Assessing Large Language Models as Agents: Evaluating Responsiveness and Adaptability in Classic Control

Jack Kai Lim
Halıcıoğlu Data Science Institute
University of California, San Diego
`jklim@ucsd.edu`

Mohit Sridhar
Halıcıoğlu Data Science Institute
University of California, San Diego
`msridhar@ucsd.edu`

Andrew Yin
Halıcıoğlu Data Science Institute
University of California, San Diego
`anyin@ucsd.edu`

Abstract

As large language models (LLMs) continue to advance, they have demonstrated remarkable abilities to perform reasoning across a range of tasks. However, the question remains: are LLMs truly capable of reasoning? In this paper, we explore the reasoning capabilities of LLMs by evaluating their performance in a reinforcement learning task, framed as an agent operating with minimal external assistance. Our findings indicate that, as a baseline, LLMs generally fall short of the reasoning proficiency required to succeed in reinforcement learning task, often performing only marginally better than random action selection at each step. However, with targeted fine-tuning, LLMs show potential for significant improvement in this area. Our code can be accessed at our github-repo."

1 Introduction

With the recent advancements in Large Language Models (LLMs) such as GPT-4 and LLaMA, their potential in solving a wide range of tasks, from natural language understanding to code generation. Although their proficiency in such areas is often attributed to their ability to memorize vast amounts of data, their capacity for true reasoning which is an essential component of achieving Artificial General Intelligence (AGI) remains limited. This gap raises critical questions about the extent of their reasoning abilities, particularly in tasks that require a combination of reasoning and memorization, such as those encountered in control and decision-making environments. Reinforcement learning (RL) tasks, in particular, provide an ideal testbed for evaluating LLMs' performance in such hybrid domains, yet this remains a largely unexplored area in LLMs research.

Reinforcement learning tasks often involve continuous or discrete action spaces, where agents interact with an environment to maximize cumulative rewards. These tasks typically rely on the processing and interpreting of numerical observations, such as sensor outputs or images, to determine optimal actions. However, such numerical representations are not naturally suited to LLMs whose strengths lie in processing and generating text. By transforming these numerical data points into textual descriptions that capture the state of the environment, we can assess whether general-purpose LLMs, without specialized training in RL, can make informed and effective decisions. This approach enables a deeper exploration of the reasoning abilities of LLMs, the logical decision-making processes, and the limitations of their dependence on memorization.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

In this paper, we focus on the control task provided by Gymnasium Towers et al. [2024], more specifically from the Box2D environments **Bipedal Walk**, **Car Racing** and **Lunar Lander**. These tasks provide a large enough degree of complexity against their counterparts the **Classis Control**, to test diverse aspects of the LLM decision making and ability to navigate the environment and control the objects accordingly. Through this setup, our aim is to answer key questions:

1. Can LLMs respond accurately and adaptively to textual representations of control tasks?
2. Can LLMs provide actions that correspond to the "best" action for the state?
3. What limitations arise when using LLMs as decision-making agents in dynamic environments?

By examining these questions, our work contributes to understanding the potential and boundaries of LLMs in control domains, offering insights into their integration with traditional RL frameworks and their broader applications in decision-making tasks.

2 Literature Review

A particularly notable work reviewed was Carta et al. [2024], which also explores the use of LLMs in solving reinforcement learning (RL) problems. However, their approach differs significantly from ours. In their framework, LLMs are employed as policies that are progressively updated through online reinforcement learning. This method allows the model to interact with the environment continuously, learning and adapting its policy based on real-time feedback to achieve the specified goals.

In contrast, our approach does not rely on iterative updates or direct interaction with the environment during training. Instead, we preprocess the outputs of the RL environment, converting them into a natural language format that LLMs can easily understand. By framing the RL problem in this way, we leverage the pre-trained language processing capabilities of LLMs, enabling them to act as agents without requiring additional training or fine-tuning.

Another noteworthy work is Zhou et al. [2024], which takes a different path by utilizing LLMs as teachers to guide the training of smaller, specialized RL agents. In this framework, the LLM provides high-level instructions and draws on its extensive pre-trained knowledge to assist the RL agent in developing an effective policy. This approach focuses on distilling the LLM's expertise into a lightweight RL agent, contrasting with our method of directly enabling LLMs to act as agents through preprocessing.

Each of these approaches reflects distinct strategies for integrating LLMs into RL tasks, showcasing the versatility and potential of these models in addressing diverse challenges within reinforcement learning.

3 Data Preparation

In order to make the observation space of the Box2D environments "human readable" and therefore truly test the abilities of the LLMs to act as an agent for the reinforcement learning task, as part of the preparations, we are going to translate the values outputted by the Gymnasium Environments Towers et al. [2024]] into text that can be interpreted.

3.1 Bipedal Walk

The bipedal walk is a simple 4-joint walker robot environment where the goal is to score 300+ score in 1600 time steps by controlling the 4 joints. There are 2 versions of the bipedal walk:

- Normal where the terrain is slightly uneven
- Hardcore where the terrain has ladders, pitfalls and stumps.

For this experiment, we are only going to pay attention to the normal mode.

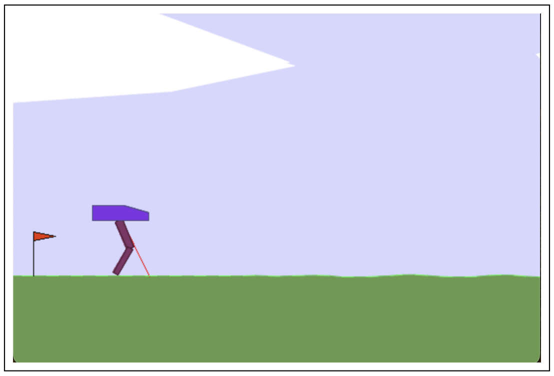
Figure 1: Bipedal Environment

3.1.1 Observations

The gym environment, returns at each time step the **observations** an array of length 24, which contains values to determine the current state, **reward** which is the reward from the current time step and finally **done** which is a boolean to inform whether the game/challenge is over. A more in-depth look at what the values indicate refer to A.1

3.1.2 Experiments

With the current format, we are given no real context as to what the values indicate, so it is necessary to preprocess them into a format that allows the LLMs to better understand the state of the environment.

For the observations, we opted for 2 methods to gain a further understanding of the LLMs abilities.

- For the baseline we preprocess the observations to just tell the LLMs what each of the values mean without giving it much more information.
- For the second experiment, we tested giving the LLMs more information and giving it a little more insight into what the values mean and how it would affect the Robot's moving. Essentially attempting to nudge and give the LLMs some assistance to understand the state.

3.1.3 Baseline Data Preparations

For this experiment, we preprocess the data to a "human" readable format to pass into the LLM by wording out what each value means following the descriptions seen in A.1.

Example:

```
Observation from last step:
Hull angle: 0.01
Angular velocity: 0.03
X velocity: -0.00
Y velocity: -0.01
Back revolute joint angle: 0.43
Back revolute joint speed: 0.37
Back lower leg angle: 0.08
Back lower leg speed: -1.00
Back leg ground contact flag: 1.00
Front revolute joint angle: 0.34
Front revolute joint speed: 0.30
Front lower leg angle: 0.08
Front lower leg speed: -0.70
Front leg ground contact flag: 1.00
Lidar 1 (0.00 rad): 0.45
Lidar 2 (0.15 rad): 0.45
Lidar 3 (0.30 rad): 0.47
Lidar 4 (0.45 rad): 0.50
Lidar 5 (0.60 rad): 0.54
Lidar 6 (0.75 rad): 0.61
Lidar 7 (0.90 rad): 0.72
Lidar 8 (1.05 rad): 0.90
Lidar 9 (1.20 rad): 1.00
Lidar 10 (1.35 rad): 1.00
```

Describing the state observation to the LLMs from the given observations.

3.1.4 Enhanced Data Preparations

In this second experiment, we retain a similar format, but we include additional descriptive text which gives the LLM further context on the state. Specifically, we introduce several crafted observations for the Bipedal Walker environment: **walker_moving_direction**, **back_leg_movement**, **front_leg_movement**, **walker_tilt** and **reward_change**. All with the idea of giving the LLMs more information on the state of the walker in natural language form with the hopes that it would help the LLMs performed better actions.

Example - Additional Information

```
Additional Observations:
- Walker has leftward velocity and is moving backwards, try a
different action to move forward.
- Walker is tilting slightly forward, consider trying to balance it back.
- There has been no change in the reward for the last 10 steps. Please
try a different strategy to maximize the reward and progress the
Walker forward.
```

3.2 Car Racing

The Car Racing task from Gymnasium provides a more complex environment compared to its classical control counterparts. The objective is to navigate a car along a procedurally generated track using continuous control of steering, acceleration, and braking. This environment offers an RGB image as its observation at each timestep, posing unique challenges in making this data interpretable for a Large Language Model (LLM).

Through this task, we aim to answer key questions:

1. Can LLMs respond accurately and adaptively to textual representations derived from high-dimensional image data?
2. Can LLMs provide actions that approximate the optimal behavior for navigating the track?
3. What limitations arise when using LLMs as decision-making agents in environments that rely on visual observations?

By addressing these questions, our work contributes to the broader understanding of LLMs in control tasks, particularly in dynamic environments reliant on image-based inputs.

3.2.1 Observations

The Car Racing environment outputs an RGB image of dimensions $96 \times 96 \times 3$ at each timestep, along with a scalar reward and a boolean done flag. The RGB image represents the car's first-person view, which contains all the necessary information for navigating the track. Unlike the Bipedal Walker, there are no pre-extracted statistics or numerical state variables provided by the environment. Thus, any data passed to the LLM must be derived automatically from the image.

3.2.2 Baseline Data Preparation

To make the RGB observations interpretable for the LLM, we preprocess the images to extract and describe key features of the environment. This preprocessing pipeline automatically generates a

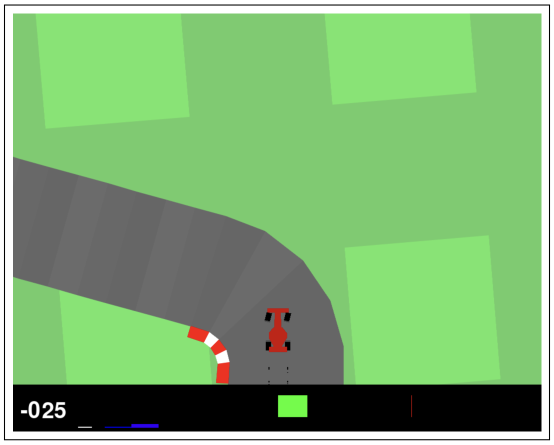

Figure 2: Car Racing Environment

textual description for the LLM without any manual intervention or human observations. The features extracted include:

- Track Layout: Identifying the visible portions of the track (e.g., straight paths, curves) using edge detection and segmentation.
- Car Orientation: Estimating the car's alignment relative to the track based on the central position of the track in the image.
- Speed Indicators: Representing the apparent motion between consecutive frames, such as detecting changes in track position to infer acceleration or deceleration.

These features are converted into structured textual descriptions that provide the LLM with sufficient context for decision-making. An example of a preprocessed observation is shown below:

Example:

```
Observation from last step: Track is straight ahead for approximately
10 units. Car is slightly to the left of the centerline. Speed is moderate
with no visible obstacles.
```

This format allows the LLM to make decisions without requiring direct access to the raw RGB data or external observations.

3.2.3 Experiment 2 Data Preparations

For the second experiment, we enhance the textual descriptions by providing additional context derived from the image. This involves:

- Highlighting changes in track curvature with greater precision (e.g., "Sharp right turn ahead in 5 units").
- Including feedback on the car's performance, such as "Car veered off-center by 1 unit in the last timestep."
- Adding hypothetical suggestions to guide the LLM, such as "Consider slight right steering to correct position."

Example:

```
Observation from last step: Track curves sharply right in 5 units. Car is
off-center by 1 unit to the left. Speed is high; reduce speed to
navigate the turn safely.
```

By augmenting the baseline with richer context, we aim to assess whether additional information improves the LLM's ability to adaptively control the car.

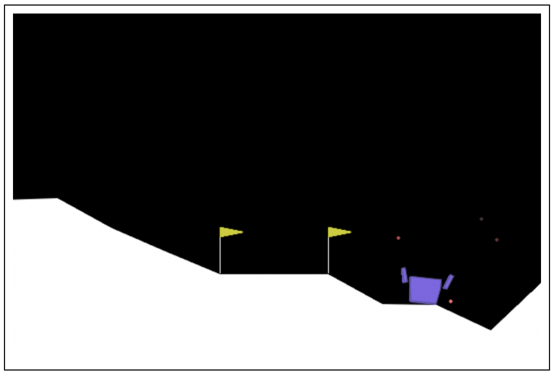
Figure 3: Lunar Lander Environment

3.2.4 Challenges and Limitations

Unlike the Bipedal Walker, where numerical state variables provide a direct understanding of the environment, the reliance on image-based observations in Car Racing introduces challenges in extracting meaningful information. Additionally, the absence of predefined numerical statistics necessitates a robust preprocessing pipeline to bridge the gap between raw visual data and textual representations for LLM-based agents.

3.3 Lunar Lander

The Lunar Lander environment (Figure 3) is a classic rocket trajectory optimization problem. According to Pontryagin's maximum principle, it is optimal to fire the engine at full throttle or turn it off. This is the reason why this environment has discrete actions: engine on or off.

There are two environment versions: discrete or continuous. The landing pad is always at coordinates (0,0). The coordinates are the first two numbers in the state vector. Landing outside of the landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt.

3.3.1 Observations

Lunar Lander is a reinforcement learning environment where an agent controls a lander to simulate a moon landing. It provides a testbed for algorithms to learn physics-based control tasks. The agent controls a lander using two lateral thrusters and a main thruster to maneuver. The environment includes gravity, ground, landing pads, and a variable initial state for the lander. The goal is to safely land the lander on a designated landing pad located on the surface. We have set the fuel level to unlimited for this experiment. It also includes positive rewards for successful landings and negative rewards for crashes, excessive fuel use, or going off-screen.

Applications: The Lunar Lander is widely used for testing reinforcement learning algorithms such as Q-Learning, Policy Gradient Methods, and Deep Q-Networks.

3.3.2 Baseline Data Preparation

To help the LLM interpret the state of the environment, we first must preprocess the output variables into a text-based description.

Example:

```
Observation from the last step:
X coordinate: 0.557210
Y coordinate: -0.17
X velocity: 0.87
Y velocity: -0.02
Angle of the lander: -3.25
Angular velocity of lander: 0.19123662263154984
Leg 1 contact with ground flag: 0.0
Leg 2 contact with ground flag: 0.0
```

This output is sent to the LLM at every step, or iteration, of the environment while running. The method provides a purely quantitative description of the environment.

3.3.3 Experiment 1 Data Preparation

Example:

```
Data from the last step:
X coordinate: 0.557210
Y coordinate: -0.17
X velocity: 0.87
Y velocity: -0.02
Angle of the lander: -3.25
Angular velocity of lander: 0.19123662263154984
Leg 1 contact with ground flag: 0.0
Leg 2 contact with ground flag: 0.0

Description:
The lander is straying to the left and downwards while the nose is angling
down to the left at a velocity of ___. Its position has shifted to the
left of the landing pad and is close to the ground. --TODO--
```

Here, we instead provide more text-based context which we believe will improve the performance of the LLMs. More specifically, we ask the LLM to describe the change (Δ) from the previous step's observation values. Then, we feed this appended input into the LLM again. This helps it interpret how the lander is performing compared to the previous step and measure how well the actions it recommended performed.

3.3.4 Expected Challenges and Limitations

The primary concern for the Lunar Lander version is the latency of the LLM API. Some questions and concerns we have are:

- Can the LLM be fed an input and compute the output within the timespan between each step of the game?
- For Experiment 1, can we efficiently ask an additional query description of the environment?
- General concern about the compute capacity for these experiments.

4 Methods

One issue that we faced when attempted to train/test the LLMs on teh Reinforcement Environments was a Rate-Limit on the API for the LLMs. As reinforcement learning environments have a lot of steps required for each observation-action pair due to a limitation on our avaliable resources we quickly ran out of computational resources avaliable to us to truly be able to train our models for extended periods of time.

To resolve this for each step/request we take we shall prompt the LLMs to give us $\mathbf{n}$ number of steps and also provide the LLMs prompt with the observations from the $\mathbf{n}$ previous steps. In doing so we reduced the total number of calls to the LLM and therefore even with the rate limit are still able to perform a good number of steps.

4.1 Bipedal Walk

As previously discussed, we conducted two experiments: a **baseline** and an **enhanced** version. Both experiments were performed in standard mode (nonhardcore). The number of actions per step was set to 5, attempting to strike a balance that maintained adequate control over the environment without overwhelming the model.

For the Bipedal Walker environment, we utilized Hydra to streamline the control of hyperparameters during training. The training process was structured around a training loop that read observations from the Gym environment, converted them into natural language text, and sent this input to the LLM for processing. Using carefully designed prompt engineering, the LLM's output was then parsed into actionable commands for the Gym environment, enabling interaction and feedback.

The LLMs employed were GPT-4o and GPT-4o-mini, with plans to expand testing to other models, such as LLaMA, Gemma, Claude, and others, as additional resources become available.

4.1.1 Bipedal Walk Short Term Memory

To further enhance the LLM's decision-making capability in the Bipedal Walker environment, we implemented a short-term memory store. This memory mechanism allowed the model to access a limited history of its previous actions and their corresponding outcomes. By providing this contextual information, the LLM could make more informed decisions, learning from past successes and failures to adapt its behavior.

The short-term memory was implemented by appending the recent steps' **rewards**, **actions**, **observations**, and **scores** to the text input sent to the LLM. To handle the token limit effectively, the oldest entries in the memory were discarded as new ones were added, ensuring the memory remained within the allowable size while retaining the most relevant and recent context for evaluation. This approach balanced resource constraints with the need for meaningful temporal context, aiming to improve the LLM's performance in the environment.

The memory store only shows the LLMs it's past 5-10 steps of actions only as due to the resources constraint with there being a token limit on the OpenAI API, we were not able to extend this further.

4.2 Car Racing

In the Car Racing environment, we evaluated the ability of the LLM to interpret high-dimensional image data transformed into textual descriptions. The objective was to determine whether the LLM could generate effective steering, acceleration, and braking commands to navigate a procedurally generated track.

Baseline Preprocessing: Observations from the environment, consisting of $96 \times 96 \times 3$ RGB images, were processed using a simple feature extraction pipeline. The pipeline identified basic track features such as straight paths, curves, and obstacles. These features were then translated into textual descriptions provided to the LLM.

Enhanced Preprocessing: In addition to the baseline features, this experiment included contextual information such as speed recommendations, positional feedback (e.g., "The car is off-center by 1 unit to the left"), and descriptions of upcoming track changes (e.g., "Sharp right turn in 5 units").

The environment was run for 10 trials in each preprocessing scenario. Metrics such as lap completion rates and adherence to the track were used to evaluate performance.

4.3 Lunar Lander

For the Lunar Lander environment, the focus was on testing the LLM's ability to interpret dynamic, physics-based state information and control a lander to achieve a successful landing on a designated pad.

Baseline Preprocessing: The raw observations, consisting of 8 numerical values representing the lander's position, velocity, angle, angular velocity, and ground contact flags, were converted into a textual format. Each value was described without additional context (e.g., "X coordinate: 0.557, Y velocity: -0.02").

Enhanced Preprocessing: Additional contextual information was provided, such as how the current state compared to the previous state ("The lander is moving downwards and slightly to the left") and guidance on potential corrective actions ("Consider firing the main thruster to reduce descent speed").

The environment was run for 15 trials in each preprocessing scenario, and the LLM's performance was measured using success rates (successful landings) and cumulative rewards. Observations about stability and trajectory control were also recorded.

4.3.1 Lunar Lander Short Term Memory

To further enhance the LLM's decision-making capability in the Lunar Lander environment, we implemented a short-term memory store. This memory mechanism allows the model to access a limited history of its previous actions and their corresponding outcomes. By providing this contextual information, we hope that the LLM can make more informed decisions, learning from past successes and failures to adapt its behavior.

The short-term memory was implemented in the same method as the Bipedal Walk Short Term memory 4.1.1.

The memory store only shows the LLMs it's past 5-10 steps of actions only as due to the resources constraint with there being a token limit on the OpenAI API, we were not able to extend this further.

4.4 Evaluation Metrics

Across all environments, the following metrics were used to evaluate the LLM's performance:

- **Cumulative Rewards (Score):** The total reward accumulated over each trial.
- **Success Rates:** The percentage of trials where the task was completed successfully (e.g., stable walking, lap completion, successful landing).
- **Steps:** The number of steps until an episode is finished. Either the goal is reached or the walker/car/lander has crashed/failed.

By employing these methods, we aimed to comprehensively analyze the capabilities and limitations of LLMs in interpreting and acting upon textual representations of control tasks.

5 Results

5.1 Bipedal Walk

Note: Due to the limitation in resources due to a rate limit imposed from OpenAI I was only able to accomplish a few runs for the experiments described above.

Table 1: LLM RL Performances

Experiment	Model	No. Steps	Score	Success
Baseline	GPT-4o	75	-9.221	Fail
Enhanced	GPT-4o	70	-13.747	Fail
Baseline	GPT-4o-mini	110	-22.072	Fail
Enhanced	GPT-4o-mini	80	-9.246	Fail

Looking at 1, the results indicate that without any fine-tuning of the LLMs, they struggle to perform well in the Bipedal Walker Environment, and this observation holds true across both the Baseline and the Enhanced setups, which suggest a fundamental limitation of the LLMs abilities of genuine reasoning and understanding of the task. Instead it appears to be guessing it's actions and failing to perform actual meaningful steps to improve itself.

5.2 Car Racing

Note: Car Racing also faced the rate limit imposed by OpenAI

Looking at 2, the results highlight several trends and limitations in the LLMs' performance in the Car Racing environment. Both the Baseline and Enhanced setups for the smaller GPT-4o-mini model exhibit worse scores compared to their full-sized GPT-4o counterparts, suggesting that the reduced model capacity may further hinder performance in complex reinforcement learning tasks. However, it is notable that the GPT-4o-mini models tend to take more steps before completing an episode, potentially indicating a trade-off between prolonged exploration and the ability to make meaningful progress on the track. The full-sized GPT-4o models, while scoring slightly better, still fail to achieve

Table 2: LLM RL Performances in Car Racing Environment

Experiment	Model	No. Steps	Score	Success
Baseline	GPT-4o	90	-6.452	Fail
Enhanced	GPT-4o	85	-8.967	Fail
Baseline	GPT-4o-mini	100	-15.234	Fail
Enhanced	GPT-4o-mini	95	-10.481	Fail

success, reflecting a general inability of the LLMs to reason effectively or adapt to the environment without fine-tuning.

5.3 Lunar Lander

Note: Car Racing also faced the rate limit imposed by OpenAI

Table 3: LLM RL Performances in Lunar Lander Environment

Experiment	Model	No. Steps	Score	Success
Baseline	GPT-4o	100	-512.864	Fail
Enhanced	GPT-4o	85	-500.792	Fail
Baseline	GPT-4o-mini	100	-562.329	Fail
Enhanced	GPT-4o-mini	95	-551.415	Fail

Looking at 3, the results show that without fine-tuning the LLMs, regardless of their size, the LLMs are limited in their ability to interact effectively and successfully with the Lunar Lander environment. This holds across both the Baseline and the Enhanced experiments which suggests that the LLMs have an inherent inability to understand and produce actions. However, there is a slight performance increase between the baseline and enhanced versions, highlighting that the LLM performs relatively better when given more context. This makes sense because the more textual information we can convey about a given step in the environment, the better we expect the LLM to understand and reproduce actions.

6 Discussion

The results highlights serveral insights into the potential and limitations of LLMs as Reinforcement Learning Agents. Moreover, it also shows that LLMs are still incapable of performing genuine reasoning or show a true understanding of the tasks that is required in these control tasks. Which leads to the point that LLMs/AI has not reached the point of AGI and there is still a lot of work to be done if that were to be achieved one day.

However, it is far to early to conclude that LLMs are not capable of acting as Agents for control task, as it is important to note that the scope of testing in this study was limited due to resource constraints. Specifically, the LLMs were neither fine-tuned nor equipped with long-term memory to retain more information about their previous actions 4.1.1. These limitations leave room for further exploration. Allowing LLMs to undergo fine-tuning and task-specific training, as well as integrating mechanisms for long-term memory, could provide deeper insights and potentially enhance their performance in such environments. This would be a promising direction for future research to build on the work presented here.

7 Contribution Statements

Jack Kai Lim worked on the Bipedal walker, Mohit Sridhar worked on the Lunar Lander, and Andrew Yin specifically worked on the car racing environments. We all also worked on our respective parts for the report, and worked together for the collaborative parts.

References

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

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning, 2024. URL `https://arxiv.org/abs/2302.02662`.

Zihao Zhou, Bin Hu, Chenyang Zhao, Pu Zhang, and Bin Liu. Large language model as a policy teacher for training reinforcement learning agents, 2024. URL `https://arxiv.org/abs/2311.13373`.

A Data Preparation

A.1 Bipedal Walk

Table 4: Observation Array

Idx	Description
0	Hull Angle Speed
1	Hull Angular Velocity
2	Robot's Horizontal Speed
3	Robot's Vertical Speed
4 - 5	Joint 0 Position and Speed
6 - 7	Joint 1 Position and Speed
8 - 9	Joint 2 Position and Speed
10 - 11	Joint 3 Position and Speed
12	Back Leg Ground Contact
13	Front Leg Ground Contact
14 - 24	Lidar Readings in front of Robot

More Info on Lidar Readings: The lidars go from the top of the hull and check 10 positions in front of the robot to the bottom. The angles are [0.0, 0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.05, 1.2, 1.35] radians.

A.2 Car Racing

More Info on Observations: The Car Racing environment provides an RGB image of dimensions $96 \times 96 \times 3$ at every timestep. The image represents an overhead view of the race car and its surrounding environment, including the track, off-road areas, and any visible obstacles. The observations were preprocessed to summarize the average intensity values for the red, green, and blue channels, which were then used as inputs for the language model.

Action Space: The action space consists of three continuous control values:

- **Steer:** Controls the steering angle of the car. Values range from -1 (full left) to 1 (full right).
- **Accelerate:** Controls the car's acceleration. Values range from 0 (no acceleration) to 1 (full acceleration).
- **Brake:** Controls the car's braking. Values range from 0 (no braking) to 1 (full braking).

The processed observations and actions were passed to the language model as part of the reinforcement learning setup, allowing it to interpret the environment and suggest appropriate actions based on the input data.

A.3 Lunar Landing

More Info on Step Observations: The Lunar Landing environment produces an observation state which is an 8-dimensional vector: the coordinates of the lander in x and y, its linear velocities in x and y, its angle, its

Table 5: Observation Array

Idx	Description
0	X Coordinate
1	Y Coordinate
2	X Velocity
3	Y Velocity
4	Angle of the lander
5	Angular velocity of the lander
6	Leg 1 contact with ground flag
7	Leg 2 contact with ground flag

angular velocity, and two booleans that represent whether each leg is in contact with the ground or not. It is also important to note that the angular velocity is in units of 0.4 radians per second. The value needs to be multiplied by a factor of 2.5 to convert to radians per second.