents face language understanding, commonsense reasoning, and large combinatorial action spaces (Hausknecht et al., 2020). These environments established text as an interface to worlds in which actions change what the agent can observe and do next. GlyphBench extends this use of text through observations that place objects in a two-dimensional grid and supply a legend and status display. By exposing adjacency and geometry directly while retaining a discrete action vocabulary, the interface makes spatial reasoning a central part of language-based interaction.

Using pretrained knowledge during interaction. ReAct interleaves reasoning with actions and feedback, providing a general pattern for adapting plans as observations arrive (Yao et al., 2023). In Crafter, SPRING organizes reasoning around game knowledge extracted from the environment's paper (Wu et al., 2023); in Minecraft, Voyager maintains an expanding library of executable skills (Wang et al., 2024). ELLM uses a language model to propose meaningful goals that guide another agent's exploration, illustrating a different role for pretrained knowledge (Du et al., 2023). CODE-

SHARP studies continual skill discovery and composition through hierarchical reward programs in Craftax (Bornemann et al., 2026). Across these approaches, pretrained knowledge becomes useful through mechanisms that connect it to the current game state: plans, memory, skills, or exploration goals. GlyphBench's harness experiments examine this connection by varying the support available to a model during play, while trajectory inspection helps reveal how the model uses it.

From evaluating agents to training them. BALROG brings existing RL games to language and vision-language models and reports substantial weaknesses in long-horizon decision-making and visual grounding (Paglieri et al., 2025). Agentick evaluates multiple agent paradigms on purpose-built tasks with several observation modalities, finding large effects of reasoning harnesses and observation format (Creus Castanyer et al., 2026). lmgame-Bench studies prompt sensitivity, perception, and memory in a shared game interface (Hu et al., 2025). On the learning side, GLAM updates language-model policies through online environment interaction (Carta et al., 2023). RAGEN examines training instability and the effects of rollout construction, and AgentGym-RL provides a modular framework for multi-turn RL across diverse interactive tasks (Wang et al., 2025; Xi et al., 2026). GlyphBench builds on this combined evaluation and training agenda: a researcher can hold the task interface fixed while changing observations, harnesses, optimization settings, or training mixtures, then inspect the resulting policy behavior.

Transfer beyond the training environment. Automatically verifiable mathematics and coding tasks provide direct outcome feedback for language-model learning (Hendrycks et al., 2021; Chen et al., 2021; Guo et al., 2025). Reasoning Gym broadens this approach through procedurally generated reasoning tasks and verifiers (Stojanovski et al., 2025). Games raise a related question: can rewards from sequential interaction improve performance when the model later faces a different kind of problem? The initial lmgame-Bench study reports transfer from single-game RL to unseen games and external planning tasks (Hu et al., 2025); ViGaL finds that visual-game RL improves performance on multimodal mathematical and spatial reasoning evaluations (Xie et al., 2026). We study this question by training a shared policy on GlyphBench and comparing its transfer with a math-trained baseline and a code-trained baseline. Held-out game seeds measure learning on new instances, while external reasoning evaluations test capabilities outside the training environment. This combination supports broader research on how language models learn from rewards and generalize across problem formats.

B Environment and Interface Details

Each **GlyphBench** task specifies the actions, rewards, termination conditions, and observations in the decision process of Section 3.1. The harness determines which observations, history, and memory the model receives. Standard tasks cap episodes at 512 steps and bound cumulative returns to $[-1, 1]$. The same seed and action sequence reproduce an episode, including its stochastic events.

Table 1: The 362 GlyphBench tasks. Standard tasks bound episodes to 512 steps and returns to $[-1, 1]$. Extended tasks support longer episodes and are excluded from the standard aggregate score.

Suite	Tasks	Track	Task scope
Classics	50	Standard	Puzzles and familiar games for planning, memory, and control.
MiniGrid	71	Standard	Navigation, keys, doors, obstacles, and memory.
MiniHack	63	Standard	Dungeon navigation, items, combat, and skill composition.
Craftax	60	Standard	Focused survival, crafting, combat, and progression tasks.
MiniAtari	43	Standard	Arcade control with compact dynamics and short horizons.
Procgen	16	Standard	Procedural platformers, shooters, and mazes.
Atari	57	Extended	Long-horizon arcade adaptations.
Full Craftax	1	Extended	Full-game survival and progression.
NetHack	1	Extended	Full-game exploration and progression.

B.1 Training interfaces and model actions

Tasks follow Gymnasium's interaction convention (Towers et al., 2024): an episode begins with an initial observation, and each action produces a new observation and reward. The common interface lets researchers apply the same training and evaluation procedure across tasks with different rules and objectives.

A Verifiers adapter presents episodes as multi-turn model interactions with reward scoring (Brown, 2025). The model chooses named actions from the task's vocabulary. Invalid responses consume a turn without advancing the agent's goal. Harnesses can maintain memory through separate model calls; the multitask study in Appendix G uses conversation history without a separate memory store.

B.2 Inspecting an episode through replay

We provide a replay UI that aligns observations, generated reasoning, actions, rewards, and optional memory. In Figure 9, we show the final decision of a successful MiniAtari BankHeist episode. Viewing the observation alongside the agent's reasoning and action makes it possible to examine what information the agent used and how the environment responded.

Researchers can inspect the decisions behind a learning curve: whether an agent navigates more directly, conserves resources, recovers from a mistake, or repeats actions that leave it stuck. Connecting model outputs to their consequences helps turn changes in aggregate return into hypotheses about how the policy learns and interacts.

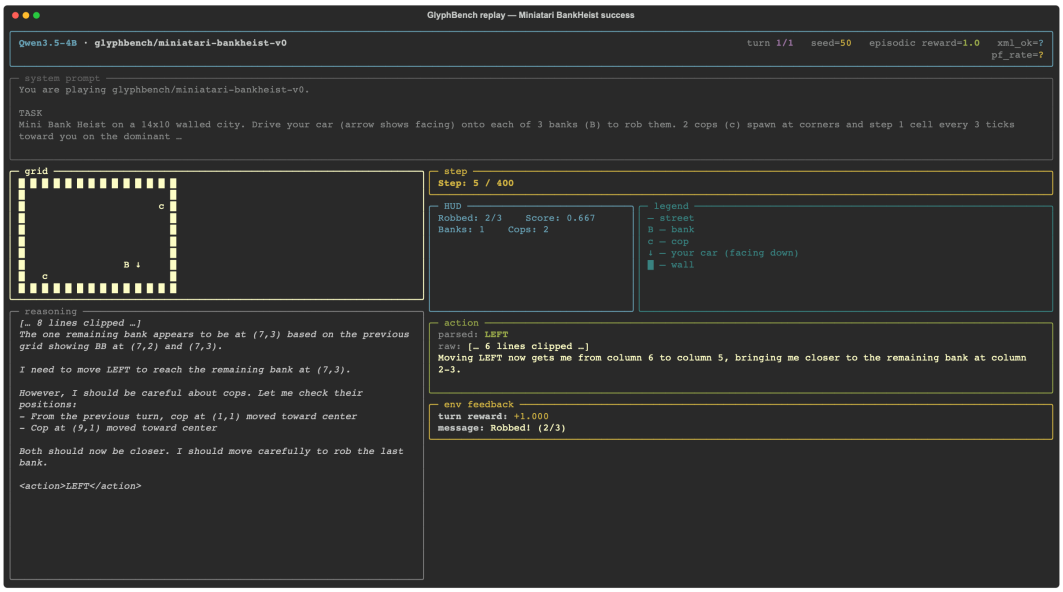

Figure 9: The final decision of a successful MiniAtari BankHeist episode. Replay connects the agent's observation and reasoning to its chosen action and the resulting environment feedback.

C Craftax Observations and Agent Harnesses

Our full Craftax wrapper exposes the original nine-floor game through the same action interface as the standard tasks. Progress requires agents to connect decisions across floors: they must find resources, remember useful locations, and revisit them as their equipment and goals change. We use this setting to study how observations and memory help agents plan, track progress, and adapt as an episode unfolds. Figure 2 compares the three observation formats, and Figure 10 shows how the glyph interface represents each floor. The following sections describe the interfaces and harnesses used in Section 5.1.

C.1 Three observation formats

Pixels. The native renderer supplies a local view with sprites, lighting, and an inventory/status panel. From this image, the model must identify terrain and entities, determine which way the player faces, and read equipment and resource levels. Only the current view is visible.

Native text (original Craftax renderer). We use Craftax's original text renderer as a comparison interface; it is not a GlyphBench observation format. It lists each cell in the local 9×11 window using coordinates relative to the player: $(x, y) = (0, 0)$ is the current cell, positive x points right, and positive y points down. A cell can describe an actor or item together with its underlying terrain. Cells outside the light mask are reported as `Darkness`. Separate fields report inventory, equipment, vitals, attributes, facing, floor, and status flags.

Glyphs. The glyph renderer places one character at each location in the same 9×11 window, preserving the spatial arrangement of visible objects. A directional player glyph shows facing, and a legend identifies the visible symbols. The accompanying HUD reports vitals, inventory, equipment, floor, and absolute row/column position. Like native text, the renderer respects the local light mask. The compact representation merges some distinctions: grass and path share a glyph, for example, and an actor occupies the character that would otherwise show the terrain beneath it.

These design choices offer different ways to recover the game state. Native text explicitly names terrain and overlapping objects; glyphs make adjacency and relative position visible in the grid and supply absolute position in the HUD. Pixels convey appearance and continuous lighting, while both text formats apply a visibility threshold. Together, the interfaces let us examine how agents use spatial structure and explicit state descriptions during play.

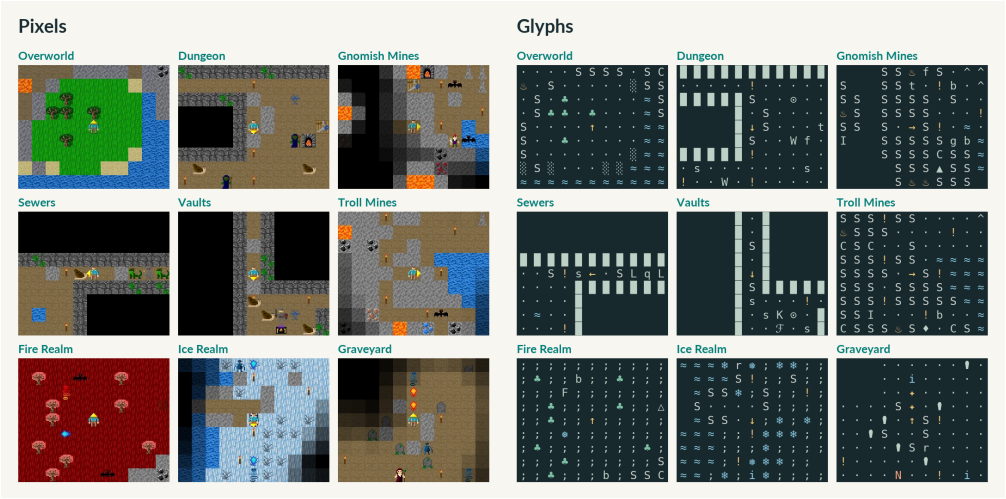

Figure 10: The nine Craftax floors in pixels and glyphs. Each pair shows the same local state, with HUDs and legends omitted. Blank glyph regions are unobserved cells; colors aid visualization only. The paired views show how the glyph interface represents the terrain and entities encountered during progression.

C.2 Basic and Guided Memory

We compare two agent harnesses to study how models use information accumulated during play. The *Basic* harness combines standard game instructions with the growing conversation, allowing the model to consult previous observations and actions. The *Guided Memory* harness organizes this information into a structured scratchpad, remembered landmarks, recent actions, and an exploration summary. It also supplies strategic advice for the current floor and uses separate model calls to update memory. Table 2 details the instructions and state available through each harness. Both choose one native game action per turn and cannot execute scripts. Memory updates store information without acting on the environment.

Table 2: Basic and Guided Memory in the Craftax harness comparison. Both use glyph observations, native game actions, and the selected model reasoning-effort setting. Neither requests an explicit output-token cap.

Component	Basic	Guided Memory
Instructions	Task, core mechanics, observation conventions, and action format	Extended game guide, action format, memory protocol, and current-floor guidance
Current view	Glyph grid and standard wrapper HUD	Glyph grid and expanded inventory/status HUD
History	Growing conversation, with the last eight observation/action frames rendered in each turn	Last eight actions and rewards, plus persistent memory
External memory	No separate memory store	Six scratchpad sections and a floor-keyed landmark database
Exploration	Prior observations in the conversation	Summary of observed tiles, visited coordinates, frontiers, and remembered features
Memory calls	None	Every five steps and after relevant events; separate from action selection

The Guided Memory scratchpad records strategy, plans, lessons, tactical state, and floor notes to help agents revise their decisions as the game changes. Remembered landmarks and an exploration summary connect these plans to locations the agent has visited. The agent updates memory every five steps and after events that may require a revised plan, such as earning a reward, unlocking an achievement, or changing floors.

(a) GLYPHBENCH GLYPHS

```
····SSSS·SC
♨·S·····▒SS
·S·♣·····≈S
·S♣♣·♣···≈S
S····↑···≈≈
S··♣·····≈≈
S·····▒≈≈≈
▒S▒···▒≈≈≈
≈≈≈≈≈≈≈≈≈≈
```

```
HP:9/9 Food:9/9 Drink:9/9 Energy:9/9 Mana:9/9 XP:0
  Pos:(24,24) Facing:up
Str:1  Dex:1  Int:1  Floor:0(Overworld)  T:0
[Inv] Wood:0 Stone:0 Coal:0 Iron:0 Diamond:0 Sapphire:0
  Ruby:0 Sapling:0
[Equip] None Pickaxe | No Sword | No Bow | Arrows:0 |
  Torches:0 | Books:0
[Armor] None
[Potions] Red:0 Green:0 Blue:0 Pink:0 Cyan:0 Yellow:0
[Status] Sleeping:no | Resting:no | Fireball:no |
  Iceball:no | Ladder:OPEN | Boss:no | Day
```

```
--- Legend ---
C — coal ore          · — grass or path     ≈ — water          ♣ — tree
S — stone             ↑ — you, facing up    ▒ — sand           ♨ — lava
```

(b) ORIGINAL CRAFTAX TEXT

```
Map:
-5, -4: path
-4, -4: path
-3, -4: path
-2, -4: path
-1, -4: stone
0, -4: stone
1, -4: stone
2, -4: stone
3, -4: path
4, -4: stone
5, -4: coal
-5, -3: lava
-4, -3: path
-3, -3: stone
-2, -3: grass
-1, -3: grass
0, -3: grass
1, -3: grass
2, -3: grass
3, -3: sand
4, -3: stone
5, -3: stone
-5, -2: path
-4, -2: stone
-3, -2: grass
-2, -2: tree
-1, -2: grass
0, -2: grass
1, -2: grass
2, -2: grass
3, -2: grass
4, -2: water
5, -2: stone
```

```
(continued)
-5, -1: path
-4, -1: stone
-3, -1: tree
-2, -1: tree
-1, -1: grass
0, -1: tree
1, -1: grass
2, -1: grass
3, -1: grass
4, -1: water
5, -1: stone
-5, 0: stone
-4, 0: grass
-3, 0: grass
-2, 0: grass
-1, 0: grass
0, 0: grass
1, 0: grass
2, 0: grass
3, 0: grass
4, 0: water
5, 0: water
-5, 1: stone
-4, 1: grass
-3, 1: grass
-2, 1: tree
-1, 1: grass
0, 1: grass
1, 1: grass
2, 1: grass
3, 1: grass
4, 1: water
5, 1: water
```

```
(continued)
-5, 2: stone
-4, 2: grass
-3, 2: grass
-2, 2: grass
-1, 2: grass
0, 2: grass
1, 2: grass
2, 2: grass
3, 2: sand
4, 2: water
5, 2: water
-5, 3: sand
-4, 3: stone
-3, 3: sand
-2, 3: grass
-1, 3: grass
0, 3: grass
1, 3: sand
2, 3: sand
3, 3: water
4, 3: water
5, 3: water
-5, 4: water
-4, 4: water
-3, 4: water
-2, 4: water
-1, 4: water
0, 4: water
1, 4: water
2, 4: water
3, 4: water
4, 4: water
5, 4: water
```

```
Inventory:
Wood: 0
Stone: 0
Coal: 0
Iron: 0
Diamond: 0
Sapphire: 0
Ruby: 0
Sapling: 0
Torch: 0
Arrow: 0
Book: 0
Red potion: 0
Green potion: 0
Blue potion: 0
Pink potion: 0
Cyan potion: 0
Yellow potion: 0
Health: 9.0
Food: 9
Drink: 9
Energy: 9
Mana: 9
XP: 0
Dexterity: 1
Strength: 1
Intelligence: 1
Direction: up
Light: 0.7969252467155457
Is Sleeping: False
Is Resting: False
Learned Fireball: False
Learned Iceball: False
Floor: 0
Ladder Open: True
Is Boss Vulnerable: False
```

Figure 11: Complete environment observations for the initial Craftax state shown in Figure 2. (a) GlyphBench's glyph grid, full HUD, and symbol legend. (b) Craftax's original text renderer: all 99 cells, inventory, and status information. The map listing continues down three columns. Line wrapping and column placement are adjusted for readability.

For the main comparisons, we evaluate each condition on five shared environment seeds using the model identifiers `gpt-5.6-luna`, `gpt-5.6-terra`, and `gpt-5.6-sol`. Both harnesses use the selected reasoning-effort setting without an additional output-token cap. Guided Memory also makes separate calls to maintain its memory, allowing us to examine whether models can use this additional support to improve progression.

Exploratory Astra run. We report one exploratory Astra run with glyph observations, Guided Memory, and maximum reasoning effort to illustrate how far a frontier model can progress with this combination. Astra reaches 51.4% of maximum reward after 2,304 steps, shown by the star in Figure 4.

An action turn in each harness. For illustration, an agent returning to a crafting table on its right might output `<action>MOVE_RIGHT</action>`. Basic chooses this action from the observation and conversation history. Guided Memory also consults its saved plan, landmarks, recent actions, and floor guidance. Either way, the environment executes one move and returns a new observation. A separate memory call can record the table's location for later decisions.

C.3 Prompt excerpts

The following excerpts show how the harnesses guide reasoning and maintain persistent state. We reproduce the instructions verbatim, with line wrapping adjusted for typesetting. The full prompts also describe game mechanics, available actions, and observation conventions.

Basic: reasoning and action format.

> "No explicit output-token cap is imposed: use the model's native reasoning budget and take as much reasoning as is useful."
>
> "After reasoning, end your response with exactly one final action tag on its own line."

Guided Memory: separating action selection from memory.

> "Discuss candidate actions as plain text while reasoning; only the final `<action>...</action>` tag is executed. Do NOT update memory on this turn."
>
> "Do all of your strategic bookkeeping here: update the plan, record lessons, note what you explored, and maintain the landmark map."
>
> "GROUNDING: the [Observation] is always ground truth. Read it before trusting memory; memory may be stale. Use [Recent Actions] to detect loops — if an action keeps failing or you are oscillating, change approach."

Guided Memory: an example of floor-specific guidance.

> "Progress promptly: once you have a stone pickaxe, stone sword, `>=4` spare wood, and safe food/drink/energy, locating the open ladder and descending is the main objective. Coal/iron upgrades are useful but OPTIONAL on floor 0; do not overfarm common stone or delay indefinitely for iron armour because floor 2 is far richer in ore."

The following excerpt from the example memory update in the harness instructions shows how the agent records its plan and a landmark. It is an instructional example, not an evaluated episode.

```
PLAN:
1. [x] Place crafting table
2. [ ] Mine 3 stone for a furnace
MAP_ADD: area=0, type=water, row=20, col=30, note=large lake
MAP_ADD: area=0, type=crafting_table, row=24, col=24, note=mine
```

The floor-specific guidance helps the agent decide when preparation is sufficient and further resource collection would delay progress. The agent can combine this guidance with its memory of earlier decisions to choose its next objective.

D BALROG OBSERVATION INTERFACES

BALROG presents game states through environment-specific language interfaces (Paglieri et al., 2025). Table 3 summarizes the original formats used in our comparison. Our glyph variant presents spatial information through two-dimensional grids of symbols, following GlyphBench's observation design. The position of each symbol directly expresses its location relative to the surrounding cells.

Table 3: Original BALROG observation formats for the four environments in the interface comparison. These are BALROG's environment-specific representations, distinct from GlyphBench's shared glyph format.

Environment	Original BALROG language interface
BabyAI	Descriptions of visible objects and their positions relative to the agent, including door state and carried objects, together with the language task.
Baba Is AI	Active rules and descriptions of objects and movable rule-word blocks using horizontal and vertical offsets from the controlled object.
Crafter	Descriptions of nearby terrain and entities by distance and direction, the object directly ahead, survival statistics, and inventory.
MiniHack	Language-wrapper descriptions of surroundings, inventory, and player statistics, supplemented by a two-dimensional ASCII map.

For BabyAI, Baba Is AI, and Crafter, glyph observations make spatial relationships explicit in the layout of the grid. MiniHack already supplies an ASCII map, so its comparison tests an alternative symbolic rendering. Figure 12 reports the resulting progression scores for each model and environment.

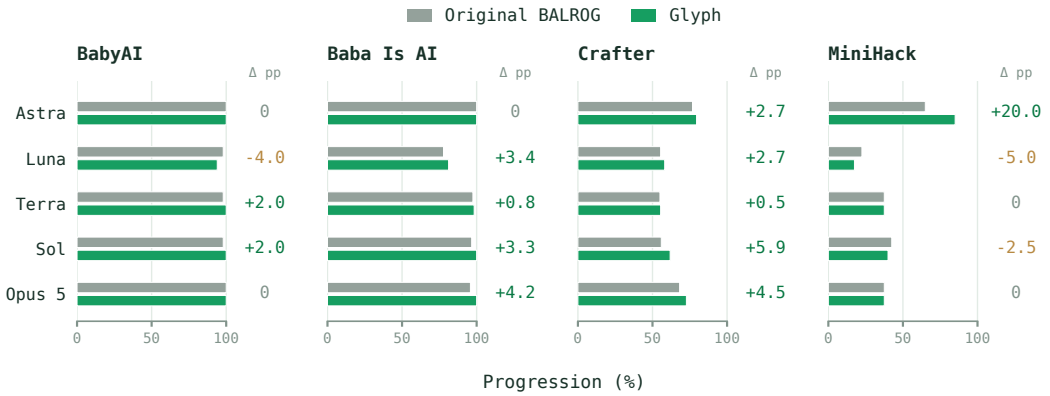

Figure 12: We compare the original BALROG interfaces with glyph observations on four environments. Bars show mean progression over paired episodes; differences are percentage points. Table 3 describes each original interface, including MiniHack's existing ASCII map.

Glyph rendering increases 12 of 20 model-environment means, leaves five unchanged, and lowers three. All five models improve in mean progression on Crafter, while the largest increase occurs for Astra on MiniHack, from 65% to 85%. These comparisons show that the effect of the observation interface depends on both the environment and the model.

E TRAINING OBJECTIVES AND SETTINGS

We start all training experiments from Qwen3.5-4B and use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, zero weight decay, gradient-norm clipping at 1, and a constant learning rate. Table 4 summarizes the settings used to establish single-task learnability, study optimization in Snake, and train on the multitask mixture.

Table 4: Training settings. Output budgets apply per action-generation call; sequence length is the trainer's token limit. Snake learning rates and group sizes are swept as described in Section 4.

Setting	Single-task runs	Snake	Multitask
Learning rate	10^{-6}	Sweep	5×10^{-6}
Action output tokens	8,192	4,096	4,096
Trainer sequence length	32,768	32,768	65,536
Trainer loss	`default`	`ipo`	`ipo`

E.1 GROUP-RELATIVE ADVANTAGES AND POLICY UPDATES

Snake and multitask training use Prime-RL v0.9.0's group-relative estimator and the loss named `ipo` in that implementation (Prime Intellect, 2025b). Within the eligible trajectories of a rollout group $\mathcal{G}$, the advantage is

$$A_i = R_i - \frac{1}{|\mathcal{G}|} \sum_{j \in \mathcal{G}} R_j. \tag{2}$$

We assign this scalar to each trainable sampled token in trajectory i, so actions and reasoning tokens share the trajectory-level learning signal. Returns are mean-centered without standard-deviation normalization or a length penalty.

For sampled token y_{ik} with model context h_{ik}, let $p_{ik} = \pi_\theta(y_{ik} \mid h_{ik})$ and let q_{ik} be its recorded rollout-policy probability. Define $\rho_{ik} = p_{ik}/q_{ik}$ and let m_{ik} select trainable RL tokens. The loss is

$$\mathcal{L} = \frac{1}{N} \sum_{i,k} m_{ik} \left[-\mathbf{1}\{|p_{ik} - q_{ik}| \le 0.1\} \, A_i \rho_{ik} + 10^{-3} (\log \rho_{ik})^2 \right], \tag{3}$$

where $N = \sum_{i,k} m_{ik}$ counts trainable RL tokens across the batch. To limit the contribution of tokens whose probabilities have changed substantially, the probability-difference mask applies to the policy-gradient term; all trainable tokens still contribute to the penalty and denominator. The penalty measures change from the rollout policy. For the single-task learnability experiments, we use the trainer's `default` loss, with `dppo_mask_low` and `dppo_mask_high` both 0.2, $\tau_{\text{adv}} = 1$, and $\tau_{\text{KL}} = 0.001$.

E.2 TASK SAMPLING AND INTERACTION HISTORY

In Snake and multitask training, each rollout group contains trajectories from the same task. We cycle through the selected tasks and vary the game instances sampled within each group, giving the policy repeated exposure to each task's rules while changing the situations it encounters. Held-out evaluation tests new instances of those tasks.

During these runs, the policy conditions on conversation history and generates actions without separate memory-update calls. We sample at temperature 1 and treat invalid actions as no-ops. To study how best to allocate a fixed rollout budget, the Snake experiments vary group size and the resulting number of updates together. The multitask run instead uses 32 groups per update to combine learning signals from different tasks. We use these complementary settings to examine interactions between the learning rate, rollout grouping, and policy-update frequency.

F SINGLE-TASK LEARNING: FULL RESULTS

We train separate policies across all six standard suites to check that the shared interface and task rewards support learning. In Figure 13, we show all 25 learning curves. For the main-text figure, we select the alphabetically first and last task within each suite, giving 12 examples chosen independently of their learning outcomes.

The curves reveal both rapid improvement and slower, noisier learning. Because the vertical scales are task-specific, their heights should not be used to rank tasks or suites. Pale lines show the batch-level variation, while centered seven-point means make the overall trajectory easier to follow. Each curve measures training reward from one run. Together, they illustrate the range of learning dynamics that researchers can study with a common training pipeline.

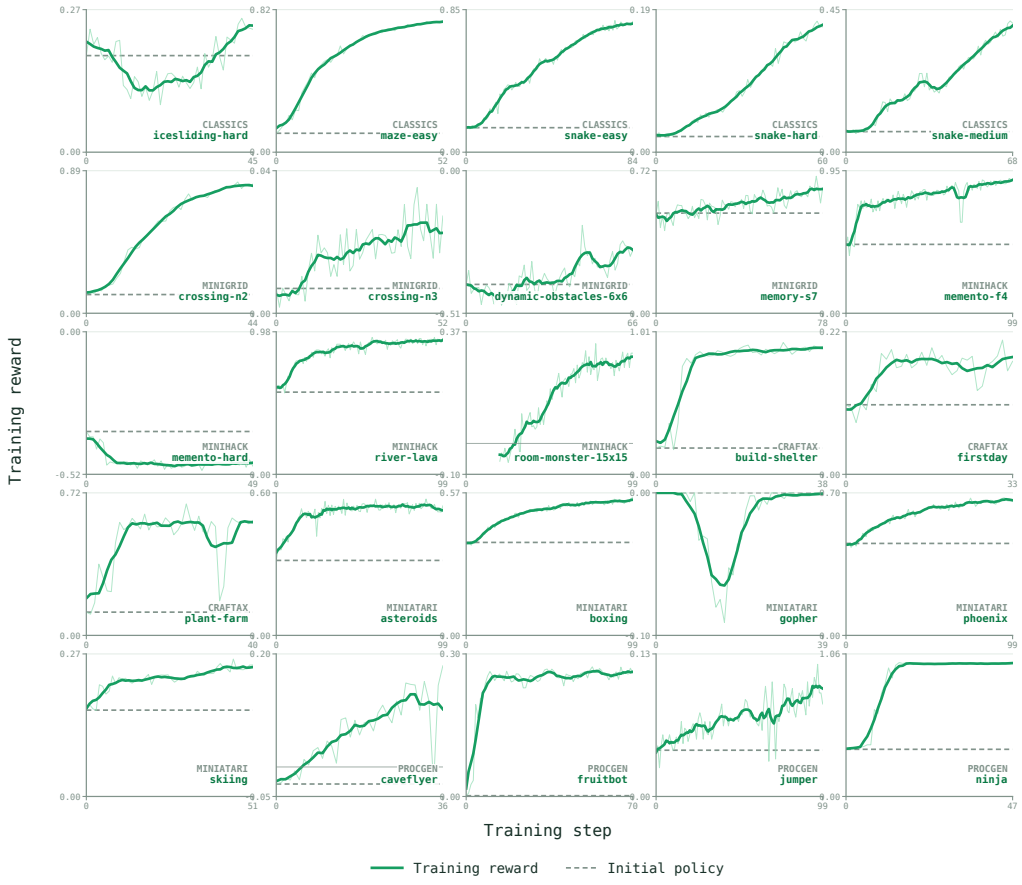

Figure 13: All 25 single-task learning curves, ordered by suite and task. Pale traces show training rewards and dark traces show centered seven-point means. Dashed lines show initial-policy rollout means where available. Endpoint ticks indicate each task's training-step and reward range. All runs start from Qwen3.5-4B and use one training seed.

G Multitask Training and Evaluation

G.1 Task mixture and optimization

We train a shared policy on a mixture of 100 tasks: 18 each from Classics, Craftax, MiniAtari, and MiniGrid, 19 from MiniHack, and nine from Procgen. Evaluation uses the same 100 task IDs with held-out seeds, measuring generalization to new instances of the training tasks.

Each update contains 1,024 rollouts in groups of 32. We use AdamW with learning rate 5×10^{-6}, $(\beta_1, \beta_2) = (0.9, 0.95)$, gradient clipping at 1, no weight decay, and a constant schedule. The IPO-style loss uses $\tau_{\mathrm{KL}} = 0.001$, $\tau_{\mathrm{adv}} = 1$, and threshold 0.1. Appendix E defines the loss and task sampling. The sequence length is 65,536, with an output budget of 4,096 tokens per turn and no separate persistent memory store.

G.2 Evaluation and learning curves

We evaluate every 50 gradient steps on 200 examples drawn from four held-out seeds and weight the 100 task returns equally in the aggregate score. Here we show the full learning curves and use only observations within each displayed interval for smoothing.

In Figure 14, we show how held-out performance develops in each suite. We then plot all 100 task-level training curves in Figures 15–16 to examine differences in the pace and stability of learning. These curves track the same shared policy throughout training.

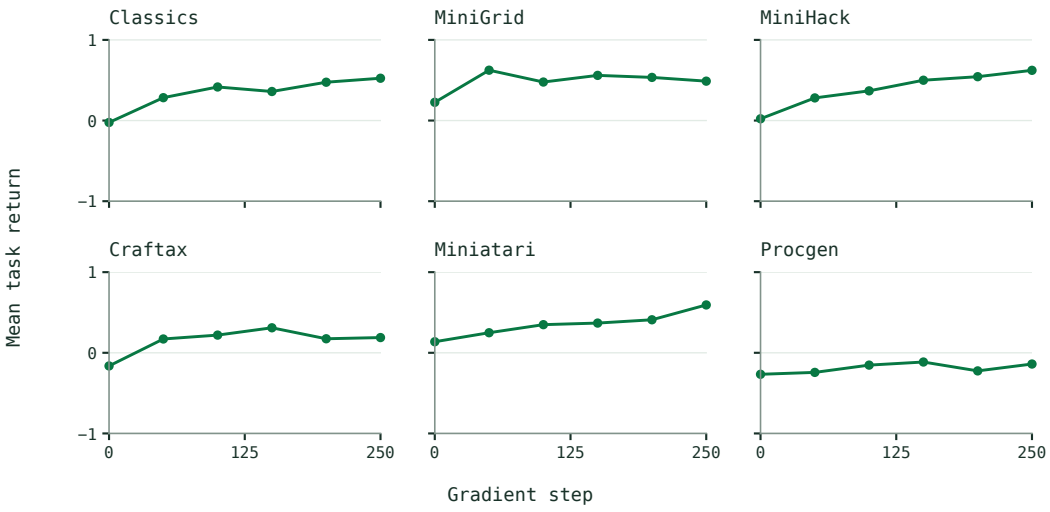

Figure 14: We evaluate the multitask policy on new environment seeds across all six training suites, using a common return scale. Each point averages task means within a suite. All suites improve over the displayed interval, while Procgen remains below zero.

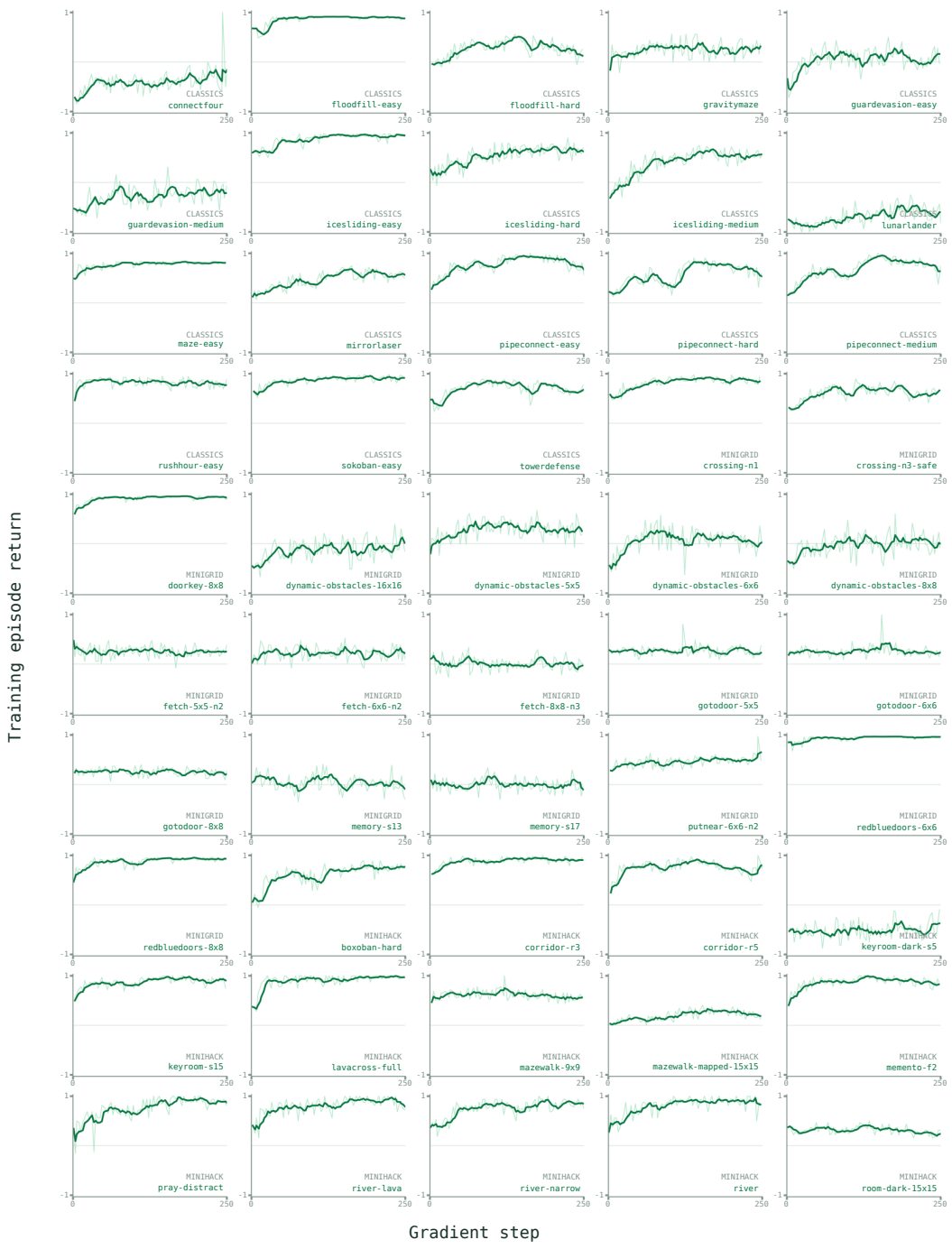

Figure 15: The multitask policy learns jointly on all 100 tasks; here we show tasks 1–50, ordered by suite and task. Pale lines show task-mean batch returns; dark lines show trailing 15-gradient-step means. All panels share steps 0–250 and returns $[-1, 1]$.

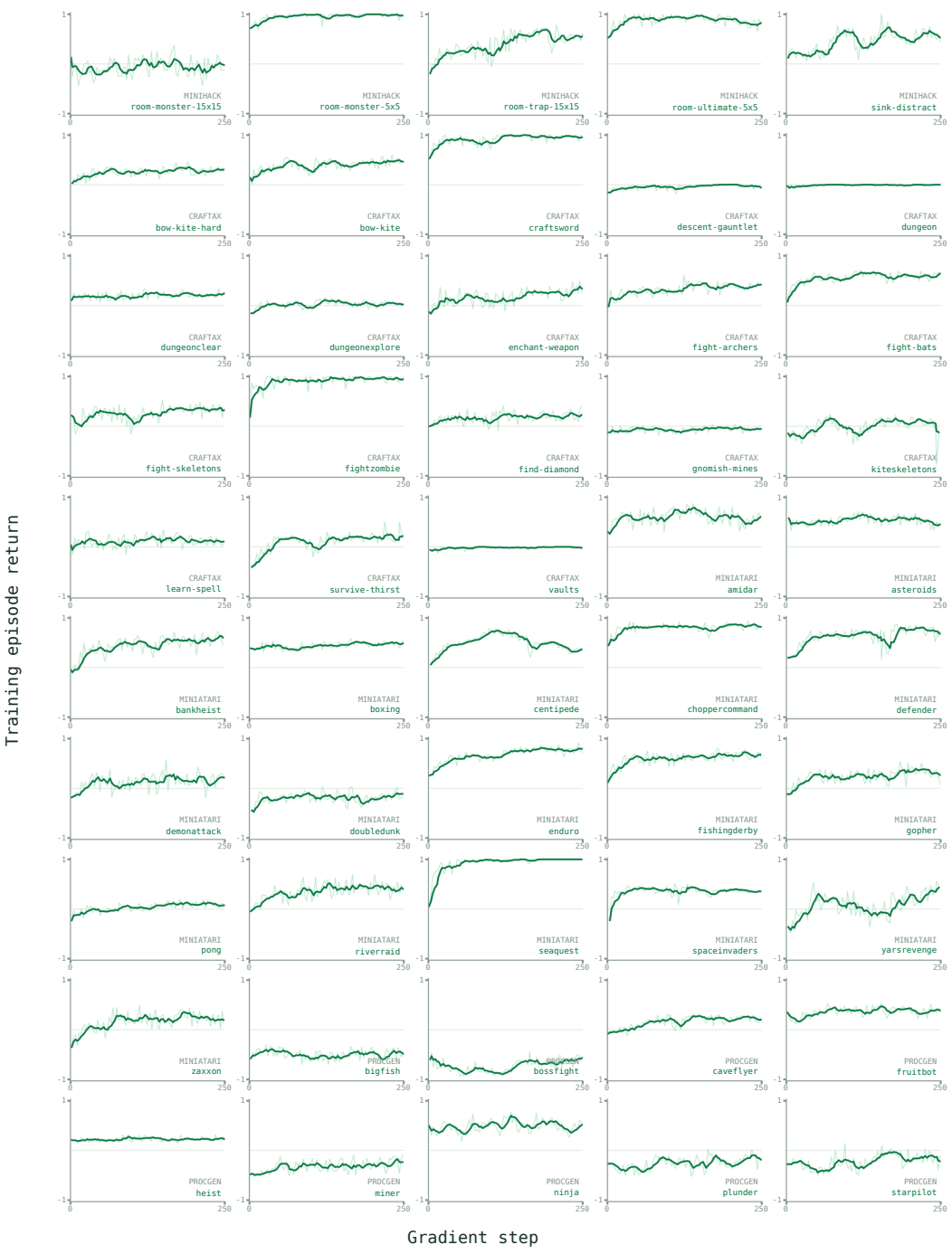

Figure 16: Training curves for the remaining 50 tasks of the same multitask policy. Axes and smoothing match Figure 15; together, the figures show each task once.

H External Transfer: Protocols and Full Results

To measure how game training affects performance in other domains, we compare the base Qwen3.5-4B and GlyphBench RL policy on reasoning, mathematics, and software-engineering benchmarks excluded from **GlyphBench** RL training.

H.1 Sampling and inference budgets

Reasoning Gym. We evaluate 31 tasks with 100 problems each and three completions per problem, totaling 3,100 problems and 9,300 attempts per model at each budget. Both models receive identical problems at output budgets of 2K, 8K, and 16K tokens. Solve rate averages success over all attempts, counting malformed or unparseable answers as failures.

Mathematics. MathArena evaluation covers seven AIME/HMMT datasets: 183 problems with four generations each, totaling 732 attempts per model. The output ceiling is 65,500 tokens, and accuracy is measured over all attempts without best-of-four selection. The 2K/8K/16K sweep applies only to Reasoning Gym.

Software engineering. We evaluate both policies on the same 500 SWE-bench Verified instances and 576 SWE-bench Pro instances, denoted Pro-576. Each episode allows at most 50 agent turns and 32,768 output tokens per model call. A patch succeeds if it passes the benchmark's tests.

H.2 Paired uncertainty

We compute paired bootstrap intervals from 10,000 resamples. Reasoning Gym resamples problems within tasks, keeping each problem's three attempts together. MathArena likewise groups each problem's four generations; SWE-bench resamples paired instances. These intervals capture evaluation uncertainty for one checkpoint, not variation across training seeds. Changes are computed from unrounded rates and may differ from the subtraction of displayed values.

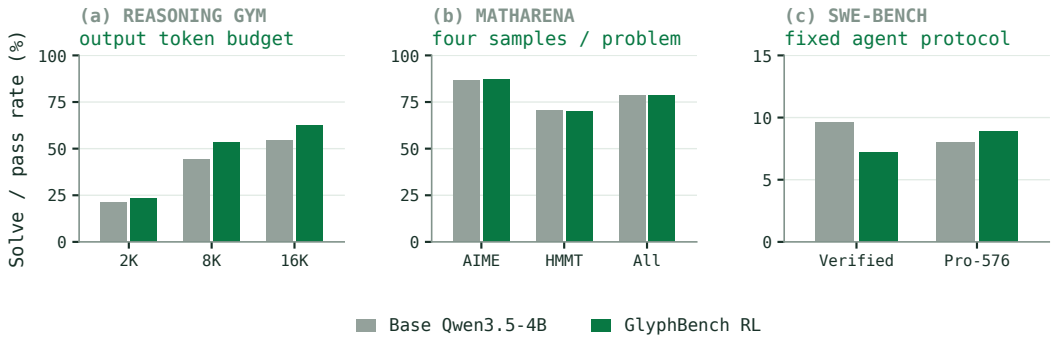

Figure 17: We evaluate transfer beyond gameplay by comparing performance before and after GlyphBench RL. (a) Reasoning Gym solve rates at three output budgets. (b) MathArena accuracy over all attempts. (c) SWE-bench pass rates. Bars start at zero; panel (c) spans 0–15%. Table 5 reports paired changes and uncertainty.

H.3 Results across domains

We also evaluate transfer on the 23 non-game Reasoning Gym tasks. At 8K, their solve rate improves from 48.52% to 58.12%, a gain of 9.59 points with a 95% interval of $[8.54, 10.65]$. The improvement extends beyond the benchmark's game category. Tables 5–6 report aggregate and category results.

We find little aggregate change on MathArena, where a small AIME increase is offset by a small HMMT decrease. On SWE-bench, Verified declines in its point estimate and Pro-576 increases slightly; neither change establishes an improvement. We report the numerical breakdown and paired intervals in Table 5. Exact two-sided McNemar tests give $p = 0.0807$ for Verified and $p = 0.5758$ for Pro-576. In these evaluations, the clearest transfer gains occur on Reasoning Gym.

Table 5: External evaluation of the base and GlyphBench RL models. Rates are percentages; changes and paired 95% intervals are percentage points. N counts attempts for mathematics and reasoning, and instances for coding. Resampling clusters attempts by problem.

Benchmark	N	Base	GlyphBench RL	Δ (pp)	Paired 95% CI (pp)
MathArena aggregate	732	78.55	78.42	-0.14	$[-2.46, +2.19]$
AIME 2024 I	60	90.00	93.33	$+3.33$	$[+0.00, +10.00]$
AIME 2024 II	60	90.00	90.00	$+0.00$	$[-5.00, +5.00]$
AIME 2025	120	85.00	85.00	$+0.00$	$[-5.00, +5.00]$
AIME 2026	120	85.00	85.00	$+0.00$	$[-5.83, +5.83]$
HMMT Feb 2025	120	70.00	64.17	-5.83	$[-14.17, +2.50]$
HMMT Nov 2025	120	77.50	80.83	$+3.33$	$[-1.67, +8.33]$
HMMT Feb 2026	132	65.15	65.15	$+0.00$	$[-4.55, +4.55]$
SWE-bench Verified	500	9.60	7.20	-2.40	$[-4.80, +0.00]$
SWE-bench Pro-576	576	7.99	8.85	$+0.87$	$[-1.56, +3.30]$
Reasoning Gym 2K	9,300	21.22	23.40	$+2.18$	$[+1.51, +2.88]$
Reasoning Gym 8K	9,300	44.11	53.33	$+9.23$	$[+8.38, +10.09]$
Reasoning Gym 16K	9,300	54.23	62.58	$+8.35$	$[+7.43, +9.29]$

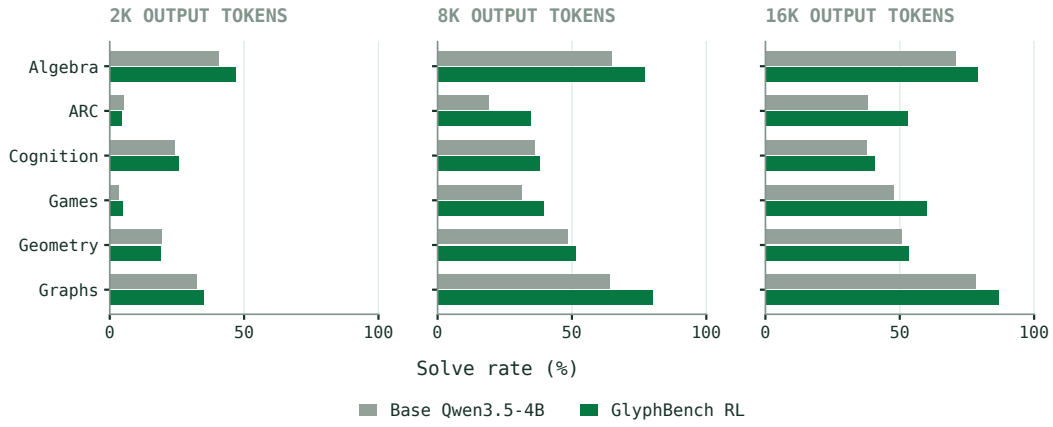

Figure 18: Reasoning Gym performance before and after GlyphBench RL by category and output budget, on a common 0–100% scale. Every category improves at 8K and 16K; the smaller 2K budget yields a weaker and less consistent benefit. Table 6 reports the corresponding values.

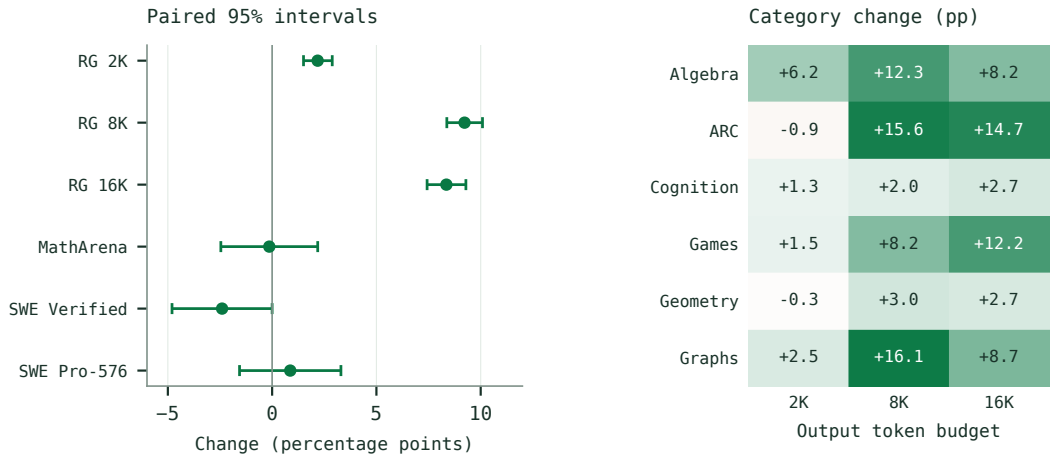

Figure 19: Changes after GlyphBench RL on external benchmarks, in percentage points. The left panel shows aggregate changes with paired 95% bootstrap intervals; the right separates Reasoning Gym changes by category and output budget. Positive estimates whose intervals include zero do not establish an improvement.

Table 6: Reasoning Gym solve rates (%): base / GlyphBench RL, with changes in percentage points. Categories contain 6, 3, 7, 8, 2, and 5 tasks in row order, each with 100 problems and three attempts per problem.

Category	2K: base / ours (Δ)	8K: base / ours (Δ)	16K: base / ours (Δ)
Algebra	40.61 / 46.78 (+6.17)	64.72 / 77.06 (+12.33)	71.00 / 79.22 (+8.22)
ARC	5.22 / 4.33 (−0.89)	19.22 / 34.78 (+15.56)	38.22 / 52.89 (+14.67)
Cognition	24.29 / 25.62 (+1.33)	36.05 / 38.00 (+1.95)	37.86 / 40.57 (+2.71)
Games	3.38 / 4.92 (+1.54)	31.42 / 39.58 (+8.17)	47.88 / 60.08 (+12.21)
Geometry	19.33 / 19.00 (−0.33)	48.50 / 51.50 (+3.00)	50.67 / 53.33 (+2.67)
Graphs	32.53 / 35.00 (+2.47)	64.13 / 80.20 (+16.07)	78.20 / 86.93 (+8.73)

I Math and Code Baselines for Reasoning Transfer

We compare GlyphBench RL with a math-trained baseline and a code-trained baseline to test how reasoning gains from gameplay transfer relative to these common RL domains. All policies start from the same Qwen3.5-4B checkpoint. We evaluate the trained policies on identical Reasoning Gym problems using a shared generation and scoring procedure within each output budget.

I.1 Training data and learning curves

Math RL uses 100 fixed problems from DeepMath-103K (He et al., 2026), restricted to high-difficulty questions with scalar numerical answers. We stratify the selection by topic and difficulty. Ambiguous, proof-based, and multipart formats are excluded; rewards use deterministic Math-Verify grading without an LLM judge, and teacher solutions are not supplied to the policy.

Code RL uses 100 fixed problems from the code portion of INTELLECT-3-RL (Prime Intellect, 2025a). Filtering removes labelled LiveCodeBench rows, duplicate questions, interactive tasks, unrendered visual dependencies, and tasks requiring arbitrary valid outputs. Sampling stratifies by historical solve rate, source, and prompt length. A response receives reward only if it passes every supplied test in the pinned execution environment.

Both baselines use the group-relative advantages and IPO-style objective described in Appendix E, with learning rate 5×10^{-6}, groups of 32, and batches of 1,024 rollouts. Baseline training uses single-turn responses with a 32,768-token output allowance and 65,536-token context. Figure 20 shows the recorded learning curves and held-out diagnostics in each training domain.

The math policy rapidly improves its domain reward and then levels off, whereas code learning is slower and noisier. Both also reduce the average length of responses in their held-out domain diagnostics. Reward heights across the two domains are not directly comparable: the verifiers, problem distributions, and opportunities for improvement differ. We compare their reasoning capabilities on the shared external evaluation below.

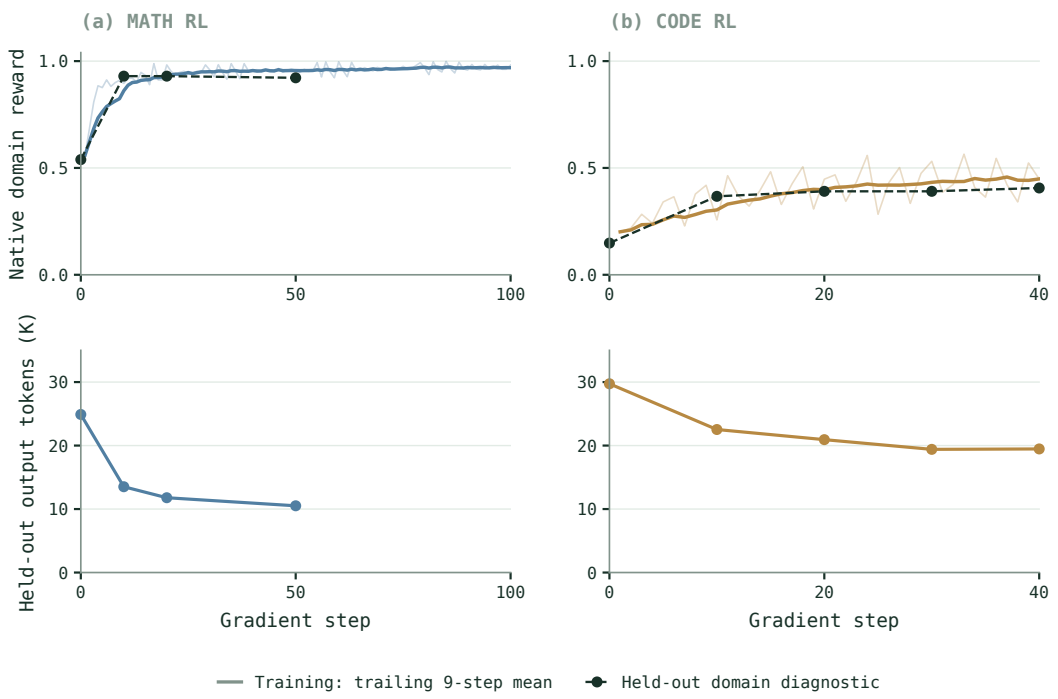

Figure 20: Math and code RL improve rewards in their respective training domains. The top panels show training reward (pale), trailing nine-step means (solid), and held-out domain reward (dashed). The bottom panels show average output tokens on held-out problems from each domain. These rewards measure learning within each domain; we evaluate reasoning transfer separately on Reasoning Gym.

I.2 Reasoning Gym protocol

We use 31 tasks with 100 problems per task and three completions per problem. Each model receives the same 3,100 problems and produces 9,300 attempts at each cap. Sampling uses temperature 0.6, top-p 0.9, the DeepSeekZero system prompt, and `<answer>` extraction.

Strict per-completion accuracy counts a native verifier score of at least 1 as success and divides by all attempts. Partial credit, malformed answers, and grading exceptions do not count as successes. Observed pass@3 instead counts a problem as solved if any of its three recorded completions succeeds. Overall accuracy weights tasks equally; because all tasks have the same number of attempts, this is also the mean over completions. It is not the unweighted mean of the six category means.

The main comparison uses a 32,768-token output cap with a 65,536-token model context. The supplementary 2K, 8K, and 16K evaluations use a 32,768-token context. Each cap is a separate sampling run, rather than a truncated prefix of the same responses. Within a cap, all four models use the same inference configuration.

I.3 Accuracy, paired differences, and category results

At 32K, GlyphBench RL achieves the highest accuracy. Relative to the base model, its gain is 7.44 percentage points; relative to Math RL and Code RL, the gains are 1.19 and 4.51 points. Table 7 reports paired uncertainty, computed by resampling problems within tasks and retaining each problem's three completions together. The 10,000 bootstrap resamples quantify evaluation uncertainty for the trained policies; variability across independent training runs requires repeated experiments.

Table 7: GlyphBench RL minus each comparison policy at the 32K output budget. Differences and paired 95% bootstrap intervals are percentage points.

Reference policy	Accuracy difference	Paired 95% interval
Base	$+7.44$	$[+6.55, +8.34]$
Math RL	$+1.19$	$[+0.38, +2.02]$
Code RL	$+4.51$	$[+3.62, +5.40]$

GlyphBench RL outperforms the math- and code-trained baselines on algebra, ARC, games, and graphs, while Math RL leads on cognition and geometry. In Figure 21, we compare the policies at additional output budgets. Math RL leads at 2K and 8K, and its difference from GlyphBench RL is inconclusive at 16K. These patterns suggest that the reasoning capabilities learned during RL interact with the compute available at evaluation.

I.4 Response length and multiple attempts

Figure 22 separates three aspects of generation: whether any of three samples succeeds, how often generation reaches its length cap, and how many output tokens are used on average. At 32K, GlyphBench RL reduces truncation from 22.55% to 6.54% and mean output length from about 12.1K to 7.7K tokens relative to the base model. Higher accuracy therefore does not require longer average responses in this comparison. Math and code training also reduce truncation, so this behavior alone does not explain the difference between training domains.

The choice of metric matters as well. At 32K, Math RL has a slightly higher observed pass@3 than GlyphBench RL (74.74% versus 74.19%), although the paired interval for their difference includes zero. This measure uses the same three completions as the accuracy analysis and introduces no additional generation. A model can have higher average per-completion accuracy without solving more distinct problems across three tries.

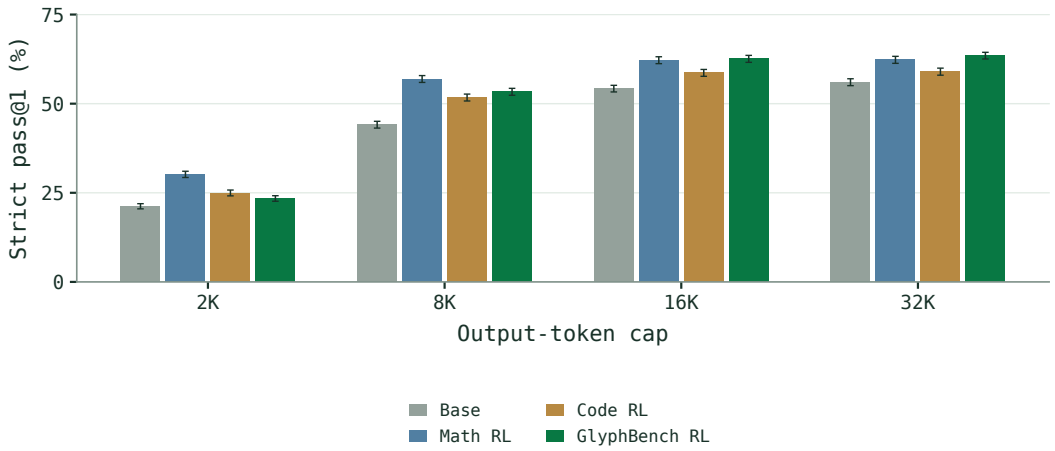

Figure 21: We compare Reasoning Gym accuracy at additional output-token budgets. Each cap uses independently sampled responses to the same problems. Whiskers show 95% problem-bootstrap intervals; all three attempts for a problem remain in the same resampled cluster.

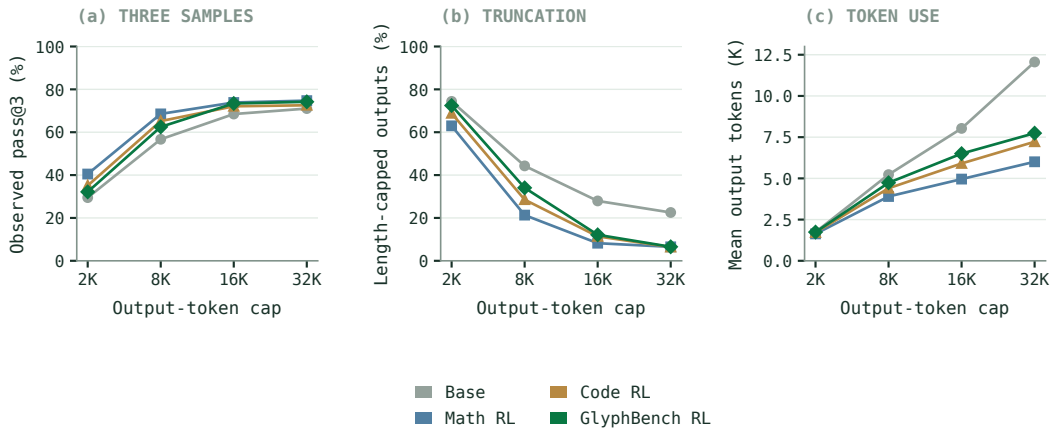

Figure 22: Reasoning Gym generation diagnostics. (a) Problems with at least one successful completion among three attempts. (b) Responses ending at the output-token limit. (c) Mean output tokens per completion. Truncated responses remain in the denominator and can still count as correct if the answer is extractable.

J SNAKE: LEARNING-RATE STABILITY

We use the Snake experiments to examine why higher learning rates can produce promising early improvements followed by collapse. Comparing each configuration's peak and final return shows which gains survive continued optimization.

The main Snake study covers 60 conditions: three difficulties, five group sizes, and four learning rates from 10^{-6} to 10^{-5}. Figure 6 shows the 45 conditions at the three lower rates. We also evaluate six exploratory conditions at 10^{-4}.

Figure 23 compares peak and final observed returns within the 6,144-rollout budget, both measured from trailing 512-rollout means.

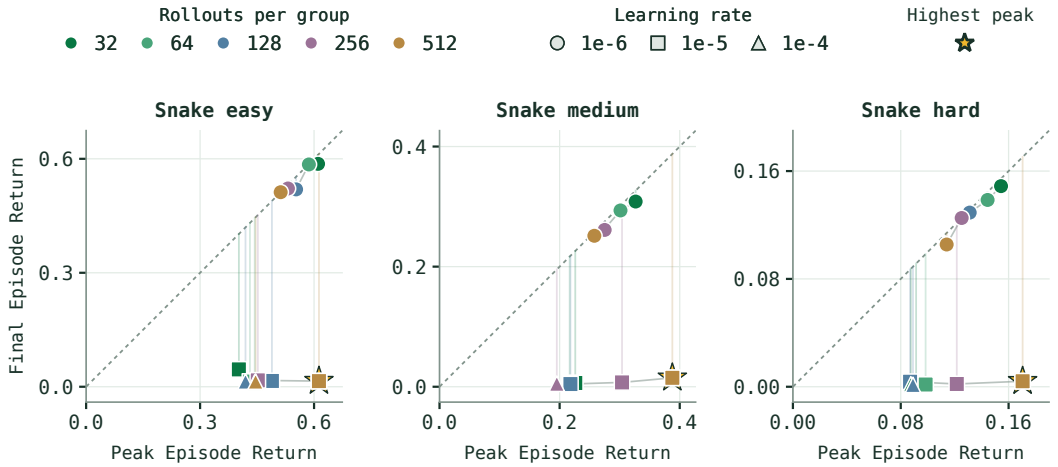

Figure 23: Peak versus final observed Snake return at 10^{-6}, 10^{-5}, and 10^{-4} within 6,144 rollouts. Both coordinates use trailing 512-rollout means. Colors indicate group sizes, shapes indicate learning rates, and gray paths connect increasing group sizes within a rate. The diagonal marks full retention of the peak; vertical stems show the drop to final return. Gold stars mark each difficulty's largest peak among plotted conditions. The 10^{-4} sweep is exploratory.

The stability comparison reveals different effects of group size across learning rates. At 10^{-6}, larger groups generally reach lower peak and final returns. At 10^{-5}, group 512 reaches the highest peak among the plotted conditions on every difficulty, yet ends near zero. On easy Snake, for example, return falls from a peak of 0.613 to 0.015. Selecting a configuration by peak return alone would obscure this instability: an apparent gain may not survive continued optimization. These collapses motivated the intermediate learning rates in Figure 6.

K SNAKE: EXTENDING THE ROLLOUT BUDGET

To test whether the larger group catches up with more interaction, we compare groups 32 and 512 at learning rate 10^{-6} over 25,600 rollouts on all three difficulties. These six runs form a separate study from the 6,144-rollout sweep.

With this larger budget, group 32 achieves the higher final return on all three tasks. The gap is largest on easy Snake (0.824 versus 0.585) and remains positive as difficulty increases. Thus the small-group advantage at this learning rate persists beyond the early phase of learning. This comparison extends the fixed-interaction analysis; it still couples group size to the number of policy updates.

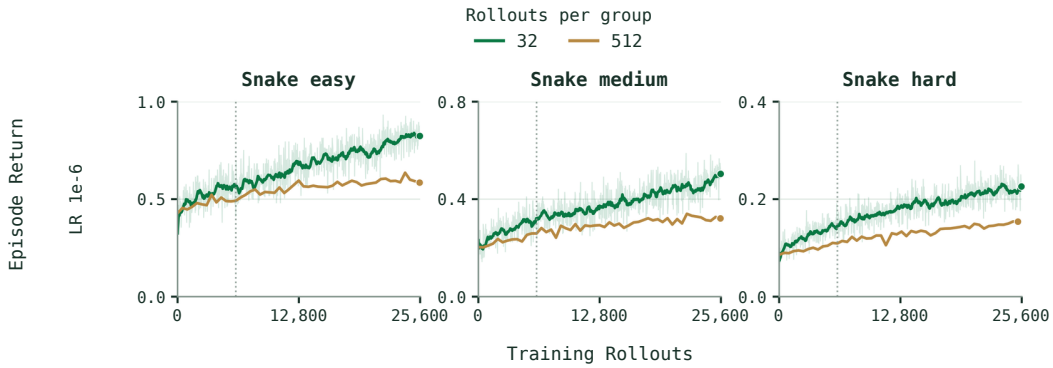

Figure 24: Snake training over 25,600 rollouts at 10^{-6}, comparing groups 32 and 512 across three difficulties. Pale traces show batch returns, solid curves show trailing 512-rollout means, and dots mark final observations. The dotted line marks 6,144 rollouts for reference. Group 32 achieves the higher final return on all three tasks.

L LIMITATIONS AND SCOPE OF THE FINDINGS

The game abstraction. GlyphBench represents game states as spatial text and bounds the horizons of standard tasks to make language-model RL experiments practical. The grid preserves cell layout and named entities while abstracting away texture, perspective, and some visual ambiguity. Standard-task scores measure progress within these adaptations; evaluating mastery of a source game requires its native mechanics and horizons. Common return bounds make tasks easier to combine but do not equalize reward density, difficulty, or opportunities for improvement.

Agent-design comparisons. We compare observations, models, and harnesses as working agent configurations. Each observation format exposes different state information, and Guided Memory combines richer instructions with persistent memory and additional model calls. Focused ablations and fixed-budget comparisons would help separate these contributions to progression and cost. Similarly, the model comparisons evaluate whole systems without isolating parameter count or architecture.

Transfer. Our results show that reasoning gains from gameplay can transfer to held-out Reasoning Gym problems and exceed those of the math- and code-trained baselines in the main accuracy comparison. This evidence concerns the evaluated policies and protocol. The tasks may share skills with gameplay or overlap with model pretraining. Response-length diagnostics show less truncation and shorter average responses after training but do not identify the cause of improved accuracy. The policy ranking also changes with the output budget and metric: the accuracy leader need not lead on observed pass@3. We do not find comparably clear gains on the mathematics and software-engineering evaluations.