SPEEDRUNBENCH: CHALLENGING LLM AGENTS ON SPEEDRUNNING GAMES

Yoshinari Fujinuma, Keisuke Kamahori, Abdelrahman Madkour, Varun Prashant Gangal, Monty Bichouna, Martyna Markiewicz, Shivani Jain, Duncan Curtis, Rebecca Qian, Anand Kannappan
Patronus AI University of Washington
{yoshinari.fujinuma,varun.gangal,abdelrahman.madkour,anand}@patronus.ai
kamahori@cs.washington.edu

ABSTRACT

Frontier LLM agents are increasingly evaluated on various tasks, yet still fall short on tasks that are complex, long-horizon, and require discovering creative approaches. Most recent frontier models have been shown to successfully complete video games. We go one step further and ask an even more challenging task: are these frontier LLM-based agents able to optimize gameplay under a speedrunning setting, i.e., minimizing the frames needed to reach the goal? In this paper, we introduce **SPEEDRUNBENCH**, a benchmark that measures the capability of LLM-based agents to optimize the number of frames needed to reach a given goal across 10 different games on 10+ different models. Unlike prior work on game benchmarks, which simply tests whether a LLM-based agent can find a successful trace to complete a game, SPEEDRUNBENCH measures various capabilities of LLM agents by repeatedly playing the games to minimize the time or frames needed to complete the game, which requires exploration of new strategies, speed bottleneck discovery, multi-modal understanding, and long-horizon capability, which coincides and potentially transfer to long-horizon agentic coding tasks. Our experiments show that SPEEDRUNBENCH is far from saturated. Agents approach the human world record on the simplest games, but on longer and more complex games they remain several times slower than it, and most improve a given trace only marginally.

1 INTRODUCTION

Games have served as a traditional go-to testbed (similar to how the fruitfly has been used in biology) for models since the earliest days of AI and ML, with memorable milestones such as Atari (Mnih et al., 2015), Doom (Wydmuch et al., 2019), Go (Silver et al., 2016), and StarCraft II (Vinyals et al., 2019). This owes to their exact scoring, unlimited repeatability, and natural ladders of difficulty. [1] These characteristics are also well-suited for evaluating modern LLM agents, addressing the practical obstacles that arise when frontier models are placed in game environments directly: unreliable visual perception, prompt sensitivity, and potential training data contamination (Hu et al., 2026; Zhang et al., 2026). The evaluation objective, however, has remained largely unchanged. In nearly all cases, the question is whether the agents can complete the game, and whether successfully.

Evaluating solely on whether games are completed has two limitations as a measurement instrument. First, a binary outcome saturates: once frontier models complete a game, the metric no longer separates them, and prior to saturation it assigns identical scores to runs of substantially different quality. We instead modify the objective, motivated by the fact that many retro games have been completed many times by human players, to measure how fast the game can be completed. We measure not whether an agent completes them, but *how quickly* they can complete the game, i.e., *speedrunning*.

[1] So much so that there has been an effort to cast even non-game tasks in a similar game like mode, such as in the bAbi style (Dodge et al., 2015) of grounded textual reasoning benchmarks.

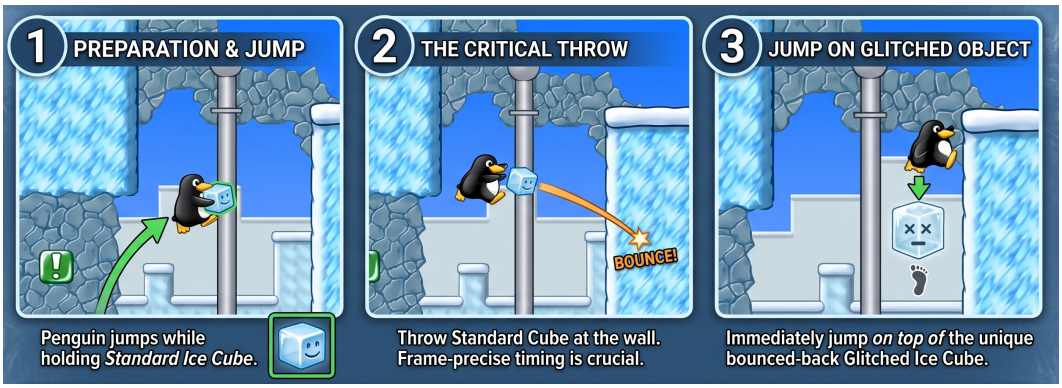


Figure 1: One example of a complex technique necessary for speedrunning SuperTux, an open-source 2D scroll action game. This technique is unnecessary to complete the game itself, but for speedrunning, this is critical to cut down the frame to reach the goal.

Speedrunning a game is an extremely difficult and challenging problem, with structural characteristics reminiscent of, if not identical to, finding new bugs in well established open source repos or progressively lowering the bounds on mathematical upper bounds such as matrix multiplication complexity and the prime number gap. The community of human players of a game, have spent decades steadily improving their time to reach the goal or a milestone by discovering new techniques while using all of the previously found ones (e.g., Figure 1) with various categories (e.g., glitchless, unwarp) (Snyder, 2017; Groß et al., 2022). This makes it ideal for evaluating whether LLM agents can explore, discover, and maintain long-horizon coherence during speedrunning. We evaluate agents under three settings, where we two of those are newly proposed in this paper. In the ONLINE setting, the one prior game benchmarks adopt, the agent observes the current frame and predicts the next action or a short chunk of actions while the game runs. In the OFFLINE setting, the agent instead edits a complete frame-indexed action trace, replays it, and reads back the frame at which the goal is reached, repeating this over a fixed number of iterations , and it separates the ability to optimize a run from the perception and reaction speed the online setting demands. We further distinguish OFFLINE-SEED, in which the agent starts from a reference trajectory, from OFFLINE-SCRATCH, in which it must construct one itself. Each setting emphasize on a different ability: ONLINE demands multimodal understanding, OFFLINE-SCRATCH demands discovery, since a working route has to be found unaided, and OFFLINE-SEED demands exploration since the agent think outside the box. All settings demand long-horizon coherence, since the agent has to keep improving a route it already holds, over many turns, without breaking it.

In this paper, we evaluate a range of LLM agents across these settings and report three findings. First, speedrunning separates models that completion alone does not: agents that can readily reach the goal still differ widely in how fast they do it. Second, the effect of seed traces dominates the offline optimization loop. Agents able to reach the goal unaided finish well ahead of the seeded runs, while weaker agents depend on the seed to reach the goal at all. Third, the evaluation budget available within a turn affects the outcome more than model choice, and with a sufficient budget the strongest agent approaches the human world record. Our contributions are SPEEDRUNBENCH itself, a suite of games spanning several genres in which commercial titles are paired with open-source counterparts of far lower pretraining exposure, the online and offline settings that let the same game be scored under different demands, and the analysis above. Even the strongest agents close little of the gap: Claude Opus 5 reaches Pokémon Blue's first badge in 160,021 frames against a human record of $\sim$40,435 frames.

2 RELATED WORK

Benchmarks for LLM Agents Seeing & Playing Games Games have been serving as evaluation benchmarks for machine learning (ML) models well before LLMs became popular, from text adventures (Côté et al., 2018) to 2D navigation (Guruprasad et al., 2026) and abstract rea-

soning tasks (Foundation, 2026). Games were also the setting for some of the field's most visible results before LLMs, from Atari from pixels (Mnih et al., 2015) to superhuman play in Go (Silver et al., 2016) and StarCraft II (Vinyals et al., 2019). Recent benchmarks extend the setting to let frontier LLMs play games directly. LMGame-Bench (Hu et al., 2026) supplies perception and memory scaffolds, standardizes prompting, and screens for training data contamination, while VideoGameBench (Zhang et al., 2026) asks whether vision-language models (VLMs) can complete popular titles at all. Taesiri et al. (2024) pairs images with questions to evaluate whether a VLM can identify what is wrong in a rendered screen. Individual titles have also served as milestones in their own right, with Gemini 2.5 Pro clearing Pokémon[2] in 406.5 hours over two attempts (Gemini Team, Google, 2025), and Hu et al. (2024) testing LLM capability on Pokémon battles specifically. Minecraft is the main setting where the objective is not completion: MineDojo (Fan et al., 2022) and Voyager (Wang et al., 2024) score an agent on how much it acquires in an open world, by items obtained and skills learned. These benchmarks therefore measure breadth, how many different things an agent can do, whereas SPEEDRUNBENCH measures depth, how far an agent can push a single goal it has already reached. SPEEDRUNBENCH scores how quickly it was completed, which continues to separate models after completion is reached by challenging them to learn and discover speedrun-related skills.

Game Speedrunning as a Task Optimizing the speed outside of games are also explored such as speedrunning LLM training (Zhao et al., 2026) and inference (Kamahori et al., 2026). The closest work to ours is the Pokeagent Challenge from NeurIPS 2025 (Karten et al., 2026a), in which speedrunning is one of the tracks, followed by Karten et al. (2026b) on online adaptation. Both evaluate an agent that observes a game frame and predicts the next action or a short chunk of actions, and both focus on a single title: Pokémon. SPEEDRUNBENCH differs in scope and in setting: our suite spans several genres, and beyond the online setting it evaluates an offline one in which the agent edits a complete frame-indexed trajectory, replays it, and scores on the resulting frame count, which turns the task from finding a working trajectory into improving one that already works.

Long-Horizon Benchmarks Long-horizon evaluation outside games takes a similar shape. DeepSWE (Huang et al., 2026) measures agents on original engineering tasks in real repositories, and Long-Horizon-Terminal-Bench (Li et al., 2026) on terminal tasks with distant end states, both graded on reaching a goal state.

3 SPEEDRUNNING WITH AN LLM AGENT

We first review what agent capabilities are required to perform well on speedrunning (§3.1), followed by introducing the different setups for LLM agents speedrunning (§3.2). Using SPEEDRUNBENCH, we explore the following research questions:

- RQ1. Is SPEEDRUNBENCH challenging for modern LLM agents? Under what setup a modern LLM-based agent can and cannot speedrun?
- RQ2. Can LLM agents optimize their gameplay with an autoresearch-like loop? Even for a near-optimal Tool-Assisted Speedrun (TAS) run?
- RQ3: What are the critical features for an LLM agent to speedrun?

3.1 AGENT CAPABILITIES MEASURED IN SPEEDRUNBENCH

Exploration: The agent needs to "think outside the box" to explore new paths and approaches to come up with an action sequence to minimize the frames to the goal. This capability is tested especially in an OFFLINE-SCRATCH setup where given seed gameplay trajectories are not necessarily fast gameplays.

Discovery: For example, a glitch in Super Mario Brothers was discovered 40 years after the original game is released by a speedrunner (Colantonio, 2026).

Multimodal understanding (ONLINE only): Similar to prior work (Hu et al., 2026; Zhang et al., 2026), understanding the game play screen, dynamics of each game, etc

[2]Pokémon is a trademark of Nintendo Co., Ltd., Creatures Inc., and Game Freak Inc.

Table 1: Games explored in SPEEDRUNBENCH, spanning several consoles and genres.

GAME	CONSOLE	GENRE
Super Mario Brothers	NES	Platformer
Super Mario World	Gameboy	Platformer
Mario Kart 64	Nintendo 64	Racing
Pokémon Blue/Red	Gameboy	RPG
Civilization I	MS-DOS/SNES	Strategy
Wolfenstein 3D	MS-DOS/Browser	Shooting
Astray	Browser	Puzzle
SuperTux	C++/Browser	Platformer
TuxeMon	Python/Browser	RPG
TuxCart	C++/Browser	Racing

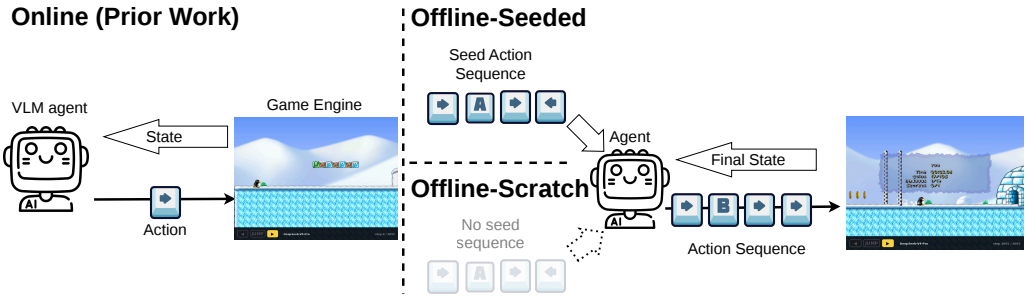


Figure 2: Overview of different settings in explored in SPEEDRUNBENCH: ONLINE, OFFLINE-SEED, and OFFLINE-SCRATCH. ONLINE setup predicts next action (or small action chunks) from a current state, whereas OFFLINE setup predicts the full action trace from the start to the goal.

Long-horizon Coherence in playing and speedrunning: Traces run large number of actions, and a single wrong edit anywhere in one costs the whole run. The agent has to keep what already works while changing part of it, and to do so over hundreds of iterations.

For OFFLINE setups, the agent needs to handle long-horizon task of optimizing the game speed by playing it multiple times.

3.2 ONLINE VS. OFFLINE SETTING FOR SPEEDRUNNING

A game is a deterministic environment with state space $\mathcal{S}$ and a finite action space $\mathcal{A}$, the set of controller inputs available on that console. Play begins from a fixed initial state $s_0 \in \mathcal{S}$, a power-on reset, and each action advances the game by one frame, so a sequence of actions in $\mathcal{A}$ fully determines a run. We explore three settings per game: *Online*, *Offline-Scratch*, *Offline-Seed* for measuring different capabilities of each model (Figure 2). We further explain these three different setups in the following paragraphs.

Online (Action prediction from screenshots) Following prior work (Hu et al., 2026; Zhang et al., 2026), given a current state $\mathcal{S}$ of a game, the agent decides the next action $a \in \mathcal{A}$. Prior work has shown that this is possible with a limited subset of frontier models (Opus 5, GPT-5.6-Sol, Grok 3.6), but this setup only measures one aspect agent's capability such as multimodal understanding. Since this setup can produce a baseline seed reference trajectory, we optimize it in the offline or trajectory editing setting.

Offline (Trace Editing) This setting mimics human speedrunning by playing the game over and over again throughout the decades. A run is represented as a frame-indexed action trace: a sequence of n actions $\{a_i\}_{i=1}^{n} \in \mathcal{A}$ covering an entire attempt, from reset to goal. On each turn the agent

submits an edited trace, evaluated, and the agent reads back the frame at which the goal was reached, or a failure if the edited trace no longer achieves the goal. The agent repeats this for a fixed number of turns and keeps the fastest trace it has found. This can be regarded as action chunking i.e., predicting next k actions (Zhao et al., 2023) with an extremely long chunk, but without image as an input. We explore the following two variant settings:

- **OFFLINE-SCRATCH**: The agent has to come up with a valid action sequence from scratch i.e., without a reference trace. This is harder than seeded setting due to a working full trajectory not given in advance.
- **OFFLINE-SEED**: The agent is given a seed reference trajectory to optimize from. Seed reference trajectory are either from humans, scripted by breadth-first-search (BFS), from online setup, which are all far from the best run for each game except for §4.5 where we analyze the effect of seed trajectory. Seeds published by the speedrunning community[3] are recorded against a particular software environment, so they do not always replay in our setup as given, and have to be ported and verified before use.

Note that *vision inputs are not required* under OFFLINE settings, and therefore, more open-weight models can be evaluated under these settings. This is because, especially for simple games, no vision inputs are needed, smaller open-weight models can optimize.

3.3 AUTO-RESEARCH LOOP-BASED OPTIMIZATION

For offline setup, we use a self-improving loop inspired by auto-research (Karpathy, 2026) to 1) mimic humans speedrunning by playing the game over and over again.

4 EXPERIMENTS

4.1 GAMES

Table 1 lists the ten titles in SPEEDRUNBENCH, spanning six genres and multiple consoles. The suite is assembled along two axes. The first is genre, since the capability a speedrun exercises depends on what the game asks for: a platformer rewards frame-precise movement, an RPG rewards routing across a long horizon, and a puzzle game rewards search over a discrete state space. The second is pretraining exposure due to popularity. Alongside widely known commercial titles we include open-source games of the same genre, so that each of the three most popular and documented genres has a counterpart whose routes and records are far less likely to appear in a pretraining corpus: SuperTux beside Super Mario Brothers, Tuxemon beside Pokémon, and TuxCart beside Mario Kart. Section 5.2 uses one such pair directly, and the design is what makes that comparison possible.[4]

Super Mario Brothers and Super Mario Land 2D side-scrolling platformers games, where both have decades of documented routing and frame-level world records, which supplies an external speedrun reference. The goal state is reaching the level end, and the metric is the frame at which it is reached. These are the titles where an agent is most likely to have exposure during pretraining data, which is what makes the open-source counterparts necessary rather than decorative.

SuperTux SuperTux (The SuperTux Team) is an open-source 2D side-scrolling platformer. It makes the same demands as the Mario titles, precise movement and route discovery toward a level exit. We use the `welcome_antarctica` level, referred to as SuperTux 1-1 throughout.

Pokémon Blue/Red and Tuxemon (Tuxemon Contributors, 2026) Turn-based RPGs, and the longest-horizon tasks in the this benchmark. A run reaches the first gym badge only after a sequence of thirteen milestones, and the frame counts involved are one to two orders of magnitude larger than

[3] `https://tasvideos.org/` where all submissions are shared under CC-BY-2.0 license `https://tasvideos.org/WelcomeToTASVideos#LinksAndRedistribution`

[4] The commercial titles listed here are named only to identify the evaluation setting. Users must obtain lawful copies themselves and nothing in this work grants rights beyond those already permitted by the relevant licenses, terms of service, and law.

in the platformers, which stresses long-context tracking rather than execution precision. Tuxemon is the open-source counterpart, matched in genre and structure but estimated to be less representative in the pretraining data in general.

Mario Kart 64 and TuxCart are racing games, where the objective is lap time rather than a goal state, and where optimization is continuous rather than a search for discrete route improvements.

Astray A browser-based maze puzzle. Unlike the action titles, it is bound by search over a discrete state space rather than by control, and it is small enough that near-optimal solutions are computable, giving an exact rather than a community-established reference point. We use the implementation from Ouyang et al. (2026).

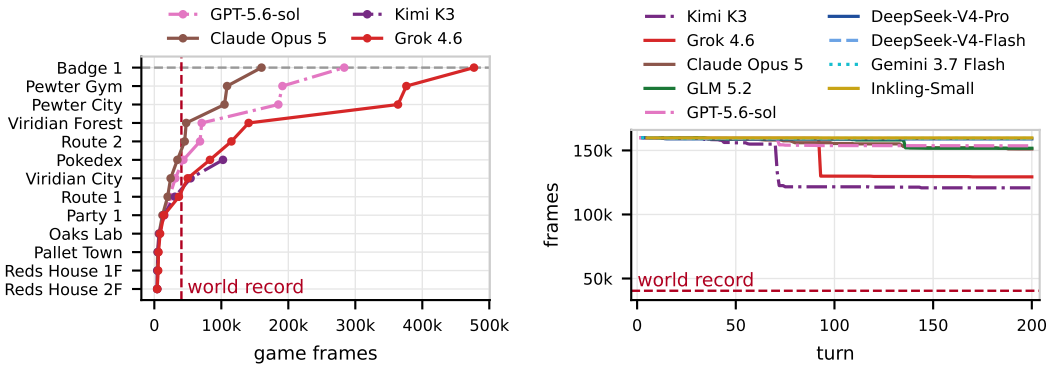
Civilization I and Wolfenstein 3D Strategy and first-person shooting games. These broaden genre coverage beyond the platformer and RPG settings that dominate prior game benchmarks.

4.2 Setup

Models: We use frontier models both from proprietary models such as Claude Opus 5 (Anthropic, 2026), GPT-5.6-sol (OpenAI), Grok 4.6 (SpaceXAI), Gemini-3.7-Flash (Google DeepMind, 2026), and open-weight models such as Kimi-K3 (Team et al., 2026), GLM 5.3, GLM 5.2 (GLM-5-Team et al., 2026), Deepseek-v4-pro-0813, Deepseek-v4-flash-0731 (DeepSeek-AI, 2026), Incling-Small (Thinking Machines Lab). All models ran with Opencode (OpenCode Developers, 2026) for experimenting on the same harness.

Seed Reference Trajectory for OFFLINE-SEED: For traces to seed the agent in OFFLINE-SEED, GPT-5.6-sol/Opus-5 online run, scripted, human. All the seed trajectories are far from the optimal speedrun play for leaving the space for agents optimize their play.

4.3 Online Play Results



(a) ONLINE: milestones reached over game frames.

(b) OFFLINE-SEED: frames to the first badge.

Figure 3: Pokemon Blue results. **(a)** ONLINE run on models reaching Pokemon Blue milestones. With $\sim$ 60 fps, Opus 5 reached Badge 1 in roughly 44m 27s, whereas the human world record is $\sim$40,435 frames, or about 11m 14s (SlNNED, 2025): the record is $\sim 4\times$ faster than the best model result. **(b)** OFFLINE-SEED speedrun until the first badge, starting from the reference trajectory of Opus 5's online run. That seed blacked out twice (Viridian Forest and Pewter Gym), and Kimi-K3's large improvement around turn 90 came from avoiding the black out at Pewter Gym.

Figure 3a shows ONLINE Pokémon play. Only the three vision-capable proprietary models reach the first badge at all; Kimi K3 stalls at the Pokédex. Reaching it is not the difficulty. Every one of them is far off the pace: Claude Opus 5 takes 160,021 frames, GPT-5.6-sol 283,474, and Grok 4.6 477,140, against a human record of 40,435, so the best agent is roughly $4\times$ slower than the record and the worst nearly $12\times$. Two blackouts account for much of the loss even in the best trajectory. Under

a completion metric these three runs are indistinguishable, which is the gap SPEEDRUNBENCH is built to measure.

4.4 OFFLINE SPEEDRUNNING RESULTS

OFFLINE-SCRATCH outperforms OFFLINE-SEED for agents able to reach the goal SuperTux 1-1 permits a paired comparison, as all eleven agents are evaluated under both settings and the OFFLINE-SEED arms are initialized from a common 9498-frame trajectory produced by GPT-5.6-sol's ONLINE run. Among the eight agents that produced a clear without a seed, each attained a lower frame count than it did under OFFLINE-SEED, with improvements up to $\sim 5.1\times$ (Figures 5a and 5b). DeepSeek-V4-Pro attains 1844 frames from scratch against 9428 when seeded, GLM 5.3 attains 1845 against 8412, and the smallest margin, obtained by GLM 5.2, remains 2405 against 8887. The two distributions do not overlap: the best OFFLINE-SEED result across all eleven agents, 8412 frames, exceeds the worst OFFLINE-SCRATCH result, 2405 frames.

This advantage of OFFLINE-SCRATCH over OFFLINE-SEED is conditioned on whether the model is capable of finding one. Three models, namely Gemini 3.7 Flash, Inkling-Small, and Inkling, failed to reach the goal from scratch within the turn budget and therefore yield no measurement under OFFLINE-SCRATCH. For these agents the seed is what renders the task tractable, and its removal eliminates the run entirely. OFFLINE-SCRATCH is preferable conditional on a capability for which the seed otherwise substitutes. The same mechanism accounts for the dispersion of the two sets of curves, with seeded runs remaining closely clustered below 9498 frames while from-scratch runs spread over 1844 to 2405 frames. We interpret this as evidence that an agent supplied with a working trajectory is optimized locally around it and inherits its route, whereas an agent that constructs its own route is not constrained by a slow one.

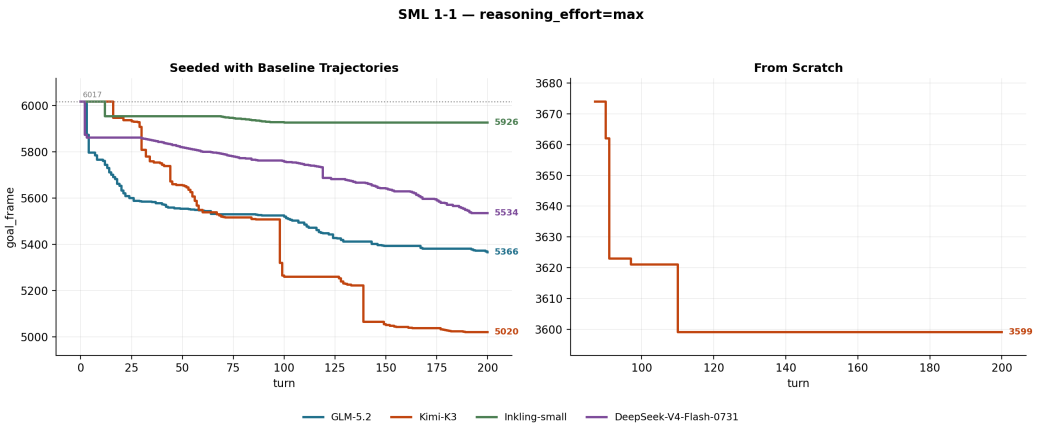
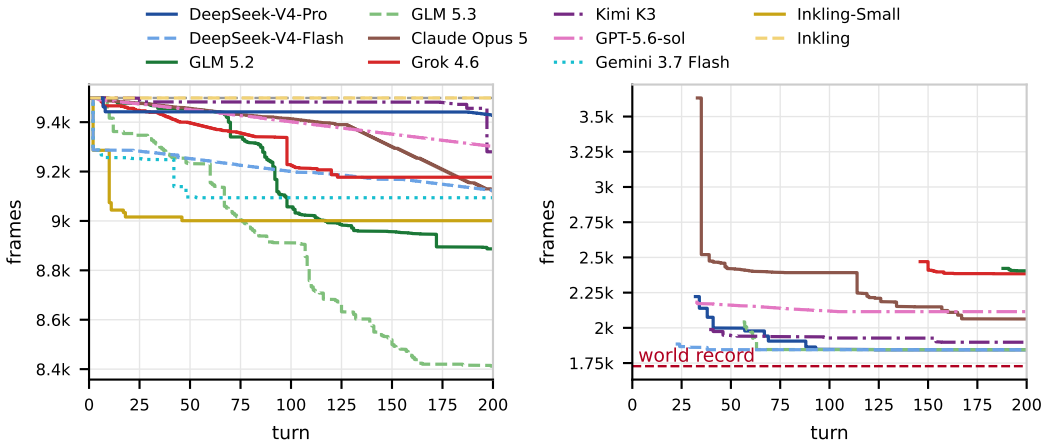


Figure 4: OFFLINE-SEED and OFFLINE-ONLINE results on Super Mario Land 1-1. Each curve tracks the best goal frame (lower is faster) found by an agent over 200 optimization turns. **Left:** agents seeded with a baseline reference trajectory (dotted line) all improve, but the final frame count varies widely across models. **Right:** the only run started from scratch converges to 3599 frames after a short burst of improvements.

Takeaways from offline optimization loop:

- **In OFFLINE-SEED, the seed reference trajectory strongly reflects the outcome**. The optimization loop is highly dependent on the seeded reference trajectory. If the model can find seed trajectory themselves, that is usually better than benign player.
- **Vision inputs are unnecessary** all models do not obtain screenshots under offline setting yet they successfully optimize the frame to reach goal from the seed trajectory.



(a) OFFLINE-SEED, from a 9498-frame seed.

(b) OFFLINE-SCRATCH, no seed.

Figure 5: SuperTux 1-1 speedrun results with an agent backed with various LLMs. Lines in OFFLINE-SCRATCH start from the turn an agent finds a valid trace to complete the goal.

4.5 Case Study on Improving Near-Optimal Speedrun Trace

The OFFLINE-SEED settings above start from seed traces far off the optimum, which leaves large room to improve for an agent. We report two experiments that instead start from a near-optimal trace on Super Mario Brothers: the first is asks an agent to improve one which is same as OFFLINE-SEED By seeding instead from an 18834-frame full-game Super Mario Brothers trace (Bisqwit, 2003) and given 200 turns, GPT-5.6-sol returned a trace three frames faster. The second is merging two near-optimal traces[5] which saved 26 frames from the fastest trajectory out of the two. These results suggest using multiple near-optimal trajectory has the potential to further improve those.

5 Analysis

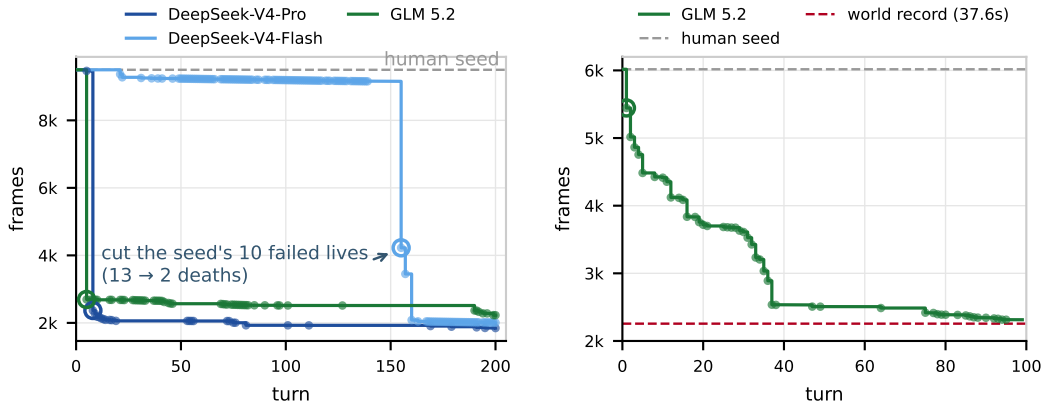
5.1 Expanding the Evaluation Budget Per Turn

In this setup, agents can evaluate and get feedback on the action traces up to 200 runs per turn. Figure 6b shows GLM 5.2 achieves a near world record run for Super Mario Land 1-1. Figure 6a shows Deepseek-v4-pro/flash, Kimi-K3 models being way better at experiments with unlimited evaluations of the trace. The number of times the agent can run the game largely influences how fast it can complete the game.

5.2 Contamination from Pretraining: Case Study on Pokemon vs. Tuxemon

That famous titles leave traces in model parameters is already established: Hu et al. (2026) reports game content memorised during pretraining , and Gemini 2.5 Pro has been observed confusing items across Pokémon versions, expecting tea where the game requires soda (Zhang, 2025). Both are observations about a single title. We therefore add a controlled pair: Pokémon and Tuxemon hold genre and setting fixed while varying how much a title is likely to appear in a pretraining corpus. Both are turn-based RPGs, run under OFFLINE-SEED with the same iteration budget, and scored on frames to the first gym. Speedrunning on the open-source title is generally more challenging: the best improvement over the seed falls from 24.4% on Pokémon (Kimi K3) to 5.9% on Tuxemon (GLM 5.3, Figure 7), and where every model on Pokémon at least reached the badge once, four of eleven models never reached the Tuxemon gym at all thus no improvement upon the benign user run. The ranking changes as well, which is harder to explain by difficulty alone: Kimi K3 leads Pokémon by a wide margin and is not among the arms that improve on Tuxemon, while Inkling-Small is last on

[5] https://tasvideos.org/UserFiles/Info/638242850875487965 and https://tasvideos.org/UserFiles/Info/638586818800241580



(a) SuperTux 1-1, top 3 models under this setup.

(b) Super Mario Land 1-1, GLM 5.2.

Figure 6: OFFLINE-SEED speedrun results with an agent allowed to submit an unlimited number of runs per turn. Super Mario Land world record is the 1-1 portion from EiP25 (2026).

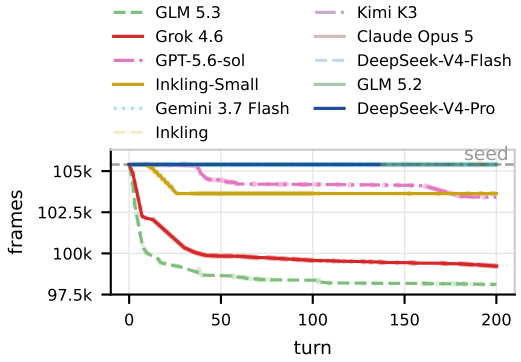


Figure 7: OFFLINE-SEED results on Tuxemon, the open-source counterpart to Pokémon, scored on frames to the first gym. All eleven arms start from the same 105,400-frame scripted route, so an arm that never improves stays flat on the seed line; four never reached the gym at all.

Pokémon and third on Tuxemon. We infer that popular titles are available in the pretraining corpus and suggest future game-based benchmarks to also experiment on less popular games.

6 Conclusion

We introduce SPEEDRUNBENCH, a benchmark that evaluates not whether an agent can finish a game but how fast. Agents can optimize a trace, and in the OFFLINE settings they do so without ever seeing the screen, which lets smaller open-weight models be evaluated alongside frontier ones. They remain far from human speedrunners on all but the simplest levels. Two things matters: the trace an agent starts from, and how many times it is allowed to evaluate a candidate within a turn. Both change results more than model choice does, and neither is visible to a benchmark that scores only completion.

Ethics Statement

SPEEDRUNBENCH is released for research and benchmarking purposes only. This paper and associated assets conveys no right to access, reproduce, distribute, modify, or commercially exploit any third-party title or its assets beyond what applicable licenses, terms of service, and law already permit. Users are responsible for lawfully obtaining the titles they evaluate on and for securing any

permissions their intended use requires, whether for evaluation, dataset construction, model development, or downstream deployment.

REFERENCES

Anthropic. System card: Claude Opus 5. https://www-cdn.anthropic.com/ceaf5c7ff2783855203fde8208ec311252dced5b/Claude%20Opus%205%20System%20Card.pdf, July 2026. Accessed: 2026-08-24.

Bisqwit. NES Super Mario Bros. "warps" in 05:15.650. TASVideos publication #665, December 2003. URL https://tasvideos.org/665M. Tool-assisted speedrun; Famtasia 5.1; obsoleted.

Giovanni Colantonio. Super mario bros. players uncover the biggest glitch in the game's 40-year history. 2026. URL https://www.polygon.com/super-mario-bros-ace-trick-discovered/.

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. *CoRR*, abs/1806.11532, 2018.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

EiP25. Super mario land – any% in 12:21.183. Speedrun.com, 2026. URL https://www.speedrun.com/sml1/runs/m7j0kgem. Verified run, 1st place, Super Game Boy 2 (JPN/NTSC).

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 (NeurIPS) Datasets and Benchmarks Track*, 2022.

ARC Prize Foundation. Arc-agi-3: A new challenge for frontier agentic intelligence, 2026. URL https://arxiv.org/abs/2603.24621.

Gemini Team, Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang

Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.7 Flash Model Card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-7-Flash-Model-Card.pdf, August 2026. Accessed: 2026-08-24.

Matthias Groß, Dietlind Zühlke, and Boris Naujoks. *Automating Speedrun Routing: Overview and Vision*, pp. 471–486. Springer International Publishing, 2022. ISBN 9783031024627. doi: 10.1007/978-3-031-02462-7_30. URL http://dx.doi.org/10.1007/978-3-031-02462-7_30.

Pranav Guruprasad, Sean Rivera, Helen Lu, Arushi Jain, Hangliang Ren, and Harshvardhan Sikka. Multinet 2.0 preview: Goal progress decays with task horizon for frontier vlms in interactive 2d environments, 2026. URL www.fig.inc/blog/multinet-v2-preview. MultiNet 2.0 Preview: Interactive 2D Mazes.

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? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qeziG97WUZ.

Sihao Hu, Tiansheng Huang, and Ling Liu. Pokellmon: A human-parity agent for pokemon battles with large language models, 2024. URL https://arxiv.org/abs/2402.01118.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. Deepswe: Measuring frontier coding agents on original, long-horizon engineering tasks, 2026. URL https://arxiv.org/abs/2607.07946.

Keisuke Kamahori, Shihang Li, Simon Peter, and Baris Kasikci. Vibeserve: Can ai agents build bespoke llm serving systems?, 2026. URL https://arxiv.org/abs/2605.06068.

Andrej Karpathy. autoresearch: An autonomous LLM research loop. https://github.com/karpathy/autoresearch, 2026. Accessed: 2026-05-06.

Seth Karten, Jake Grigsby, Tersoo Upaa Jr, Junik Bae, Seonghun Hong, Hyunyoung Jeong, Jaeyoon Jung, Kun Kerdthaisong, Gyungbo Kim, Hyeokgi Kim, Yujin Kim, Eunju Kwon, Dongyu Liu, Patrick Mariglia, Sangyeon Park, Benedikt Schink, Xianwei Shi, Anthony Sistilli, Joseph Twin, Arian Urdu, Matin Urdu, Qiao Wang, Ling Wu, Wenli Zhang, Kunsheng Zhou, Stephanie Milani, Kiran Vodrahalli, Amy Zhang, Fei Fang, Yuke Zhu, and Chi Jin. The pokeagent challenge: Competitive and long-context learning at scale, 2026a. URL https://arxiv.org/abs/2603.15563.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents, 2026b. URL https://arxiv.org/abs/2605.09998.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading, 2026. URL https://arxiv.org/abs/2607.08964.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

OpenAI. Gpt-5.6 system card. URL https://deploymentsafety.openai.com/gpt-5-6.

OpenCode Developers. Opencode: An open-source platform for code execution. https://github.com, 2026. Accessed: 2026-08-25.

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

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

SlNNED. Pokémon red/blue – any% glitchless in 1:43:40. Speedrun.com, 2025. URL https://www.speedrun.com/pkmnredblue/runs/y4n7endm. Verified run, 1st place, GBA emulator.

David Snyder. *Speedrunning: Interviews with the Quickest Gamers*. Studies in Gaming. McFarland, Jefferson, NC, 2017. ISBN 9781476670805.

SpaceXAI. Introducing Grok 4.6. URL https://x.ai/news/grok-4-6.

Mohammad Reza Taesiri, Tianjun Feng, Anh Nguyen, and Cor-Paul Bezemer. Glitchbench: Can large multimodal models detect video game glitches?, 2024. URL https://arxiv.org/abs/2312.05291.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang,

Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

The SuperTux Team. SuperTux. URL https://github.com/SuperTux/supertux/releases/tag/v0.6.3. Open-source 2D sidescrolling platform game.

Thinking Machines Lab. Inkling-small model card. URL https://thinkingmachines.ai/model-card/inkling-small/.

Tuxemon Contributors. Tuxemon: Open source monster-fighting rpg. GitHub repository, 2026. URL https://github.com/Tuxemon/Tuxemon. Commit abc1234.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

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.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. ViZDoom Competitions: Playing Doom from Pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019. doi: 10.1109/TG.2018.2877047. The 2022 IEEE Transactions on Games Outstanding Paper Award.

Alex L. Zhang, Thomas L. Griffiths, Karthik R. Narasimhan, and Ofir Press. Videogamebench: Can vision-language models complete popular video games?, 2026. URL https://arxiv.org/abs/2505.18134.

Joel Zhang. The making of Gemini Plays Pokémon. Blog post, Joel's Developer Blog, August 2025. URL https://blog.jcz.dev/the-making-of-gemini-plays-pokemon. Accessed: 2026-08-24.

Bingchen Zhao, Despoina Magka, Minqi Jiang, Xian Li, Roberta Raileanu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Kelvin Niu, Shagun Sodhani, Michael Shvartsman, Andrei Lupu, Alisia Maria Lupidi, Karen Hambardzumyan, Martin Josifoski, Edan Toledo, Thomas Foster, Lucia Cipolina-Kun, Derek Dunfield, Abhishek Charnalia, Alexander H Miller, Oisin Mac Aodha, Jakob Nicolaus Foerster, and Yoram Bachrach. The automated LLM speedrunning benchmark: Reproducing nanoGPT improvements. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=w98hMEjzu8.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.