Investigating Iterative Hyperparameter Refinement by Large Language Models for Reinforcement Learning

David Xu (s3997308)

Supervisors:
Mike Preuss & Matthias Müller-Brockhausen

Bachelor Thesis– Informatica

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

Abstract

Hyperparameter selection is important in order to achieve good performance of reinforcement learning algorithms, yet finding good hyperparameter configurations often requires a lot of manual tuning or automated search methods. This thesis investigates whether Large Language Models (LLMs) can act as iterative hyperparameter optimization agents by suggesting new configurations based on feedback from previous evaluations. An LLM-guided framework was developed in which multiple open source LLMs iteratively generate hyperparameter configurations for Proximal Policy Optimization (PPO) on the LunarLander-v3 environment. The generated configuration were evaluated and their performance was used for subsequent prompts, allowing optimization through in context learning without model fine tuning. The results showed that all evaluated LLms were able to discover good hyperparameter configurations within a fixed optimization budget. However, noticeable difference were found in how the model explored the search space. Some models preferring to exploit successful bootstrap configurations, while others showed a more balanced exploration-exploitation strategy by refining newly discovered configurations while avoiding the poor performing configurations. These results suggests that the usefulness of LLM guided hyperparameter optimization depends on the optimization framework and the behaviour of the underlying language model. Although the proposed approach showed that LLMs can function as iterative optimization agents, the observed variability across models and repeated experiments suggests that further research is needed to improve the stability and generalizability.

Contents

1 Introduction

As of 2026, reinforcement learning (RL) has achieved success in a wide range of domains, such as robotics, game playing([8],[11]), autonomous systems and resource management. Modern algorithms such as Proximal Policy Optimization (PPO)[10], Soft Action Critic (SAC)[5] and Deep Q networks (DQN) [9] have shown to have the ability to learn complex behaviors by interacting with the environment. However, their performances of these algorithms rely on the choice of hyperparameters set up at the start. Parameters such as learning rate, discount factor, batch size and gamma can significantly affect the sample efficiency and final performance[2]. Furthermore, because of the stochastic nature of RL, the usefulness of a hyperparameter configuration may vary across runs and environment.

To address these challenges, researchers have developed numerous hyperparameter optimization techniques. However, many of those approaches require having large computation budgets or extensive manual configuration. At the same time, recent advances in Large Langue Models (LLM) have been shown to be valuable in practical domains[6]. Modern LLMs can perform reasoning, planning and iterative decision making, which suggests that they can be used as optimization agents ([13],[15]).

Recent studies have shown that LLMs can iteratively improve candidate solutions by analyzing previous results and generating refined proposals[14]. This iterative refinement paradigm resembles the process of how human experts analyze experimental outcomes and adjust future configuration accordingly. This raises a question of whether LLMs can be used to guide the optimization of reinforcement learning hyperparameters? Unlike traditional optimizers, LLMs can use both numerical and natural language, which could lead to more informed optimization decisions.

This thesis investigates whether LLMs can improve reinforcement learning hyperparameter through iterative feedback. An LLM guided optimization framework is developed in which the LLM receives information about previous runs and proposes new hyperparameter configurations. These configurations are afterwards evaluated using a reinforcement learning algorithm, and the resulting performance metrics are fed back to the model to guide future recommendations.

The primary research question addressed in this thesis is: **Can Large Language models improve the hyperparameters of reinforcement learning algorithms through iterative feedback?** To answer this question, this thesis evaluates the LLM guided hyperparameter optimization framework on reinforcement learning environments and compare it against simple reference configurations. Rather than solely focusing on the final performance, this thesis investigates the behaviour of the optimization process itself, including how the model explore and reuse configurations over time.

The main contribution of this thesis are:

- The development of an iterative LLM-based framework for reinforcement learning hyperparameter optimization.
- An empirical evaluation of LLM-guided optimization in the PPO and LunarLander environment.
- An analysis of the search behaviour of different LLMs, including exploration, exploitation and reuse of previously evaluated configurations.

- An assessment of the stability and limitations of LLMs as optimization agents in stochastic reinforcement learning environments.

1.1 Research Questions

The primary research question addressed in this thesis is: **RQ1: Can Large Language Models improve the hyperparameters of reinforcement learning algorithms through iterative feedback?** To provide a more detailed understanding of the behaviour and effectiveness of the proposed framework, this thesis further investigates:

- **RQ2:** How do different large language models differ in their hyperparameter search behaviour?
- **RQ3:** Can LLM-guided optimization produce high-performing configurations within a fixed computational budget?

2 Related Work

2.1 Automated Reinforcement Learning

The growing complexity of RL systems has motivated research into Automated Reinforcement Learning (AutoRL), which aims to reduce human involvement in the design and optimization of RL pipelines. The paper [1] provides a overview of AutoRl and highlights hyperparameter optimization as one the main problems. This survey shows the increasing importance of automation in reinforcement learning and establishes the broader context in which this thesis is situated.

2.2 Hyperparameter Optimization for Reinforcement Learning

Hyperparameter optimization plays a critical role in RL due to the stochastic nature of RL algorithms to parameter choices. Traditional approaches include grid search, random search, Bayesian optimization, evolutionary algorithms and population based methods. These approaches attempt to find hyperparameter configurations that maximized the agent's performance while minimizing the compute costs. A recent work proposed a optimization method called "HyperController"[4]. HyperController models hyperparameter optimization as a Linear Gaussian Dynamical System and uses Kalman filtering to estimate promising hyperparameter configurations during training. Experimental results shows that HyperController can achieve better performance while maintaining computational efficiency. However, this approach relies probability and does not use the reasoning capabilities of modern language models.

2.3 Large Language Models for Hyperparameter Optimization

Recent works has explored using LLMs as hyperparameter optimization agents. The paper [17] proposed an iterative framework in which an LLM receives descriptions of previous runs and gives hyperparameter configurations. Their results showed that LLM based optimization can have similar performance compared to traditional hyperparameter optimization techniques under limited search budgets. AgentHPO extends this by introducing an LLM agent that performs hyperparameter

optimization on machine learning tasks [7]. The system analyzes previous runs, reason about the performance trends, and proposes new configurations in an iterative optimization loop. Similarly, the paper[12] showed the use of an LLM agent for optimizing hyperparameters in communication systems. Their results showed that LLMs can help the optimization processes in specialized domains. Although these studies showed the capability of LLM-based hyperparameter optimization, their experiments focused on general machine learning or communication optimization problems rather than reinforcement learning. Another recent work evaluated whether LLM based optimization can outperform classical hyperparameter optimization algorithms [3]. Their results showed that pure LLM based optimization struggle to consistently outperform classical hyperparameter optimization algorithms, while hybrid approaches that combine both LLMs and traditional optimization methods can achieve stronger results. These results suggests that there is further investigation needed into the conditions under which LLM based optimization is effective.

2.4 Large Language Models in Reinforcement Learning

Researchers have also begun integrating LLMs directly into the RL systems. Prompted Policy Search (ProPS) being one of the most recent examples [18]. Rather than relying on RL optimizers, ProPS places an LLM within the policy optimization loop. The LLM receives reward feedback and task specific information, which then proposes policy updates. Experimental results showed that combining both numerical and natural language can improve learning performance across benchmark environments. While ProPS uses an LLM to optimize policy parameters directly, the focus of this thesis is different. Instead of replacing the policy update rule of the reinforcement learning algorithm, this thesis retains the existing RL algorithms and investigates whether an LLM can optimize their hyperparameters.

3 Methodology

This thesis investigates whether LLMs can improve the hyperparameters of a RL algorithm through iterative feedback. To evaluate this, an LLM guided hyperparameter optimization framework was developed in which a LLM suggests a new hyperparameter configurations based on the performance of previously evaluated configurations. The generated configuration is then used to train a reinforcement learning agent, in which the results of the performance is fed back to the LLM for further optimization. This optimization loop continues for a fixed number of iterations.

3.1 Experimental setup

All experiments were conducted locally using Ollama to run open source LLMs. The experiments were ran on a desktop computer equipped with an NVIDIA RTX4060 GPU (8Gb VRAM), which is capable to run the following models:

- Llama 3.1 8B
- Qwen 2.5 7B
- DeepSeek-R1 8B

- Mistral 7B
- DeepSeek-R1 14B

These models were chosen because they can represent a diverse set of open source LLMs with different architectures and reasoning capabilities while being small enough to be ran locally. This makes it possible to compare how the LLMs choose their hyperparameters. The RL algorithm used for this study was Proximal Policy Optimization (PPO). Experiments were performed on the LunarLander-v3 environment from Gymnasium. The LunarLander environment requires the agent to safely land a spacecraft while balancing the fuel efficiency and control stability.

3.2 Hyperparameter Search Space

The optimization process focused on four PPO hyperparameters:

- Learning rate: Controls how the parameters during training are updated. A high value may cause unstable learning, while a too low value can slow convergence.
- Discount factor (γ): Determines the importance of future rewards relative to immediate rewards. Higher values encourages long term planning, while lower values emphasize short term rewards.
- Batch size: Determines the number of training samples used for each gradient update and influences the stability of learning.
- n_steps: Defines the number of environment interactions collected before each policy update. This affects the amount of experience gathered for learning and the frequency of updates.

These hyperparameters were chosen because they were the most commonly used hyperparameters for PPO [16]. Parameters such as the Generalized Advantage Estimation (GAE) parameter and clipping range were kept fixed to reduce dimensionality of the hyperparameter search space

3.3 Bootstrap Phase

Figure 1 shows the workflow of the bootstrap phase. A bootstrap phase was performed before optimization began. Ten manually selected hyperparameter configurations were evaluated using PPO. For each bootstrap configuration, a model was trained and then evaluated over 100 episodes. The resulting mean reward and configuration was recorded and appended to the optimization history. The purpose of this phase is to provide the LLM with examples of both successful and unsuccessful configurations, which helps the model to identify regions of interests before the iterative optimization begins.

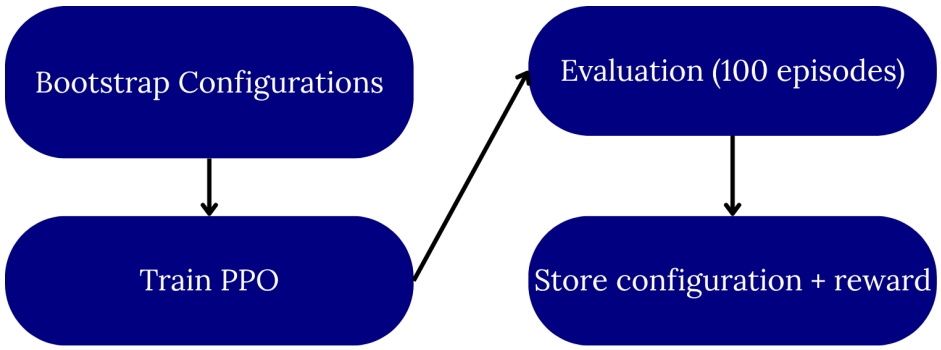


Figure 1: Bootstrap phase workflow used to initialize the optimization history. Ten manually selected PPO configurations are trained and evaluated over 100 episodes and their results are stored as the initial history for the LLM guided optimization loop.

3.4 LLM-Guided Optimization

The complete iterative optimization process is shown in figure 2. Unlike a random search, each hyperparameter configuration generated by the LLM depends on the optimization history accumulated so far. As a result the search process is iterative, which allows the LLM to refine previously successfully configurations while avoiding regions of search space that produce poor results. After the bootstrap phase, the iterative optimization process begins. At each iteration, the LLM receives:

- The environment name.
- The reinforcement learning algorithm being optimized.
- Results from the bootstrap phase.
- A complete history of previously evaluated configurations and their corresponding rewards.

The LLM weights remain fixed throughout all the experiments. Learning occurs solely through in context prompting, where the optimization history is provided as part of the prompt. No fine tuning or parameter updates are applied to the language models. The LLM was given the instruction to maximize the evaluation reward while balancing exploration and exploitation. Specifically, it is instructed to:

- Identify regions of interest of the hyperparameter space.
- Refine configurations that achieved high rewards.
- Avoid repeatedly exploring poor performing regions of interests.
- Occasionally explore new configurations to prevent premature convergence.

Using this information, the model generates a new hyperparameter configuration in JSON format. The generated configuration is then used to train the PPO agent. After training, the agent is evaluated over 100 episodes and the mean reward is recorded. The newly evaluated configuration and reward are appended to the optimization history and provided to the LLM in the next iteration.

This process was limited to 30 iterations for all methods to keep the total experimental runtime manageable and that the optimization budgets are equal for all methods.

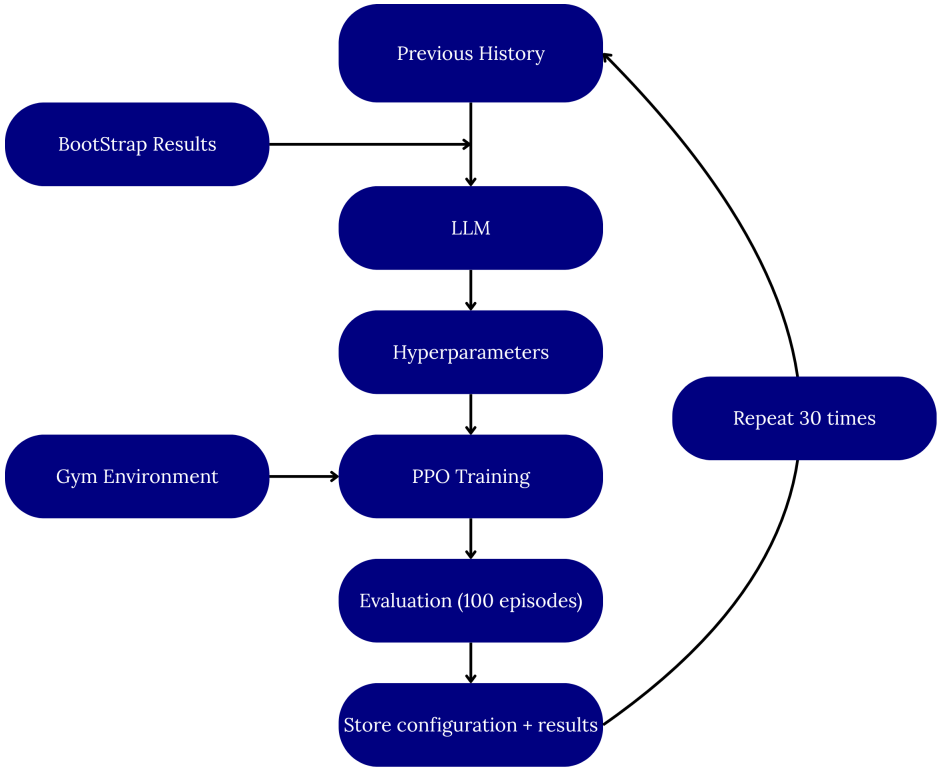


Figure 2: Overview of the LLM-guided hyperparameter optimization loop. At each iteration, the LLM generates a configuration based on the full optimization history, which is trained and evaluated using PPO on a gym environment. The resulting reward is appended to the history for subsequent iterations.

3.5 Baselines

To investigate the effectiveness of the proposed approach, two simple baseline methods were included.

3.5.1 Default PPO

The first baseline uses the default PPO hyperparameters provided by Stable-Baselines3. This baseline represents a commonly used configuration requiring no additional tuning. The default PPO hyperparameters are:

- Learning rate: 0.0003,
- Discount factor: 0.99
- Batch size: 64
- Number of n_steps: 2048

3.5.2 Random Search

The second baseline uses random search. At each iteration, hyperparameter values are sampled independently from predefined candidate sets:

- Learning rate: 1e-3, 3e-4, 1e-4, 3e-5
- Discount factor: 0.95, 0.98, 0.99, 0.995
- Batch size: 32, 64, 128, 256
- Number of n_steps: 128, 512, 2048, 4096

Random search was chosen as a baseline because it required no prior knowledge and does not use information from previous evaluations.

3.6 Evaluation

An experiment was performed to choose an appropriate number of evaluation episodes. Using the default hyperparameters of PPO, ten models were trained, each model was then evaluated using 10,20,50,100,200,500 and 1000 evaluation episodes. Based on this experiment, 100 evaluation episodes was selected for evaluating the PPO. The details are presented in the Results section 4.
Each generated configuration was evaluated using the same procedure. A PPO agent was trained using the generated hyperparameter configuration and then evaluated over 100 episodes. The performance metric used was the mean reward obtained after 100 evaluations. For each LLM, the optimization process was performed for 30 iterations. At each iteration, the LLM generated a new hyperparameter configuration based on the optimization history. The resulting mean reward obtained from the 100 evaluation episodes was recorded.
The optimization performance was visualized by plotting the mean evaluation reward obtained at each iteration. These learning curves were used to compare the convergence behaviour of different LLMs and baselines methods.

3.7 Experimental Objectives and Success Criteria

The objective of this study is to evaluate whether LLMs can be effective iterative optimization agents for reinforcement learning hyperparameters using only in context feedback. The experiments are designed to assess three aspects:

- Performance improvement: whether LLMs are able to generate configurations that improve over time.
- Adaptation: whether LLMs adjust their suggestions based on observed performance feedback.
- Search behavior: whether LLMs show a balance of exploration and exploitation rather than random sampling.

A successful outcome is not defined by solely outperforming the baseline methods. Instead, it is characterized by:

- Consistent generation of good configurations across independent runs.
- Evidence that suggestions change in response to observed performance.
- Non-random and structured evolution of hyperparameter choices over time.

This allows the performance to be evaluated and whether LLMs show meaningful optimization behavior.

4 Results

This sections presents the results of the LLM guided hyperparameter optimization experiments. The analysis focuses on the optimization performance over time and the behaviour of the models in terms of exploration and exploitation and the reuse of previously evaluated configurations. The goal is not to only assess whether high performing configurations are found, but also to understand how the different models chooses hyperparameters.

4.1 Preliminary Experiments

4.1.1 Evaluation Variance

In figure 3 the results for the preliminary experiment are shown to determine an appropriate number of evaluations. Although increasing the number of evaluation episodes should generally produce more stable reward estimates, the standard deviation does not change at all and the variance continue to fluctuate across different evaluation counts. However, when only considering the mean reward, the results obtained using 100 and 200 evaluation were similar to those obtained using 1000 evaluation episodes. This means that increasing the number of evaluation episodes beyond around 100 provides limited benefits despite the increased computational cost. Based on these observations, 100 evaluation episodes were selected for all experiments. This choice is a compromise between computational cost and measurement reliability, while reducing the evaluation time required for each hyperparameter configuration

4.2 Bootstrap Phase

In the figure 4 and table 1, the results of the bootstrap phase are shown. In the figure 4, each boxplot represents one of the ten manually selected hyperparameter configurations evaluated before the start of the optimization process. Based on the graph the most promising configurations are 1,3,5. These configurations represents potentially promising regions of the hyperparameter search space. The other configurations that produced significantly lower rewards, represents the less promising regions. The results of the bootstrap phase are then appended to the prompt. This initial information about the hyperparameter space should help the model to begin the optimization process with knowledge of previously successful and unsuccessful configurations rather than starting from no information. Further details the configurations are found in the table 1.

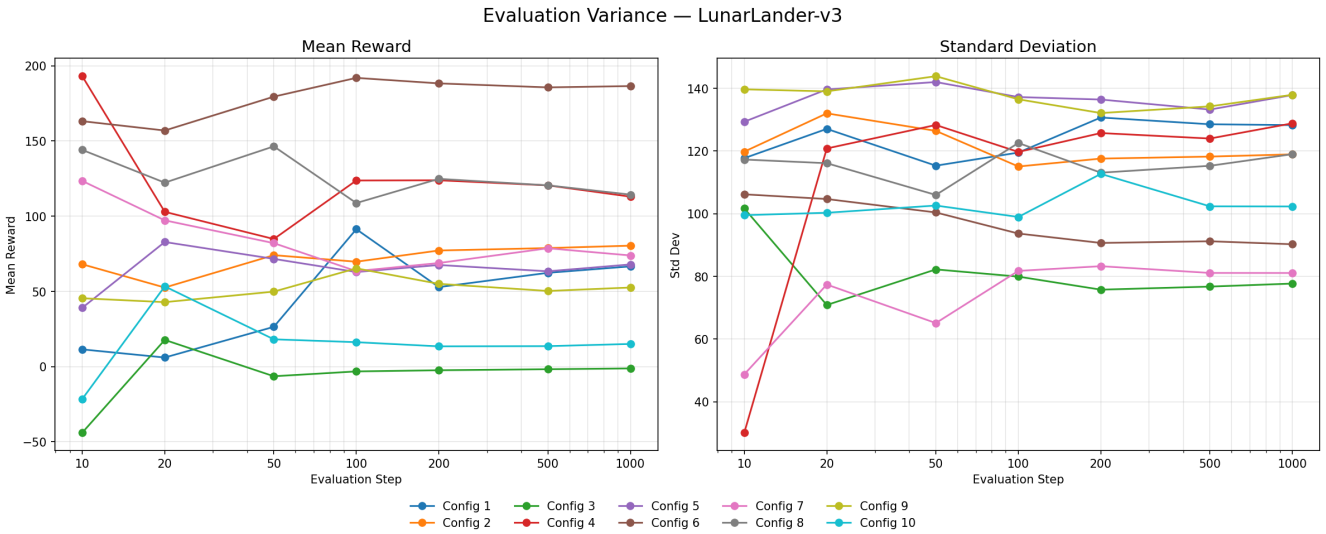


Figure 3: In the left figure, the x-axis represents the evaluation episodes and the y-axis represents the estimated mean reward. In the right figurem, the axis represents the evaluation episodes and the y-axis represents the standard deviation.

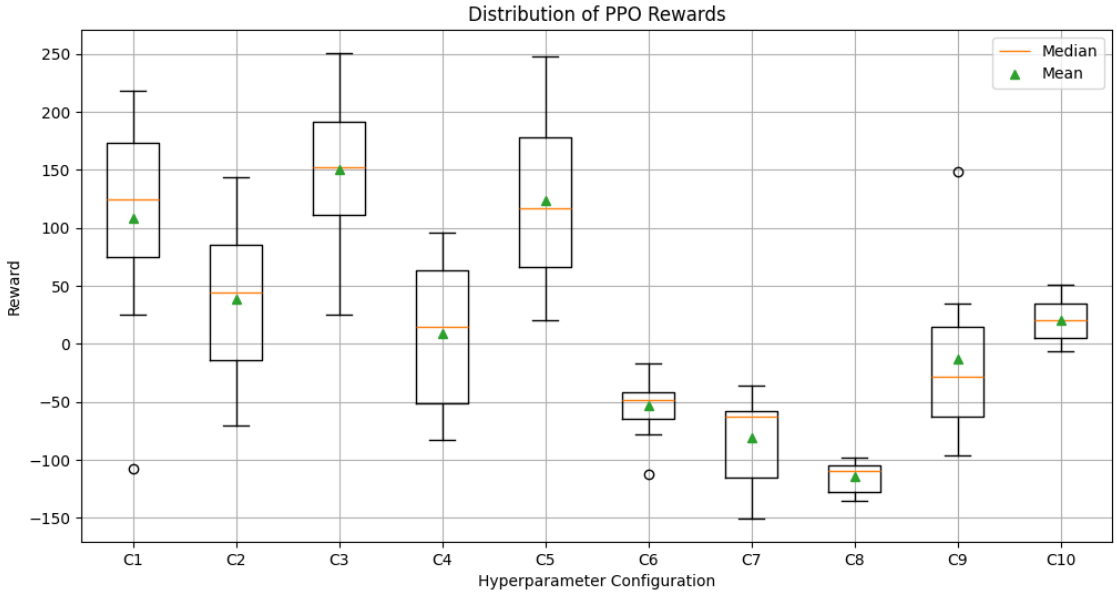


Figure 4: Evaluation rewards obtained by the ten manually selected bootstrap configurations. Each boxplot represents one configuration evaluated over 100 episodes. The orange line inside the boxplot represents the median and the green triangle represents the mean. The circles outside the boxplot represents the outliers. These results provide the initial optimization history for the LLM before the iterative process begins.

learning rate	gamma	batch size	n steps	mean reward	std reward
0.000300	0.990000	64	2048	108.339145	92.279931
0.000100	0.990000	128	1024	38.245202	67.617521
0.000500	0.995000	64	4096	149.911441	62.208809
0.000250	0.980000	256	2048	8.955670	64.031963
0.001000	0.990000	32	512	123.363788	69.636896
0.000030	0.999000	128	8192	-52.695665	26.851753
0.000700	0.970000	64	1024	-80.844463	37.871523
0.000050	0.995000	512	4096	-114.032035	13.672795
0.000200	0.980000	128	2048	-13.189102	67.031754
0.000100	0.999000	64	16384	20.456823	18.100513

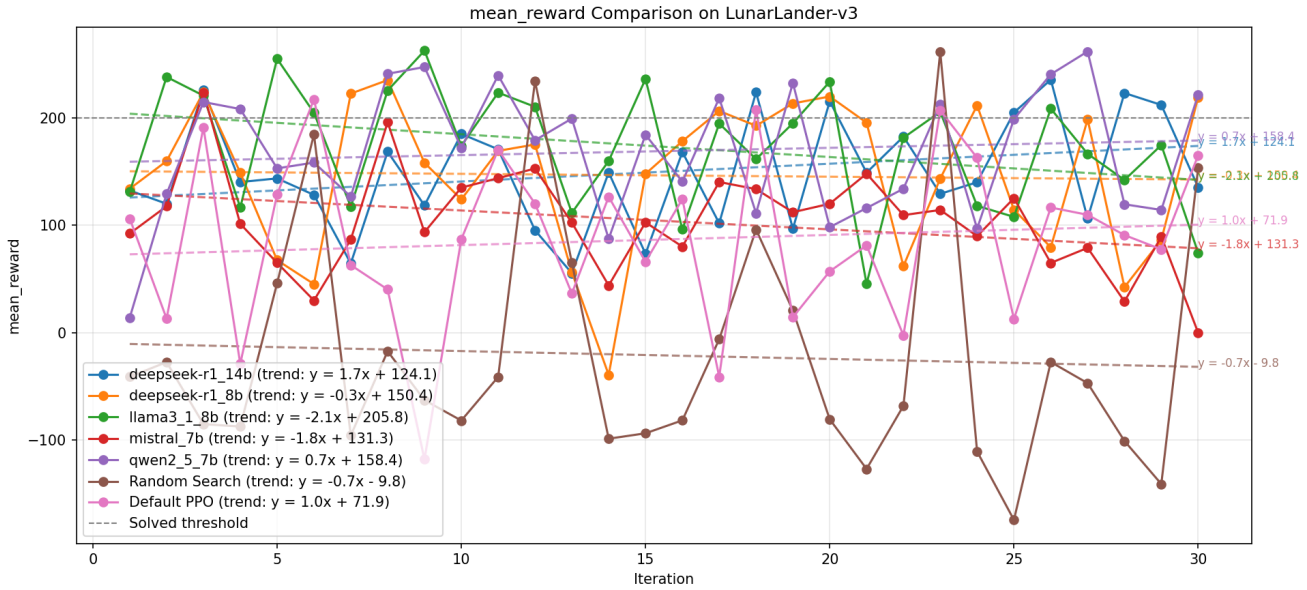
Table 1: Each row represents the hyperparameters used for each configuration with their corresponding mean rewards and standard deviations.

4.3 LLM guided Optimization Performance

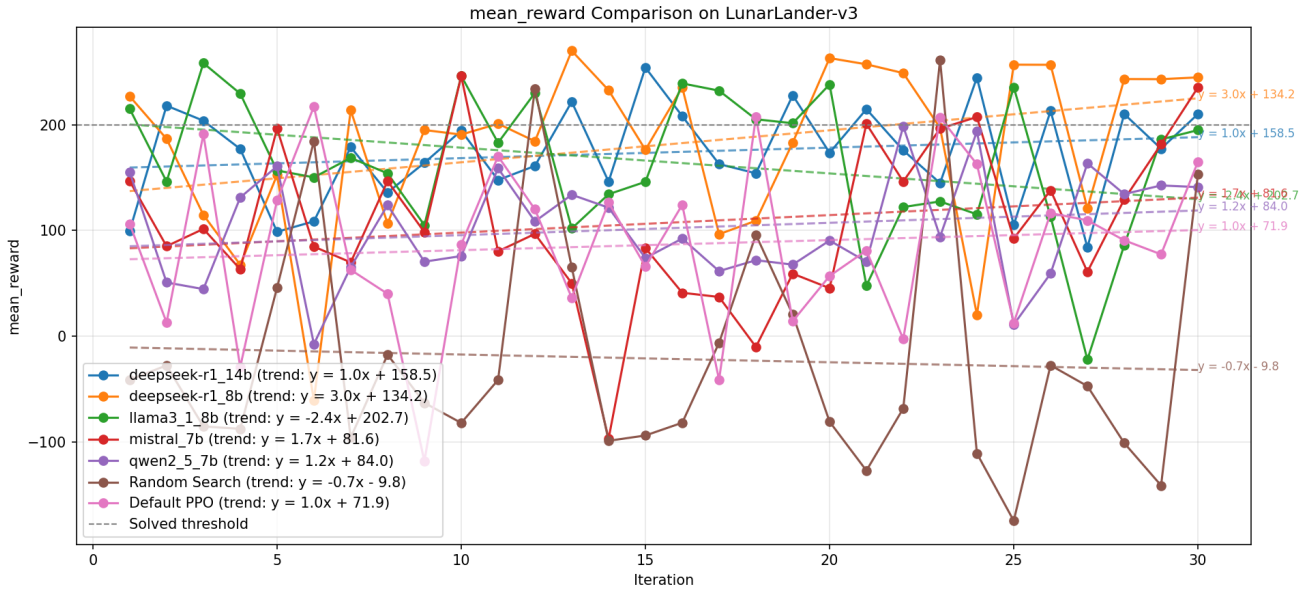
Due to both reinforcement learning training and LLM generated hyperparameter suggestions being stochastic, the complete optimization process was repeated twice for each model. This allows us to know whether observed performance trends are reproducible rather than the result of a single lucky run.

The figure 5 shows the optimization performance of the evaluated LLMs across 30 iterations. The figures includes trend lines to show the overall direction of the performance. Among the evaluated models, Deepseek-R1-14B was the only one with a consistent positive coefficient trend across both experimental runs. DeepSeek-R1-8B also shows stable behavior, with performance remaining close to the zero coefficient across both experiments. In contrast, the other models showed greater variability between experiments and did not consistently show positive improvement trends. While some individual iteration showed good configurations, for example the llama model in the first experiment 5a had the highest mean reward at one of the iterations but ultimately the overall optimization behaviour was less stable.

When compared to the baselines methods, all evaluated LLMs achieved higher rewards than both the default configurations and the random search baselines.



(a) First experiment



(b) Second experiment

Figure 5: Mean evaluation reward obtained at each optimization iteration for the five evaluated LLMs over two independent optimization runs. The dashed lines represents the linear trend lines summarizing the overall performance trend of each model.

4.4 Hyperparameter search behaviour

The complete optimization histories of all models and from both experiments are provided in appendix A and appendix B.

4.4.1 Configuration Diversity

Table 2 shows the number of unique hyperparameter configurations generated by each model across boths experiments.In contrast, Qwen and Llama generated fewer unique configurations. Both DeepSeek models produced a moderate number of unique configurations while maintaining relatively consistent behaviour across both experiments.

Model	Exp 1 unique params	Exp 2 unique params
deepseek-r1-14b	9	6
deepseek-r1-8b	9	6
llama3-1-8b	3	8
mistral-7b	24	20
qwen2-5-7b	3	1

Table 2: Number of unique hyperparameter configurations generated by each LLM during the 30 optimization iterations for both experimental runs. Higher values means that the LLM preferred exploring the hyperparameter search space and lower values means that the LLM preferred exploiting known hyperparameters

4.4.2 Reuse of Bootstrap Configurations

Table 3 shows how often each model rely on reusing configurations from the bootstrap phase. DeepSeek-R1-14B, Llama and Qwen (first experiment) keeps returning to configurations that had already been evaluated during the bootstrap phase. In contrast, DeepSeek-R1-8B and Mistral did not reuse any bootstrap configurations in either experiments.

Model	Experiment 1	Experiment 2
deepseek-r1-14b	22/30	24/30
deepseek-r1-8b	0/30	0/30
llama3-1-8b	27/30	21/30
mistral-7b	0/30	0/30
qwen2-5-7b	24/30	0/30

Table 3: Number of optimization iterations in which the LLM reused one of the bootstrap configurations. Larger values means that the LLM relied on the initial bootstrap phase and therefore a stronger exploitation strategy.

4.5 Summary of Observations

Across all experiments, the following observations are made:

- Models generate varying numbers of unique hyperparameter configurations.
- The frequency of reuse of bootstrap configurations differs across models and experiments.

- Performance trajectories vary between models and between independent runs.
- High-reward configurations occur at different points during optimization across models.

5 Discussion

This section interprets the observed results by linking the behaviour of the LLMs to its implications as an iterative hyperparameter search method. The discussion focuses on explaining the observed optimization dynamics rather than restating numerical results.

5.1 Optimization Performance

The optimization curves showed that good hyperparameter configurations can be found through iterative prompting. However, none of the models showed a clear and consistent improvement throughout the optimization process. Instead, performance generally fluctuated across iterations. This behaviour suggests that the optimization does not do gradual convergence, but rather repeatedly searching for promising regions of the hyperparameter space. The repeated experiments further showed that optimization behaviour is not entirely reproducible. Although similar performing configurations were found in both runs, they were identified at different iterations, meaning that there is variability in the optimization trajectory. This means that both the stochastic nature of reinforcement learning and the generative nature of LLMs influence the optimization trajectory. Additionally, finding a good configuration does not mean that the model will continue refining that region in subsequent iterations. These observations suggest that iterative prompting alone is sufficient to guide the search towards good solutions, but does not guarantee stable convergence. Instead, the optimization process should be viewed as an adaptive search procedure whose usefulness depends on how the model interprets and how it uses the optimization history.

5.2 Exploration And Exploitation

The experimental results showed that each model behave differently in how the LLMs chooses their hyperparameters. It is either by exploring new hyperparameter configurations or reusing previously evaluated configurations. Some models relied on reusing previously successful configurations, particularly configurations form the bootstrap phase. In these cases, the optimization process becomes reliant on early high performing configurations. This means that the optimization trajectory is heavily reliant on early context, which leads to a more conservative search behaviour over time.
Other models generated more unique configurations, meaning a broader exploration of the hyperparameter space. In these cases, previously evaluated configurations are less reused and new unique configurations are continuously made. This means that these models do not rely on bootstrap information and does a more exploratory search strategy.
A third behaviour pattern is also observed, where models initially explore new configurations but gradually shift towards exploiting configurations that give higher rewards. This indicates that the optimization process can dynamically transition between exploration and exploitation depending on the observed performance feedback.

5.3 Model-Specific Optimization Behaviour

The observed behavioural patterns differ across models, despite all models using the same prompt structure and optimization framework. DeepSeek-R1-14B showed a strong tendency to exploiting previously identified high-performing configurations, particularly configurations from the bootstrap phase. Newly generated configurations rarely influence the search direction, which means that the model places high weight on early successful examples when constructing subsequent proposals. In contrast, DeepSeek-R1-8B showed a more balanced optimization pattern. It does not reuse bootstrap configurations and instead does more exploration of the hyperparameter space. Successful configurations discovered during the optimization process are reused temporarily, after which the model continues exploration. This indicates a more dynamic interaction between exploration and exploitation. Llama3-1-8B and Qwen2.5-7B relied on previous configurations. In these cases, the optimization process keeps returning to a small subset of configurations, meaning that the search process is constrained and less diverse over time. Mistral-7B showed a more exploratory pattern, generating large unique configurations. However, these explored configurations are not always retained in subsequent iterations even if it produced a high reward, meaning the model does not prefer exploiting.

5.4 Influence of bootstrap Configurations

Across models, the bootstrap phase plays a significant role in how subsequent decisions are made. Configurations that are used during this phase are frequently reused, particularly in models that prefers exploitation. This suggests that early configurations act as anchors that influence subsequent decisions of the models. However, the degree of reliance on bootstrap configurations varies across models. Some models heavily reuse these initial configurations, while others gradually diverge from them as new promising configurations are found and some not even reuse the bootstrap configurations at all.

5.5 Stability and reproducibility across runs

Comparing independent experimental runs shows that the optimization trajectories are not fully stable. While the general behaviour of each model remain visible across runs, the exact sequence of generated configurations and their associated performance varies. This means that the opimization process is sensitive to stochastic variation in both model generation and feedback ordering. As a result, the same model can produce different search trajectories under identical experimental conditions.

5.6 System-level interpretation

Overall, the results suggest that LLM-guided hyperparameter optimization acts as a context dependent search process rather than a fixed optimization algorithm. The optimization behaviour comes from the interaction between the model, the prompt structure and the accumulated optimization history. The observed differences between models showed that different LLMs have distinct and relatively consistent behaviour tendencies. Some models have a stronger tendency to exploit more while others maintain a more exploratory search throughout the optimization process. These

differences suggest that the behaviour is not only determined by the evaluation feedback itself, but also by how each model processes and weights their information. This includes how strongly past configurations influence subsequent decisions and how the model shifts between exploration and exploitation. While these differences may be influenced by their underlying architectural and training differences between models, the experimental design does not isolate architecture as an independent variable. Therefore, the observed behaviour should be interpreted as an emergent property of the interaction between model design, prompting strategy, and feedback structure rather than a direct consequence of architecture alone. Overall, the results showed that LLM-based optimization systems are highly sensitive to both model choice and prompt context, and may not exhibit stable or predictable convergence behaviour in their current form.

6 Conclusions and Future Research

This thesis investigated whether LLMs can improve RL hyperparameters through iterative feedback. An optimization framework was developed in which the LLM receive results from previous proposed configurations, which then proposes a configuration. The results suggest that LLMs are capable of producing hyperparameter configurations that reaches promising regions. However, this capability is not stable across models or across repeated repeated runs. While successful configurations were found in all experiments, the optimization trajectories differ significantly in how these configurations are reached and whether they are reused or refined over time. A key observations is that different models behave in different ways. Some models rely on previous configurations, which results in a more exploitative search behaviour. Other models prefer exploring more, which generate a broad range of configurations. These differences means that the optimization behaviour rely on how each model process and prioritize previous evaluations. Overall, the results showed that LLM-based optimization behaves less like a traditional optimizer and more of a context driven decision process. The performance and stability of this process depend not only on the feedback signal but also on the model's internal architecture and its sensitivity to prior configurations. Several limitations affect the generalisability of these findings. The study is only done in a single RL algorithm and environment and the optimization budget is relatively small. Additionally, the reliance on prompt based learning makes the system sensitive to the prompt structure, which may influence behaviour of the model. The stochasticity of both RL training and LLM generation further introduces variability across runs. Despite these limitations, the study showed that LLMs can be used as functional, in context optimization agents capable of guiding hyperparameter search without gradient based updates or explicit optimization rules. However, their behaviour is not stable enough to replace traditional optimization methods, particularly in settings where reproducibility and convergence reliability are required.

6.1 Future Research

Future work could explore mechanisms to improve stability and consistency, such as hybrid optimization frameworks combining LLMs with classical search methods or evaluation across a wider range of environments, model scales and different reinforcement learning algorithm.

References

[1] Reza Refaei Afshar, Joaquin Vanschoren, Uzay Kaymak, Rui Zhang, Yaoxin Wu, Wen Song, and Yingqian Zhang. Automated reinforcement learning: An overview, 2026.

[2] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 9104–9149. PMLR, 23–29 Jul 2023.

[3] Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch, 2026.

[4] Jonathan Gornet, Yiannis Kantaros, and Bruno Sinopoli. Hypercontroller: A hyperparameter controller for fast and stable training of reinforcement learning neural networks, 2025.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[6] Walid Hariri. Unlocking the potential of chatgpt: A comprehensive exploration of its applications, advantages, limitations, and future directions in natural language processing, 2025.

[7] Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization, 2025.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas 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*, 518:529–33, 02 2015.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[11] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George 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*, 529:484–489, 01 2016.

[12] Wanzhe Wang, Jianqiu Peng, Menghao Hu, Weihuang Zhong, Tong Zhang, Shuai Wang, Yixin Zhang, Mingjie Shao, and Wanli Ni. Llm agent for hyper-parameter optimization, 2025.

[13] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[14] Eric Xue, Ke Chen, Zeyi Huang, Yuyang Ji, and Haohan Wang. Improve: Iterative model pipeline refinement and optimization leveraging llm experts, 2025.

[15] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024.

[16] Yunbo Yang and Francesco Goia. Hyperparameter optimization impact and tuning guidelines for decentralized multi-agent reinforcement learning in multi-energy neighborhoods. *Applied Energy*, 414:127833, 2026.

[17] Michael R. Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization, 2024.

[18] Yifan Zhou, Sachin Grover, Mohamed El Mistiri, Kamalesh Kalirathnam, Pratyush Kerhalkar, Swaroop Mishra, Neelesh Kumar, Sanket Gaurav, Oya Aran, and Heni Ben Amor. Prompted policy search: Reinforcement learning through linguistic and numerical reasoning in llms, 2025.

A Experiment 1 Results

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000500	0.995000	64	4096	132.241561
2	0.000500	0.995000	64	8192	119.926346
3	0.000500	0.995000	64	4096	225.901998
4	0.000500	0.995000	64	4096	140.148234
5	0.000250	0.995000	64	4096	143.515984
6	0.000500	0.997000	128	4096	128.165267
7	0.000500	0.995000	64	4096	64.385229
8	0.000500	0.995000	64	4096	168.368744
9	0.000500	0.995000	64	4096	118.380808
10	0.000500	0.995000	64	4096	184.883196
11	0.000500	0.995000	64	4096	170.651852
12	0.000300	0.997000	128	4096	95.262355
13	0.000500	0.995000	64	4096	55.299827
14	0.000500	0.995000	128	4096	149.077268
15	0.000500	0.995000	64	4096	73.342551
16	0.000500	0.995000	64	4096	168.104428
17	0.000600	0.997000	128	8192	101.733834
18	0.000500	0.995000	64	4096	223.788909
19	0.000500	0.995000	64	4096	96.740235
20	0.000500	0.995000	64	4096	215.279877
21	0.000500	0.995000	64	4096	148.865316
22	0.000500	0.995000	64	4096	182.411485
23	0.000600	0.995000	128	4096	129.157263
24	0.000500	0.995000	64	4096	139.962474
25	0.000500	0.995000	64	4096	204.799788
26	0.000500	0.995000	64	4096	235.402243
27	0.000500	0.995000	64	4096	106.218894
28	0.000500	0.995000	64	4096	222.958366
29	0.000500	0.995000	64	4096	211.890428
30	0.000600	0.995000	64	8192	135.318977

Table 4: Hyperparameter results for deepseek-r1 14b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000600	0.996000	128	2048	133.921637
2	0.000600	0.995000	64	4096	159.627456
3	0.000600	0.995000	128	4096	223.768211
4	0.000600	0.996000	128	4096	148.799779
5	0.000600	0.995000	128	4096	67.443614
6	0.000600	0.995000	128	4096	45.137458
7	0.000600	0.995000	128	4096	222.451507
8	0.000550	0.995000	128	4096	235.016115
9	0.000600	0.995000	128	4096	158.075948
10	0.000600	0.995000	128	4096	123.986538
11	0.000500	0.995000	128	4096	169.143321
12	0.000500	0.995000	128	4096	175.106684
13	0.000550	0.995000	128	4096	56.290050
14	0.000550	0.995000	128	4096	-39.319257
15	0.000550	0.995000	128	4096	147.766780
16	0.000600	0.995000	128	4096	178.148954
17	0.000550	0.995000	128	4096	206.284875
18	0.000650	0.995000	128	4096	192.515098
19	0.000600	0.995000	128	4096	213.297552
20	0.000600	0.995000	128	4096	219.762982
21	0.000550	0.995000	128	4096	195.378706
22	0.000550	0.995000	128	4096	61.913038
23	0.000560	0.995000	128	4096	143.406255
24	0.000575	0.995000	128	4096	211.400496
25	0.000550	0.995000	128	4096	114.344276
26	0.000550	0.995000	128	4096	79.462365
27	0.000600	0.995000	128	4096	198.465063
28	0.000575	0.995000	128	4096	42.371125
29	0.000600	0.995000	128	4096	86.322578
30	0.000650	0.995000	128	4096	218.954525

Table 5: Hyperparameter results for deepseek-r1 8b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000300	0.995000	64	4096	130.994089
2	0.000300	0.995000	64	4096	237.957665
3	0.000500	0.995000	64	4096	220.565720
4	0.000300	0.995000	64	4096	116.404451
5	0.000500	0.995000	64	4096	254.859470
6	0.000500	0.995000	64	4096	204.605039
7	0.000500	0.995000	64	4096	117.156737
8	0.000500	0.995000	64	4096	225.301879
9	0.000500	0.995000	64	4096	262.400140
10	0.000500	0.995000	64	4096	173.841290
11	0.000500	0.995000	64	4096	223.358799
12	0.000500	0.995000	64	4096	210.155507
13	0.000500	0.995000	64	4096	111.428004
14	0.000500	0.995000	64	4096	159.748491
15	0.000500	0.995000	64	4096	236.134056
16	0.000500	0.995000	64	4096	96.000333
17	0.000500	0.995000	64	4096	194.571761
18	0.000500	0.995000	64	4096	161.459766
19	0.000500	0.995000	64	4096	194.811026
20	0.000500	0.995000	64	4096	233.543849
21	0.000500	0.995000	64	4096	45.380353
22	0.000300	0.990000	64	2048	181.355078
23	0.000500	0.995000	64	4096	205.463791
24	0.000500	0.995000	64	4096	118.004777
25	0.000500	0.995000	64	4096	107.713908
26	0.000500	0.995000	64	4096	208.428637
27	0.000500	0.995000	64	4096	166.429844
28	0.000500	0.995000	64	4096	142.058900
29	0.000500	0.995000	64	4096	174.515288
30	0.000500	0.995000	64	4096	74.343173

Table 6: Hyperparameter results for llama3 1 8b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000300	0.990000	64	4096	92.202983
2	0.000400	0.992000	128	8192	117.622511
3	0.000500	0.990000	64	8192	223.515625
4	0.001000	0.980000	32	4096	101.252026
5	0.000600	0.990000	128	4096	64.950011
6	0.000700	0.990000	128	4096	29.956113
7	0.001000	0.992000	256	8192	86.644552
8	0.000600	0.990000	128	4096	195.724718
9	0.000550	0.990000	128	2048	93.517421
10	0.000500	0.980000	128	4096	134.976737
11	0.000600	0.990000	256	8192	143.724482
12	0.000600	0.990000	128	8192	152.716977
13	0.000700	0.990000	256	8192	102.411600
14	0.000600	0.990000	256	4096	43.907531
15	0.000600	0.990000	256	8192	102.582178
16	0.000600	0.992000	256	8192	80.039897
17	0.000600	0.985000	128	4096	140.171243
18	0.000580	0.990000	256	4096	133.542537
19	0.000600	0.990000	256	8192	112.060198
20	0.000620	0.990000	128	4096	119.929615
21	0.000700	0.990000	256	8192	147.935831
22	0.000650	0.990000	128	4096	109.346549
23	0.000650	0.990000	128	8192	114.156853
24	0.000650	0.990000	256	8192	89.804969
25	0.000650	0.985000	256	4096	125.107892
26	0.000650	0.985000	256	16384	64.870814
27	0.000650	0.985000	256	4096	78.883396
28	0.000620	0.990000	192	32768	28.984658
29	0.000650	0.985000	256	16384	89.314407
30	0.000630	0.989000	256	7000	-0.059243

Table 7: Hyperparameter results for mistral 7b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000300	0.990000	64	2048	13.925630
2	0.000500	0.995000	64	4096	129.075603
3	0.000500	0.995000	64	4096	214.619157
4	0.000500	0.995000	64	4096	208.078980
5	0.000500	0.995000	64	4096	152.846824
6	0.000500	0.995000	64	4096	158.267085
7	0.000500	0.995000	64	4096	126.575908
8	0.000500	0.995000	64	4096	240.879819
9	0.000500	0.995000	64	4096	247.195806
10	0.000500	0.995000	64	4096	171.887257
11	0.000500	0.995000	64	4096	239.457014
12	0.000500	0.995000	64	4096	178.997443
13	0.000500	0.995000	64	4096	199.212315
14	0.000500	0.995000	64	4096	87.281105
15	0.000500	0.995000	64	4096	184.205058
16	0.000500	0.995000	64	4096	140.449285
17	0.000500	0.995000	64	4096	218.266674
18	0.000500	0.995000	64	4096	111.133776
19	0.000500	0.995000	64	4096	232.390431
20	0.000500	0.995000	64	4096	98.148229
21	0.000500	0.995000	64	4096	115.942172
22	0.000500	0.995000	64	8192	133.885778
23	0.000500	0.995000	64	8192	212.413691
24	0.000500	0.995000	64	4096	96.749057
25	0.000500	0.995000	64	8192	198.694866
26	0.000500	0.995000	64	4096	240.396462
27	0.000500	0.995000	64	4096	261.657145
28	0.000500	0.995000	64	8192	119.424592
29	0.000500	0.995000	64	8192	114.186229
30	0.000500	0.995000	64	8192	221.373910

Table 8: Hyperparameter results for qwen2 5 7b on LunarLander-v3

B Experiment 2 Results

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000500	0.995000	128	8192	99.769485
2	0.000500	0.995000	64	4096	218.010074
3	0.000500	0.995000	64	4096	203.770928
4	0.000500	0.995000	64	4096	177.078083
5	0.000500	0.997000	64	4096	98.786715
6	0.000500	0.995000	64	4096	108.722945
7	0.000600	0.995000	128	4096	178.816481
8	0.000500	0.995000	64	8192	135.891956
9	0.000500	0.995000	64	4096	164.002606
10	0.000500	0.995000	64	4096	194.312147
11	0.001000	0.995000	128	8192	147.729746
12	0.000500	0.995000	64	4096	161.052714
13	0.000500	0.995000	64	4096	221.698761
14	0.000500	0.995000	64	8192	146.208699
15	0.000500	0.995000	64	4096	254.022264
16	0.000500	0.995000	64	4096	208.108327
17	0.000500	0.995000	64	4096	162.785389
18	0.000500	0.995000	64	4096	154.090397
19	0.000500	0.995000	64	4096	227.806902
20	0.000500	0.995000	64	4096	173.459556
21	0.000500	0.995000	64	4096	214.703198
22	0.000500	0.995000	64	4096	175.771842
23	0.000500	0.995000	64	4096	144.838894
24	0.000500	0.995000	64	4096	244.597976
25	0.000500	0.995000	64	4096	105.397741
26	0.000500	0.995000	64	4096	213.026286
27	0.000500	0.995000	64	4096	84.031179
28	0.000500	0.995000	64	4096	210.106039
29	0.000500	0.995000	64	4096	177.355243
30	0.000500	0.995000	64	4096	210.100671

Table 9: Hyperparameter results for deepseek-r1 14b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.001000	0.995000	64	2048	226.833454
2	0.001000	0.995000	64	2048	186.671171
3	0.001000	0.995000	64	2048	114.783495
4	0.000300	0.995000	256	8192	67.605918
5	0.000700	0.985000	128	3072	152.420123
6	0.000150	0.985000	256	4000	-60.457881
7	0.001000	0.995000	64	2048	213.691115
8	0.001000	0.995000	64	2048	106.466549
9	0.000300	0.995000	64	8192	194.995343
10	0.001000	0.995000	64	2048	190.655944
11	0.001000	0.995000	64	2048	200.951896
12	0.000300	0.995000	64	8192	184.403504
13	0.001000	0.995000	64	2048	269.884659
14	0.001000	0.995000	64	2048	232.622806
15	0.001000	0.995000	64	2048	176.280684
16	0.001000	0.995000	64	2048	235.345567
17	0.001000	0.995000	64	2048	96.545655
18	0.001000	0.995000	64	2048	109.244745
19	0.001000	0.995000	64	2048	183.127937
20	0.001000	0.995000	64	2048	262.954476
21	0.001000	0.995000	64	2048	257.088795
22	0.001000	0.995000	64	2048	248.808327
23	0.001000	0.995000	64	2048	196.975544
24	0.000700	0.985000	256	4000	20.430271
25	0.001000	0.995000	64	2048	256.733658
26	0.001000	0.995000	64	2048	256.624626
27	0.000300	0.995000	64	8192	120.994893
28	0.001000	0.995000	64	2048	243.160583
29	0.001000	0.995000	64	2048	243.038526
30	0.001000	0.995000	64	2048	244.728233

Table 10: Hyperparameter results for deepseek-r1 8b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000500	0.995000	64	4096	214.933323
2	0.000300	0.995000	64	4096	146.261129
3	0.000500	0.995000	64	4096	258.221934
4	0.000500	0.995000	64	4096	229.358727
5	0.000500	0.995000	64	4096	156.752191
6	0.000500	0.995000	64	4096	150.252668
7	0.000500	0.995000	64	4096	168.691321
8	0.000500	0.995000	64	4096	153.988973
9	0.000500	0.995000	64	4096	105.095388
10	0.000500	0.995000	64	4096	246.141010
11	0.000500	0.995000	64	4096	182.915041
12	0.000500	0.995000	64	4096	230.115730
13	0.000500	0.995000	64	4096	101.924772
14	0.000250	0.990000	128	2048	134.295790
15	0.000300	0.990000	128	2048	146.035036
16	0.000500	0.995000	64	4096	239.030468
17	0.000500	0.995000	64	4096	232.178685
18	0.000500	0.995000	64	4096	205.152179
19	0.000500	0.995000	64	4096	201.404840
20	0.000500	0.995000	64	4096	238.097086
21	0.000200	0.999000	256	8192	47.881070
22	0.000500	0.995000	64	4096	122.175085
23	0.000300	0.990000	64	2048	127.501245
24	0.000300	0.990000	64	2048	115.328770
25	0.000500	0.995000	64	4096	235.576780
26	0.000350	0.994500	192	3072	113.070687
27	0.000250	0.990000	256	2048	-21.993796
28	0.000350	0.994500	192	3072	86.243634
29	0.000350	0.994500	192	3072	186.139347
30	0.000350	0.994500	192	3072	194.905223

Table 11: Hyperparameter results for llama3 1 8b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000300	0.990000	128	4096	146.699465
2	0.000400	0.990000	128	4096	85.184389
3	0.000500	0.995000	256	4096	101.507237
4	0.000500	0.990000	128	4096	63.473769
5	0.000700	0.990000	128	4096	196.345704
6	0.000700	0.990000	256	16384	84.771447
7	0.000700	0.980000	256	4096	69.927455
8	0.000500	0.990000	128	4096	146.704016
9	0.000500	0.990000	256	4096	98.485960
10	0.000600	0.997000	128	8192	245.967631
11	0.000500	0.990000	128	4096	80.352706
12	0.000450	0.995000	128	3072	96.823395
13	0.000300	0.990000	128	4096	50.186938
14	0.001000	0.980000	256	2048	-96.728890
15	0.000500	0.990000	256	4096	83.402282
16	0.000500	0.990000	128	4096	41.192116
17	0.000600	0.980000	128	4096	37.199136
18	0.000700	0.980000	256	4096	-9.946714
19	0.000500	0.990000	128	4096	59.237927
20	0.000700	0.980000	128	4096	45.549576
21	0.000600	0.990000	256	4096	200.806215
22	0.001000	0.990000	128	4096	146.301671
23	0.000700	0.990000	64	4096	196.335188
24	0.000700	0.990000	128	4096	207.266245
25	0.000700	0.980000	128	4096	92.426419
26	0.000500	0.990000	128	4096	138.038655
27	0.000500	0.990000	128	2048	60.963519
28	0.001000	0.990000	256	4096	129.414552
29	0.000500	0.990000	128	8192	181.540404
30	0.000500	0.990000	64	5120	235.242740

Table 12: Hyperparameter results for mistral 7b on LunarLander-v3

iteration	learning rate	gamma	batch size	n steps	mean reward
1	0.000300	0.995000	128	4096	155.034226
2	0.000300	0.995000	128	4096	50.910353
3	0.000300	0.995000	128	4096	44.652484
4	0.000300	0.995000	128	4096	131.239134
5	0.000300	0.995000	128	4096	160.727811
6	0.000300	0.995000	128	4096	-7.338953
7	0.000300	0.995000	128	4096	65.493347
8	0.000300	0.995000	128	4096	124.005809
9	0.000300	0.995000	128	4096	70.567965
10	0.000300	0.995000	128	4096	75.886599
11	0.000300	0.995000	128	4096	158.816637
12	0.000300	0.995000	128	4096	108.656734
13	0.000300	0.995000	128	4096	133.811063
14	0.000300	0.995000	128	4096	121.633002
15	0.000300	0.995000	128	4096	73.995602
16	0.000300	0.995000	128	4096	92.419016
17	0.000300	0.995000	128	4096	61.652965
18	0.000300	0.995000	128	4096	71.966127
19	0.000300	0.995000	128	4096	67.775559
20	0.000300	0.995000	128	4096	90.551885
21	0.000300	0.995000	128	4096	70.528718
22	0.000300	0.995000	128	4096	198.530725
23	0.000300	0.995000	128	4096	94.154173
24	0.000300	0.995000	128	4096	193.719851
25	0.000300	0.995000	128	4096	11.165392
26	0.000300	0.995000	128	4096	59.343617
27	0.000300	0.995000	128	4096	163.471027
28	0.000300	0.995000	128	4096	134.453773
29	0.000300	0.995000	128	4096	142.606359
30	0.000300	0.995000	128	4096	141.035910

Table 13: Hyperparameter results for qwen2 5 7b on LunarLander-v3