ALE-NL: The Arcade Learning Environment in Natural Language

Roger Creus Castanyer
Mila Québec AI Institute
Université de Montréal
roger.creus-castanyer@mila.quebec

Abstract

Large Language Models (LLMs) have demonstrated increasingly general capabilities across a wide range of tasks. However, when deployed as agents that must reason and act over time, they often struggle with even simple forms of sequential decision-making. To study these limitations, we introduce ALE-NL, a natural language interface to the Arcade Learning Environment (ALE), designed for evaluating LLMs as agents in interactive settings. ALE-NL translates symbolic game states into natural language descriptions, allowing LLMs to engage with classic Atari games using text-based prompts. The framework supports a variety of prompting strategies, integrates seamlessly with existing APIs and includes tools for logging, visualization, and analysis, accelerating research in agentic LLM systems. By enabling language-based interaction in a long-standing reinforcement learning benchmark, ALE-NL offers a lightweight yet powerful testbed for studying reasoning, generalization, and the agentic capabilities of LLMs. ALE-NL is open-source and publicly available at https://github.com/roger-creus/ale-nl.

1 Introduction

Over the past decade, Reinforcement Learning (RL) has become a dominant framework for training autonomous agents capable of complex behavior [31]. Grounded in a rich theoretical foundation, RL has been central to some of the most impressive achievements in AI, including superhuman performance in games like Go [29] and Atari [26], as well as in robotics [9], operations research, and beyond [7, 3]. More recently, RL has played a crucial role in large-scale language models through Reinforcement Learning from Human Feedback (RLHF), enabling alignment and fine-tuning of foundation models [27].

Despite these advances, deep RL remains notoriously difficult to scale and deploy in practice [30]. Training effective policies often requires extensive tuning, large amounts of compute, and millions of environment interactions. These challenges are further compounded by common issues such as learning instabilities [21], sample inefficiency [12], and brittleness to small changes in environment dynamics or reward structure.

In parallel, Large Language Models (LLMs) have emerged as powerful, general-purpose systems with increasingly broad capabilities across domains. Their ability to incorporate vast knowledge priors, follow complex instructions, and generalize to new tasks has led to their rapid integration into downstream applications [2]. Importantly, LLMs are now widely accessible and can be deployed with minimal setup through APIs or open-source model libraries.

Motivated by these trends, recent research has explored the use of LLMs as decision-making agents, bypassing the training difficulties of RL by leveraging prompting techniques such as zero-shot reasoning, chain-of-thought, or few-shot in-context learning [25, 24]. These LLM-based agents have

Preprint. Under review.

demonstrated promising results in a variety of domains, from complex games to web navigation [33, 28].

However, many of these works operate under simplified or heavily abstracted settings, making it difficult to fairly assess the true decision-making capabilities of LLMs. Common simplifications include reduced or templated action spaces, handcrafted observations, and limited interaction depth. As a result, it is often unclear whether the observed performance reflects genuine reasoning and control ability, or simply clever prompt engineering over toyified versions of the problem.

To address this gap, we propose evaluating LLM agents in a rigorous and well-established benchmark: the Arcade Learning Environment (ALE) [1]. ALE has long been a gold standard for the RL community, offering diverse and challenging games with well-documented baselines for human, random, and RL agents [1, 19, 20, 32]. We introduce ALE-NL, a natural language interface to ALE that allows LLMs to interact with these environments through language alone. ALE-NL translates structured game states into natural language descriptions and feeds them into a customizable prompting pipeline. This enables the evaluation of LLM agents under consistent and transparent conditions, facilitating fair comparisons across LLMs, RL agents, and humans.

Our framework aims to enable deeper research into the agentic capabilities of LLMs, providing tools for benchmarking, visualization, and diagnostic analysis. It opens the door to studying how prompting strategies, model scale, and environment complexity interact in shaping LLM behavior in sequential decision-making settings.

2 Related Work

Since the introduction of LLMs, there has been rapid progress in their capabilities, sparking a growing need for thorough evaluation methods [4]. A wide range of benchmarks have emerged to assess different skills, such as mathematical reasoning [10], programming [5], software engineering [11], and multi-step problem solving [6]. These benchmarks are useful not only for comparing model performance but also for understanding what these systems are capable of in specific domains.

Beyond measurement, benchmarks also play a central role in shaping the research landscape. By offering shared tasks and metrics, they provide a framework for iterative improvement, making it easier to identify gaps and drive progress through competition and comparison.

More recently, there has been increasing interest in specialized benchmarks for evaluating LLMs as agents [15, 17, 16, 35]. These settings typically involve sequential decision making, planning, and sometimes collaboration with other agents. Games, in particular, have become a popular testbed, drawing from their long-standing use in RL, as they naturally support interactive and dynamic decision-making [23, 22]. When adapted for LLMs, games can help assess low-level control abilities, which are essential for building more general-purpose, autonomous systems that operate in open-ended environments [34].

At the same time, many existing agentic benchmarks simplify interaction through abstractions like code-based interfaces, templated instructions, or curated APIs [33, 25]. While these approaches are practical, they often bypass the harder problem of learning fine-grained control from raw observations or more natural interaction channels. To really assess the capabilities of LLMs as autonomous agents, it's important to design benchmarks that focus on these lower-level decisions and the challenges they pose.

To support this direction, we introduce ALE-NL, an open-source platform that connects LLMs to classic Atari games through a natural language interface. ALE-NL removes the need for symbolic APIs or intermediate abstractions, and instead encourages exploration of prompting strategies, decision-making, and reasoning grounded in textual observations and actions. It provides a reproducible and extensible environment to study LLMs as agents, and serves as a foundation for developing and evaluating more capable, grounded decision-making systems.

3 ALE-NL: Framework Overview

ALE-NL is a natural language interface built on top of the widely used ALE [1]. ALE-NL leverages OCAtari [8] as an intermediary layer to enable the evaluation of LLMs in classic Atari games.

ALE-NL provides an effective and fair platform for comparing the control capabilities of LLM agents to those of traditional RL agents and humans. It is designed to accelerate research in language-driven agent systems and foster a deeper understanding of how LLMs perform in sequential decision-making tasks. An overview of ALE-NL is shown in Figure 2.

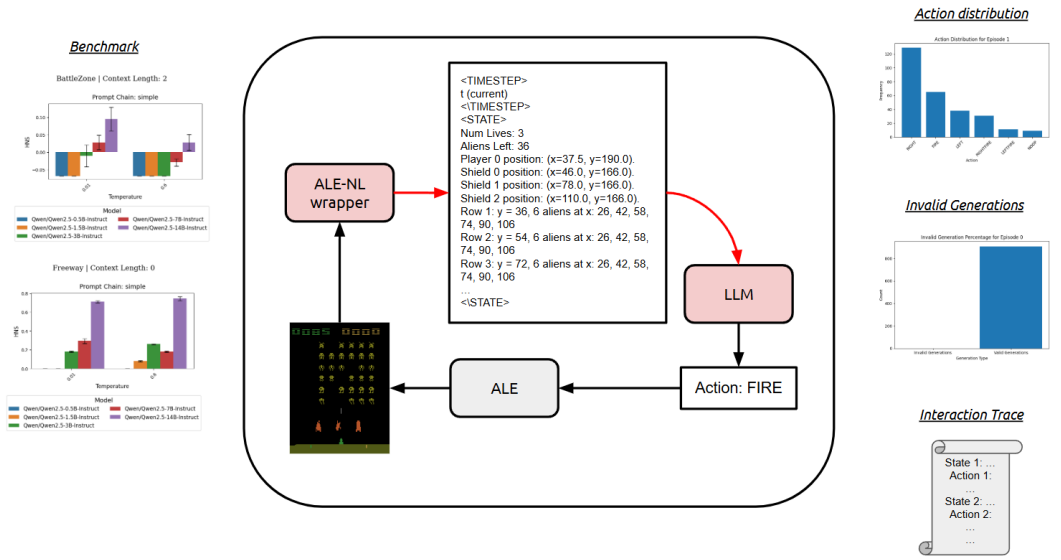

Figure 1: Overview of the ALE-NL framework. ALE-NL extends the ALE by providing a natural language interface that translates game states into descriptive text, enabling LLMs to interact with and control Atari games. The framework is fully customizable, supporting flexible prompting strategies, sampling parameter ablations, and compatibility with both HuggingFace and OpenAI models. ALE-NL also includes built-in tools for automated logging, benchmarking, and qualitative analysis of agent behavior through high-level statistics and visualizations.

3.1 Key Features of ALE-NL

ALE-NL includes a range of features that make it a comprehensive tool for benchmarking LLM agents across a variety of tasks and environments. Some of the core features are as follows:

- **12 Atari Games Supported**: ALE-NL currently supports twelve classic Atari games, with plans to add more. These include well-known games like SpaceInvaders, MsPacman, Asterix, and KungfuMaster. This diverse set of games allows for a broad evaluation of agent performance across different types of control challenges.
- **Support for HuggingFace and OpenAI Models**: ALE-NL allows users to run any text-generation model from HuggingFace locally, providing flexibility for researchers to test different models on their systems. Additionally, users can also run OpenAI's models via API, enabling the use of cutting-edge LLMs such as GPT variants.
- **Modular and Customizable Prompting Strategies**: ALE-NL supports several prompting techniques, including Chain-of-Thought (CoT), zero-shot, and few-shot strategies [14]. The framework provides a modular pipeline that allows users to easily customize the way models interact with the environment, ensuring a variety of testing conditions.
- **Ablation of Sampling Parameters**: Researchers can experiment with different sampling parameters (e.g., temperature, context length) to see how these factors influence agent behavior. This flexibility helps in understanding the underlying decision-making processes of LLM agents and enables more granular control over model behavior.
- **One-Click Benchmarking**: ALE-NL comes with a built-in benchmarking tool, allowing users to instantly generate and visualize performance metrics. This makes it easier to compare models on multiple metrics and draw meaningful conclusions about their effectiveness.

- **Visual and Statistical Debugging**: ALE-NL includes tools for inspecting and debugging agent behavior. Users can visualize action distributions, generation errors, and review full interaction traces (see Figure 5). These tools provide insights into agent performance, highlighting where agents might be failing or underperforming.

3.2 Prompting in ALE-NL

ALE-NL supports a flexible and modular prompting framework, which is essential for enabling language-driven agent behavior. The prompts used to guide LLMs are dynamically composed from three key components:

1. **Game Descriptions**: Each game in the ALE-NL suite comes with a detailed description of its mechanics and objectives. These descriptions are loaded from the official ALE documentation. The descriptions help the LLMs understand the environment and the context in which they are operating. For example, in the game SpaceInvaders, the description includes information about the player's goal (destroying alien invaders) and the mechanics of the game (movement, shooting).
2. **Prompt Chains**: The system supports various prompting strategies via prompt chains. These chains define how the LLM interacts with the environment. Users can customize the chain to employ different strategies like zero-shot, where the model receives no prior information beyond the game description, or Chain-of-Thought (CoT), which enables the model to break down the problem into intermediate reasoning steps before selecting an action [14].
3. **State Descriptions**: Each game has specific state descriptions that define how the game state is translated into natural language. These descriptions are customizable per game. For instance, in MsPacman, the state description might include the position of walls, food, and enemies, as well as the current score. This allows the LLM to receive consistent and structured input about the environment, which is critical for effective decision-making.

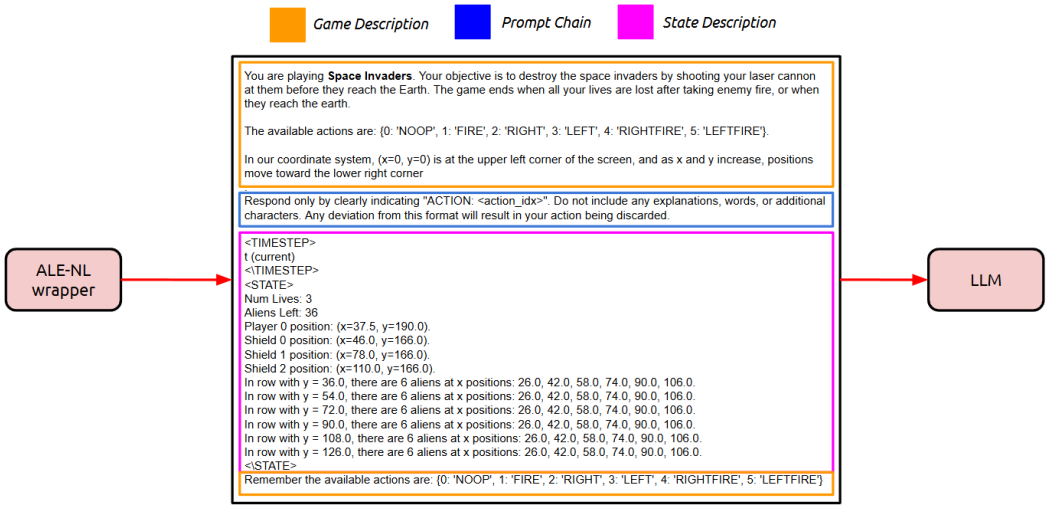

Figure 2: Prompt construction in ALE-NL. Each prompt is modularly composed of three components: the game description (sourced from the ALE documentation), a user-defined prompt chain (e.g., zero-shot, few-shot, chain-of-thought), and a programmatically generated state description based on the current game state. This structure enables flexible and interpretable interaction with LLMs.

4 Benchmarking with ALE-NL

The primary goal of ALE-NL is not to advocate for a particular model, prompting strategy, or specific performance result. Instead, this work aims to provide a flexible and open-source platform that

accelerates research into the agentic capabilities of LLMs. By standardizing the environment, input modality, and evaluation tools, ALE-NL allows researchers to probe the limits and behaviors of LLMs in sequential decision-making tasks under a fair and transparent setting. While the main contribution is

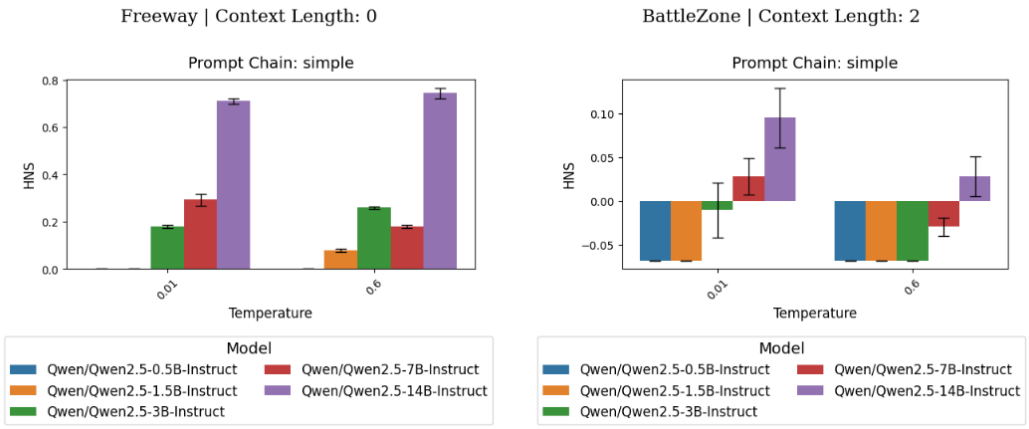

Figure 3: Example benchmark results in Freeway and Battlezone using the Qwen model family. Bars represent the mean Human Normalized Score (HNS), with error bars denoting standard deviation across three independent runs. ALE-NL enables effortless generation of such visualizations, allowing users to compare LLM performance across environments, model sizes, and sampling parameters (context length indicated in the title, temperature along the x-axis). In both games, larger models consistently achieve higher performance. Results shown use zero-shot prompting.

the framework itself, we have conducted a set of benchmark experiments using the Qwen model family (`Qwen-0.5B`, `1.5B`, `3B`, `7B`, `14B`) across a subset of the supported Atari games. These benchmarks are not meant to draw definitive conclusions about model performance, but rather to illustrate how ALE-NL enables structured experimentation, interpretability, and reproducibility when evaluating LLM agents. Following standard practice in the ALE for RL research, we evaluate LLM agents across 3 independent runs for each experimental configuration and report the Human-Normalized Score (HNS). The HNS is computed by scaling raw game scores relative to the performance of a random agent and a human player. An HNS of 1.0 corresponds to human-level performance, while scores above or below this threshold indicate superhuman or subhuman behavior, respectively. A negative HNS suggests that the agent performs worse than a random policy, highlighting potential failure cases in understanding or control. This metric allows for consistent comparisons across agents, games, and prompting setups.

Interestingly, Freeway, a game where the player controls a chicken attempting to cross a busy road, is considered a hard-exploration environment in the reinforcement learning literature [32, 1]. This is due to the fact that the agent only receives a reward upon successfully reaching the other side, making it difficult for traditional RL algorithms to learn meaningful behaviors from sparse feedback. Despite the conceptual simplicity of the task, many RL agents struggle to explore effectively. In contrast, LLMs can leverage their built-in world knowledge and common-sense reasoning to infer, even in a zero-shot setting, that the optimal behavior likely involves repeatedly choosing the `UP` action over alternatives like `DOWN` or `NOOP`. This insight allows models like Qwen-14B to achieve a high HNS of 0.8, reaching 80% of human-level performance in this game without any fine-tuning or task-specific training.

A broader evaluation across additional games is presented in Table 1. For conciseness, we report results only for *temperature = 0.6* and *context_length = 0*. Each score reflects the average over three independent runs.

5 Discussion: Future Work

The primary goal of ALE-NL is not to declare any specific language model or prompting strategy as universally superior, but to empower and accelerate research on the agentic capabilities of large

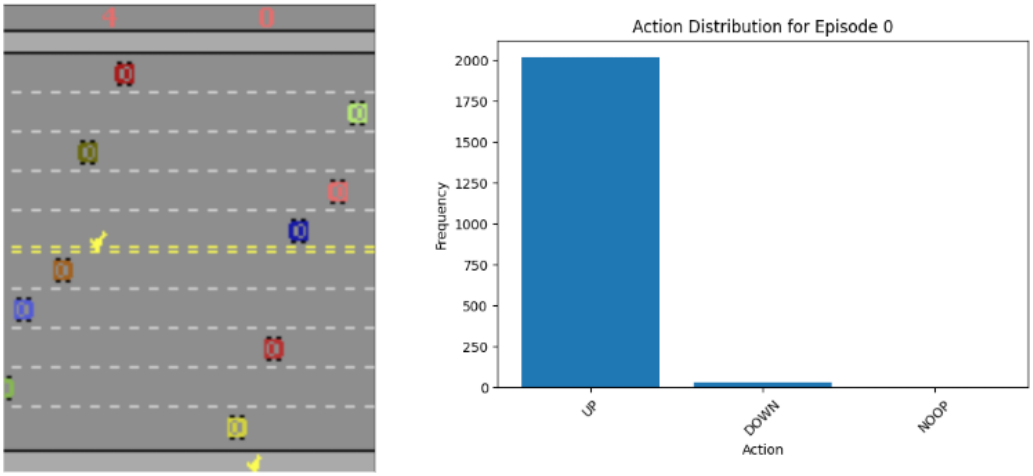

Figure 4: **(Left)** Screenshot of the game Freeway, where the player controls a yellow chicken attempting to cross a busy road while avoiding oncoming cars. **(Right)** Action distribution of the Qwen-14B model, which correctly identifies that the `UP` action should be taken far more frequently than alternatives, demonstrating its ability to infer the objective through zero-shot reasoning.

language models. By offering a flexible and transparent platform for evaluating LLMs in classic control environments, ALE-NL bridges a critical gap between RL and language-based reasoning systems.

Looking forward, we highlight several promising directions for future research enabled by ALE-NL:

- **Fine-tuning LLMs for Low-Level Control.** Current experiments in ALE-NL rely on zero-shot performance from general-purpose LLMs. A natural next step is to fine-tune models on control-centric tasks or on interaction data from ALE, potentially improving their consistency and performance in low-level decision making.
- **Better Evaluating Reasoning Models:** Future work could focus on improving the evaluation of reasoning models, such as DeepSeek-R1 [27], in complex decision-making environments. Understanding how these models reason in dynamic, sequential settings could be crucial for advancing agentic LLM systems.
- **Incorporating Vision-Language Models (VLMs).** While ALE-NL translates game states into text to leverage the reasoning capabilities of language models, future extensions may explore the use of VLMs that operate directly from visual inputs. Despite recent progress, current VLMs often lag behind LLMs in structured reasoning tasks, especially in multi-step decision making. ALE-NL could serve as a diagnostic tool to probe and compare their relative strengths.
- **LLM-Guided RL Agents.** Combining the high-level generalization and priors of LLMs with the fine-grained control learned by RL agents is a promising hybrid approach [13, 18]. ALE-NL can facilitate research into architectures and interfaces that enable language models to guide or modulate RL policies, acting as abstract planners or environment advisors.
- **Expanding the Benchmark Suite.** A crucial next step is to populate the ALE-NL benchmark with results from a broader range of LLM baselines. This will enable more rigorous comparisons, enable leaderboard-style evaluation, and help contextualize progress in agentic LLM systems.
- **Studying Prompt Engineering at Scale.** The modularity of ALE-NL makes it possible to conduct large-scale studies of prompting strategies across a variety of games and model families. Understanding the sensitivity and transferability of prompts in control settings remains an open research problem [14].
- **Safe and Interpretable Decision Making.** The transparency of text-based control in ALE-NL opens avenues for studying interpretability and safety in LLM decisions. For instance,

tracing generation errors or hallucinated actions can provide insights into failure modes and help build more robust control agents.

In summary, ALE-NL lays the groundwork for a new class of research at the intersection of language modeling, decision making, and embodied reasoning. We hope it serves as a catalyst for more rigorous and reproducible work in this emerging space.

6 Conclusion

We introduce ALE-NL, a lightweight and extensible framework for evaluating large language models as control agents in the classic Atari Learning Environment [1]. By leveraging natural language interfaces, ALE-NL enables a transparent and reproducible setup for studying the agentic behavior of LLMs across a diverse set of environments, without requiring visual inputs or complex wrappers.

Rather than aiming to benchmark and rank models conclusively, this work focuses on enabling the broader research community to explore LLMs in interactive, multi-step decision-making tasks. ALE-NL is highly modular and supports any HuggingFace or OpenAI language model, customizable prompting strategies, and automated tools for benchmarking, logging, and analysis.

We open-source ALE-NL to accelerate research at the intersection of reinforcement learning, language modeling, and embodied AI. We hope the community will use and extend this platform to probe the limits of LLM reasoning, understand their behavioral dynamics, and develop more capable agentic systems.

Code: `https://github.com/roger-creus/ale-nl`

References

[1] 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.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Pablo Castro, Ajit Desai, Han Du, Rodney Garratt, and Francisco Rivadeneyra. Estimating policy functions in payment systems using reinforcement learning. *ACM Transactions on Economics and Computation*, 13(1):1–31, 2025.

[4] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[5] 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.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[8] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. Ocatari: Object-centric atari 2600 reinforcement learning environments. *arXiv preprint arXiv:2306.08649*, 2023.

[9] Shixiang Gu, Ethan Holly, Timothy P Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1610.00633*, 1(1), 2016.

[10] 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. *arXiv preprint arXiv:2103.03874*, 2021.

[11] 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? *arXiv preprint arXiv:2310.06770*, 2023.

[12] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[13] Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

[14] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

[16] Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. Agentsims: An open-source sandbox for large language model evaluation. *arXiv preprint arXiv:2308.04026*, 2023.

[17] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[18] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[19] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[21] Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36:37142–37159, 2023.

[22] Munan Ning, Bin Zhu, Yujia Xie, Bin Lin, Jiaxi Cui, Lu Yuan, Dongdong Chen, and Li Yuan. Video-bench: A comprehensive benchmark and toolkit for evaluating video-based large language models. *arXiv preprint arXiv:2311.16103*, 2023.

[23] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

[24] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[25] Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*, 2023.

[26] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

[28] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

[29] 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.

[30] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pages 32145–32168. PMLR, 2023.

[31] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[32] Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

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

[34] Tom Ward, Andrew Bolt, Nik Hemmings, Simon Carter, Manuel Sanchez, Ricardo Barreira, Seb Noury, Keith Anderson, Jay Lemmon, Jonathan Coe, et al. Using unity to help solve intelligence. *arXiv preprint arXiv:2011.09294*, 2020.

[35] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Benchmark Results

Table 1: Human-Normalized Scores (HNS) for various LLMs across Atari environments.

Model	Environment	HNS
Qwen2.5-0.5B	Asterix	-0.0032
Qwen2.5-1.5B	Asterix	-0.0012
Qwen2.5-3B	Asterix	-0.0193
Qwen2.5-7B	Asterix	**0.0068**
Qwen2.5-14B	Asterix	-0.0133
Qwen2.5-0.5B	BattleZone	-0.0390
Qwen2.5-1.5B	BattleZone	**0.1332**
Qwen2.5-3B	BattleZone	0.0567
Qwen2.5-7B	BattleZone	-0.0103
Qwen2.5-14B	BattleZone	0.1045
Qwen2.5-0.5B	BeamRider	0.0152
Qwen2.5-1.5B	BeamRider	-0.0140
Qwen2.5-3B	BeamRider	-0.0220
Qwen2.5-7B	BeamRider	**0.0161**
Qwen2.5-14B	BeamRider	-0.0105
Qwen2.5-0.5B	Boxing	-1.5361
Qwen2.5-1.5B	Boxing	**-0.2306**
Qwen2.5-3B	Boxing	-2.7583
Qwen2.5-7B	Boxing	-3.2306
Qwen2.5-14B	Boxing	-1.3972
Qwen2.5-0.5B	Breakout	**0.0567**
Qwen2.5-1.5B	Breakout	-0.0359
Qwen2.5-3B	Breakout	0.0104
Qwen2.5-7B	Breakout	-0.0359
Qwen2.5-14B	Breakout	-0.0590
Qwen2.5-0.5B	Freeway	0.0000
Qwen2.5-1.5B	Freeway	0.0788
Qwen2.5-3B	Freeway	0.2590
Qwen2.5-7B	Freeway	0.1802
Qwen2.5-14B	Freeway	**0.7432**
Qwen2.5-0.5B	KungFuMaster	**-0.0041**
Qwen2.5-1.5B	KungFuMaster	-0.0115
Qwen2.5-3B	KungFuMaster	-0.0115
Qwen2.5-7B	KungFuMaster	-0.0115
Qwen2.5-14B	KungFuMaster	-0.0115
Qwen2.5-0.5B	MsPacman	-0.0146
Qwen2.5-1.5B	MsPacman	0.0215
Qwen2.5-3B	MsPacman	-0.0372
Qwen2.5-7B	MsPacman	**0.0742**
Qwen2.5-14B	MsPacman	-0.0372
Qwen2.5-0.5B	SpaceInvaders	-0.0217
Qwen2.5-1.5B	SpaceInvaders	0.0035
Qwen2.5-3B	SpaceInvaders	0.0561
Qwen2.5-7B	SpaceInvaders	0.0583
Qwen2.5-14B	SpaceInvaders	0.0419
GPT-4 Turbo (2024-04-09)	SpaceInvaders	**0.1230**
GPT-4o (2024-08-06)	SpaceInvaders	0.0046
GPT-4o Mini	SpaceInvaders	0.0030
GPT-3.5 Turbo (0125)	SpaceInvaders	0.0605

B Interaction Trace

Below, we present a single step from an interaction between ALE-NL and an LLM. While ALE-NL logs the complete interaction trace for each episode, we display only one step in Figure 5 for clarity.

Interaction Trace

USER: You are playing **SpaceInvaders**. Your objective is to destroy the space invaders by shooting your laser cannon at them before they reach the Earth. The game ends when all your lives are lost after taking enemy fire, or when they reach the earth. The available actions are: 0: 'NOOP', 1: 'FIRE', 2: 'RIGHT', 3: 'LEFT', 4: 'RIGHTFIRE', 5: 'LEFTFIRE'. In our coordinate system, (x=0, y=0) is at the upper left corner of the screen, and as x and y increase, positions move toward the lower right corner.

Respond only by clearly indicating "ACTION: <action_idx>". Do not include any explanations, words, or additional characters. Any deviation from this format will result in your action being discarded. E.g. ACTION: 3

<TIMESTEP>
t (current)
<TIMESTEP>
<STATE>
Num Lives: 3
Aliens Left: 36
Player 0 position: (x=37.5, y=190.0).
Shield 0 position: (x=46.0, y=166.0).
Shield 1 position: (x=78.0, y=166.0).
Shield 2 position: (x=110.0, y=166.0).
In row with y = 36.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
In row with y = 54.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
In row with y = 72.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
In row with y = 90.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
In row with y = 108.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
In row with y = 126.0, there are 6 aliens at x positions: 26.0, 42.0, 58.0, 74.0, 90.0, 106.0.
<STATE>

LLM: RIGHTFIRE

USER:

...

Figure 5: Example of an interaction logged by ALE-NL in the SpaceInvaders environment. The log captures the user prompt, which includes the game description, a user-defined prompt chain (in this case, zero-shot: 'Respond only by indicating the immediate action, without extra explanations'), and the game state description wrapped in <STATE> tags. This automatically generated log provides transparency into the decision-making process of the LLMs during control tasks.