# Breaking Boundaries: Large Language Models as Atari Breakout Agents

Nicholas Paul
*Dept. Mathematics and Computer Science*
*Lawrence Technological University*
Southfield, USA
npaul@ltu.edu

Siri sri Churakanti
*Dept. Mathematics and Computer Science*
*Lawrence Technological University*
Southfield, USA
schurakan@ltu.edu

Nazeer Uddin Syed
*Mathematics and CS*
*Lawrence Technological University*
Southfield, USA
schurakan@ltu.edu

***Abstract*—The present study examines the viability of Large Language Models (LLMs) to act in the game of Atari Breakout, demonstrating their versatility in capabilities beyond a primarily text-based task. This research investigates LLMs, supplied with textual descriptions of the game state, such as those describing the game environment on a frame-by-frame basis, in their capacities of both predicting the ball's movement trajectory and controlling the paddle. We then evaluate the efficacy of LLMs to constitute prompt-based approaches with deterministic rule-based algorithms and human performance.**

**Given experimental studies with varied prompts—simple and complex—it is observed that larger language models like GPT-4o-mini are superior to smaller ones in terms of prediction accuracy and scoring, although at the cost of increased computations. Each of the rule-based AIs has baseline performance with LLM-generated rules and obtains varying performance, showing a great time difference to evaluation of deterministic measures. A high score of 405, from the LLM-desired rule-based process, indicates hope for hybrid models that mix LLM-conceived reasoning with the efficiency of a rule-enforcing algorithm.**

**The present work has illustrated dynamic environments such as Atari gaming and offers perspectives on their possible application, as well as the enabling strengths and limitations they express. These findings help further AI's development in real-time decision-making tasks and provide a basis for future research, aiming at integrating LLMs with the more established AI methodologies.**

## I. Introduction

Atari Breakout is a classic arcade game designed by Steve Wozniak inspired by Nolan Bushnell and Steve Bristow's vision based on *Pong*, the Atari arcade game released in 1972. Designed with a layer of bricks fixed onto the top third of the screen, the objective of Breakout is to destroy all the bricks by a ball that continuously bounces off the paddle into them. This simplicity combined with dynamic challenges has made it a standard benchmark for testing the prowess of artificial intelligence.

AI has since developed remarkably, revealing a plethora of opportunities to experiment with, precisely when applied to dynamic versus interactive tasks. The simplicity of the applications along with clearly defined objectives made Breakout particularly suitable for trying out various styles of AI. While traditional methods like reinforcement learning (RL) tend to dominate the research in this regard, they have drawbacks; RL often demands extra processing power, training that may take a long time, as well as reward functions either pre-defined or learned from experience. Our work diverges from the works mentioned above and bases the paddle control mechanism on LLMs such as GPT-4o and Llama. By relating textual descriptions of the game state, LLMs predict actions without a prior training phase, which proves they can be learned zero-shot.

Large language models are deep learning networks, often transformer-based, trained on vast amounts of text data to generate human-like text. Although they primarily work with language, the structure and principles of these models can be adapted to solve other sequential problems, such as decision-making in games. Large language models (LLMs), particularly transformer-based architectures, excel in processing sequential data, making them adaptable to domains like gaming. By modeling gameplay as a sequence of actions, similar to tokens in text, LLMs can predict optimal moves based on prior actions. This approach leverages the contextual nature of decision-making in games, enabling LLMs to handle sequential outcomes and generate coherent gameplay strategies.

Moreover, this study integrates rule-based AI—a deterministic, computationally efficient technique able to calculate the course trajectories of the balls from thus defining the function of the paddle. The AI's achievement can be put into context against that of a human baseline. By comparing these methods, the aim of the project was to check how effectively each predicts the ball's trajectory, optimizes paddle movements, and leads to high scores. This hybrid approach of combining LLMs with traditional methods provides a fresh outlook as to how dynamic decision-making tasks could efficiently be solved.

The document inquired into the methods used—the rule-based algorithms and the LLM-prompt-based methods—and their comparative performance against human players. The analysis highlighted practical uses for LLMs in non-textual, interactive tasks, thereby relieving the overhead of traditional AI methods and providing further avenues for the integration of LLMs into gaming, including other dynamic systems.

By allowing the combination of more conventional AI techniques and newer language models, this research bears on the role AI presents in multimodal and dynamic settings. The results indicate not only the prospects for LLM in the realm of gaming but also in broader contexts of AI that can provide insights for researchers and practitioners alike.

### A. Background

The application of Large Language Models (LLMs) as low-level policy controllers has garnered significant interest in recent research, marking a shift from traditional reinforcement learning methods. A foundational study in this area, ”Playing Atari with Deep Reinforcement Learning”[1] established the effectiveness of deep Q-networks for learning directly from high-dimensional sensory inputs. Building on this, ”Atari-GPT” Jiang et al[2]. demonstrated the feasibility of using multimodal LLMs to process visual game frames and generate actions, highlighting their potential in game-playing tasks.

Further exploring the versatility of LLMs, Liang et al. illustrated how LLMs can generate code-based policies for robotic control, translating high-level instructions into executable actions. [3] Similarly, Shentu et al. proposed using LLMs to create latent codes that connect high-level commands to low-level actions in robotic systems [4].

Our study builds upon these advancements by evaluating LLMs as low-level policy controllers in Atari Breakout, specifically utilizing textual descriptions of the game state as input. This approach aims to leverage the natural language understanding capabilities of LLMs while exploring their effectiveness in game-playing scenarios without direct visual input processing.

## II. Methodology

In this study, we evaluate two methods for action selection via an LLM: LLM as an agent and LLM as an agent generator.

When using an LLM as an agent directly, we generate a text-based description of the current world state, embed the state into a prompt, and then request the LLM to select an action for that frame. We evaluate two prompt formats, a simple prompt that asks for an action directly and a more complex one which encourages chain of thought[5] reasoning. This direct usage of LLM as a policy selection agent is similar to the method used by [2]. However, in our study, we provide the information in text rather than ask the LLM to evaluate the frame directly.

When using an LLM as an agent generator, we ask the LLM to generate a rule based agent in Python with inputs from the world state such as ball position and paddle position of the previous and current frame. Two rule based policies are compared. The first iteration and the final iteration after human feedback.

### A. Prompt Generation

For the prompt based agents, we evaluated two prompts, a simple one and a complex one. Both prompts contained some information about the context (”We are playing Atari breakout...”) and a description of the frame.

The simple prompt template was generated by Llama-8B[6]. It contains a short sentence about the context, a textual description of the frame, and a direct question asking the LLM to generate an action: left, right, or stay. For the simple prompt, the textual description of the frame contained information about the relative location of the ball and paddle (Full sample prompts are provided in the appendix).

*Frame Update:*
*Ball position (X,Y): (120, 120)*
*Paddle position (X): 156*
*Paddle Position: Right side of the screen*
*Ball Position: Far to the left of the paddle*
*Ball Direction: Moving down and to the left*
*Ball height relative to paddle: Mid-range*

Unlike the simple prompt, the complex prompt does not give any extrapolated information about the ball and paddles relative locations. It only provides $X$ and $Y$ coordinates directly and tasks the LLM with figuring out the rest on its own. The complex prompt template was generated by GPT-4o and includes information about to approach the problem. GPT-4o was tasked with creating a prompt a smaller AI could use to figure out how to move the ball. The full prompt includes information hinting the AI to account for the momentum of the paddle and encourages the LLM to calculate the target intersection location of the ball and paddle.

### B. Rule Based Agent Generation

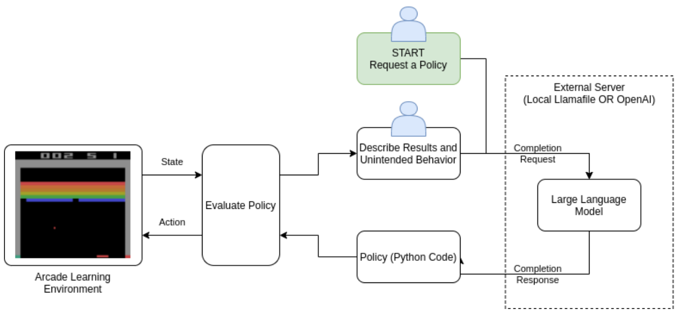

Fig. 1. Using an LLM as an agent generator with a human in the loop.

When using LLMs as an agent generator, we asked the LLM to generate a rule based policy based on information about the game. The LLM was given feedback about how well it's policy did and was asked to update the rule based algorithm to improve the score. The iteration process involved a human in the loop to provide feedback. The results discussed in this paper are based off the first and final policies after a few iterations. A full comparison across iterations is not discussed in this paper[1], only the first and final iteration.

### C. Experimental Setup

In order to evaluate each approach, we used the Arcade Learning Environment[7] based off of OpenAI's gym[8]. For each frame, we extract relevant information such as the ball position and the paddle position from the system virtual RAM (Figure 2).

For the prompt based agents, we insert these values directly into the prompt template and then send a completion request

[1]Code samples for all iterations are preserved but they are not evaluated here primarily due to the limited scope of this work.

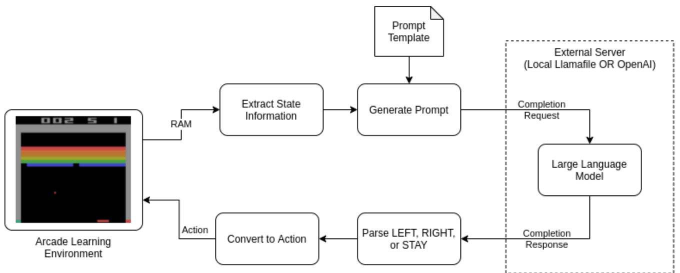
Prompt Template
External Server (Local Llamafile OR OpenAI)
RAM
Extract State Information
Generate Prompt
Completion Request
Large Language Model
Action
Convert to Action
Parse LEFT, RIGHT, or STAY
Completion Response
Arcade Learning Environment

Fig. 2. Experimental setup for using an LLM as an agent directly. Information from the game state is extracted each frame and formatted using a prompt template. An LLM generates an action based on the prompt. The action is used to advance the game until the next frame.

to a LLM server. We parse LEFT, RIGHT, or STAY from the response and use that as the action for that frame. This process continues until the episode completes. We evaluate several LLMs including small locally running models. All Llama 3.2 models were running locally using a llamafile[2] server.

- Llama 3.2 1 Billion (Simple Prompt Only)
- Llama 3.2 3 Billion
- Llama 3.2 8 Billion (Simple Prompt Only)
- GPT-4o Mini

## III. Results

### A. Model Performance

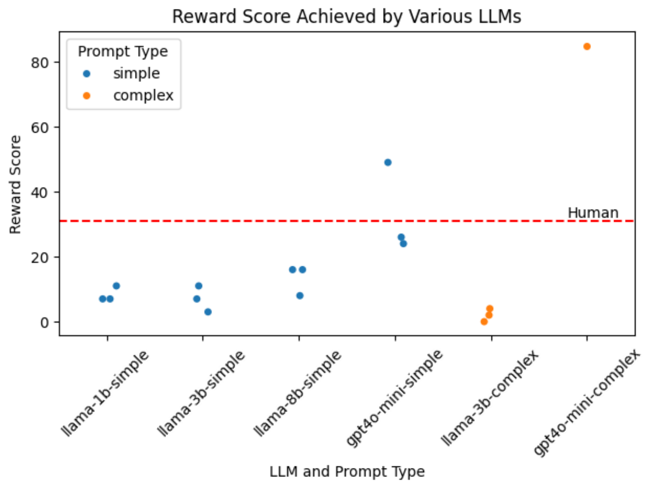

Fig. 3. LLM Results.

The score for each model and prompt type is shown in Figure 3. For the simple prompt, our results demonstrate a clear correlation between model size and performance in the task. Smaller models like Llama 1B and 3B achieved scores around 10 points. Llama 8B showed marginal improvement over its smaller counterparts, but the difference was not substantial. The GPT4o-mini model surpassed the human baseline on average across three trials.

[2]https://github.com/Mozilla-Ocho/llamafile

Llama 3B was evaluated using both the simple and complex prompts. When evaluated using the complex prompt, its performance degraded significantly compared to the simple prompt. With the simple prompt, the model was able to pick up on key phrases such as "the ball is to the left of the paddle" and quickly reason that it should also move the paddle left. But this type of information was not included in the complex prompt. The complex prompt required additional reasoning in order to figure out the result. This additional reasoning provided more flexibility but was also significantly more complicated to get right.

The most impressive results came from GPT4o-mini when provided with the complex prompt. In this single trial, the model significantly outperformed the human baseline. Notably, it demonstrated the ability to anticipate intersection points with the paddle several frames in advance, a behavior specifically requested in the prompt. This showcases the model's capacity to not only understand complex instructions but also to execute them effectively in a dynamic environment. While these results are promising, it's important to note that the complex prompt configuration has only been tested once due to its time-intensive nature, and further trials would be necessary to confirm its consistency.

| type | Score |
|---|---|
| DQN | 168 |
| Rule Based Policy (Final Iteration) | 145 |
| GPT4o-mini (Complex Prompt) | 85 |
| Rule Based Policy (First Iteration) | 46 |
| GPT4o-mini (Simple Prompt) | 33 |
| Human | 31 |
| Llama-8B (Simple Prompt) | 13 |
| Llama-1B (Simple Prompt) | 8 |
| Llama-3B (Simple Prompt) | 7 |
| Atari-GPT | 4 |
| Llama-3b (Complex Prompt) | 2 |

TABLE I
Average scores for various agents evaluated for a maximum of 1000 steps.

In Table III-A, we compare the prompt based approaches to the DQN[1] and [2], and Human baseline[1]. No model, prompt based or rule based, was able to outperform the DQN implementation. When given simple prompts, even the smallest 1B models were able to outperform Atari-GPT. This is likely due to the fact that Atari-GPT uses vision as input which proved to be a difficult task for breakout though it was capable of performing well on other games. GPT-4o-mini was able to outperform the human baseline with both the simple and complex prompts. The simple prompt had a slight advantage while the complex prompt scored more than twice as high as human players.

Figure 4 shows that by using methods similar to [4] to create policies from high level descriptions, we see that we are able create a rule based policy that outperforms all models except DQN.

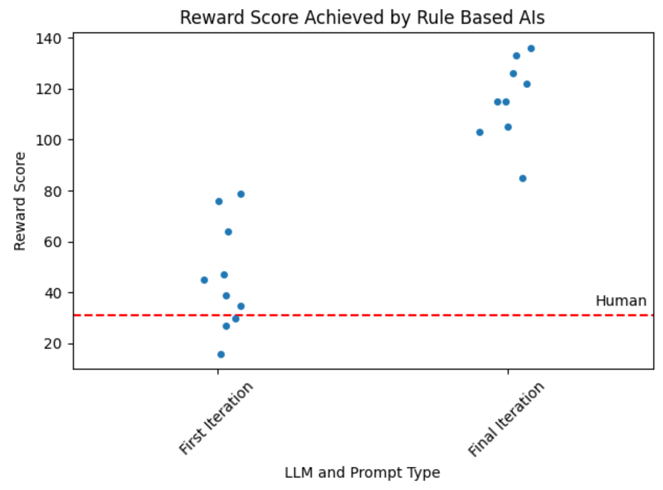

Fig. 4. Reward scores for the first and final iteration of a rule based algorithm

## IV. Discussion & Conclusion

### A. Future Research Ideas Addressing our Limitations

By enhancing complexity, such as introducing procedurally generated levels and multi-objective gameplay, to create richer challenges. Improving state representation by combining pixel data with game-specific variables like ball speed can offer a more comprehensive understanding of the environment.

Efforts to refine evaluation metrics should prioritize metrics beyond score, such as paddle efficiency and trajectory optimization, ensuring robustness under varying conditions. Addressing sparse rewards through reward shaping and techniques like Temporal Difference learning can mitigate feedback limitations.

To overcome agent-specific drawbacks, focus on transferable skills through multi-game training and enhance adaptability via ensemble learning. Optimizing LLM architectures for real-time decision-making and employing hierarchical learning can address computational challenges.

Developing tools for explainability and policy debugging will aid in transparency and iterative improvements. Expanding Breakout with complex variants and integrating human-AI collaboration can broaden its utility as a benchmark.

Atari Breakout, a classic arcade game, has long served as a benchmark for artificial intelligence research in game-playing agents. While traditional reinforcement learning techniques have achieved notable success in mastering such games, the emergence of Large Language Models (LLMs) presents a novel opportunity to approach gameplay from a new perspective. Originally designed for natural language processing, LLMs have demonstrated remarkable adaptability to various tasks that can be framed as text-based problems.

This paper explores the potential of LLMs in playing Atari Breakout by processing textual descriptions of the game state frame by frame. By investigating how effectively LLMs can interpret these text-based representations and generate appropriate actions, we aim to compare their performance with established reinforcement learning methods and gain insights into their reasoning capabilities in a game context. This approach not only tests the limits of LLMs in decision-making scenarios but also explores new possibilities in bridging natural language understanding with game AI, potentially opening avenues for more intuitive and flexible game-playing agents.

### B. Discussion

This study highlights the potential of Large Language Models (LLMs) as agents in dynamic tasks like Atari Breakout. GPT-4o-mini demonstrated strong reasoning and action prediction, outperforming smaller LLMs and human players in specific cases. However, its performance lagged behind reinforcement learning (RL) techniques like DQN, which are more efficient for precise, computationally intensive tasks.

The hybrid approach of combining LLM-generated insights with rule-based policies showed promise, balancing adaptability and efficiency. Challenges such as the time-intensive nature of complex prompts and reduced performance with increased reasoning complexity underscore the need for better prompt engineering and integration with traditional methods.

### C. Conclusion

This study demonstrates the feasibility of LLMs as agents in dynamic gaming environments by using textual descriptions to predict actions and control gameplay in Atari Breakout. Despite their limitations, LLMs like GPT-4o-mini achieved remarkable results, particularly when guided by well-designed prompts. The introduction of hybrid models, combining the reasoning power of LLMs with the efficiency of rule-based algorithms, offers a promising avenue for future research.

The potential applications of LLMs extend beyond gaming into broader domains requiring real-time decision-making and multimodal understanding. Future work should focus on enhancing model efficiency, improving state representation, and exploring hybrid frameworks that capitalize on the strengths of both LLMs and traditional AI methods. By addressing these challenges, LLMs could play a pivotal role in advancing AI's capabilities in complex, dynamic environments.

## References

[1] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] N. R. Waytowich, D. White, M. Sunbeam, and V. G. Goecks, *Atari-gpt: Investigating the capabilities of multimodal large language models as low-level policies for atari games*, 2024. arXiv: 2408.15950 `[cs.AI]`. [Online]. Available: https://arxiv.org/abs/2408.15950.

[3] J. Liang, W. Huang, F. Xia, *et al.*, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9493–9500.

[4] Y. Shentu, P. Wu, A. Rajeswaran, and P. Abbeel, "From llms to actions: Latent codes as bridges in hierarchical robot control," *arXiv preprint arXiv:2405.04798*, 2024.

[5] J. Wei, X. Wang, D. Schuurmans, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[6] A. Dubey, A. Jauhri, A. Pandey, *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.

[7] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[8] G. Brockman, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

## Appendix

### Sample *Simple* Prompt

```
Lets play atari breakout. I will describe what is
    happening on the screen and you will tell me
    which direction to the move the paddle or to
    keep it still. In order to win, you will need to
     move the paddle so that it is aligned with the
    ball. You may also choose not to move the paddle
    . Here is the current frame:
Frame Update:
 - Ball position (X,Y): (120, 120)
 - Paddle position (X): 156
 - Paddle Position: Right side of the screen
 - Ball Position: Far to the left of the paddle
 - Ball Direction: Moving down and to the left
 - Ball height relative to paddle: Mid-range
Based on where the ball is, where should I move the
    paddle so it is aligned with the ball? Say LEFT,
     RIGHT, or STAY
```

### Sample *Complex* Prompt

```
Instructions: We are playing a game of Atari
    Breakout. The task is to decide how to move the
    paddle based on the current and previous ball
    and paddle positions, as well as the speed of
    the paddle and ball. The goal is to prevent the
    ball from missing the paddle and achieve the
    highest score.

Inputs (provided for each frame):

   ball_curr (tuple): The current position of the
       ball (x, y).
   ball_prev (tuple): The previous position of the
       ball (x, y).
   paddle_curr (int): The current x-position of the
       paddle.
   paddle_prev (int): The previous x-position of the
        paddle.
   prev_action (str): The action taken in the
       previous frame.

Important Considerations:

   Paddle Width: The paddle is 16 pixels wide. The
       ball should ideally land within this range,
       not necessarily in the center of the paddle.
       We need to aim for a target range that spans
       +/-8 pixels from the paddle's center.

   Braking Distance: The paddle moves quickly, and
       the AI must prevent it from overshooting the
       predicted ball position. This is the braking
       distance. The paddle will stop moving if it's
        within a reasonable distance from the target
        area (defined by the predicted ball position
        and the 16-pixel target range).

   Ball Prediction:
      Use the ball's velocity (based on the
          difference between ball_curr and ball_prev
          ) to predict where the ball will be when
          it reaches y=180 (the y-position of the
          paddle).
      If the ball is moving towards the paddle,
          predict the x position at which the ball
          will land on the paddle.
      If the ball is moving away from the paddle,
          anticipate where the ball will move based
          on its trajectory.

   Paddle Movement Logic:
      Stay if the ball is within +/-8 pixels of the
          paddle's center.
      Move left if the predicted ball position is to
           the left of the paddle's target range.
      Move right if the predicted ball position is
          to the right of the paddle's target range.
      The paddle should avoid overshooting. If the
          ball is within the braking distance, the
          paddle should stop moving to avoid missing
           it.

   Screen coordinates:
      The left and right bounds of the screen are
          from x=50 to x=200 pixels
      The paddle is located at y=180
      As the ball moves down it gets closer to 180,
          as it moves up it gets closer to 0

Task:

   Based on the above inputs and considerations,
       decide whether the paddle should move LEFT,
       RIGHT, or STAY in order to prevent missing
       the ball and achieve the highest score. Do
       not provide a python function to evaluate
       this result. You must figure it out yourself
       and provide a final answer for this frame in
       your response.

Here are the inputs for the current frame:

   ball_curr: (120, 120)
   ball_prev: (124, 116)
   paddle_curr: 156
   paddle_prev: 156
   prev_action: STAY

Which action do I take? The last line of your
    response must have the following format:

Final answer: LEFT, RIGHT, or STAY
```