

Fast or Slow: Adaptive Decision-Making in Reinforcement Learning with Pre-Trained LLMs

Katrine Linnea Nergård, Kai Olav Ellefsen, Jim Torresen

Abstract—This study investigates the use of Large Language Models (LLMs) in Reinforcement Learning (RL) to mimic human adaptive reasoning, and hence the ability to adjust cognitive effort to task complexity. Inspired by the dual-system theory of cognition, our approach employs a pre-trained LLM to guide an RL agent between “fast” and “slow” thinking modes. These modes are policies that correspond to either rapid, intuitive responses or more calculated, deliberate actions.

Our main contribution is demonstrating the LLM’s ability to dynamically select the appropriate mode for varying situations in a custom grid-world environment, without requiring additional training or fine-tuning. Moreover, we demonstrate that the LLM’s performance is comparable to that of a Q-learning agent trained for the same purpose. This research presents a proof-of-concept integration of LLMs in RL, offering preliminary insights into their potential for more complex decision-making.

I. INTRODUCTION

Just as humans apply different levels of mental effort depending on the task, computational systems can also benefit from adapting their resource allocation to task complexity. This adaptability holds significant potential across various fields. Kahneman’s dual-system theory [1] frames cognition as a balance between fast intuition (System 1) and slow deliberation (System 2), coordinating speed and accuracy in human decision-making.

Several previous studies have investigated the potential of training Reinforcement Learning (RL) agents to choose between “fast” and “slow” thinking modes [2], [3]. Such meta-agents learn to map their current environmental state to the most effective reasoning mode for that specific situation. We hypothesize that much of the information relevant to this mapping is already embedded within the “common sense” knowledge of modern Large Language Models (LLMs). Therefore, we explore the potential of bypassing meta-agent training by directly querying the LLM about the appropriate reasoning mode for the current situation. This concept has the benefit of reducing total training time and may offer a more generalizable, environment-independent meta-agent.

Our approach prompts the LLM to suggest a thinking mode based on the current state of a meta-agent within a grid world environment. The meta-agent, guided by this recommendation, adopts either a “fast-thinking” or “slow-thinking” policy. This strategy utilizes the LLM’s inherent, human-like capabilities to understand the complexity of described states, allowing the agent to select the appropriate

All authors are with the Department of Informatics, University of Oslo, Norway. Emails: katrner@uio.no, kaiolae@uio.no, jimtoer@uio.no

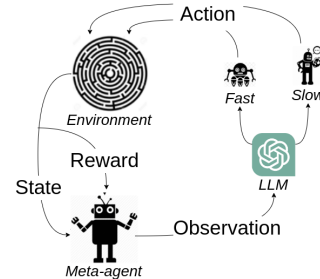


Fig. 1: Building on the work of Strand et al. [2], the meta-agent inquires the LLM before selecting the appropriate policy.

level of deliberation for each task without additional training or fine-tuning. A conceptual illustration is shown in Figure 1.

In summary, our main contributions are:

- We propose utilizing pre-trained LLMs as dynamic reasoning models to adaptively allocate cognitive effort based on situational demands.
- We demonstrate this approach in a simple, custom environment as an initial proof of concept.

II. BACKGROUND

A. Dual-system theory of cognition

Daniel Kahneman’s dual-process theory, outlined in his book *Thinking, Fast and Slow* [1], describes two systems of thought: System 1 and System 2. System 1 operates automatically and quickly, with little to no effort, and is responsible for intuitive and everyday decisions. System 2 is slower, conscious, and analytical, and is used for more complex situations that require attention and mental effort. Transitions from System 1 to System 2 are typically triggered by:

- **Novelty:** Encountering unfamiliar situations that System 1 cannot quickly process.
- **Error detection:** Noticing inconsistencies or mistakes prompts System 2 to reassess initial judgments.
- **High stakes:** Situations involving significant consequences or stress.
- **Cognitive demand:** Tasks requiring focus, like complex calculations or learning, engage System 2.
- **Planning:** Anticipating future outcomes or evaluating action sequences.
- **Conscious decision:** Choosing to focus or think carefully about a problem activates System 2 intentionally.

We aim to mimic this natural switching between modes by utilizing an LLM in a navigation environment. When

navigating a maze, some decisions can be made quickly and efficiently, while others demand more thoughtful deliberation, depending on the scene.

B. Q-learning

The conventional RL agents used in our experiments were trained using Q-learning, a reinforcement learning algorithm that estimates the expected cumulative reward (Q-value) for taking a given action in a particular state [4]. These estimates are updated iteratively using the Bellman equation:

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_a Q(s', a) - Q(s, a)] \quad (1)$$

Here, $Q(s, a)$ is the current estimate for state s and action a , α is the learning rate, r is the immediate reward, and $\max_a Q(s', a)$ is the best estimated value for the next state s' . The discount factor, γ , controls how much future rewards are valued. In our experiments, we varied γ to simulate fast (short-sighted) and slow (far-sighted) reasoning strategies.

III. RELATED WORK

A. Adaptive Reasoning

1) *Dual-System Thinking*: Research into dual-system approaches has led to the development of RL agents designed to mirror human-like reasoning. The Expert Iteration (EXIT) algorithm [5] combines a neural network (System 1) with tree search (System 2), showing improved performance in games like Hex over single-system methods.

Similarly, SOFAI [3] integrates a "fast" solver (S1) and a "slow" solver (S2), outperforming each solver alone in a grid environment. Control gradually shifts from S2 to S1 as the system gains experience, reflecting how humans tend to rely more on intuition as familiarity increases.

2) *Dynamic Discounting in Reinforcement Learning*: Tuning the discount factor offers a way to model different reasoning strategies in RL. Lower values of γ emphasize immediate rewards and align with more reactive behavior (System 1), while higher values place greater importance on future outcomes, reflecting more deliberate reasoning (System 2).

Recent studies have explored ways to adapt this parameter dynamically. Fedus et al. [6] showed that learning across multiple time horizons with varying discount rates can improve performance. Sharma et al. [7] proposed adjusting the discount factor based on state transitions, leading to faster convergence by prioritizing certain transitions.

Strand et al. [2] extended this concept by training a meta-agent to switch between pre-trained policies, each defined by a different discount rate and associated level of deliberation. In our work, we build on this idea but replace the trained meta-agent with a pre-trained LLM that selects between modes using textual observations. By doing so, we avoid the need for meta-agent training and enable a potentially more general, environment-independent decision-making mechanism.

B. Large Language Models Utilized in Reinforcement Learning

Navigating ambiguous and complex environments remains a central challenge for RL agents. While capable of solving difficult tasks, they often require extensive training and struggle to generalize to new environments. As highlighted by [8], conventional RL methods depend heavily on direct action-reward feedback, which restricts adaptability across tasks. Incorporating natural language understanding, particularly through pre-trained language models, offers a promising path to make RL more flexible and broadly applicable.

Recent studies have demonstrated how LLMs can inject abstract world knowledge into RL systems, enabling more flexible and context-aware behavior. For example, [9] proposed an algorithm in which a pre-trained LLM suggests achievable, context-relevant goals that guide exploration toward diverse and human-relevant objectives. Similarly, [10] used LLMs to identify tasks that are both learnable and novel, steering agents toward goals that are appropriately challenging and meaningful.

A three-component system (Planner-Actor-Reporter) was introduced in [4], where the Planner (an LLM) generates natural-language commands, which an RL-trained Actor executes. Feedback from the Reporter closes the loop. In a related approach, the LID framework [5] incorporates a pre-trained LLM into a policy network by encoding observations, goals, and history into a sequence of embeddings. This setup enables effective decision-making even in the absence of explicit language inputs or outputs.

GLAM (Grounded Language Models) [11] further explored LLM-based policies grounded in environment interactions. Here, the model evolves through online RL to meet language-specified goals. This work also introduced BabyAI-Text [12], a text-only version of BabyAI that returns linguistic observations instead of symbolic ones. We used this textual translation in our experiments.

Together, these efforts demonstrate that LLMs can enhance RL by improving generalization, reducing training, and enabling more human-aligned reasoning. Our work builds on these ideas by proposing a novel integration: using an LLM as a meta-agent to guide adaptive policy selection. This approach is complementary to prior work and can be combined with existing methods to further improve decision-making in complex environments.

IV. METHODS

In this concept study, we employ a pre-trained LLM to directly guide a meta-agent in selecting between two cognitive modes, "fast" and "slow," represented by two distinct pre-trained policies. Mode selection is based on the descriptive linguistic observations provided by the meta-agent at each timestep. While our work is limited to these two modes, the methodology can potentially include multiple modes in future research.

To demonstrate feasibility, we apply our method in a simple grid-world environment designed to test the meta-agent's ability to adapt its reasoning mode to situational demands.

To evaluate basic generalization across maze instances, we also test the approach in an extended version of the layout. We detail the method in the following paragraphs and direct the reader to Figure 2 for a schematic view.

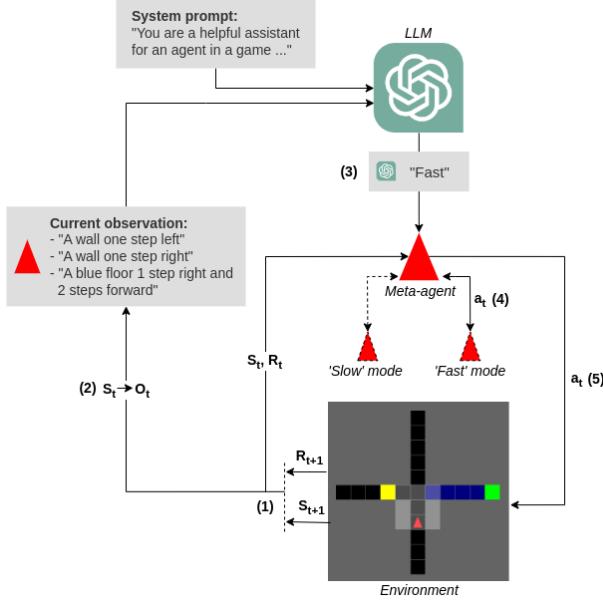


Fig. 2: Schematic view of the concept. (1) The meta-agent receives a partial and encoded observation from the environment (state, s_t). (2) This observation is translated to text and prompted to the LLM. (3) Instructed by the system prompt, the LLM suggests a mode of thinking suitable for the described situation. (4) Based on this response, the meta-agent gathers an action (given s_t) from the policy of the corresponding sub-agent. (5) The action is executed, a reward is returned, and the cycle continues.

A. Implementation details

We modified the RADAR algorithm [2] by substituting their meta-agent policy with a pre-trained LLM, see Algorithm 1. The remainder of the algorithm is consistent with the original paper. With our modification, we sought to address the same problem while removing the necessity of training the meta-agent.

First, two sub-agents were trained using Q-learning with different discount rates. The resulting two Q-functions, referred to as Q_{γ_i} in the algorithm, were then stored for later deployment by the meta-agent. The second part of the algorithm involved the meta-agent. For each step, the meta-agent queried the LLM on which mode (i) to activate and executed an action sampled from Q_{γ_i} based on the response. The state space for the meta-agent was the same as for the sub-agents, but the action space was reduced to the number of available sub-agents.

B. The test environment

We used the same maze environment as Strand et al. [2], referred to here as Maze 1, to evaluate our concept. This layout facilitates direct comparison to their Q-learning-based

Algorithm 1 LLM-guided meta-agent

Training the sub-agents:
Initialize the set Q_γ of Q-functions.
for γ_1, γ_2 , where each $\gamma_i \in [0, 1]$ **do**
 for episode = 1, 2, ... **do**
 for step = 1, 2, ... **do**
 Take action a using policy π , induced by Q_{γ_i} .
 Update Q_{γ_i} as in (1), using γ_i .
 end for
 end for
 $Q_\gamma \leftarrow Q_\gamma \cup Q_{\gamma_i}$.
end for

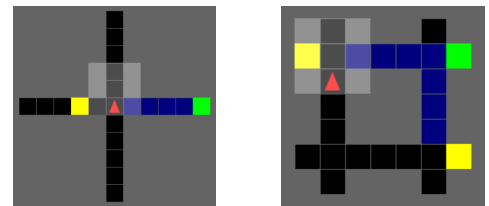
Applying the sub-agents:

Initialize LLM-guided meta-agent.

for episode = 1, 2, ... **do**
 Initialize system-prompt to LLM.
 for step = 1, 2, ... **do**
 Send observation prompt to LLM.
 Choose $Q_{\gamma_i} \in Q_\gamma$ according to response from LLM.
 Execute an action a , sampled from the policy induced by Q_{γ_i} .
 end for
end for

meta-agent and is specifically designed to test varying levels of deliberation. It is built using the MiniGrid platform [13], which offers modular and customizable environments for goal-oriented tasks.

To further evaluate the robustness of our approach, we introduce a second, more complex maze, referred to as Maze 2, within the same grid-based framework. This test examines whether the LLM/Q-meta agent can adapt to environmental variation rather than overfitting to a single, fixed layout. While the main analysis focuses on Maze 1, results from Maze 2 provide additional insight into the flexibility of the approach. See Figure 3a for Maze 1 and Figure 3b for Maze 2.



(a) Maze 1, the primary layout used for experiments.

(b) Maze 2, a second layout for evaluating adaptability.

Fig. 3: The red triangle represents the agent. The lighter tone around it represents its observable area. A larger reward is received when entering the green cell and a smaller one when entering the yellow one. The blue tiles offer an indirect indication of the proximity of the higher reward.

In both mazes, all non-gray tiles are accessible to the agent, which begins each episode from a random starting position. Yellow and green tiles mark goal states, reaching either ends the episode. The green goal yields a larger reward than the yellow one. Blue tiles provide no direct reward but indicate proximity to a high-reward state. Gray cells are walls, and the red arrow represents the agent.

The agent’s observable area is highlighted in a lighter tone and defines its state. State representations are orientation-invariant, treating identical observations from different angles as the same state. The agent can turn left, turn right, or move forward but cannot pass through walls.

Rewards depend on steps taken relative to the maximum allowed steps. A 0.0 reward is assigned for all actions except reaching a goal, where rewards are defined by Equations (2) and (3).

We extended MiniGrid by integrating textual descriptions for the agent’s partial observations, replacing its default symbolic representation. The text descriptions are based on BabyAI-Text [11]. See Section V for details.

C. Sub-agents representing reasoning modes

Two low-level agents were trained using Q-learning with identical hyperparameters and random seed, differing only in their discount factors: $\gamma = 0.1$ for the “fast mode” agent and $\gamma = 0.9$ for the “slow mode” agent. While these values serve as proxies for fast and slow reasoning, we emphasize that this is a simplified abstraction. Our focus is not on modeling the modes themselves, but on enabling adaptive switching between them. At each state, the agent could choose one of three actions: turn left, turn right, or move forward.

The resulting Q-tables for each sub-agent (Q_{γ_i}) contained 36×3 state-action pairs $Q(s_t, a_t)$, representing all combinations of the 36 unique states (as defined by the agent’s partial observation; see Figure 6) and the three possible actions. Both agents were trained with a decaying ϵ -greedy policy, same as in [2].

D. LLM-guided meta-agent

With the two reasoning modes (sub-agents) defined, an LLM was used to dynamically select between them based on the current observation. Our approach utilized a meta-agent that provided the LLM with a task description and a textual representation of the current environmental observation. The LLM’s linguistic responses were then converted into specific actions for the meta-agent. These actions entailed choosing the policy of one of the sub-agents rather than directly engaging with the environment. The chosen sub-agent’s policy subsequently dictated the interaction with the environment.

The LLM served as a decision-maker without any fine-tuning, using a few-shot prompting setup rather than explicit training to guide mode selection. Details on the prompting strategy are provided in Section V.

E. Q-learning trained meta-agent

To establish a baseline for comparing the performance of the LLM-guided meta-agent, we reproduced the meta-agent from [2]. This agent was trained using mostly the

same parameters as the sub-agents, with a discount rate of 0.9 and an identical state space. However, its action space was confined to selecting between the policies of the two sub-agents, matching the configuration for the LLM-guided meta-agent. Adjustments were made to the reward functions for both meta-agents (LLM and Q), and these are described in the following subsection.

F. Deliberation cost

The results after training the two sub-agents alone for 5000 episodes are shown in Figure 4.

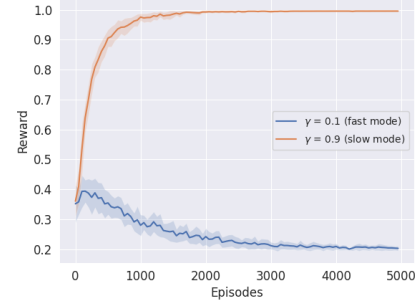


Fig. 4: Convergence of the policies of the two sub-agents after 5000 episodes. The policy of the ‘slow thinking’ agent converges towards the higher reward (green cell), and the ‘fast thinking’ agent towards the lower reward (yellow cell). Shaded areas indicate standard deviation over 25 runs.

As illustrated, the ‘fast mode’ agent typically opts for the lower, immediate reward (yellow), while the ‘slow mode’ agent targets the higher, more distant, reward (green). Their reward functions were defined as [2]:

$$R = \begin{cases} R_{\text{Green}} = 1.0 - 0.9 \frac{\text{step_count}}{\text{max_steps}} \\ R_{\text{Yellow}} = 0.2 - 0.18 \frac{\text{step_count}}{\text{max_steps}} \end{cases} \quad (2)$$

Where R_{Green} and R_{Yellow} reflect the reward received by the agent when entering the green or yellow goal state, respectively.

With the agents converging towards different goal states, we demonstrate varying results when depending solely on one mode of reasoning. However, based on this illustration, the ‘slow’ mode seems to be the obvious choice in any given scenario. In the real world, there would be a cost involved relative to the cognitive or computational effort, which is why the ‘slow’ mode can not always be activated.

This cost was modeled in both of the meta-agents by scaling their reward functions with a deliberation penalty factor (ζ), depending on the chosen reasoning modality. The penalized reward function was defined as:

$$R' = R - \gamma_i \times \zeta \quad (3)$$

The larger the discount rate of the chosen sub-agent, the larger the subtraction of the reward given to the meta-agent. A higher discount rate reflects a slower thinking mode. With this extended reward model, the meta-agents should be

encouraged to choose the lower level of deliberation in most scenarios, only applying the slow mode when the complexity of the task demands it. The deliberation penalty factor was set at 0.01, consistent with the value used in [2].

V. EXPERIMENTS

In our experiments with the LLM-guided meta-agent, we used OpenAI’s gpt-4o-mini API. The prompt template is given below.

- 1) A **system prompt**, containing a task description, initializing the message history for each episode:
You are assisting an agent navigating a grid-world environment. At each timestep, you receive a list of observations describing objects and their relative positions, including walls and color-coded goals and floor tiles. The agent can move forward, turn left, or turn right. It cannot walk into walls. The environment contains two goal squares: one with a lower reward and one with a higher reward. The objective is to maximize total reward. Based on the current observation, determine whether the agent should rely on intuition ("fast") or careful, deliberate reasoning ("slow"). An action will be taken based on your response, and observations will be updated at each timestep.
- 2) A **user prompt**, that is updated each timestep based on the partial observation made by the agent:
Current observation: <textual observation>. Should the agent think fast or slow? Respond with one word only; 'fast' or 'slow'.
- 3) **The models’ response** is then appended to the message history before a new user prompt is presented after an action is made.

The task description in the system prompt is based on the information provided about each environment in the MiniGrid documentation. The level of detail is what a human could expect before playing the game for the first time. Lack of grounding can limit functional competence and cause misalignment between knowledge of the LMM and the environment. Hence, some information about the environment is necessary for the LLM to perform as expected.

The <textual observation> given in the prompt template is a list of descriptions following the structure given in [11]: "You see a <object> <location>", where:

- The <object> is an adjective followed by a noun. The adjective is a color. The noun is a non-empty tile, e.g., a goal, floor, or wall.
- The <location> is given as the number of steps right, left, and forward from the agent to the object.

An example prompt following this structure is given in the 'current observation' box in Figure 2, with walls on each side of the agent and a blue floor one step right and two steps forward.

We used a few-shot prompting approach, providing the model with a set of example inputs and responses at the start of each episode via the system prompt. This setup improved

performance. The message history was maintained throughout each episode. Notably, small variations in observation translation and task descriptions led to differences in the model’s responses, highlighting the sensitivity to prompt design. More careful prompt refinement could further enhance consistency and performance. The LLM was configured with a temperature of 0.3, while all other parameters were kept at their default settings.

VI. RESULTS AND DISCUSSION

A. LLM-guided Meta-agent

A bar plot illustrating the results for the LLM-guided meta-agent (LLM-meta) in Maze 1, averaged over 50 runs with random starting positions, is shown in Figure 5. The plot displays the frequency with which the LLM selected a specific mode at each state. States with no bars were not visited, either because they were unreachable due to restricted starting positions (left of the yellow goal or on a goal tile), or because the agent simply did not encounter them during the runs. To assess whether the model’s mode selections were appropriate in different situations, we examined which mode was chosen for each unique partial observation (Figure 6).

The bar chart analysis and initial discussion focus on Maze 1, while a performance comparison involving both Maze 1 and Maze 2 is presented later in this section.

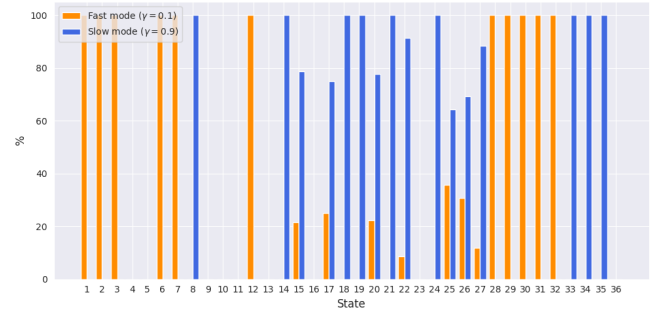


Fig. 5: Mode selection frequencies by state, averaged over 50 runs in Maze 1, using the LLM-guided meta-agent. Starting positions were randomized in each run.

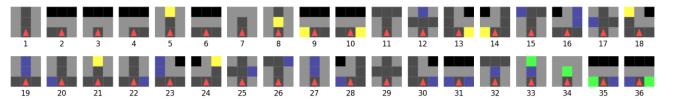


Fig. 6: Compilation of all unique partial observation states encountered by the agent in Maze 1.

As per the mode-switching criteria detailed in section II, the states in Maze 1 can be categorized based on their deliberation demands (see Figure 6 for reference):

- A. States with only walls: 1, 2, 3, 4, 6, 7, 11.
- B. States featuring a yellow goal: 5, 8, 9, 10, 13, 14, 18, 21, 24.
- C. States featuring a green goal: 33, 34, 35, 36.
- D. States with blue floor present and no goal: 12, 15, 16, 17, 19, 20, 22, 23, 25, 26, 27, 28, 30, 31, 32.

In category A, the agent’s partial view shows only walls and no intersections. This simplicity makes the fast mode, which relies on intuition, the optimal approach. As illustrated in Figure 5, LLM-meta consistently selects the fast mode for all visited states in this category.

In categories B and C, the presence of a goal, either yellow or green, dictates the strategy. Ideal mode selection depends on whether the agent knows in advance which goal yields a higher reward. Without this knowledge, it would be logical for the agent to ponder various aspects, like the significance of blue tiles, the global positions of each goal, and the count of steps taken so far. In our experiments, the meta-agent’s task description did not specify which goal offered the greatest reward, leading to the conclusion that the slow mode is preferable in these situations. The LLM-meta reliably makes this optimal choice in every state of these categories.

Category D, encompassing states with blue tiles, introduces a more intricate scenario. Key considerations include whether it is the agent’s first encounter with a blue tile and whether the agent possesses prior knowledge about their significance. The choice of mode is contingent upon these factors. The LLM-meta was not explicitly informed about the blue tiles in its task description, and, as expected, the mode selection varies in this category. It opts for the fast mode in five instances and the slow mode in eight.

B. Meta-agent trained with Q-learning

Figure 7 presents a bar plot of mode selections made by the Q-learning-trained meta-agent (Q-meta) in Maze 1, following the format introduced in [2]. As in Figure 5, states with no bars were not visited. While the LLM-meta plot displays selection frequency, the y-axis here instead represents Q-values, averaged over 25 training iterations.

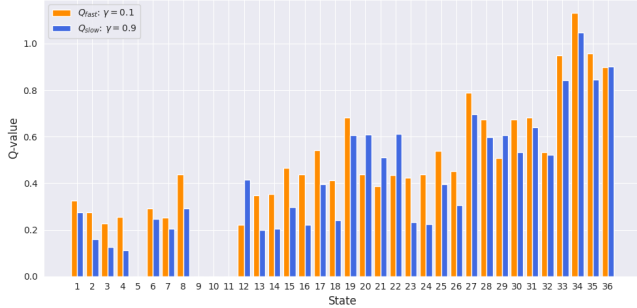


Fig. 7: Bar plot of average Q-values for Q-meta’s optimal policy across all unique states in Maze 1, factoring in a deliberation penalty. Results are averaged over 25 iterations.

In [2], special attention is drawn to states 19, 20, 21, and 22, as they represent critical situations where the agent is placed at the center of an intersection. The authors emphasize the need for planning ahead in these scenarios. As shown in Figure 7, Q-meta primarily engages its slow mode in states 20 through 22, while switching to fast mode in state 19. This suggests that Q-meta values deliberation at intersections but occasionally opts for quick decisions.

In contrast, LLM-meta consistently selects the slow mode in all four intersection states. These differences are summarized in Table I.

TABLE I: Mode selection in intersection states in Maze 1.

State	LLM-meta	Q-meta
19	Slow	Fast
20	Slow	Slow
21	Slow	Slow
22	Slow	Slow

C. Comparison of performance

To evaluate overall performance, we compared the total rewards acquired by LLM-meta and Q-meta. The box plot in Figure 8 shows the results across 50 runs with randomized starting positions in both Maze 1 and Maze 2.

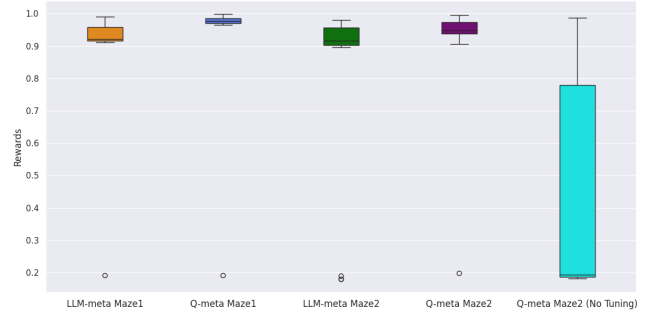


Fig. 8: Box plot of total rewards over 50 runs, comparing LLM-meta and Q-meta in Maze 1 and Maze 2. Also shown is Q-meta in Maze 2 using parameters from Maze 1 (no tuning), which resulted in reduced performance.

All API parameters for the LLM were kept constant across environments, including the system prompts and the textual representations of the state. In contrast, Q-meta required parameter tuning to reliably converge toward higher rewards.

When appropriately tuned, Q-meta achieved higher average rewards in both mazes. Nonetheless, LLM-meta remained competitive despite requiring no environment-specific training or parameter adjustment.

Interestingly, both agents occasionally chose the lower-reward (yellow) goal, indicating that neither performs optimally in all situations. This behavior appears to stem from a suboptimal action executed by the ‘slow’ sub-agent’s policy in a specific state.

The ability of LLM-meta to perform well across both environment instances without tuning highlights its potential as a general-purpose meta-agent. These results serve as an initial proof of concept, suggesting that pre-trained language models can support adaptive decision-making in reinforcement learning without additional training.

VII. CONCLUSION

This paper presents a novel approach to dynamic reasoning in reinforcement learning, leveraging a large language

model’s human-like capabilities to guide a meta-agent in selecting the appropriate level of deliberation for different situations. The LLM effectively functions as the meta-agent’s policy, enabling it to choose suitable reasoning modes at each timestep. Our experiments, conducted in environments designed to favor specific modes in particular states, demonstrate the feasibility of this approach.

We show that the LLM-guided meta-agent can navigate the environment effectively, activating higher levels of deliberation in the appropriate scenarios. It performs on par with a Q-learning meta-agent despite requiring no fine-tuning or task-specific training. Notably, it adapts to a more complex environment layout without parameter adjustment, whereas Q-meta required retuning to maintain performance. However, this remains a proof-of-concept in a constrained setting, and broader validation is necessary.

While traditional approaches require the meta-agent to learn decision-making strategies through extensive training, our results indicate that a pre-trained LLM can take on this role without environment-specific optimization. This offers a promising direction for future research. Nonetheless, computational demands may limit real-time applicability. Inference time could become a bottleneck in complex or time-sensitive tasks. One possible optimization is to restrict LLM queries to key intervals or decision points rather than querying at every timestep. Alternatively, the LLM could serve as an expert demonstrator, generating trajectories to train a lightweight, fast-switching meta-agent. This hybrid approach may combine the flexibility of LLM-driven reasoning with the efficiency required for real-time use.

Beyond computational considerations, future work could replace the current discount-based mode definition with a more principled distinction between model-free and model-based RL agents. This would allow the switching mechanism introduced here to operate over more cognitively grounded modes. Another direction is to investigate whether LLMs can arbitrate between modes using non-linguistic observations, such as visual input, extending applicability beyond text-derived environments. Finally, examining the effects of prompt engineering and scaling to more complex tasks with varied deliberation demands could offer deeper insights into the robustness of the approach.

REFERENCES

- [1] D. Kahneman, *Thinking, fast and slow*, 1st ed. Farrar, Straus and Giroux, 2011.
- [2] O. Strand, D. S. Reilstad, Z. Wu, B. C. Da Silva, J. Torresen, and K. O. Ellefsen, “RADAR: Reactive and Deliberative Adaptive Reasoning - Learning When to Think Fast and When to Think Slow,” in *2022 IEEE International Conference on Development and Learning (ICDL)*. London, United Kingdom: IEEE, Sep. 2022, pp. 184–189.
- [3] M. B. Ganapini, M. Campbell, F. Fabiano, L. Horesh, J. Lenchner, A. Loreggia, N. Mattei, T. Rahgooy, F. Rossi, B. Srivastava, and B. Venable, “Combining Fast and Slow Thinking for Human-like and Efficient Navigation in Constrained Environments,” Feb. 2022, arXiv:2201.07050 [cs].
- [4] R. S. Sutton and A. Barto, *Reinforcement learning: an introduction*, nachdruck ed., ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014.
- [5] T. Anthony, Z. Tian, and D. Barber, “Thinking fast and slow with deep learning and tree search,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 5366–5376.
- [6] W. Fedus, C. Gelada, Y. Bengio, M. G. Bellemare, and H. Larochelle, “Hyperbolic Discounting and Learning over Multiple Horizons,” Feb. 2019, arXiv:1902.06865 [cs, stat].
- [7] A. Sharma, R. Gupta, K. Lakshmanan, and A. Gupta, “Transition Based Discount Factor for Model Free Algorithms in Reinforcement Learning,” *Symmetry*, vol. 13, no. 7, p. 1197, Jul. 2021, number: 7 Publisher: Multidisciplinary Digital Publishing Institute.
- [8] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, “A Survey of Reinforcement Learning Informed by Natural Language,” Jun. 2019, arXiv:1906.03926 [cs, stat].
- [9] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas, “Guiding Pretraining in Reinforcement Learning with Large Language Models,” in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 8657–8677, iSSN: 2640-3498.
- [10] J. Zhang, J. Lehman, K. Stanley, and J. Clune, “OMNI: Open-endedness via Models of human Notions of Interestingness,” Oct. 2023, arXiv:2306.01711 [cs].
- [11] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, “Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning,” in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 3676–3713, iSSN: 2640-3498.
- [12] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, “BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning,” Dec. 2019.
- [13] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, “Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks,” Jun. 2023.