2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

PRLL: Policy Regularization and Reward Shaping Assisted by Large Language Models

Qianxia Zheng[1], Xiangfeng Luo[1,✉] and Tao Wang [1]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
{zhengqianxia, luoxf, wangt96}@shu.edu.cn

Abstract. Through continuous exploration and repeated trials, reinforcement learning (RL) enables agents to learn the optimal strategy, acquiring a certain level of behavioral intelligence. However, in complex and dynamically changing real-world environments, the state space and action spaces grow significantly larger. This implies that agents need to explore the environment more extensively to identify viable solutions. Unfortunately, such repetitive and inefficient exploration often leads to increased training time, higher costs, and greater risks. Several methods have emerged that use prior knowledge from large language models (LLMs) to assist RL training, but many of these approaches do not consider the issue of low sample efficiency. To address these challenges, we propose **P**olicy **R**egularization and reward shaping assisted by **L**arge **L**anguage models (PRLL). Firstly, PRLL calculates the similarity between LLMs-generated suggestions and the agent's actions, using this as a regularization term, to constrain the agent's exploration direction. Secondly, to efficiently align the agent's behavior with human preferences, PRLL employs LLMs to evaluate the alignment between the agent's actions and human values, translating this evaluation into an intrinsic reward signal. Experiments in both discrete and continuous action spaces demonstrate that PRLL outperforms most baseline methods while requiring fewer training time steps.

Keywords: Deep reinforcement learning, Large Language Models, Policy Regularization, Reward Shaping.

1 Introduction

Reinforcement learning (RL) enables agents to learn appropriate strategies in a completely unfamiliar and dynamically changing environment through continuous exploration and multiple trials, thereby acquiring a certain degree of intelligent decision-making ability [15, 19, 33]. Consequently, RL has become an important method for the practical application of artificial intelligence technologies [16, 18, 28], with wide applications in industrial robot control [29], intelligent medical diagnosis [3, 9], Multiplayer Online Battle Arena (MOBA) games [2], military operations [32], and more. However, the necessity of learning through continuous exploration of the environment implies that the agent needs to interact continuously with the environment until training is complete, inevitably leading to some repetitive exploration. Especially in complex

and dynamic training environments, both the action space and the solution space become larger, requiring the agent to explore the environment more extensively to obtain the solution space.

This can easily lead to rapid increases in training time, costs, and levels of danger [12, 25]. For example, in military applications, the training efficiency and decision-making accuracy of unmanned combat agents are important for gaining battlefield advantages, particularly when the battlefield environment undergoes significant changes due to natural disasters or human bombings. If the agents cannot quickly adapt to new battlefield environments and changes in enemy situations, learn new strategies, and make decisions promptly, it could delay decision-making time, affecting tactical execution and overall combat effectiveness. Therefore, addressing the issues of large solution spaces and low learning efficiency in traditional deep reinforcement learning is crucial to ensure that agents can execute new tasks quickly and accurately.

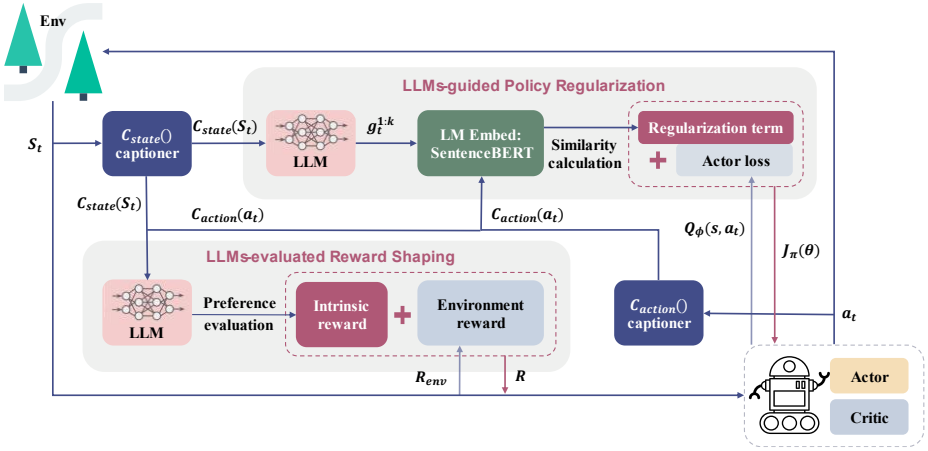

Fig. 1. The architecture of policy regularization and reward shaping assisted by large language models (PRLL). On one hand, we utilize large language models (LLMs) to provide action suggestions for the agent under the current observation, and calculating the similarity between these suggestions and the actions currently output by the agent to form a regularization term. On the other hand, we use LLMs to evaluate whether the actions conform to given human preferences, and provides corresponding intrinsic rewards to the agent.

Currently, methods for addressing the challenges of large solution spaces and low learning efficiency in RL can be roughly categorized into three groups. The first category includes offline RL and imitation learning, but acquiring a large amount of expert demonstration data is difficult for complex and costly tasks [14, 26]. Moreover, the state distribution in the data may have errors compared to the real environment. The second category involves model-based RL, but constructing models for complex and dynamically changing environments is challenging, and these algorithms lack transferability between different environments [22]. The third category is intrinsic motivation RL method, but actions that lead to new states in the environment do not necessarily mean they are solutions within the solution space [33].

Additionally, some researchers have used large language models (LLMs) to address problems in RL [4, 30]. ELLM [6] leverages LLMs for pre-training, providing agents with goal suggestions in a task-agnostic manner. However, since it is not tailored to specific tasks, some tasks may never be addressed during pre-training. LCA [5] uses Vision Language Models (VLMs) to map observations into text, which is then processed by LLMs to generate subgoals. But it is limited to the task of stacking three blocks in robotics. These methods provide agents with embeddings of subgoal texts, requiring LLMs to be called at every time step. The frequent reliance on LLMs increases time costs and fails to fully address the issue of low sample efficiency. In our algorithm, we call LLMs only at critical time steps relevant to task completion. For instance, in the unmanned ground vehicle (UGV) street-keeping scenario, LLMs are invoked only when the UGV enters the area near a traffic light intersection.

To tackle this challenge, we propose the policy regularization and reward shaping assisted by large language models (PRLL). In PRLL, there are two captioners, one of which translates the current observations of the agent into descriptive text, while the other translates the actions currently taken by the agent into descriptive text. Both pieces of text are then input into the following two modules. On one hand, we utilize a policy regularization method guided by LLMs. This method initially provides action suggestions for the agent under the current observation based on the LLMs. Subsequently, it calculates the similarity between the suggestions and the actions currently output by the agent, forming a parameterized regularization term. This term guides the policy learning towards the solution space capable of completing tasks. On the other hand, we employ a reward reshaping mechanism based on LLMs evaluation. We first use the LLMs to evaluate whether the actions align with given human preferences. Then, based on the degree of importance assigned to these human preferences, it provides the agent with a certain proportion of intrinsic rewards, assisting the policy in learning the solution space with human preferences. By combining these two parts, we form the PRLL to address the challenges of large solution spaces and low learning efficiency. The architecture is shown in Fig. 1.

We conducted experiments in the discrete action space of Crafter and the continuous action space of the Unity3D UGV street-keeping scenario to validate the effectiveness of PRLL. Additionally, we carried out ablation experiments to evaluate the influence of policy regularization and reward reshaping on the PRLL algorithm. The results indicate that our algorithm consistently outperforms most baseline methods across various experimental environments while requiring fewer training time steps, highlighting a greater learning efficiency for our approach.

The main contributions of this work are summarized as follows:

- We propose a policy regularization method guided by a LLMs to help the policy get the solution space necessary for completing tasks.
- We propose a reward reshaping mechanism based on LLMs evaluation to assist the policy in learning the solution space with abstract human preferences.
- Our algorithm outperforms most baseline methods in both discrete and continuous action space environments, demonstrating greater sample efficiency for the agent by requiring significantly fewer training time steps.

2 Related Work

The current methods for addressing the challenges of large solution spaces and low learning efficiency in RL can be roughly categorized into three groups. The first involves providing demonstration data, the second entails building environment models, and the third involves designing intrinsic rewards. Offline RL and imitation learning both involve learning directly from expert demonstration data [14, 26], utilizing the prior knowledge in the data to help the agent narrow down the solution space. However, both methods require the collection of expert demonstration data in advance, which can be challenging for complex or costly tasks. Moreover, the state distribution in expert demonstration data may have errors compared to the real environment, leading to the agent learning incorrect action values. Model-based RL improves the efficiency of learning policies by constructing environment models, including state transition models and reward models, to predict the state and reward of the environment given the agent's current state and action. However, constructing models for complex and dynamically changing environments can be difficult. Additionally, this method lacks transferability between different environments and may have limited generalization capabilities [22].

Intrinsically motivated RL methods primarily provide rewards based on the novelty and predictability of the current state of the agent [33]. This encourages the agent to explore parts of the environment that were previously unknown, thereby increasing the coverage of exploration and speeding up policy learning. However, taking actions that lead to new states in the environment does not necessarily mean discovering solutions within the solution space. For instance, in the scenario of an UGV navigating through streets, the frequent swaying of tree branches due to strong winds may provide prolonged novelty in the observations for the vehicle agent, but it may not significantly affect the agent's driving behavior.

With the rapid development of LLMs, some researchers have also utilized their integration with traditional RL to address their respective shortcomings [17]. ChatGPT [23] utilizes RL in the training process of LLMs to align the model with human preferences, thoughts, and abstract goals. However, it lacks the capability to directly control UGVs at the most granular level, making it impractical for direct real-world applications. SayCan [1] utilizes LLMs to decompose natural language commands issued by users into a sequence of robot skills, and then multiplies the probability of skill success judged by the feasibility function with the probability of skill utility judged by LLMs. EUREKA [20] takes the source code of the environment and task description text as inputs to a LLM. It then generates executable reward function code, and iterates between sampling reward functions, evaluating rewards, and reflecting on rewards to improve the reward function.

Moreover, some researchers also utilize LLMs to address issues in RL [4, 30]. ELLM [6] embeds suggestions generated by LLMs into the input of the policy. However, calling LLMs for suggestions at each step consumes a significant amount of time and resources. LCA (The Language-Centric Agent) [5] is centered around language, leveraging LLMs and visual language models to comprehend the environment, decompose tasks, and then output actions based on subtasks and the current state of the

environment using RL policies. However, this is limited to the field of robotics and requires providing examples of the desired language outputs.

Different from these methods, the main idea of our approach is to address the challenges of large solution spaces and low learning efficiency through policy regularization and reward shaping assisted by LLMs. We leverage the commonsense knowledge inherent in LLMs, which are zero-shot and do not require fine-tuning. Our approach is lighter-weight compared to LCA and more efficient compared to EUREKA. Unlike offline RL and imitation learning, we do not require the collection of demonstration data. Additionally, we introduce a reward shaping mechanism based on LLMs evaluation to imbue the agent with abstract human preferences such as safety and urgency, without the need for designing corresponding reward functions.

3 Background

The RL problem can be formulated as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ represents the state space and $\mathcal{A}$ represents the action space. $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the transition probability function that describes the dynamics of the environment, while $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to R$ is the reward function and $\gamma \in (0,1]$ is the discount factor that determines the importance of future rewards relative to immediate rewards.

In RL, the policy that outputs actual actions for the agent and the policy that updates values can be the same (on-policy) or different (off-policy). In off-policy algorithms, a classic algorithm is TD3, which integrates the idea of DDQN into the DDPG algorithm, addressing the issue of overestimation and suitable for continuous control tasks [8]. The TD3 consists of six networks: the actor network π_θ, critic1 network Q_{ϕ_1}, critic2 network Q_{ϕ_2}, target actor network $\pi_{\theta'}$, target critic1 network $Q_{\phi_1'}$, and target critic2 network $Q_{\phi_2'}$. Compared to the two critic networks, the updates of the actor network and the three target networks are delayed. The loss function for the actor network is as follows, where $\mathbb{D}$ represents the experience pool.

$$J_\pi(\theta) = E_{s\sim\mathbb{D}}\left[-Q_{\phi_1}(s,a)\right], a = \pi_\theta(s) \tag{1}$$

The use of policy regularization in RL has already been employed. For example, considering maximum entropy during policy optimization [11] helps to enhance the exploratory nature of the policy to some extent. The loss function of SAC is as follows:

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha\mathcal{H}\left(\pi(\cdot\,|s_t)\right)\right] \tag{2}$$

where $\mathcal{H}\left(\pi(\cdot\,|s_t)\right)$ is the entropy. In the PPO-Penalty algorithm [13], the stability of training is enhanced by dynamically adjusting the coefficient β of the KL divergence penalty term, thereby restricting the magnitude of policy optimization. The loss function is as follows, where A represents the advantage function [33].

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t) - \beta\mathrm{KL}\left(\pi_{\theta'}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right)\right] \tag{3}$$

To successfully train an agent in environments with sparse rewards, it often requires the design of specific exploration mechanisms to boost rewards. This is referred to as intrinsic reward, in contrast to the rewards provided by the environment, known as extrinsic reward. Some intrinsic rewards encourage the agent to explore more states, others to explore more state-action pairs, and some to learn more diverse objectives (like ELLM and PRLL). RND calculates the error of the state encodings predicted by the predictor network $\hat{f}(s_{t+1})$ and the target network $f(s_{t+1})$ to measure the novelty of states [19], providing the agent with a corresponding intrinsic reward.

$$R_i = ||f_\theta(s_{t+1}) - \widehat{f_{\hat{\theta}}}(s_{t+1})||^2 \tag{4}$$

The predictor network $\hat{f}$ is trained simultaneously with PPO, while the target network f is initialized with random parameters θ and kept fixed, with both neural networks sharing the same structure.

4 Policy Regularization and Reward Shaping Assisted by LLMs

This section presents our proposed method, PRLL, designed to address the challenges of expansive solution spaces and low learning efficiency. Our method comprises two main components: (a) utilizing a LLMs-based policy regularization method to guide policy learning towards solution spaces conducive to task-solving, and (b) employing reward reshaping mechanisms assessed by LLMs to support policy learning towards solution spaces aligned with human preferences.

4.1 LLMs-guided Policy Regularization

Regarding the use of LLMs for goal proposals in RL, ELLM [6] embeds the suggestions generated by LLMs into the input of the policy. However, we propose a more efficient approach to utilize suggestions generated by LLMs. As shown in Fig. 2, we calculate the similarity between the suggestions from the LLMs and the actions selected by the agent, and then We choose the highest suggestion similarity to form a regularization term. In other words, we utilize the inherent prior knowledge in the LLMs to assist the policy in narrowing down the solution space efficiently for task completion.

Two description captioners are used: one describing the state and one describing the action. The state captioner C_{obs} is responsible for converting the agent's current state O_t into text descriptions $C_{obs}(O_t)$ following a fixed format. This includes the current observations of the agent (e.g., in UGV street-keeping scenario: crosswalks, traffic lights, grassy areas, other vehicles), the current status (whether it is driving or stopped), and the available actions (e.g., possible driving directions on the current road: straight, right, left). All this information can be obtained in Unity, and then utilized by the state captioner to input the agent's current state in text form to the LLMs, helping the model to comprehend the agent's current state. On the other hand, the action captioner C_{action} converts the action a_t selected by the agent's policy at the current timestep into text $\mathrm{C_{action}(a_t)}$ (e.g., "stop in front of the crosswalk").

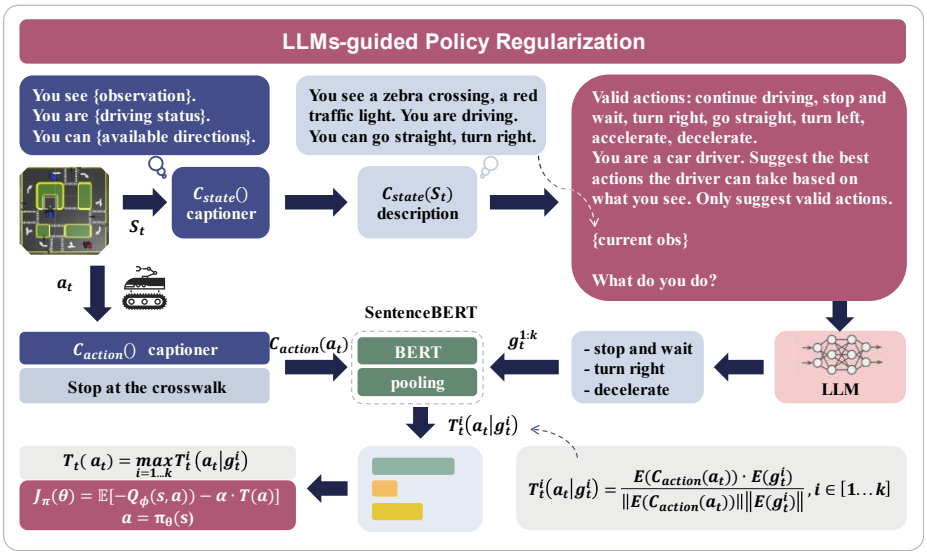

Fig. 2. The architecture of LLMs-guided Policy Regularization. We calculate the similarity between the suggestions from the LLMs and the actions selected by the agent, and then choose the highest suggestion similarity to form a regularization term.

In this module, besides describing the current state of the agent, scene descriptions and task descriptions also need to be input to the LLMs. Scene descriptions refer to the actions the agent can take in the current scene. For example, in UGV street-keeping scenario, the agent's action space is continuous and includes only two dimensions: steering and speed. Therefore, it is necessary to input the legal actions (continue driving, stop and wait, turn right, go straight, turn left, slow down, speed up) to the LLMs to constrain the output of the LLMs. Task descriptions refer to a detailed introduction of the agent's current role and task objectives. For instance, instructing the LLMs to act as a car driver and provide the best action suggestions based on the current situation. In this way, the LLMs can generate k pieces of coarse-grained suggestions $g_t^{1:k}$ at the policy level to guide the agent, rather than precise numbers such as degrees of steering or specific speeds. In fact, LLMs cannot provide such highly precise numbers.

After obtaining the action description text $C_{action}(a_t)$ and suggestions $g_t^{1:k}$ from LLMs, we use SentenceBERT E [27] to calculate the cosine similarity between each suggestion g_t^i and the action description text $C_{action}(a_t)$, resulting in k similarities $T_t^i(a_t|g_t^i)$:

$$T_t^i(a_t|g_t^i) = \frac{E(C_{action}(a_t))E(g_t^i)}{||E(C_{action}(a_t))||||E(g_t^i)||}, \mathrm{i} \in [1 \ldots k] \tag{5}$$

where $E(C_{action}(a_t))$ is the embedding of the action description text and $E(g_t^i)$ is the embedding of the suggestions. Higher similarity implies that the agent is attempting the suggested action in a way that is beneficial for achieving the task, even though we do not directly inform the agent of the suggestions. Then, we take the highest similarity as

the similarity $T_t(a_t)$ between the current action and the overall task objective at the current time step:

$$T_t(a_t) = \max_{i=1\ldots k} T_t^i\left(a_t \middle| g_t^i\right) \tag{6}$$

The reason is that LLMs often provide multiple viable approaches to complete a task, and the agent only needs to follow one of these suggestions to achieve success. For example, at a red light, the agent can either stop and wait for the green light before going straight, or it can make a right turn to go around. The agent only needs to be executing one of these actions, rather than driving onto the grassy area on the side of the road.

Next, we use this similarity $T_t(a_t)$ between the action and the task objective to form a regularization term:

$$J_\pi(\theta) = E\left[Q_\phi(s,a) - \alpha \cdot T(s)\right], a = \pi_\theta(s) \tag{7}$$

where the parameter α represents the importance given to this similarity regularization term. In this way, by adding the similarity between the current action and the task objective during policy optimization, the agent can leverage the general prior knowledge from LLMs to more quickly narrow down the solution space and efficiently achieve the predefined goals.

4.2 LLMs-evaluated Reward Shaping

Regarding the use of LLMs for reward design in RL, EUREKA [20] involves inputting the environment's source code and task description text into the LLMs, allowing it to generate executable and complete reward function code. We propose a novel reward approach that supplements rewards with evaluation results from the LLMs, enabling the trained agent to exhibit human preferences without the need to define reward functions for each preference explicitly. The overall framework is illustrated in the Fig. 3.

The human preferences we refer to include safety, urgency, and other human-specific characteristics in decision-making. For instance, some individuals are cautious and prioritize safety when driving. They strictly adhere to traffic signals and speed limits, even slowing down before a crosswalk when the light is green to prevent potential accidents with pedestrians or vehicles. However, in urgent situations such as police pursuits or military support missions, the agent must reach the destination as quickly as possible, disregarding traffic signals and speed limits but ensuring no collisions with other vehicles or pedestrians.

Using the same state captioner and action captioner as in the previous module, we obtain text descriptions of the state and action, which are then input to the LLMs. In addition, we input a task description with a specific human preference (for example, "you are a conservative and safety-oriented driver"), along with a query asking whether the current action satisfies this human preference, and We constrain the LLMs to only output "Yes" or "No". This way, the LLMs can evaluate whether the current action aligns with our desired human preference. If the LLMs output "Yes", we consider the reward given to the agent by the LLMs $R\_\{llm\}$ as 1. If the output is "No", the reward from the LLMs to the agent $R\_\{llm\}$ is considered as 0.

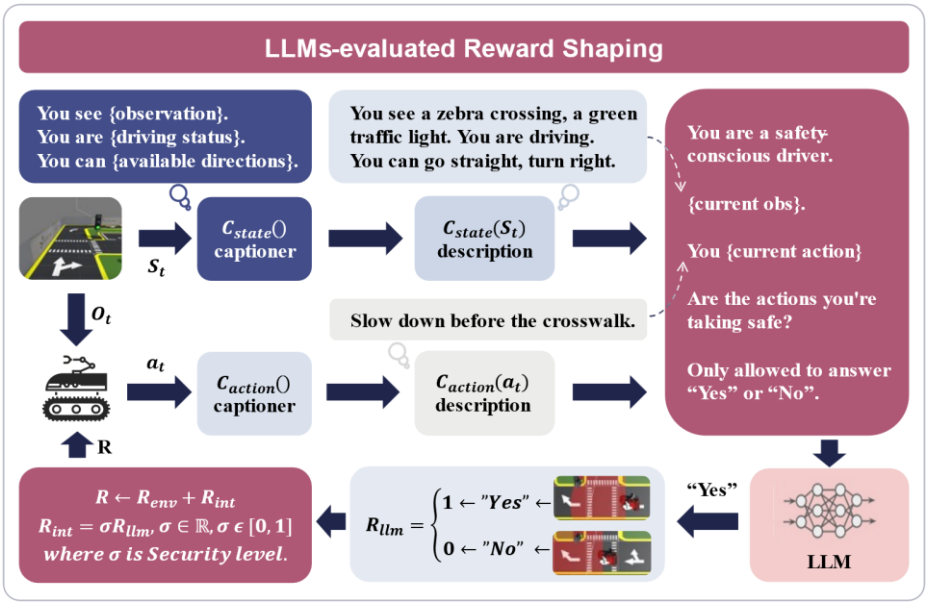

Fig. 3. The network architecture of LLMs-evaluated Reward Shaping. Given state descriptions, action descriptions, and task descriptions with human preferences, we use LLMs to evaluate whether the agent's behavior meets these human preferences. This evaluation is used as part of the reward.

$$R_{llm} = \begin{cases} 1 \leftarrow "Yes" \\ 0 \leftarrow "No" \end{cases} \tag{8}$$

In different scenarios, the emphasis on the human preferences possessed by the agent varies, hence the need for a parameter σ to control the magnitude of the intrinsic rewards R_{int} actually given to the agent. This parameter σ is a constant between 0 and 1, allowing for flexible adjustment according to different human preference requirements. Subsequently, the intrinsic reward R_{int} is added to the environmental reward R_{env} to form the total reward R for the agent.

$$R_{int} = \sigma R_{llm}, \sigma \in \boldsymbol{R}, \sigma \in [0,1] \tag{9}$$

$$\mathrm{R} \leftarrow \mathrm{R_{env}} + \mathrm{R_{int}} \tag{10}$$

In other words, if the LLMs deem the current action of the agent to align with the specified human preferences, an additional positive intrinsic reward is given to the agent. This encourages the agent to perform more actions in line with human preferences, facilitating the rapid narrowing of the solution space imbued with human preferences.

5 Experiments

In order to verify that the strategy regularization and reward reshaping mechanisms based on LLMs can effectively improve sample efficiency, we conducted experimental validation in the benchmark Crafter environment [6] and in a virtual simulation UGV street-keeping scenario built on Unity3D, as shown in Fig. 4.

5.1 Experiment setting

Crafter is a partially observable 2D open-world survival environment inspired by Minecraft, originally introduced by Danijar Hafner [6, 24]. In the Crafter experimental environment, the initial landscape is randomly generated, requiring the agent to explore its surroundings, gather essential resources such as food and water, and craft specific tools for survival. In each round, the agent earns a sparse reward of +1 for unlocking an achievement. For every health point lost, the reward decreases by -0.1, while for each health point restored, the reward increases by +0.1.

The UGV street-keeping scenario is a virtual simulation environment we developed in Unity3D. This scene includes various elements such as traffic lights, crosswalks, other vehicles, and lawns. The UGV needs to learn to navigate autonomously along the road while avoiding collisions with other unmanned vehicles, all while considering specific human preferences (e.g., conservative safety or emergency support). The UGV earns a reward of +1 each time it passes a checkpoint or crosses through a green traffic light. Conversely, if it collides with another vehicle, a wall, or a lawn, it receives a penalty of -7, which results in the immediate termination of the current episode.

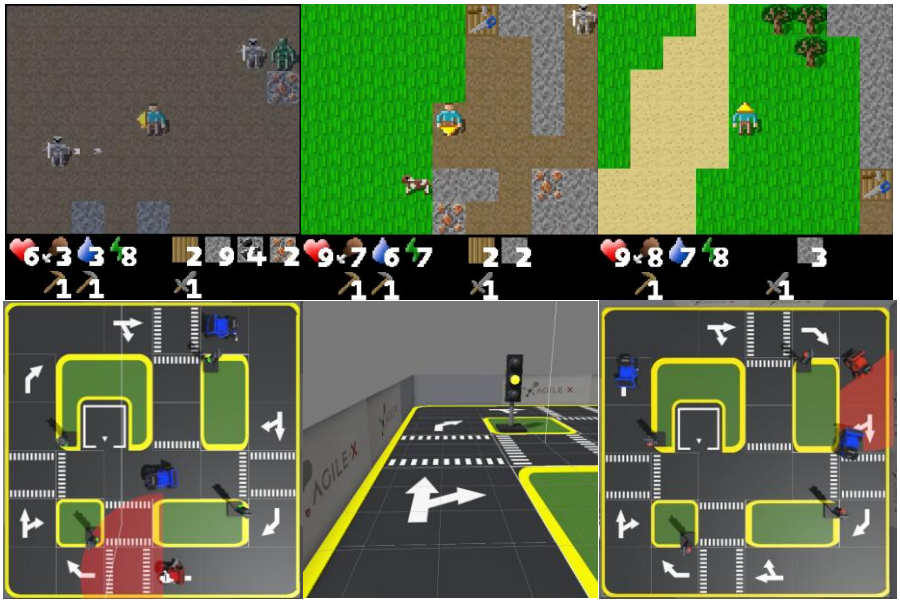

Fig. 4. Diagrams of the PRLL experimental environments, including Crafter (top row) and UGV street-keeping scenario (bottom row).

We selected LLaMA 3.3 (70.6B) [7, 10, 31] as the LLMs for the PRLL algorithm. Given that Crafter features a one-dimensional discrete action space represented by action IDs, we built upon the DQN [21] algorithm to implement the PRLL. In contrast, the UGV street-keeping scenario involves a two-dimensional continuous action space, encompassing direction and speed, for which we utilized the SAC algorithm [11]. The experimental hyperparameters for the PRLL algorithm are provided in Table 1.

Table 1. Hyperparameter table for PRLL algorithm experiment.

hyperparameters	Crafter	UGV street-keeping
discount factor	0.99	0.99
max steps	2000000	150000
update coefficient	0.005	0.005
update steps	8000	1
learning rate	0.0000625	0.0003
evaluate steps	25000	1000
evaluate episodes	10	3

5.2 Overall performance

We compare our algorithm with the following baselines: ELLM [6], DQN [21], and SAC [11], where ELLM is also an algorithm that combines LLMs with RL. In the Crafter environment, we use the Crafter score as the evaluation metric, which is influenced by the number of achievements unlocked by the agent, lost or recovered health points, and the difficulty of the unlocked achievements. Additionally, we selected two challenging achievements with preconditions, "deforestation" and "make wood sword", as human preferences. To fine-tune the trained policy for these downstream tasks, we trained a new agent while replacing 50% of the random sampling with actions sampled from the trained policy. The performance of the agent was then evaluated using achievement scores. In the UGV street-keeping scenario, we utilize the cumulative rewards from the environment as the evaluation metric, which does not include the sparse rewards provided by PRLL, and safe driving is the human preference. In this experiment, we employ the authors' implementation of ELLM in the Crafter environment, along with our own re-implementation of ELLM in the UGV street-keeping environment, using the recommended hyperparameters reported in their paper.

Table 2. Average results of the final 10 evaluations.

algorithms	Crafter	deforestation	make wood sword	UGV street-keeping
ELLM	7.5	2.3	0.7	39.2
SAC	\	\	\	24.6
DQN	4.8	0.2	0.1	\
PRLL(Ours)	7.9±0.6	2.9±0.2	1.3±0.1	51.3±2.4

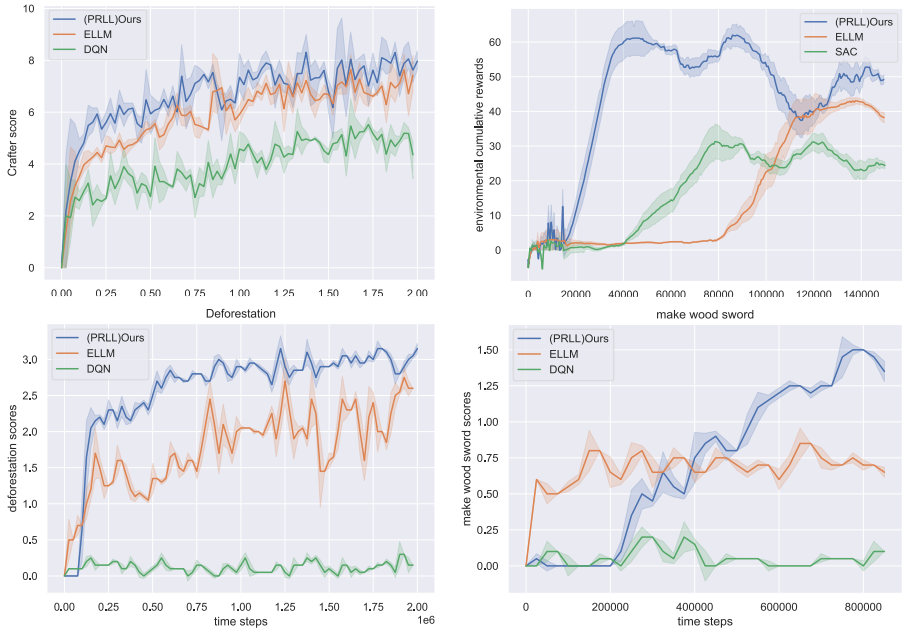

Fig. 5. PRLL and the baselines' learning curves in the Crafter environment (top left, bottom left, bottom right) and the UGV street-keeping environment (top right).

The results of our algorithm, along with all the considered baseline methods, are summarized in Table 2. Additionally, as shown in Fig. 5, we plot the learning curves of PRLL, DQN, SAC, and ELLM in the Crafter environment and the UGV street-keeping environment. Obviously, PRLL demonstrates superior final performance with enhanced training stability, as evidenced by higher convergence trajectories and attenuated oscillation patterns compared to baseline methods. Overall, our method performs best in terms of average scores, regardless of whether the environment has discrete or continuous action spaces. Moreover, our method requires fewer time steps to achieve comparable scores compared to the other approaches.

5.3 Ablation

As shown in Fig. 6, we evaluate the impact of using or not using policy regularization and reward reshaping on the performance of the PRLL algorithm. "PRLL without PR" refers to PRLL without policy regularization, relying solely on the actor loss of the RL algorithm for training. Meanwhile, "PRLL without RS" indicates that PRLL does not use reward reshaping, depending instead on the original rewards from the environment and a manually defined preference reward function to learn specific preferences.

We record the scores of these three algorithms after training for 2 million steps and 60,000 steps in the two environments, respectively. The results indicate that using

policy regularization can better constrain the policy to select actions that align with human common sense and are capable of completing tasks, leading to higher scores. Additionally, although "PRLL" and "PRLL without RS" exhibit similar policy constraint capabilities, "PRLL" achieves a higher score, possibly because it is difficult to design a reward function that describes abstract human preferences. Overall, relying solely on policy regularization or reward reshaping is insufficient. It is also necessary to leverage the common sense embedded within LLMs.

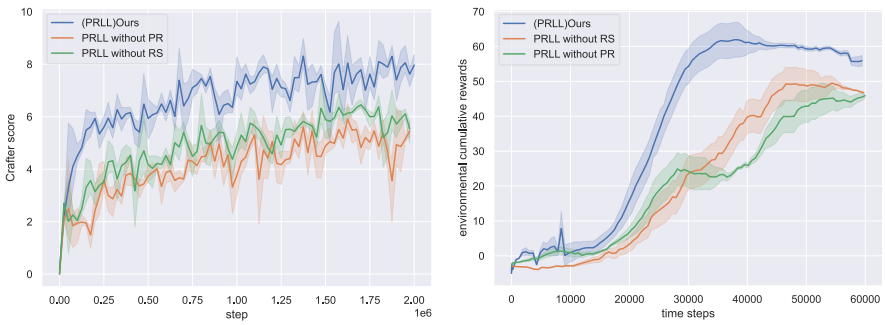

Fig. 6. The performance of PRLL, both with and without policy regularization (PR) and reward reshaping (RS), is evaluated in the Crafter environment (left) and the UGV street-keeping environment (right).

6 Conclusion

We propose the policy regularization and reward shaping assisted by large language models (PRLL), aimed at addressing the challenges of large solution spaces and low learning efficiency in traditional deep reinforcement learning. By computing the similarity between the "suggestions" from large language models (LLMs) and the "actions" of the agent, we form a regularization term to help policy get task-solving solution spaces. Additionally, we introduce the use of LLMs to evaluate whether the policy's output actions align with human preferences, providing corresponding intrinsic rewards to assist policy learning towards solution spaces with human preferences. Our experiments demonstrate that at the same time steps, our algorithm outperforms most baselines, significantly enhancing the learning efficiency of RL agents. Our method's outstanding performance in intelligent unmanned ground vehicle (UGV) systems provides a potential option for the widespread application of unmanned systems in the real world. In the future, we will conduct further research on this.

Acknowledgments. This study was funded by National Natural Science Foundation of China (grant number 62421004).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B.,Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al.: Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691 (2022)
2. Bian, H., Lu, Q.: Hlrs: A deep reinforcement learning-based hero recommendation system for moba games. In: 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 2040–2047. IEEE (2023)
3. Bui, V.H., Mohammadi, S., Das, S., Hussain, A., Hollweg, G.V., Su, W.: A critical review of safe reinforcement learning strategies in power and energy systems. Engineering Applications of Artificial Intelligence 143, 110091 (2025)
4. Chen, D., Huang, Y.: Integrating reinforcement learning and large language models for crop production process management optimization and control through a new knowledge-based deep learning paradigm. Computers and Electronics in Agriculture 232, 110028 (2025)
5. Di Palo, N., Byravan, A., Hasenclever, L., Wulfmeier, M., Heess, N., Riedmiller, M.: Towards a unified agent with foundation models. In: Workshop on Reincarnating Reinforcement Learning at ICLR 2023 (2023)
6. Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., Andreas, J.: Guiding pretraining in reinforcement learning with large language models. arXiv preprint arXiv:2302.06692 (2023)
7. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
8. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International conference on machine learning. pp. 1587–1596. PMLR (2018)
9. Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Knoll, A.: A review of safe reinforcement learning: Methods, theories and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
10. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
11. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. pp. 1861–1870. PMLR (2018)
12. Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., Wang, Z.: Exploration in deep reinforcement learning: From single-agent to multiagent domain. IEEE Transactions on Neural Networks and Learning Systems (2023)
13. Heess, N., Tb, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S., et al.: Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286 (2017)
14. Le Mero, L., Yi, D., Dianati, M., Mouzakitis, A.: A survey on imitation learning techniques for end-to-end autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems 23(9), 14128–14147 (2022)
15. Li, S.E.: Deep reinforcement learning. In: Reinforcement Learning for Sequential Decision and Optimal Control, pp. 365–402. Springer (2023)
16. Li, Y., Ma, W., Li, Y., Li, S., Chen, Z., Shahidehpour, M.: Enhancing cyber-resilience in integrated energy system scheduling with demand response using deep reinforcement learning. Applied Energy 379, 124831 (2025)

17. Liu, S., Yao, Y., Jia, J., Casper, S., Baracaldo, N., Hase, P., Yao, Y., Liu, C.Y., Xu, X., Li, H., et al.: Rethinking machine unlearning for large language models. Nature Machine Intelligence pp. 1–14 (2025)
18. Liu, W., Yao, P., Wu, Y., Duan, L., Li, H., Peng, J.: Imitation reinforcement learning energy management for electric vehicles with hybrid energy storage system. Applied Energy 378, 124832 (2025)
19. Ma, C., Li, A., Du, Y., Dong, H., Yang, Y.: Efficient and scalable reinforcement learning for large-scale network control. Nature Machine Intelligence 6(9), 1006–1020 (2024)
20. Ma, Y.J., Liang, W., Wang, G., Huang, D.A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., Anandkumar, A.: Eureka: Human-level reward design via coding large language models. arXiv preprint arXiv:2310.12931 (2023)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning (2020)
22. Moerland, T.M., Broekens, J., Plaat, A., Jonker, C.M., et al.: Model-based reinforcement learning: A survey. Foundations and Trends® in Machine Learning 16(1), 1–118 (2023)
23. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems 35, 27730–27744 (2022)
24. Paglieri, D., Cupiał, B., Coward, S., Piterbarg, U., Wolczyk, M., Khan, A., Pignatelli, E., Kuci´nski, Ł., Pinto, L., Fergus, R., et al.: Balrog: Benchmarking agentic llm and vlm reasoning on games. arXiv preprint arXiv:2411.13543 (2024)
25. Pitis, S., Chan, H., Zhao, S., Stadie, B., Ba, J.: Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In: International Conference on Machine Learning. pp. 7750–7761. PMLR (2020)
26. Prudencio, R.F., Maximo, M.R., Colombini, E.L.: A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Transactions on Neural Networks and Learning Systems (2023)
27. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
28. Rolf, B., Jackson, I., M¨uller, M., Lang, S., Reggelin, T., Ivanov, D.: A review on reinforcement learning algorithms and applications in supply chain management. International Journal of Production Research 61(20), 7151–7179 (2023)
29. Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Mart´ın-Mart´ın, R., Stone, P.: Deep reinforcement learning for robotics: A survey of real-world successes. Annual Review of Control, Robotics, and Autonomous Systems 8 (2024)
30. Tang, X., Liu, F., Xu, D., Jiang, J., Tang, Q., Wang, B., Wu, Q., Chen, C.P.: Llm-assisted reinforcement learning: Leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment. IEEE Transactions on Consumer Electronics (2025)
31. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozi`ere, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
32. Wang, N., Li, Z., Liang, X., Hou, Y., Yang, A., et al.: A review of deep reinforcement learning methods and military application research. Mathematical Problems in Engineering 2023 (2023)
33. Xiao, Y., Tan, W., Hoffman, J., Xia, T., Amato, C.: Asynchronous multi-agent deep reinforcement learning under partial observability. The International Journal of Robotics Research p. 02783649241306124 (2025)