



Measuring RL Environment Quality Across Self-Play Training

Essa Jan



Background

The Art of Scaling Reinforcement Learning Compute for LLMs

Devvrit Khatri^{2,*}, Lovish Madaan^{1,3,*},
Rishabh Tiwari⁴, Rachit Bansal⁵, Sai Surya Duvvuri², Manzil Zaheer¹,
Inderjit S. Dhillon², David Brandfonbrener¹, Rishabh Agarwal⁶

¹Meta, ²UT Austin, ³UCL, ⁴UC Berkeley, ⁵Harvard University, ⁶Periodic Labs

*Equal contribution, [†]Work done at Meta

Reinforcement learning (RL) has become central to training large language models (LLMs), yet the field lacks predictive scaling methodologies comparable to those established for pre-training. Despite rapidly rising compute budgets, there is no principled understanding of how to evaluate algorithmic improvements for scaling RL compute. We present the first large-scale systematic study, amounting to more than 400,000 GPU-hours, that defines a principled framework for analyzing and predicting RL scaling in LLMs. We fit sigmoidal compute-performance curves for RL training and ablate a wide range of common design choices to analyze their effects on asymptotic performance and compute efficiency. We observe: (1) Not all recipes yield similar asymptotic performance, (2) Details such as loss aggregation, normalization, curriculum, and off-policy algorithm primarily modulate compute efficiency without materially shifting the asymptote, and (3) Stable, scalable recipes follow predictable scaling trajectories, enabling extrapolation from smaller-scale runs. Combining these insights, we propose a *best-practice* recipe, **ScaleRL**, and demonstrate its effectiveness by successfully scaling and predicting validation performance on a single RL run scaled up to 100,000 GPU-hours. Our work provides both a *scientific framework* for analyzing scaling in RL and a practical recipe that brings RL training closer to the predictability long achieved in pre-training.

Correspondence: {lovish, brandfon}@meta.com, {devvrit.03, rishabhagarwal.467}@gmail.com



Background

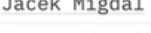
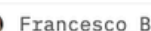
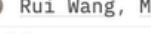
The Art of Scaling Compute for LLM

Devvrit Khatri^{2,*}, Lovish Madaan¹,
Rishabh Tiwari⁴, Rachit Bansal⁵,
Inderjit S. Dhillon², David Brandfon

¹Meta, ²UT Austin, ³UCL, ⁴UC B
*Equal contribution, [†]Work done at

Reinforcement learning (RL) has l
field lacks predictive scaling metho
rapidly rising compute budgets, th
improvements for scaling RL comp
to more than 400,000 GPU-hours,
RL scaling in LLMs. We fit sigmoi
range of common design choices t
efficiency. We observe: (1) Not all
loss aggregation, normalization, c
efficiency without materially shiftin
scaling trajectories, enabling extr
propose a *best-practice* recipe, **Sc**
predicting validation performance
provides both a *scientific framework*
training closer to the predictability

Correspondence: {lovish, brandfon}@

 Mercor	Multi-Domain	SF	-Santa Clara	Thiel Fellowship, Harvard, Geor...	 Brendan Foody, Adarsh Hiremath, Surya Midha
 Handshake	Multi-Domain	SF	-NYC, Bangalore,...	Palantir, Meta, Scale AI	 Garrett Lord, Ben Christensen, Scott Ringwelski RL envs: Pac
 Scale	Multi-Domain Code	SF	-NYC, DC, London	Stanford, Coinbase, YC	 RL envs: Chetan Rane
 Turing	Code	SF	-Palo Alto, Guru...	Stanford, Rover, Yahoo	 Jonathan Siddharth, Vijay Krishnan
 Surge	Multi-Domain	SF	-NYC, Seattle	Google, Meta, Twitter	 RL envs: Nick Heiner
 Snorkel	Multi-Domain Code	SF	-NYC	Stanford, UW, Meta	 Alex Ratner, Braden Hancock, Christopher Ré, Paroma Varma, Her
 micro1	Multi-Domain	SF		Stanford, Berkeley	 Ali Ansari
 AfterQuery	Multi-Domain	SF	-NYC, Seattle	Wharton, Penn, UBC	 Carlos Georgescu, Spencer Mateega
 Latch	Science	SF		Berkeley, Google, Asimov	 Alfredo Andere, Kenny Workman, Kyle Giffin
 Patronus AI	Multi-Domain Code	SF		Meta AI, Google, Amazon	 Anand Kannappan, Rebecca Qian
 Datacurve	Code	SF		Waterloo	 Serena Ge, Charley Lee
 Halluminate	Finance Enterprise	SF		Capital One, Meta, Cornell	 Jerry Wu, Wyatt Marshall
 Mechanize	Code	SF		Epoch AI	 Guive Assadi, Tamay Besiroglu, Ege Erdil, Matthew Barnett
 Pareto	Multi-Domain Tool Use	SF		Stanford, Scale, Turing	 RL envs: Mark Whiting, Youssef Elhady, Joe Crawley
 Proximal	Code	SF	-Bangalore	Prime Intellect, Cursor	 Justus Mattern, Navid Pour, Calvin Chen
 Aptura	Finance Enterprise	London	-SF	Lazard, McKinsey, EF	 Chirag Modi
 Bespoke Labs	Code	Mountain View	-Men...	Google DeepMind, UC Berkeley	 Mahesh Sathiamoorthy, Alex Dimakis
 Collinear	Enterprise Long Horizon	Mountain View	-Sun...	Hugging Face, Salesforce, Stanf...	 Nazneen Rajani, Soumyadeep Bakshi
 Emulated	ML Science	SF		AWS, DeepMind, Prime Intellect	 Joseph Wang, Sid Patllollu
 Quesma	Security	Warsaw		Elastic, Sumo Logic, Dynatrace	 Jacek Migdal
 ReasonCore	Science Code	SF		Turing, Aera Technology, Oracle	 Geoff Wolfe, Anshuman Lall, Satish Vutukuru
 Abundant	Code	SF		Waymo, Google, Mercor	 Jesse Hu, Ke Huang, Meji Abidoye
 Aviro	Code ML	SF		YC, Princeton, Georgia Tech	 Ayush Karupakula, Aarush Kukreja
 BenchFlow	Enterprise Browser	SF		Berkeley, Tesla	 Xiangyi Li
 Cua	Code Computer Use	SF		Microsoft, Twitter	 Francesco Bonacci, Dillon DuPont
 EdotEnv	Long Horizon ML	SF		G-Research, Etched, ETH Zurich	 Rui Wang, Michael Zhang
 General Reasoning	Finance	London	-SF	Meta, Conjecture, Aleph Alpha	 Ross Taylor, Chengxi Taylor, Kip Parker, Thomas Grady
 Markov	Computer Use Tool Use	SF		YC, IIT Madras	 Dev Mandal, Harish Ashok
 Metaphi	Code Enterprise	SF	-NYC	Waymo, Nuro, Netflix	 Abhishek Chandwani, Ishan Gupta
 Normal	Hardware Engineering	SF		Cohere, Notion, Fuse	 Hudzah, Anson Yu
 pre.dev	Code Long Horizon	Delaware		Penn M&T, Wharton	 Arjun Raj Jain, Adam Elkassas
 Refresh	Code Computer Use	SF		Uber, Amazon	 Christopher Settles, Erik Quintanilla
 Tacit Labs	Science Long Horizon	SF		MosaicML, MSR, EF	 Nicole Fitzgerald, Anne Marie Droste
 Ulam	Math	Warsaw	-London	Oxford	 Przemek Chojecki
 Vmax	ML	SF	-NYC	UCL	 Matthew Sargent, Augustine Mavor-Parker
 Uber AI	Audio	SF		Scale AI, Turing, Harvard	 Key people: Natesh Pillai
 Originator	Computer Use Code	London	-Paris	Google, Dataform, Pareto Holdin...	 Ben Cambier, Lewis Hemens

Background

The Art of Scaling Compute for LLM

Devvrit Khatri^{2,*}, Lovish Madaan¹,
Rishabh Tiwari⁴, Rachit Bansal⁵,
Inderjit S. Dhillon², David Brandfon

¹Meta, ²UT Austin, ³UCL, ⁴UC B
*Equal contribution, [†]Work done at

Reinforcement learning (RL) has field lacks predictive scaling methods rapidly rising compute budgets, the improvements for scaling RL compute to more than 400,000 GPU-hours, RL scaling in LLMs. We fit sigmoid range of common design choices to efficiency. We observe: (1) Not all loss aggregation, normalization, compute efficiency without materially shifting scaling trajectories, enabling extrapolate propose a *best-practice* recipe, Scaling predicting validation performance provides both a *scientific framework* training closer to the predictability

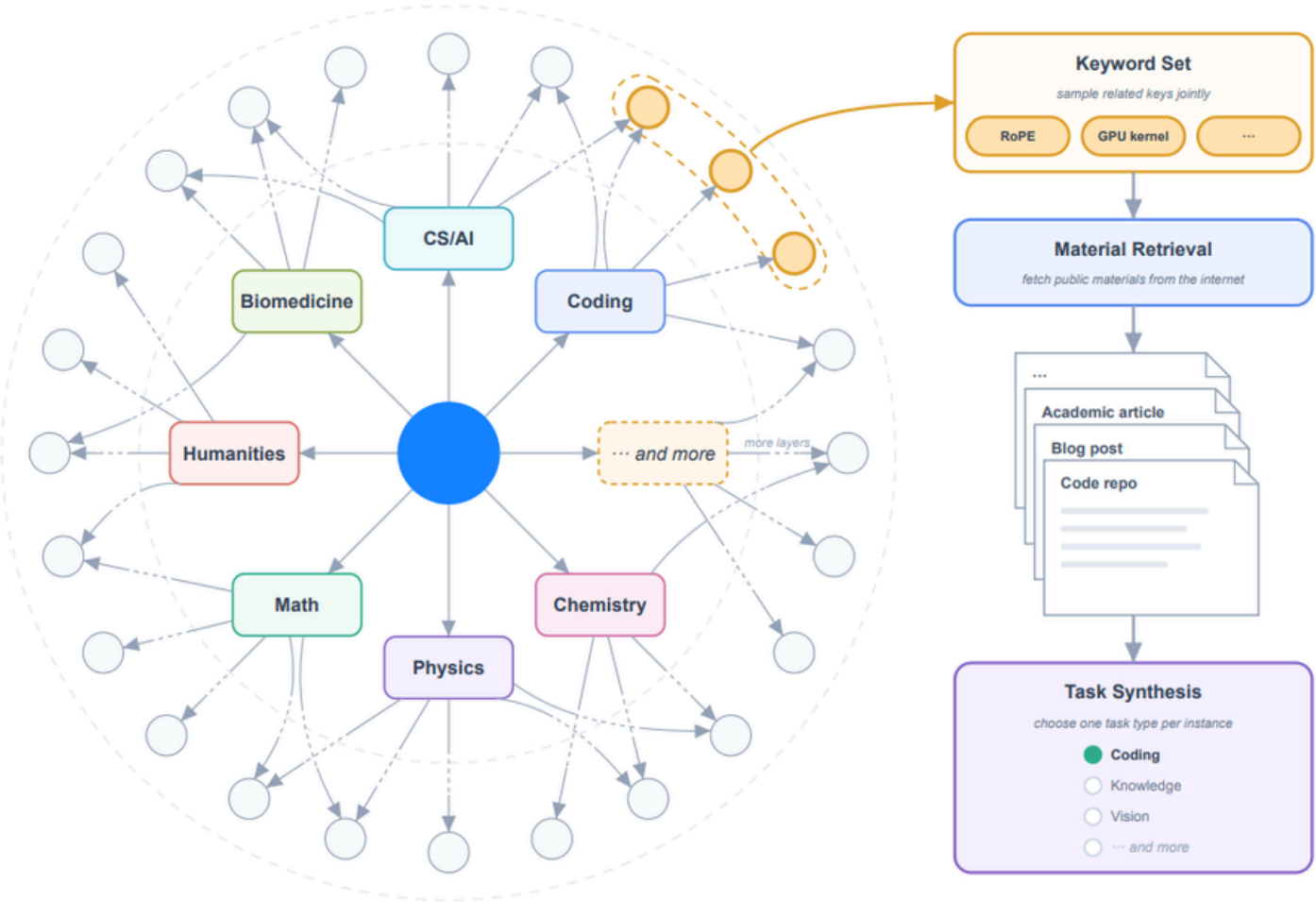
Correspondence: {lovish, brandfon}@

- Mercox
- Handshake
- Scale
- Turing
- Surge
- Snorkel
- micro1
- AfterQuery
- Latch
- Patronus AI
- Datacurve
- Halluminate
- Mechanize
- Pareto
- Proximal
- Aptura
- Bespoke Labs
- Collinear
- Emulated
- Quesma
- ReasonCore
- Abundant
- Aviro
- BenchFlow
- Cua
- EdotEnv
- General Reason
- Markov
- Metaphi
- Normal
- pre.dev
- Refresh
- Tacit Labs
- Ulam
- Vmax
- Uber AI
- Originator

construct different harness configurations for different task groups, exposing Kimi K3 to diverse combinations of these modules rather than the conventions of any single harness. The same abstraction also readily supports RL across various task domains, providing a scalable foundation for training more general-purpose agents.

4.2.2 Knowledge-Graph-Guided Task Synthesis

Motivation and overview The quality and diversity of post-training tasks are largely determined by their source materials. Retrieval guided by fine-grained concepts surfaces specialized and underrepresented knowledge, while sampling across diverse concepts broadens domain coverage. To control both granularity and coverage at scale, we build a self-evolving, hierarchically organized knowledge graph that agents continuously expand through web-scale exploration across knowledge-intensive and coding domains. Figure 9 illustrates the task synthesis pipeline.



Self Play

SPADE ♠: Self-Play in Adaptive Synthetic Executable Environments

Bo Liu^{1,2,*}, Simon Yu^{3,*}, Yiding Jiang⁴, Ao Qu⁵, Andrew Zhao, Zichen Liu⁶, Junsu Kim⁷, Zijian Zhou⁶, Seungone Kim⁴, Tongzheng Ren, Mickel Liu¹, Hanfei Yu⁸, Zhaorun Chen⁹, Weiyan Shi³, Paul Pu Liang⁵, Luke Zettlemoyer¹, Yejin Choi^{2,†}, Natasha Jaques^{1,†}

¹University of Washington, ²Stanford University, ³Northeastern University, ⁴Carnegie Mellon University, ⁵Massachusetts Institute of Technology, ⁶National University of Singapore, ⁷Seoul National University, ⁸Stevens Institute of Technology, ⁹University of Chicago

*Equal contribution, †Joint last author

Continuous self-improvement requires an ever-expanding pool of self-generated, diverse, adaptive goals. For language agents, existing training environment pools (hand-curated, statically synthesized, or frozen-verifier) keep the goal distribution fixed as the learner scales. We introduce SPADE (Self-Play in Adaptive Synthetic Executable Environments), a self-play RL framework in which a single LLM plays two roles: an **Environment Designer** that writes complete, long-horizon training environments as executable code with an OpenAI Gym-style `reset()/step()` interface, and a **Reasoning Agent** that learns to act in them. Each is a stateful, multi-turn environment (state transitions, reward functions, and verification code), so one interface spans reasoning problems and multi-step agentic tool use. The **Reasoning Agent**'s regret is estimated using the gap between its reward with and without privileged hints; in optimizing this regret signal the **Environment Designer** learns to target environments at the edge of the agent's capabilities while keeping them feasible. Through extensive experimentation, we find several components critical to success: grounding the **Environment Designer** on documents sampled from a large pretraining corpus, and giving it an accumulated environment memory. Scaling to 30B-parameter models, SPADE improves over the strongest fixed-environment baseline by +5.3 on average across eight held-out math, science, code, and reasoning benchmarks, and lifts the tool-use setting by +5.7 on BFCL v4 multi-turn and +13.9 on ACEBench-Agent; on the games setting, the margin over the strongest baseline grows with model scale. By making environment design itself a learnable component, SPADE takes a concrete step toward open-ended self-improvement.

Correspondence: benjaminliu.eecs@gmail.com, yejinc@stanford.edu, nj@cs.washington.edu

Code: <https://github.com/spade-rl/spade>

Project: <https://spade-rl.github.io>

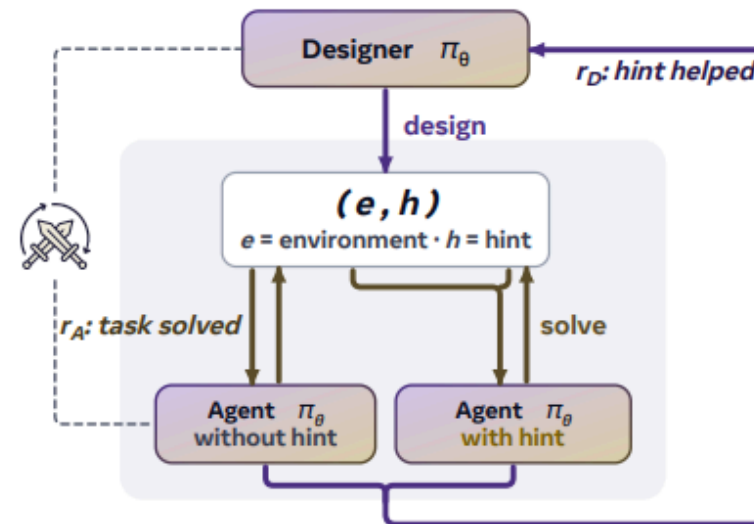


Self Play

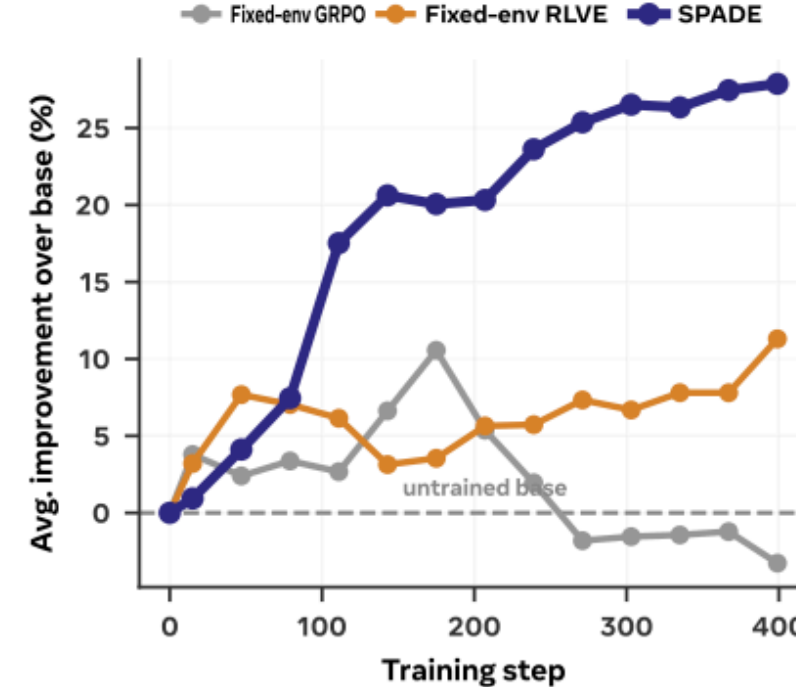
SPADE ♠: Self-Play in Adaptive Synthetic Executable Environments

Bo Liu^{1,2,*}, Simon Yu^{3,*}, Yiding Jiang⁴, Ao Qu⁵, Andrew Zhao, Zichen Liu⁶, Junsu Kim⁷, Zijian Zhou⁶, Seungone Kim⁴, Tongzheng Ren, Mickel Liu¹, Hanfei Yu⁸, Zhaorun Chen⁹, Weiyan Shi³, Paul Pu Liang⁵, Luke Zettlemoyer¹, Yejin Choi^{2,†}, Natasha Jaques^{1,†}

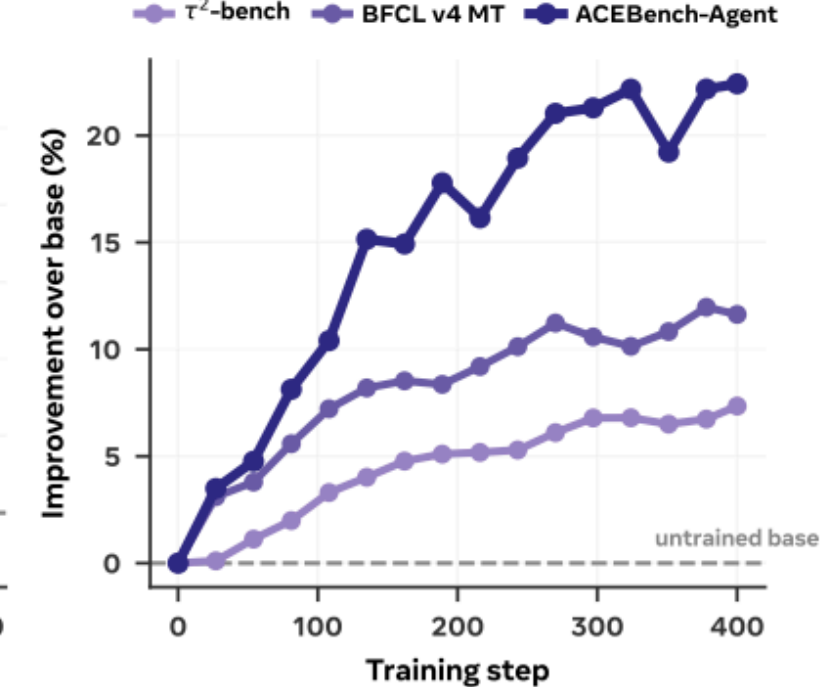
¹University of Washington, ²Stanford University, ³Northeastern University, ⁴Carnegie Mellon



Games (30B-A3B): transfer to 8 OOD evals



Tool use (30B-A3B): transfer to 3 evals



sampled from a large pretraining corpus, and giving it an accumulated environment memory. Scaling to 30B-parameter models, SPADE improves over the strongest fixed-environment baseline by +5.3 on average across eight held-out math, science, code, and reasoning benchmarks, and lifts the tool-use setting by +5.7 on BFCL v4 multi-turn and +13.9 on ACEBench-Agent; on the games setting, the margin over the strongest baseline grows with model scale. By making environment design itself a learnable component, SPADE takes a concrete step toward open-ended self-improvement.

Correspondence: benjaminliu.eecs@gmail.com, yejinc@stanford.edu, nj@cs.washington.edu

Code: <https://github.com/spade-rl/spade>

Project: <https://spade-rl.github.io>





Environment Quality in Self-Play

- It's unclear how the quality of synthetic environments improves under self-play.
- Three Dimensions of Environment Quality
 - a. Bugs/Compilation Issues
 - b. Reward Denial
 - c. Reward Hacking

Hypothesis:

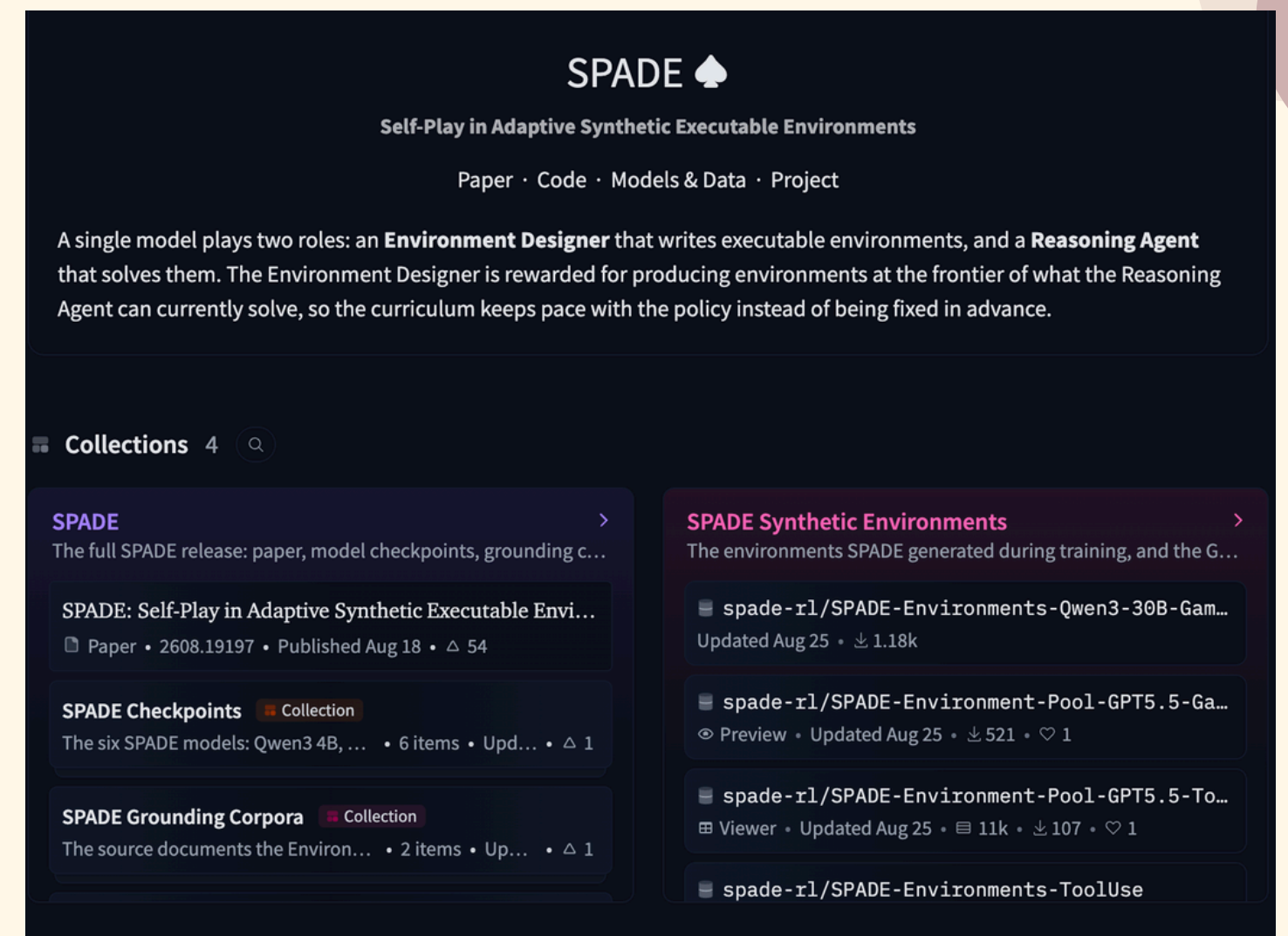
As self-play training progresses, I expect the generated environments to have fewer bugs, reward denials, and reward hacks. I expect bugs and reward denials to decrease faster than reward hacking.

.



Experimental Design

- Spades tool-use blend environments (30B)
 - ~24 envs every 32 steps.
- Measure 1 Solvability:
 - Qwen3.5-4B → Qwen3.5-9B → gpt-5.4-nano → gpt-6-luna
 - Up to 3 plays per model, 25 turns per play.
 - Monitor the rollouts
- Measure 2 Reward Denial
 - Classify unsolved environments into four classes
 - Unclear instructions
 - Wrong grader
 - Not runnable
 - No defect
- Measure 3 Hackability
 - Whitebox adversarial agent
 - Blackbox adversarial agent

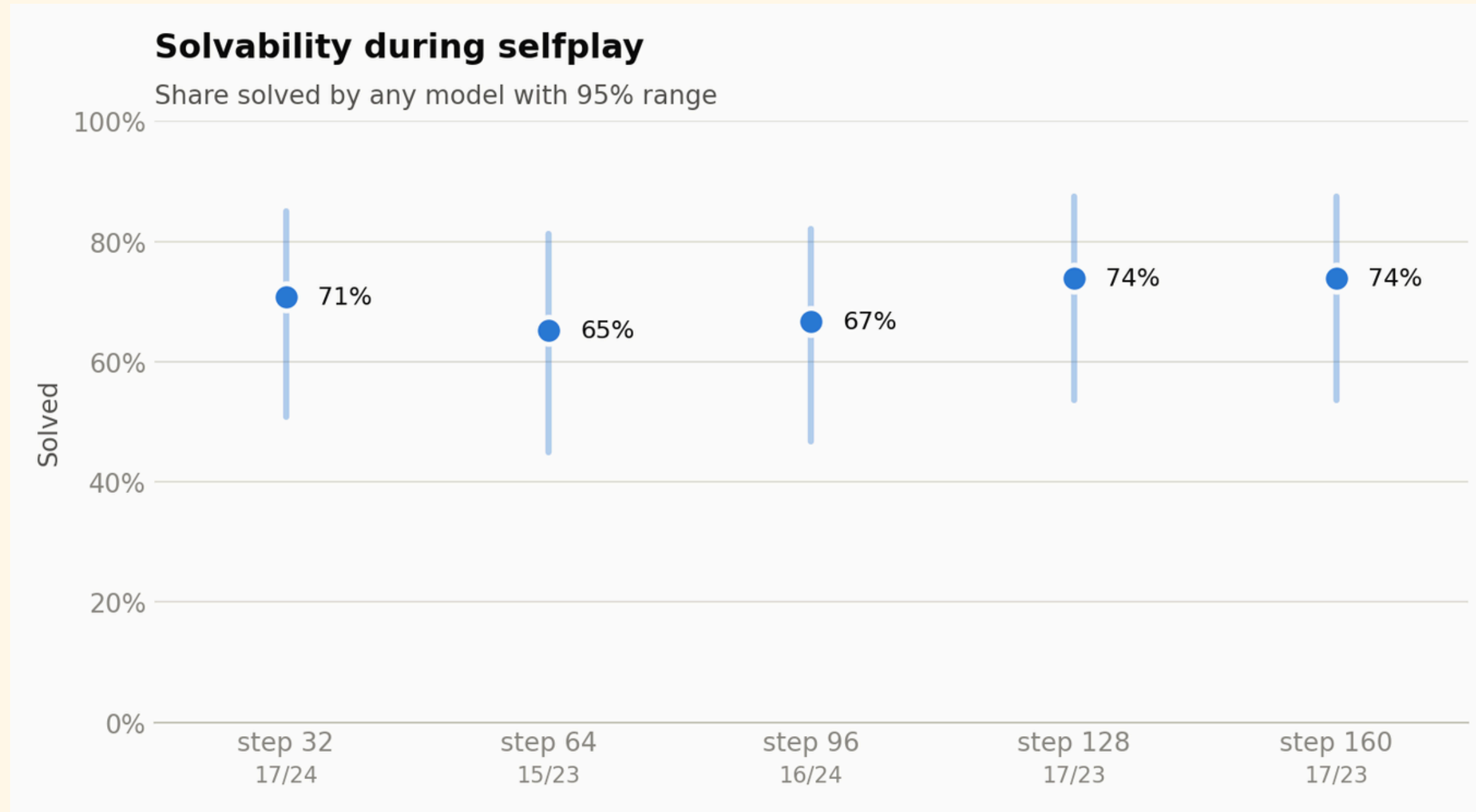


Spades Environment Filtering:

1. Smoke test reset() and 3 random inputs
2. Run the grader
3. LLM audit

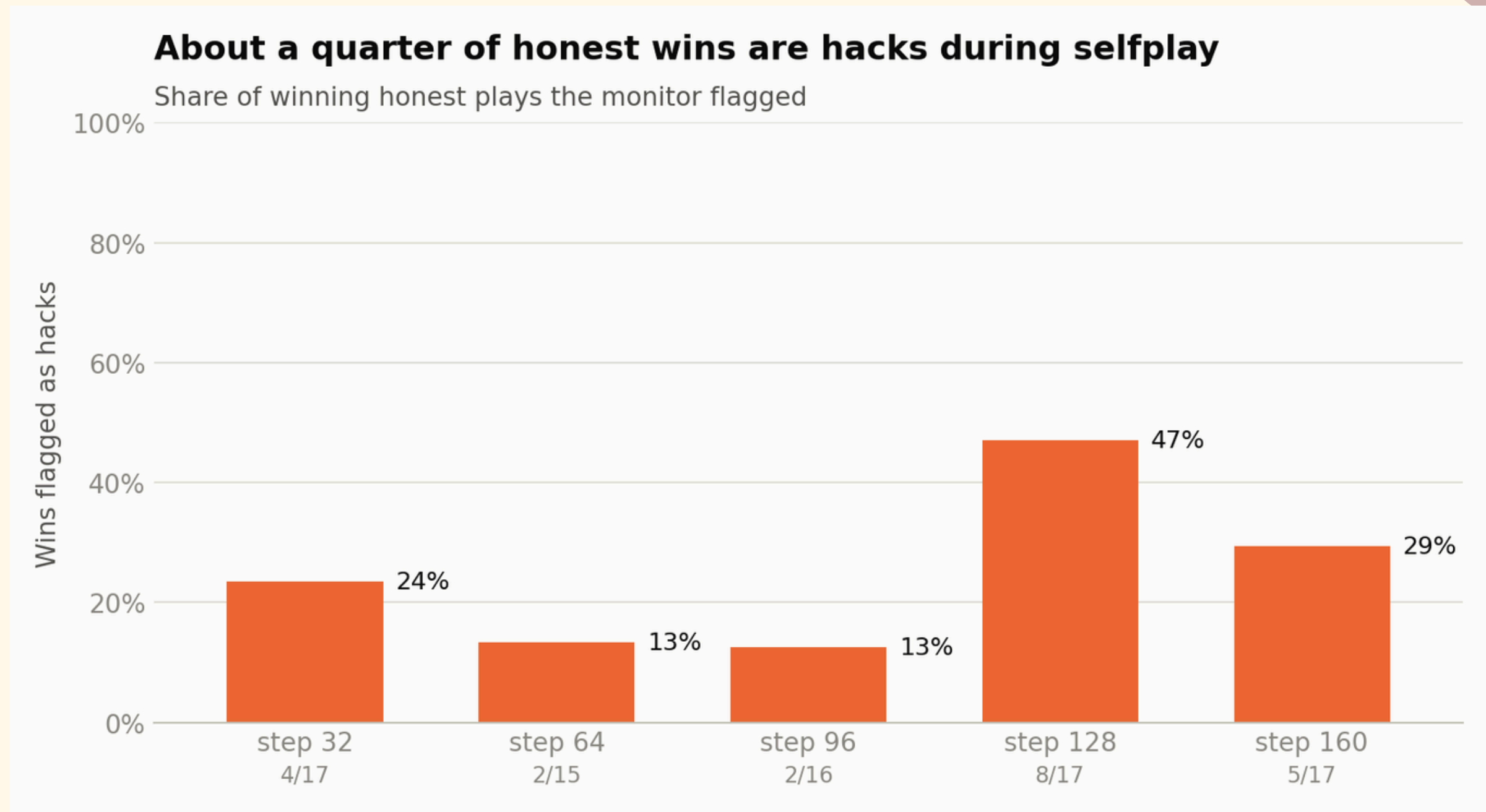
Results: Solvability

- Solvability remain stable across training.



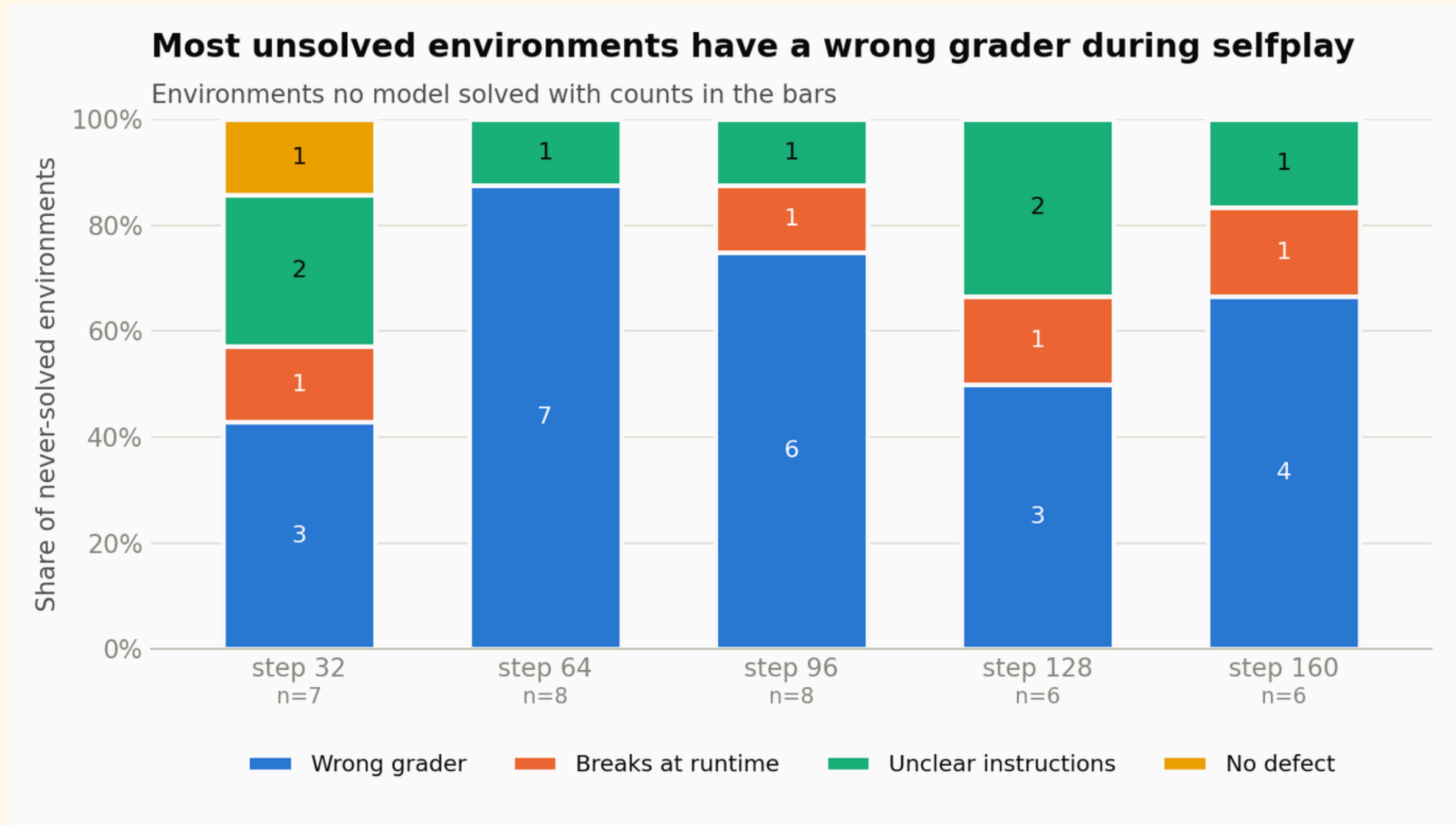
Results

- **Monitor Flag:** At least one step passed even though the task described by that step was not actually completed.
- The monitor judges the trajectory 3 times and takes the majority vote (105/114 unanimous)
- A lot of hacks are grader-side issues e.g
 - Cart-removal check passes on any item.
 - Payment accepted the wrong invoice ID.



Why Environments Remain Unsolved

- **Judge:** GPT-6-luna
 - a. Source code
 - b. Instructions
 - c. One rollout from each model (total 4)
- **Reward Denial:** Wrong grader and unclear instructions show no decrease across training.



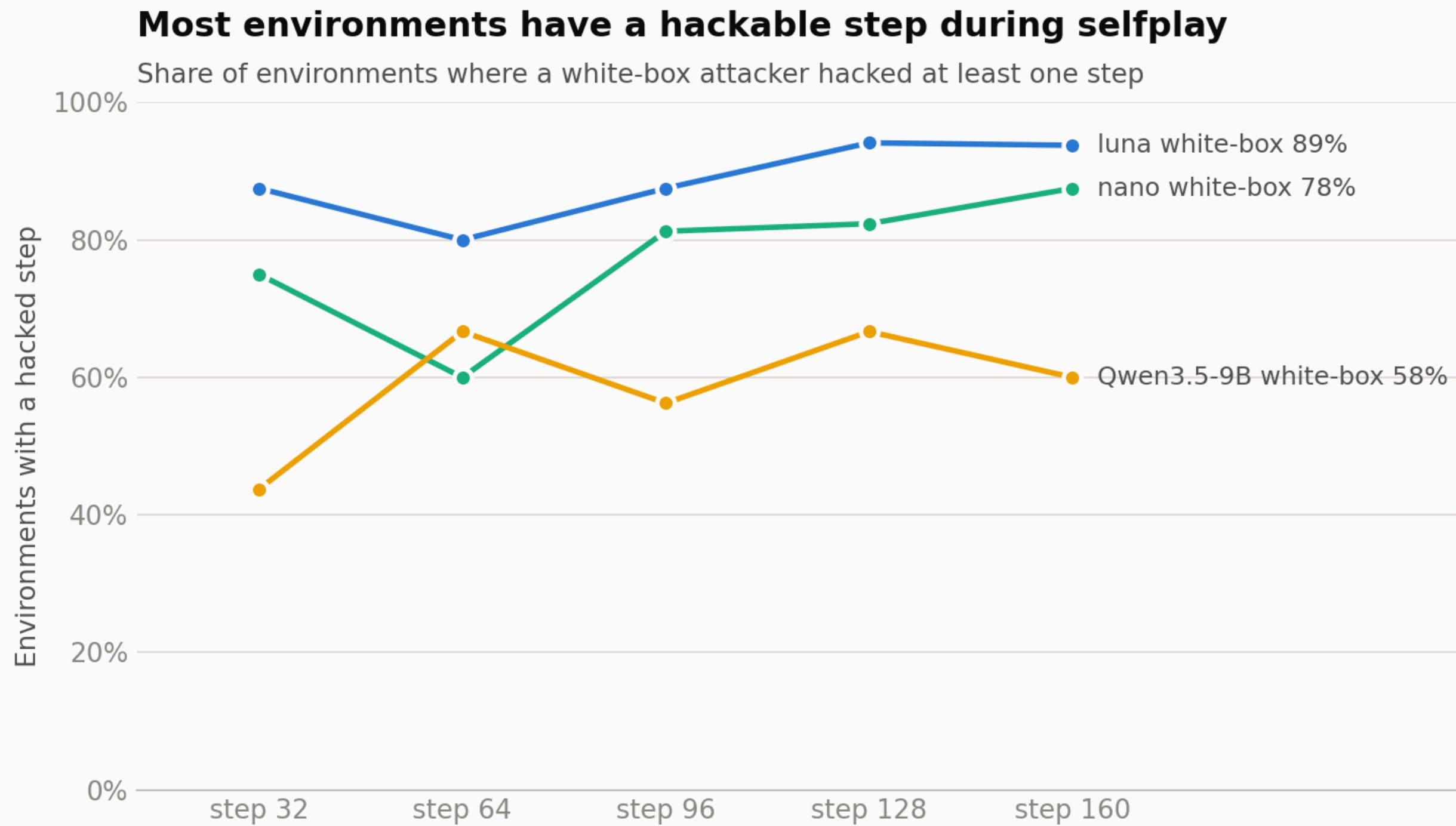


Reward Hacking

Redteaming Pipeline:

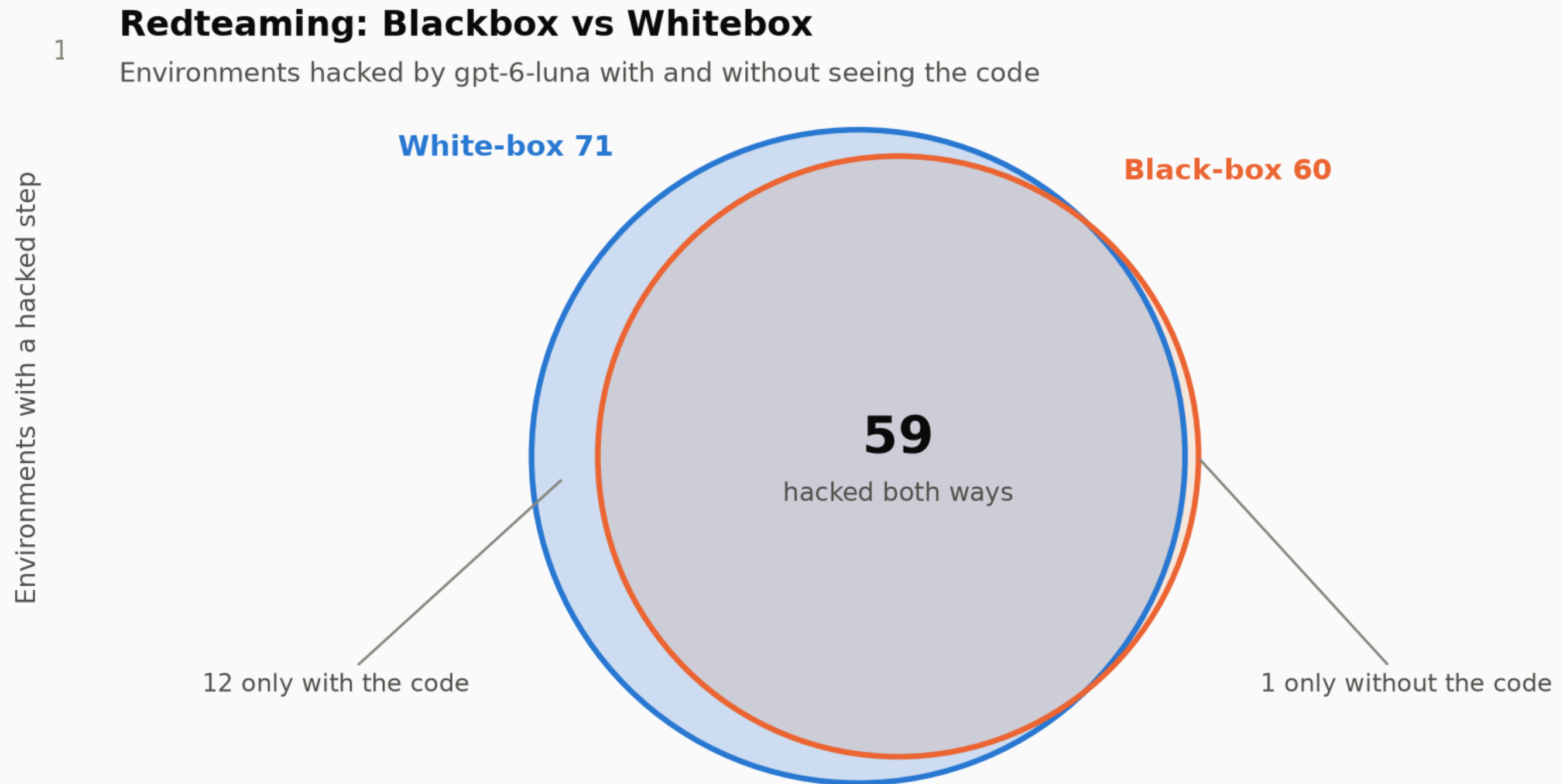
- Attackers: GPT-6-luna, GPT-5.4-nano, Qwen3.5-9B
 - Types: Black-box (tools only) and White-box (source code)
 - Judge: GPT6-luna
 - Steps:
 - a. 100 turns (honest run = 25)
 - b. Tools: env tools + reset_environment, submit_solution, finish
 - c. Submission: ordered list of env tool calls only
 - d. Replay the submitted tools on a fresh env
 - e. Many submissions allowed end rollout when finish tool called or out of turns
 - f. Keep the highest scoring submissions
 - Judging (per passed step)
 - Hacked: The check passed but the instruction was not carried out
 - Order only: work done but in another step / lookup skipped (not a hack)
 - Honest: Instruction was carried out
- 

Results: Reward Hacking



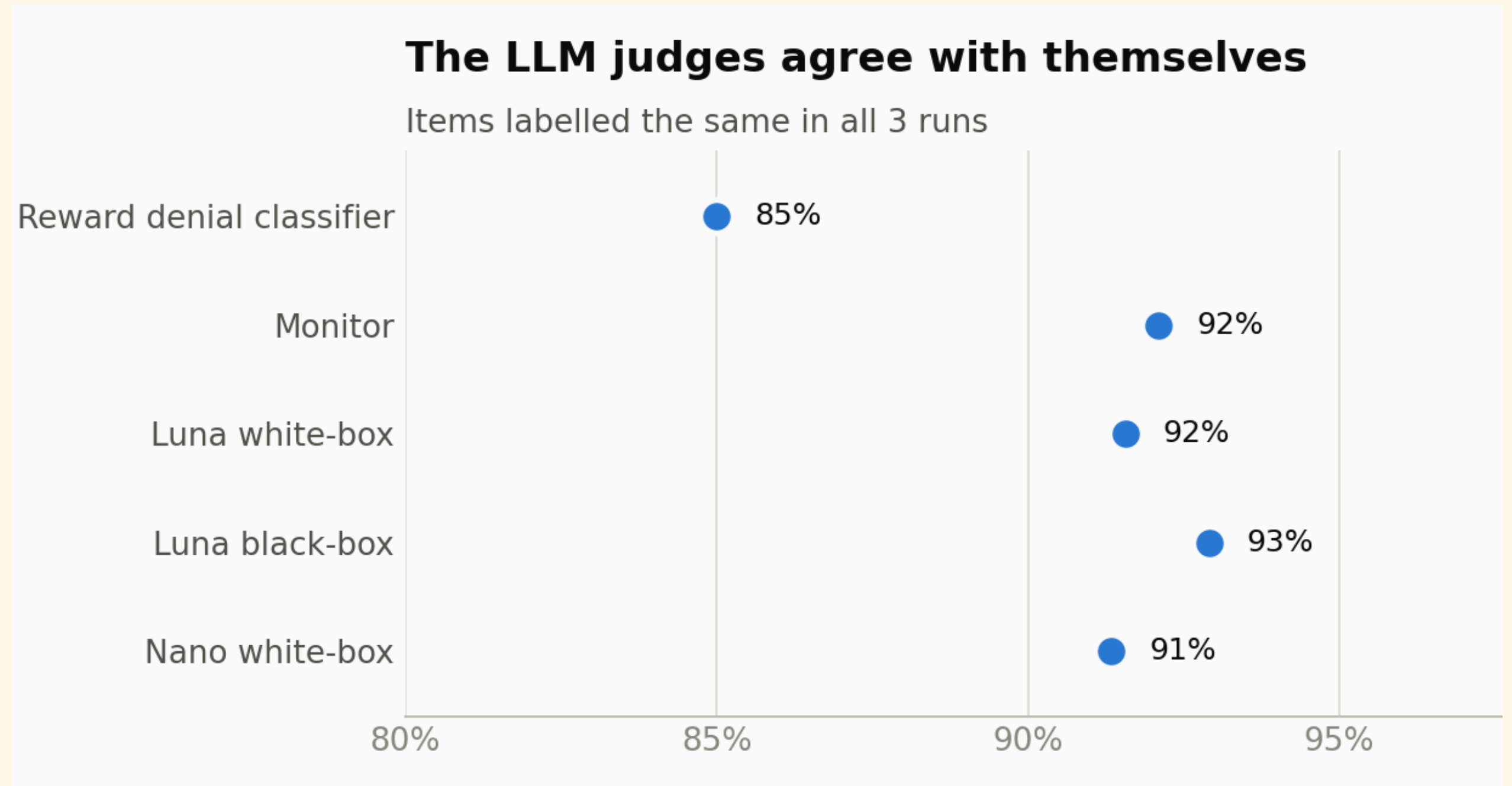
Results: Reward Hacking

- Most vulnerabilities do not require source-code access.



Judge Consistency

- Manual review: 8-10 judgements from each judge

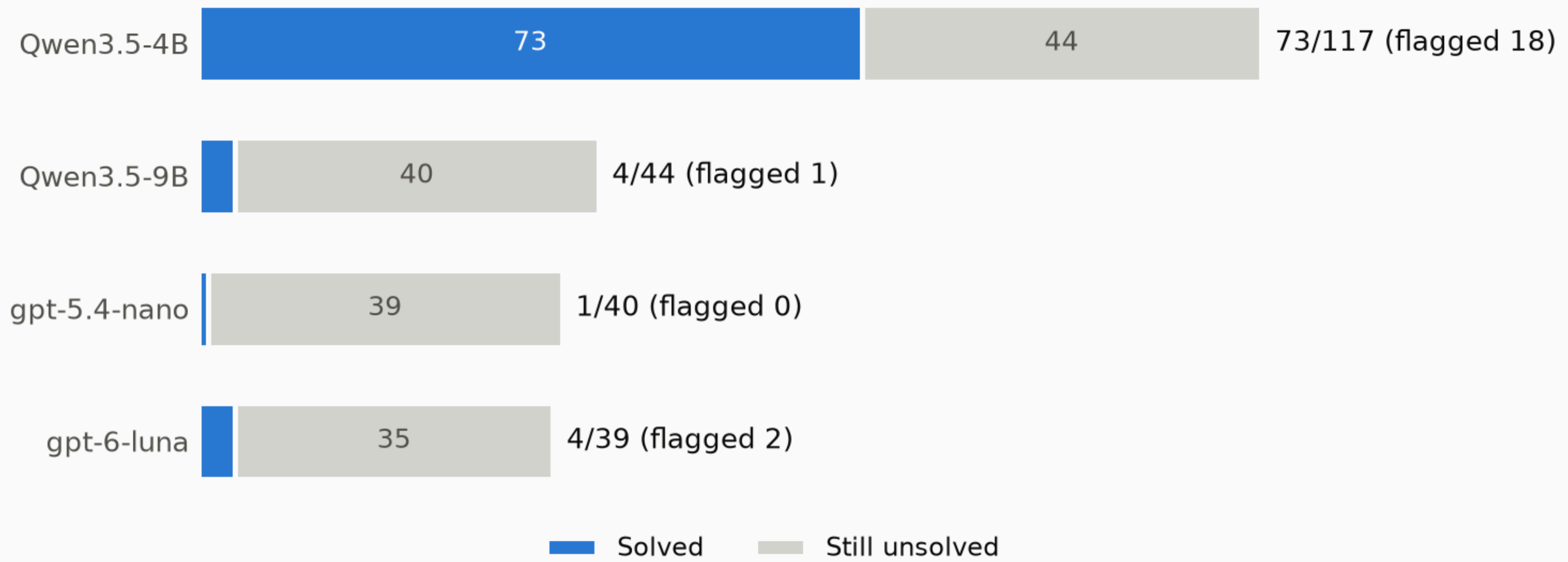


Why the Results Might Not Generalize

- For RL environments, scores should have a high correlation with capabilities.

Environments solved per model

Steps 32 to 160 (117 environments) with each model playing only what is still unsolved





Limitations

- The environments are toy-like, not realistic
 - Scripted as fixed sequences of steps
 - Simple single line instructions
 - State is checked after every tool call
 - Tasks are short and tool calls are straightforward
 - Too few environments per training step to make strong claims
 - The judge is not validated
 - Hacking is easier and more visible than in real environments
- 