LLM-Empowered State Representation for Reinforcement Learning

Boyuan Wang [*1] **Yun Qu** [*1] **Yuhang Jiang** [1] **Jianzhun Shao** [1] **Chang Liu** [1] **Wenming Yang** [1] **Xiangyang Ji** [1]

Abstract

Conventional state representations in reinforcement learning often omit critical task-related details, presenting a significant challenge for value networks in establishing accurate mappings from states to task rewards. Traditional methods typically depend on extensive sample learning to enrich state representations with task-specific information, which leads to low sample efficiency and high time costs. Recently, surging knowledgeable large language models (LLM) have provided promising substitutes for prior injection with minimal human intervention. Motivated by this, we propose LLM-Empowered State Representation (LESR), a novel approach that utilizes LLM to autonomously generate task-related state representation codes which help to enhance the continuity of network mappings and facilitate efficient training. Experimental results demonstrate LESR exhibits high sample efficiency and outperforms state-of-the-art baselines by an average of **29%** in accumulated reward in Mujoco tasks and **30%** in success rates in Gym-Robotics tasks. Codes of LESR are accessible at https://github.com/thu-rllab/LESR.

1. Introduction

Traditional reinforcement learning (RL) algorithms (Tesauro et al., 1995; Watkins & Dayan, 1992; Rummery & Niranjan, 1994) generally require a large number of samples to converge (Maei et al., 2009; Ohnishi et al., 2019). Compounding this challenge, in most cases, the intricate nature of RL tasks significantly hampers sample efficiency. (Dulac-Arnold et al., 2019). For handling complex tasks and enhancing generalization, neural networks are utilized to approximate value functions (Mnih et al., 2015; Schulman et al., 2015). However, value networks often lack smoothness even when they are trained to converge, leading to instability and low sample efficiency throughout the training process (Asadi et al., 2018). Considering that in deep RL, state vectors serve as the primary input to value networks, sub-optimal state representations can result in limited generalization capabilities and non-smoothness of value network mappings (Schuck et al., 2018; Merckling et al., 2022).

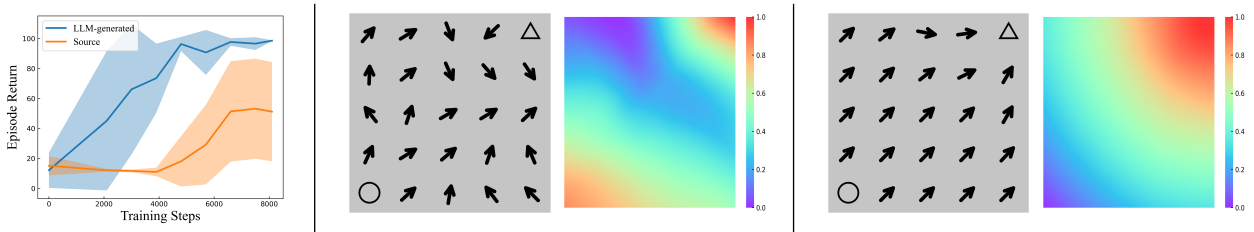
In most RL environments (Todorov et al., 2012; Brockman et al., 2016; de Lazcano et al., 2023), source state representations typically embody environmental information but lack specific task-related details. However, the learning of value networks depends on accurately capturing task dynamics through rewards (Yang et al., 2020; Yoo et al., 2022). The absence of task-related representations may impede the establishment of network mappings from states to rewards, affecting network continuity. Previous researches leverage diverse transitions, intensifying the issue of time-consuming data collection and training (Merckling et al., 2020; Sodhani et al., 2021; Merckling et al., 2022). Consequently, a question arises regarding the existence of a more efficient method for identifying task-related state representations.

Indeed, incorporating task-specific information into the state representation can be achieved through introducing expert knowledge (Niv, 2019; Ota et al., 2020). Recently, significant strides have been accomplished in the field of Large Language Models (LLM) (Touvron et al., 2023; OpenAI, 2023). The exceptional performance of LLM in various domains (Shen et al., 2023; Miao et al., 2023; Liang et al., 2023; Madaan et al., 2023; Stremmel et al., 2023), particularly in sequential decision-making tasks, showcases their extensive knowledge and sufficient reasoning abilities (Wang et al., 2023a;b; Feng et al., 2023; Wu et al., 2023; Du et al., 2023; Shukla et al., 2023; Sun et al., 2023; Zhang & Lu, 2023). This motivates the idea that the prior knowledge embedded in LLM can be exploited to enhance the generation of task-related state representations.

To preliminarily validate LLM's capacity in enhancing state representations, an illustrative toy example is provided. In Figure 1, results demonstrated that the state representation generated by LLM can help to enhance the continuity of value networks and expedite the convergence of policy learning. To substantiate this, we employ the Lipschitz constant (Jones et al., 1993) for smoothness assessment. The

[*]Equal contribution [1]Tsinghua University. Correspondence to: Xiangyang Ji <xyji@tsinghua.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) Training Curves (b) Source, $\mathrm{Lip}(Q) = 37.47$ (c) LLM-generated, $\mathrm{Lip}(Q) = 3.15$

Figure 1. Experiments conducted in the `PointMaze` in Gym-Robotics (de Lazcano et al., 2023). The agent aims to navigate from the bottom-left (depicted by '◯') to the top-right target (depicted by '△'). (a) The training curves of policies in two different state representations: (1) Source: which only involves the coordinates of the agent and the target; (2) LLM-Generated: which adds another dimension indicating the distance from the agent to the target. (b) (c): Visualizations of final policies and learned state values. Arrows show actions by final policies. Heatmaps display learned state values after just 500 training steps, and the smoother result of LLM-generated shows higher sample efficiency. The Lipschitz constant $\mathrm{Lip}(Q)$ is defined in Definition 3.1. Further details are provided in Appendix D.3.

LLM-generated state representations enhance the Lipschitz continuity of value networks, accounting for the improvement of sample efficiency and performance.

In this paper, we propose a novel method named **L**LM-**E**mpowered **S**tate **R**epresentation (**LESR**). We utilize LLM's coding proficiency and interpretive capacity for physical mechanisms to generate task-related state representation function codes. LLM is then employed to formulate an intrinsic reward function based on these generated state representations. A feedback mechanism is devised to iteratively refine both the state representation and intrinsic reward functions. In the proposed algorithm, LLM consultation takes place only at the beginning of each iteration. Throughout both training and testing stages, LLM is entirely omitted, ensuring significant time savings and flexibility.

In summary, our main contributions are:

- We propose a novel method employing LLM to generate task-related state representations accompanied by intrinsic reward functions for RL. These functions are demonstrated to exhibit robustness when transferred to various underlying RL algorithms.
- We have theoretically demonstrated that enhancing Lipschitz continuity improves the convergence of the value networks and empirically validated more task-related state representations can enhance Lipschitz continuity.
- LESR is a general framework that accommodates both continuous and discontinuous reward scenarios. Experimental results demonstrate that LESR significantly surpass state-of-the-art baselines by an average improvement of **29%** in Mujoco tasks and **30%** in Gym-Robotics tasks. We have also experimentally validated LESR's adaptability to novel tasks.

2. Related Work

Incorporating LLM within RL Architecture Since the advent of LLM, researchers have endeavored to harness the extensive common-sense knowledge and efficient reasoning abilities inherent in LLMs within the context of RL environments. Challenges have arisen due to the misalignment between the high-level language outputs of LLMs and the low-level executable actions within the environment. To address this, Qiu et al. (2023); Agashe et al. (2023); Yuan et al. (2023); Wang et al. (2023a;b); Feng et al. (2023); Wu et al. (2023) have sought to employ environments where observation and action spaces can be readily translated into natural language (Carta et al., 2023; Puig et al., 2018). Alternative approaches employ language models as the policy network with fine-tuning (Zhang & Lu, 2023; Li et al., 2022; Shi et al., 2023; Yan et al., 2023; Carta et al., 2023). Meanwhile, other endeavors focus on leveraging LLMs as high-level planners, generating sub-goals for RL agents (Sun et al., 2023; Shukla et al., 2023; Zhang et al., 2023). Nevertheless, these works encounter a common challenge: the tight coupling of LLMs with RL agents, leading to frequent communication between the two even during the testing stage, a process that proves time-consuming and inefficient.

State Representation Derived from LLM Some researchers use LLM for state representations that contain more information in partially observable scenarios. Da et al. (2023) employs LLMs to provide extra details about road conditions in traffic control areas. Similarly, Shek et al. (2023) focuses on robot locomotion in unstructured environments, using LLM as translators to extract environmental properties and generate contextual embeddings for training. Chen et al. (2023), tracks key objects and attributes in open-world household environments via LLM, expanding and updating object attributes based on historical trajectory information. These methods mainly addressed the issue of missing state information in partially observable scenarios.

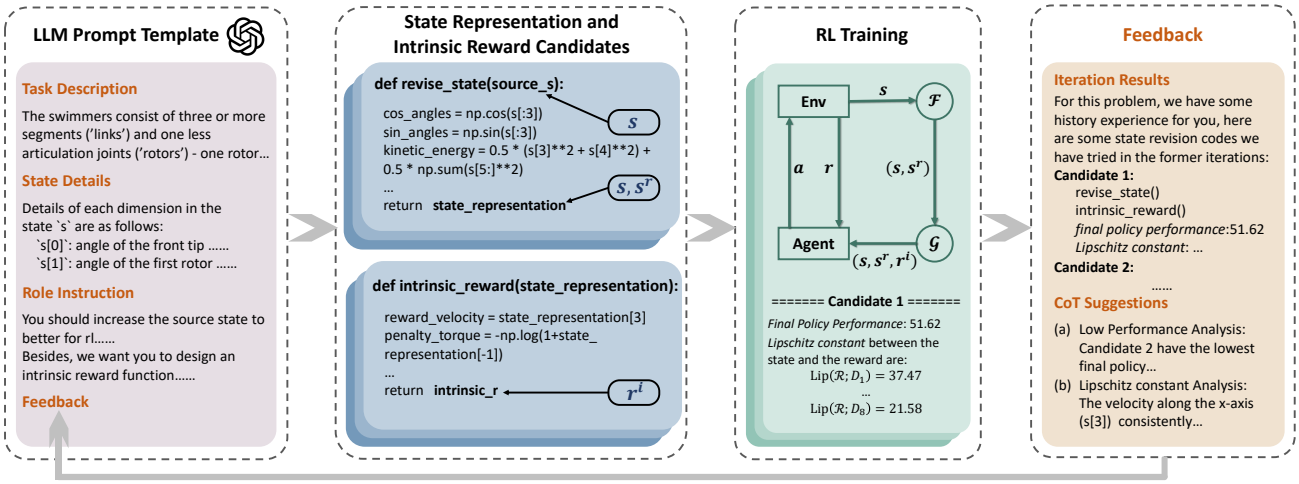


Figure 2. LESR Framework: (1) LLM is prompted to generate codes for state representation and intrinsic reward functions. Refer to Appendix C for details on all prompt templates. (2) K state representations and intrinsic rewards $\{\mathcal{F}_k\}_{k=1}^K, \{\mathcal{G}_k\}_{k=1}^K$ are sampled from LLM. (3) During RL training, function $\mathcal{F}$ and $\mathcal{G}$ are utilized to generate $s^r = \mathcal{F}(s)$ for state representations, and $r^i = \mathcal{G}(s, s^r)$ for intrinsic rewards. (4) Finally, Lipschitz constants and episode returns of each candidate serve as feedback metrics for LLM.

In contrast, our method emphasizes exploring correlations among internal state features, recognizing meaningful correlations reflecting underlying physical relationships and generating more task-related representations.

Reward Design via LLM For the purpose of effectively bridging the gap between high-level language instructions and low-level robot actions, some researchers employ rewards as the intermediate interface generated by LLM. Works in this domain can be categorized into the following three types: (1) Sparse Reward: Kwon et al. (2023); Yu et al. (2023); Sontakke et al. (2023), aiming to design sparse rewards at the trajectory level. (2) Dense Reward: Song et al. (2023); Xie et al. (2023); Ma et al. (2023); Rocamonde et al. (2023), aiming to design dense rewards for every interactive step of the agent. (3) Intrinsic Reward: Klissarov et al. (2023); Triantafyllidis et al. (2023), aiming to design intrinsic rewards for reducing ineffective exploration and improving sample efficiency. In this paper, we also utilize LLM to generate intrinsic reward function codes. The primary difference between our methodology and prior research lies in that our reward design serves as an auxiliary mechanism for encouraging the agent to better comprehend the state representations generated by LLM. This aids the policy and critic networks in establishing correlations between the state representations and intrinsic rewards.

3. Method

3.1. Problem Statement

We consider a Markov decision process (Puterman, 1990) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0, \gamma)$, where $\mathcal{S}$ denotes the source state space and $\mathcal{A}$ denotes the action space. Given a specific task, $\mathcal{R}$ is the source extrinsic reward function of the environment. $\mathcal{P}(s'|s,a)$ denotes the dynamic transition function, p_0 is the initial state distribution, and γ is the discount factor. The primary objective is to learn a RL policy $\pi(a|s)$ that maximizes the cumulative rewards expectation, which is defined as value function $Q_\pi(s_t, a_t) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_t, a_t\right]$.

In order to assess the impact of LESR on network continuity, we introduce the Lipschitz constant (Jones et al., 1993):

Definition 3.1. Denote the data space as $\mathcal{X} \subset \mathbb{R}^d$ and the label space as $\mathcal{Y} \subset \mathbb{R}$. Consider a dataset $\mathcal{X}_0 \subset \mathcal{X}$, and the label $\mathcal{Y}_0 = \{y_i | y_i = u(x_i), \text{where } x_i \in \mathcal{X}_0\} \subset \mathcal{Y}$. Here, x_i represents a sequence of i.i.d. random variables on $\mathcal{X}$ sampled from the probability distribution ρ, and $u : \mathcal{X}_0 \subset \mathcal{X} \to \mathcal{Y}$ is the Lipschitz constant of a mapping given by

$$\mathrm{Lip}(u; \mathcal{X}_0) = \sup_{x_1, x_2 \in \mathcal{X}_0} \frac{\|u(x_1) - u(x_2)\|_2}{\|x_1 - x_2\|_2}. \quad (1)$$

When $\mathcal{X}_0$ is all of $\mathcal{X}$, we write $\mathrm{Lip}(u; \mathcal{X}) = \mathrm{Lip}(u)$. A lower Lipschitz constant indicates a smoother mapping u.

3.2. LLM-Empowered State Representation

In many RL settings (Todorov et al., 2012; Brockman et al., 2016; de Lazcano et al., 2023), source state representations usually contain general environmental information, while often lacking specific details related to current tasks which is critical to the training of the value networks (Yang et al., 2020; Yoo et al., 2022). The absence of task-related representations may hinder network mappings from states to

rewards, impacting the continuity of networks. Recognizing this limitation, the identification and incorporation of additional task-related state representations emerge as a pivotal strategy. This strategic augmentation can expedite the establishment of network mappings, subsequently boosting the smoothness of networks and augmenting training efficiency.

Due to the extensive knowledge and priors embedded in LLM, utilizing it for generating task-related state representations can be promising. In this context, we present a direct and efficient method named **L**LM-**E**mpowered **S**tate **R**epresentation (**LESR**). The whole framework is depicted in Figure 2. Our methodology hinges on leveraging LLM to facilitate the generation of more task-specific state representations. Herein, we denote LLM as $\mathcal{M}$ and a descriptor translating symbolic information into natural language as d. Consequently, the input to LLM $\mathcal{M}$ is represented as $d(\mathcal{S})$, which constitutes four parts: (1) *Task Description*: information about the RL environment and the specific task. (2) *State Details*: information pertaining to each dimension of the source state. (3) *Role Instruction*: assignments that require LLM to generate task-related state representation and intrinsic reward codes. (4) *Feedback*: historical information from previous iterations. For a full comprehensive descriptions of our prompts, refer to Appendix C.

Our primary objective is to harness LLM $\mathcal{M}$ to formulate a python function $\mathcal{F} : \mathcal{S} \to \mathcal{S}^r$, where $\mathcal{S}^r$ denotes the LLM-empowered state representation space. $\mathcal{F}$ is sampled from $\mathcal{M}\Big(d(\mathcal{S})\Big)$ and $d(\mathcal{S})$ explicitly embeds the task information into LLM. The state representation function $\mathcal{F}$ utilizes source state dimensions for calculations, generating task-specific state dimensions. At each timestep t, when the agent get the current state s_t from the environment, the corresponding state representation $s_t^r = \mathcal{F}(s_t)$ will be concatenated to the source state as the input for the policy $\pi(a_t|s_t, s_t^r)$ and value $Q(s_t, s_t^r, a_t)$.

Once the state representation function $\mathcal{F}$ is obtained, LLM $\mathcal{M}$ is subsequently required to provide an intrinsic reward function $\mathcal{G} : \mathcal{S}^c \to \mathbb{R}$ in python code format based on $s_t^c = (s_t, s_t^r) \in \mathcal{S}^c$, where $\mathcal{S}^c = \mathcal{S} \times \mathcal{S}^r$ is the joint state space. More precisely, we stipulate in the prompt that LLM is obliged to incorporate the LLM-empowered state representations s_t^r to calculate the intrinsic rewards, and it also retains the option to incorporate the source state s_t for a better intrinsic reward design. We formulate the joint optimization objective as:

$$\max_{\mathcal{F},\mathcal{G}} \max_{\pi} \mathbb{E}_{\mathcal{F},\mathcal{G},\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(r + w \cdot r^i\Big) \Big| r^i = \mathcal{G}\Big(s_t, \mathcal{F}(s_t)\Big)\Big]. \tag{2}$$

where $\mathcal{F}, \mathcal{G} \sim \mathcal{M}\Big(d\left(\mathcal{S}\right)\Big)$, and w is the weight of the intrinsic reward.

3.3. Lipschitz Constant for Feedback

In practice, to enhance the robustness of state representations, we iteratively query LLM multiple times, incorporating previous training results as feedback. During each training iteration, we sample K state representation and intrinsic reward function codes $\mathcal{F}_k, \mathcal{G}_k, k = 1, \ldots, K$ from LLM $\mathcal{M}$. Subsequently, we concurrently execute K training processes over N_{small} training timesteps to evaluate the performance of each function $\mathcal{F}_k, \mathcal{G}_k$. Here, N_{small} is intentionally set to be smaller than the total timesteps N employed in the final evaluation stage.

• **Continuous Scenarios** For scenarios with continuous extrinsic reward, we maintain a Lipschitz constant array $C_k \in \mathbb{R}^{|\mathcal{S}^c|}$ for each of the K training instances. Each element of C_k signifies the Lipschitz constant of a mapping $u_i, i = 1, \ldots, |\mathcal{S}^c|$, which maps each dimension of $\mathcal{S}^c$ to the extrinsic rewards. Note: u_i is introduced to signify the Lipschitz constant computed independently for each state dimension concerning the extrinsic reward. This assessment is crucial for guiding the LLM in identifying and eliminating undesired dimensions within the state representation. Given a trajectory $T = \{s_t^c, r_t\}_{t=1}^{H}$ of length H, the current Lipschitz constant array is calculated as follows:

$$C_k^T = \Bigg[\mathrm{Lip}(u_i; T_i)\Bigg]_{i=1}^{|\mathcal{S}^c|}, \tag{3}$$

where $T_i = \{s_t^c[i], r_t\}_{t=1}^{H}$, $s_t^c[i]$ denotes the i-th dimension of the joint state representations s^c, and C_k^T denotes the Lipschitz constant array of the current trajectory. we soft-update C_k over trajectories:

$$C_k = \tau C_k + (1 - \tau) C_k^T, \tag{4}$$

where $\tau \in [0, 1]$ is the soft-update weight. At the end of each training iteration, the Lipschitz constant array C_k and policy performance ν_k are provided to LLM for CoT (Wei et al., 2022) suggestions, which, along with the training results, serve as feedback for subsequent iterations. The feedback information helps LLM to generate task-related state representations that exhibit a lower Lipschitz constant with extrinsic rewards, as elaborated in Section 3.4 where we discuss the theoretical advantages of a lower Lipschitz constant for network convergence. In the subsequent iterations, LLM leverages all historical feedback to iteratively refine and generate improved state representation and intrinsic reward function codes $\{\mathcal{F}_k\}_{k=1}^{K}, \{\mathcal{G}_k\}_{k=1}^{K}$. The whole algorithm is summarized in Algorithm 1.

• **Discontinuous Scenarios** Dealing with scenarios with discontinuous extrinsic reward using conventional RL baselines is notably challenging and constitutes a specialized research area (Vecerik et al., 2017; Trott et al., 2019; Liu et al., 2023). Despite this challenge, LESR remains effective in such scenarios. We provide two ways of estimating

Algorithm 1 LLM-Empowered State Representation

Input: Policy network π_θ, value network Q_ϕ, LLM $\mathcal{M}$, training timesteps N_{small}, total final evaluation timesteps N, sample count K, iteration count I, initial prompt $l_1 = d(\mathcal{S})$, feedback analysis prompt $l_{analysis}$.
Output: learned policy network π_θ, best state representation and intrinsic reward function $\mathcal{F}_{best}, \mathcal{G}_{best}$.

```
# Sampling and Training Stage
```
for itr = 1 to I **do**
  Sample $\mathcal{F}_k, \mathcal{G}_k \sim \mathcal{M}(l_{\text{itr}}), k = 1, \ldots, K$
  Concurrently execute K training processes over N_{small} training timesteps using $\mathcal{F}_k, \mathcal{G}_k$.
  Obtain C_k according to Equation (3), (4) and final policy performance ν_k.
  $l_{feedback} = \mathcal{M}(l_{analysis}; \{C_k\}; \{\nu_k\})$
  $l_{\text{itr}+1} = \left(l_{\text{itr}}; l_{feedback}\right)$
end for
```
# Evaluating Stage
```
Initialize π_θ and Q_ϕ.
Execute N training timesteps using $\mathcal{F}_{best}, \mathcal{G}_{best}$.
return $\pi_\theta, \mathcal{F}_{best}, \mathcal{G}_{best}$.

Lipschitz constant as feedback in discontinuous extrinsic reward settings.

LESR with Discounted Return In Equation 3, u_i initially maps each dimension of $\mathcal{S}^c$ to dense extrinsic rewards. In sparse reward settings, we substitute these extrinsic rewards with the discounted episode return $\sum_t \gamma^t r$. Thus, u_i now maps each dimension of $\mathcal{S}^c$ to the discounted episode returns, consistent with the algorithmic framework and theoretical scope proposed in LESR.

LESR with Spectral Norm Since in Theorems B.4 and B.8 show that reducing the Lipschitz constant of the reward function lowers the upper bound of $Lip(V; \mathcal{S})$ and improves the convergence of value functions. Therefore, we can use the spectral norm to estimate $Lip(V; \mathcal{S})$ (Anil et al., 2019; Fazlyab et al., 2019) as feedback to LLM. By calculating the spectral norm of the N weight matrices $W_1, \ldots, W_N$ of the value functions, the Lipschitz constant of the value function is bounded by $\prod_{i=1}^{N} \|W_i\|_2$, which is then presented to the LLM as feedback.

3.4. Theoretical Analysis

In this section, we present analysis of the theoretical implications of the Lipschitz constant on convergence in neural networks, inspired by Oberman & Calder (2018). Consider the dataset and labels $\mathcal{X}_0, \mathcal{Y}_0$ defined in Definition 3.1 with $N = |\mathcal{X}_0|$ elements. The true mapping from $\mathcal{X}$ to $\mathcal{Y}$ is denoted as $u_0^* : \mathcal{X} \to \mathcal{Y}$. Let $f : \mathcal{X} \to \mathcal{X}$ be a function transforming a source $x \in \mathcal{X}$ into a more task-related $f(x)$.

Definition 3.2. Denote $u(x; \psi)$ as a neural network mapping parameterized by ψ. Consider the empirical loss for $\mathcal{X}_0, \mathcal{Y}_0$, where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function satisfying (i) $\ell \geq 0$, (ii) $\ell(y_1, y_2) = 0$ if and only if $y_1 = y_2$:

$$\min_{u:\mathcal{X}_0 \to \mathcal{Y}} \mathcal{L}(u, \mathcal{X}_0) = \frac{1}{N} \sum_{i=1}^{N} \ell(u(x_i; \psi), y_i). \tag{5}$$

Assumption 3.3. The mapping $f : \mathcal{X} \to \mathcal{X}$ only swaps the order of x in the dataset $\mathcal{X}_0$, which means $\mathcal{X}_1 = \{f(x_i) | f(x_i) \in \mathcal{X}_0, i = 1, \ldots, N\}$ and $\mathcal{X}_1 = \mathcal{X}_0$. While for $x \in \mathcal{X}_2 = \{x | x \in \mathcal{X}, x \notin \mathcal{X}_0\}$, $f(x) = x$. Under f, the true mapping from $f(\mathcal{X})$ to $\mathcal{Y}$ is denoted as $u_1^* : f(\mathcal{X}) \to \mathcal{Y}$. It can be derived that $u_1^* = u_0^* \circ f^{-1}$. We suppose under f a lower Lipschitz constant is achieved:

$$\text{Lip}(u_1^*) \leq \text{Lip}(u_0^*). \tag{6}$$

Theorem 3.4. *Under Assumption 3.3, Given $\mathcal{X}_0, \mathcal{Y}_0$ and $\mathcal{X}_1, \mathcal{Y}_1 = \{y_i | y_i = u_1^*(x_i), x_i \in \mathcal{X}_1\}$, $u_0 \in \mathcal{U}_0$ is any minimizer of $\mathcal{L}(u, \mathcal{X}_0)$ and $u_1 \in \mathcal{U}_1$ is any minimizer of $\mathcal{L}(u, \mathcal{X}_1)$, where $\mathcal{U}_0$ and $\mathcal{U}_1$ denote the solution set of $\mathcal{L}(u, \mathcal{X}_0)$ and $\mathcal{L}(u, \mathcal{X}_1)$ in Definition 3.2 relatively, then on the same condition when $\text{Lip}(u_0) = \text{Lip}(u_1)$:*

$$\sup_{u_1 \in \mathcal{U}_1} \mathbb{E}_{x \sim \rho_f} \|u_1^* - u_1\|_2 \leq \sup_{u_0 \in \mathcal{U}_0} \mathbb{E}_{x \sim \rho} \|u_0^* - u_0\|_2, \tag{7}$$

ρ and ρ_f denote the source probability distribution on $\mathcal{X}$ and probability distribution on $f(\mathcal{X})$, relatively. In Theorem 3.4, it is demonstrated that the mapping f exhibiting a lower Lipschitz constant can attain superior convergence. This observation underscores the significance of identifying task-related state representations characterized by lower Lipschitz constants with respect to the associated rewards. Such analysis to some extent sheds light on why smoother network mappings exhibit improved convergence performance. Proofs of Theorem 3.4 can be referred to Appendix A.

We delve deeper into the significance of the Lipschitz constant of the reward concerning state representations in RL. We introduce two additional theorems, namely Theorem B.4 and Theorem B.8, establishing a strong correlation between $Lip(r; \mathcal{S})$ and $Lip(V; \mathcal{S})$ and, consequently, the convergence of RL's value functions. Theorem B.4 indicates that reducing the Lipschitz constant of the reward function lowers the upper bound of $Lip(V; \mathcal{S})$. Theorem B.8 illustrates how decreasing $Lip(r; \mathcal{S})$ can enhance the convergence of RL algorithms' value functions. These theorems collectively emphasize our focus on minimizing the Lipschitz constant of the reward function to improve RL algorithms' convergence. Detailed proofs are available in Appendix B.

4. Experiments

In this section, we will assess **L**LM-**E**mpowered **S**tate **R**epresentation (**LESR**) through experiments on two well-established reinforcement learning (RL) benchmarks: Mujoco (Todorov et al., 2012; Brockman et al., 2016) and Gym-Robotics (de Lazcano et al., 2023). For more information

about the tasks, see Appendix D. The following questions will guide our investigation:

•**Q1:** Can LESR generate task-related state representations characterized by lower Lipschitz constants in relation to extrinsic environmental rewards? (Section 4.2)

•**Q2:** Can LESR achieve higher sample efficiency and outperform RL baselines? Does each component contribute to the final performance? (Section 4.3, 4.4)

•**Q3:** Are functions $\mathcal{F}_{best}$ and $\mathcal{G}_{best}$ algorithm-agnostic, and transferable directly to other RL algorithms? (Section 4.5)

•**Q4:** Do $\mathcal{F}_{best}$ and $\mathcal{G}_{best}$ possess semantic-physical significance and exhibit consistency across different runs of LLM? Does LESR exhibit robustness to the variations in hyperparameters? (Section 4.6, 4.7)

4.1. Implementation Details

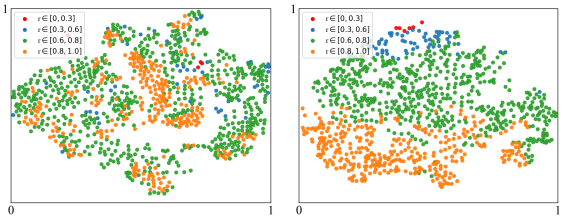
LLM and Prompts We employ the gpt-4-1106-preview as LLM to generate the state representation and intrinsic reward functions. There are three well-designed prompt templates and details of prompts are available in Appendix C.

Baseline Algorithm We employ the SOTA RL algorithm TD3 (Fujimoto et al., 2018) as the foundational Deep Reinforcement Learning (DRL) algorithm. Building upon the implementation of TD3 as provided in the source paper[1], we have formulated LESR (Ours). We also employ EUREKA (Ma et al., 2023) for comparison, which incorporates LLM for human-level reward designs. It is noteworthy that, in order to maintain comparative fairness, we have adhered to the hyperparameters of TD3 without introducing any modifications. Both EUREKA and LESR are grounded in the common RL algorithm TD3. For a comprehensive list of hyperparameters, please refer to Appendix G.

4.2. LESR Can Enhance Lipschitz Continuity

The reward function $\mathcal{R}$ commonly serves as an indicator of the specific task within the same environment (Yoo et al., 2022). In RL, value function learning is also predicated on rewards. Specifically, when the discount factor γ equals to 0, the value function directly learns the rewards. Additionally, the state representations form a part of the input to the value function. Task-related state representations might contribute to enhancing the Lipschitz continuity of the value function, thereby expediting the learning process.

Therefore, to validate whether LESR can help enhance the Lipschitz continuity between the generated state representations and the value function, we execute the final policy of LESR in the Mujoco `Ant` environment for 20 episodes of length $H = 1000$ and establish two datasets:



(a) $\mathrm{Lip}(\mathcal{R}; T_1) = 560.2$ (b) $\mathrm{Lip}(\mathcal{R}; T_2) = 168.1$

Figure 3. Visualization illustrating states post 2D dimensionality reduction via t-SNE. Details of T_1 and T_2 can be referred to Section 4.2. The reward for each state is normalized to a range of $[0, 1]$ and discretized, and the graph employs color coding to represent their respective reward values.

(a) $T_1 = \{s_t, r_t\}_{t=1}^{H}$ and (b) $T_2 = \{\mathcal{F}_{best}(s_t), r_t\}_{t=1}^{H}$. For states within each dataset, we employ t-SNE (Van der Maaten & Hinton, 2008) to visualize the states on the 2D graph with coloring the data with corresponding rewards and calculate the Lipschitz constant between states and rewards.

As shown in Figure 3, it is illustrated that in T_1 the Lipschitz constant of the mapping from state representations generated by LESR to the extrinsic environment rewards ($\mathcal{R} : \mathcal{F}(s) \to r, \mathrm{Lip}(\mathcal{R}; T_2) = 168.1$) is much lower than that of the source($\mathcal{R} : s \to r, \mathrm{Lip}(\mathcal{R}; T_1) = 560.2$). This indicates that the task-related state representations generated by LESR can enhance the Lipschitz continuity.

4.3. LESR Can Achieve High Sample Efficiency

Performance Comparison We validate LESR and the results are presented in Table 2. In Mujoco environments, LESR outperforms the state-of-art baselines in 4 out of 5 tasks. In Gym-Robotics environments, it outperforms the baselines in 5 out of 6 tasks. Particularly noteworthy are the results on challenging antmaze tasks, where TD3 fails to train, resulting in all final success rates of TD3 remaining at zero. Conversely, the performance exhibited by LESR is marked by excellence, highlighting its proficiency in overcoming the challenges posed by these intricate tasks. LESR yields an average performance improvement of 29% over the 5 tasks in Mujoco and an average success improvement of 30% over the 7 tasks in Gym-Robotics, thus substantiating the efficacy of the employed methodology. More information of training can be referenced in Appendix E.4.

Sample Efficiency For the purpose of assessing the sample efficiency of LESR, we have presented the performance for a limited scope of 300k training steps, as depicted in Table 1. The results demonstrate that LESR exhibits superior performance within significantly fewer training steps, excelling comprehensively across all tasks in comparison to the SOTA baseline, thereby manifesting heightened sample efficiency. This is further supported by the information of experimental details presented in Appendix E.4.

[1]https://github.com/sfujim/TD3

Table 1. Final Performance on Mujoco environments over only **300k** environment interaction training steps, aiming to validate the sample efficiency of LESR. **RPI =** `(accumulated_rewards - baseline)/baseline`: relative performance improvement.

Environments \ Algorithm	TD3	LESR(Ours)	RPI
HalfCheetah	7288.3±842.4	**7639.4±464.1**	**5%**
Hopper	921.9±646.8	**2705.0±623.2**	**193%**
Walker2d	1354.5±734.1	**1874.8±718.2**	**38%**
Ant	1665.2±895.5	**1915.3±885.5**	**15%**
Swimmer	49.9±5.2	**150.9±9.5**	**203%**
Mujoco Improve Mean	-	-	**91%**

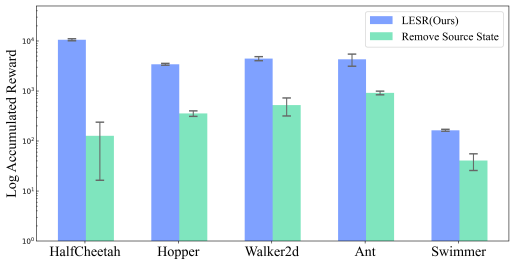


Figure 4. Comparison of Mujoco tasks between LESR and the removal of source state during training (i.e., utilizing only the ($\mathcal{F}(s)$) as input for policy and critic network training). The y-axis is on a logarithmic scale, and error bars represent 5 random seeds.

4.4. Ablation Study

Experimental Setting In this section, we conduct ablative analyses on distinct components of LESR to evaluate their respective contributions. Three ablation types are considered in total. **Ours w/o IR:** This entails the removal of the intrinsic reward component, with training relying solely on state representation. **Ours w/o SR:** Here, we exclude the state representation part. It is crucial to note that despite this ablation, the state representation function is still necessary due to the intrinsic reward calculation process $r^i = \mathcal{G}\Big(s, \mathcal{F}(s)\Big)$ outlined in Equation (2). However, in this context, only the source state s is provided as input to the policy, instead of the concatenated state s^c. **Ours w/o FB:** In this instance, the 'iteration_count' specified in Appendix G is set to 1 and the 'sample_count' is increased to 10, eliminating the feedback component.

Results and Analysis The ablation results are presented in Table 2. It is elucidated that regardless of the component subjected to ablation, a substantial decline in final performance ensues, underscoring the indispensability of all components for the final efficacy of our method. Crucially, our findings demonstrate that when only the intrinsic rewards part is ablated, namely **Ours w/o IR**, the performance results exhibit minimal influence, particularly in Mujoco environment tasks. This observation unveils the pivotal role of state representation component in our method, highlighting its significant contribution to the primary performance enhancement and further substantiating the assertions made in Section 1. Furthermore, since LESR requires that the input of networks be the concatenation of the source state and the generated state representations (i.e., $s_t^c = (s_t, s_t^r)$), we have conducted experiments to substantiate the indispensability of the source state s_t. As depicted in Figure 4, it is evident that the source state and the generated state representations work synergistically, both playing pivotal roles in achieving optimal performance.

Besides, in Appendix E, we also validate the role of the Lipschitz constant of LESR through more ablation experiments. Furthermore, we showcase LESR's robustness through experiments solely utilizing intrinsic reward functions, affirming its reliability. Additionally in Appendix E experiments on the novel tasks 'Walker Jump' and 'Walker Split Legs' underscore LESR's adaptability to new scenarios.

4.5. Directly Transfer to Other Algorithms

As there is no algorithm-specific information provided in the iteration prompts, we hypothesize that the state representation and intrinsic reward functions are algorithm-agnostic. This suggests that they can be directly transferred and integrated with other RL algorithms without the iteration process in Algorithm 1. To substantiate this hypothesis, we retain the best state representation and intrinsic reward function $\mathcal{F}_{best}, \mathcal{G}_{best}$ and combine them with two other widely employed RL algorithms **PPO** (Schulman et al., 2017) and **SAC** (Haarnoja et al., 2018) for validation.

The results in Table 3 highlight that the state representations and their associated intrinsic reward functions, acquired through the training of **TD3**, exhibit the potential to be integrated with alternative algorithms, still resulting in improved outcomes. This validates our initial hypothesis. Furthermore, these results further emphasize the efficacy and adaptability of our approach. Consequently, by simply employing a fundamental algorithm to explore state representation and intrinsic reward functions for a given task, it becomes possible to significantly diminish the training complexity associated with that task for other algorithms.

4.6. Semantic Analysis and Consistency Verification

To elucidate the precise function of the state representations and comprehend the rationale behind their superior performance, we meticulously analyze the state representation functions produced by LLM. We use the `Swimmer` task from Mujoco as an example for semantic analysis. Please refer to Appendix E.5 for details about the state representation functions generated by LLM.

Semantic Analysis In the case of the `Swimmer` task, the goal is to move as fast as possible towards the right by apply-

Table 2. Final Performance on Mujoco and Gym-robotics environments over **1M** environment interaction training steps. **Ours w/o IR:** without intrinsic reward. **Ours w/o SR:** without state representation. **Ours w/o FB:** without feedback. More details about the ablation study are elucidated in Section 4.4. **RPI =** `(accumulated_rewards - baseline)/baseline`: relative performance improvement. **PI =** `success_rate - baseline`: success rate improvement. This distinction arises because the performance is represented by the accumulative rewards in Mujoco, whereas in Gym-Robotics, it is measured by the success rate. The "mean±std" denotes values computed across five random seeds.

Environments \ Algorithm	TD3	EUREKA	RPI	Ours w/o IR	RPI	Ours w/o SR	RPI	Ours w/o FB	RPI	LESR(Ours)	RPI
HalfCheetah	9680.2±1555.8	10400.6±289.2	7%	9969.8±1767.5	3%	9463.2±796.3	-2%	9770.4±1531.3	1%	**10614.2±510.8**	**10%**
Hopper	3193.9±507.8	3346.4±423.2	5%	3324.7±191.7	4%	3159.0±466.7	-1%	2851.3±748.1	-11%	**3424.8±143.7**	**7%**
Walker2d	3952.1±445.7	3606.2±1010.0	-9%	4148.2±352.9	5%	3977.4±434.3	1%	4204.9±590.3	6%	**4433.0±435.3**	**12%**
Ant	3532.6±1265.3	2577.7±1085.6	-27%	**5359.0±336.4**	**52%**	3962.2±1332.0	12%	4244.9±1227.9	20%	4343.4±1171.4	23%
Swimmer	84.9±34.0	98.1±31.1	16%	132.0±6.0	55%	160.2±10.2	89%	85.4±29.8	1%	**164.2±7.6**	**93%**
Mujoco Relative Improve Mean	-	-	-2%	-	24%	-	21%	-	3%	-	**29%**
-	-	-	**PI**	-	**PI**	-	**PI**	-	**PI**	-	**PI**
AntMaze_Open	0.0±0.0	0.13±0.04	13%	0.0±0.0	0%	0.15±0.06	15%	0.16±0.07	16%	**0.17±0.06**	**17%**
AntMaze_Medium	0.0±0.0	0.01±0.01	1%	0.0±0.0	0%	0.07±0.05	7%	0.0±0.0	0%	**0.1±0.05**	**10%**
AntMaze_Large	0.0±0.0	0.0±0.0	0%	0.0±0.0	0%	0.06±0.03	6%	0.04±0.04	4%	0.07±0.03	7%
FetchPush	0.07±0.02	0.07±0.03	0%	0.78±0.16	71%	0.06±0.05	-1%	0.77±0.14	70%	**0.91±0.08**	**84%**
AdroitHandDoor	0.53±0.44	0.83±0.08	30%	0.46±0.46	-7%	0.63±0.39	10%	0.23±0.38	-30%	**0.88±0.12**	**34%**
AdroitHandHammer	0.28±0.32	0.32±0.45	4%	0.22±0.21	-6%	0.29±0.28	1%	0.41±0.33	13%	**0.53±0.38**	**25%**
Gym-Robotics Improve Mean	-	-	8%	-	8%	-	6%	-	10%	-	**30%**

Table 3. Gym-Robotics Results. **PPO** and **SAC** denote algorithms trained using the original state and reward functions within the tasks. **+ Ours** signifies training utilizing state representation functions and the corresponding intrinsic reward functions. **PI**: performance improvement.

Environments \ Algorithm	PPO	PPO + Ours	PI	SAC	SAC + Ours	PI
AntMaze_Open	0.006±0.005	**0.12±0.022**	**11%**	0.002±0.001	**0.12±0.033**	**12%**
AntMaze_Medium	0.0±0.0	**0.156±0.039**	**16%**	0.008±0.003	**0.108±0.03**	**10%**
AntMaze_Large	0.006±0.004	0.092±0.02	9%	0.009±0.003	**0.124±0.029**	**12%**
FetchPush	0.063±0.017	**0.812±0.13**	**75%**	0.059±0.015	**0.589±0.207**	**53%**
AdroitHandDoor	0.001±0.001	**0.948±0.047**	**95%**	0.759±0.158	**0.956±0.035**	**20%**
AdroitHandHammer	0.006±0.002	0.007±0.002	0%	0.752±0.158	**0.892±0.084**	**14%**
Gym-Robotics Improve Mean	-	-	**34%**	-	-	**20%**

Table 4. Consistency verification in `Swimmer`. $seed_i$ denotes the experiments over four random seeds. c_i signifies the state representation categories. ✓signifies the inclusion of a particular category in the corresponding seed.

	$seed_1$	$seed_2$	$seed_3$	$seed_4$
c_1	✓	✓	✓	✓
c_2	✓	✓	✓	✓
c_3	✓	✓	✓	✓
c_4		✓	✓	✓
c_5		✓		

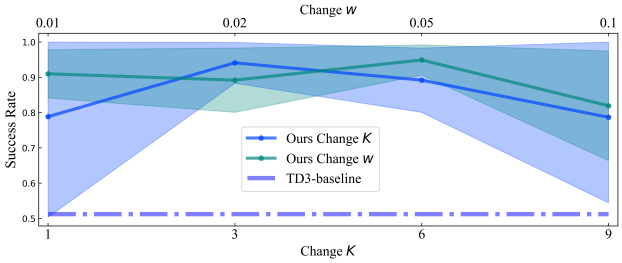


Figure 5. Experimental results for hyperparameter variations in `AdroitHandDoor`. The sample count K is adjusted to [1, 3, 6, 9] (bottom x-axis), and the intrinsic reward weight w is modified to [0.01, 0.02, 0.05, 0.1] (top x-axis).

ing torque on the rotors and using the fluids friction, without taking any drastic action. The state representations in this task can be categorized into five groups, whose names are derived from the comments provided by LLM: *cosine or sine of the angles (c1)*, *relative angles between adjacent links (c2)*, *kinetic energy (c3)*, *distance moved (c4)*, and *sum of torques applied (c5)*. It is apparent that these representations exhibit a strong correlation with the task's objective: *c1* and *c2* contribute to enhanced understanding of posture, *c3* and *c5* signify the necessity to avoid excessive actions, while *c4* corresponds directly to the objective of advancing towards the target.

Consistency of LLM Through semantic analysis, it is demonstrable that LLM is capable of generating significantly task-related state representations of a physically meaningful nature, stemming from the source state. Then, how consistent are the answers from LLM? We carried out experiments with four random seeds for the `Swimmer` task, and took out the final state representation functions in each experiment for statistics. As demonstrated in Table 4, the functions generated by LLM exhibit a pronounced level of consistency across diverse experiments, e.g. c_1, c_2 and c_3 are included in all experiments. This reaffirms the importance and universality of the extended state while ensuring the robustness and stability of our methodology.

4.7. Robustness

To validate the robustness and stability of varying hyperparameters, we conducted experiments in the `AdroitHandDoor` environment. We systematically altered the values of two key hyperparameters: the weight of the intrinsic reward w in Equation (2) and the sample count K. For each experiment, we modified one hyperpa-

rameter while keeping the others. The outcomes of these experiments are illustrated in Figure 5.

The results indicate that an increase in the sample count K can lead to a performance improvement, for instance, from $K = 1$ to $K = 3$. However, elevating the value of K from 3 to 9 does not yield further performance enhancement. Regarding the variation in the weight of the intrinsic reward w, the findings illustrate that the final performance is not significantly affected by changes in w and remains stable. In summary, the final performance across all hyperparameter tuning experiments remains consistently within a stable range, significantly surpassing the baseline. This observation underscores the stability of our approach.

5. Conclusion

In this paper, we introduce **LESR**, an algorithm leveraging the coding proficiency and interpretive capacity for physical mechanism of LLM to generate state representation and intrinsic reward function codes for reinforcement learning. Demonstrating its efficacy on benchmarks Mujoco and Gym-Robotics, we illustrate LLM's capability to produce task-specific state representations alongside meaningful intrinsic reward functions. These representations enhance Lipschitz continuity in networks, resulting in superior efficiency and outperforming SOTA baselines. In-depth ablations and additional experiments show the consistency and robustness of LESR. We believe that LESR can effectively contribute to various real-world interaction tasks.

However, our work still suffers limitations. A primary constraint lies in our attempt to derive task-related representations solely from the source state features using LLM, without incorporating external information. This approach may restrict the information available to the network in partially observable environments. Besides, the quality of state representations generated by LLM is constrained by its capabilities, and there is no absolute guarantee that it can produce more task-related representations. This limitation is anticipated to be mitigated as LLM evolves. In the future, we are interested in exploring the integration of additional information to establish a more comprehensive framework for state representations. We anticipate that our work may serve as an inspiration for further exploration in this promising area among researchers.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgment

This work was supported by the National Key R&D Program of China under Grant 2018AAA0102801.

References

Agashe, S., Fan, Y., and Wang, X. E. Evaluating multi-agent coordination abilities in large language models. *arXiv preprint arXiv:2310.03903*, 2023.

Anil, C., Lucas, J., and Grosse, R. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pp. 291–301. PMLR, 2019.

Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*, 2023.

Chen, S., Xiao, A., and Hsu, D. Llm-state: Expandable state representation for long-horizon task planning in the open world, 2023.

Coulom, R. *Reinforcement learning using neural networks, with applications to motor control*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2002.

Da, L., Gao, M., Mei, H., and Wei, H. Llm powered sim-to-real transfer for traffic signal control. *arXiv preprint arXiv:2308.14284*, 2023.

de Lazcano, R., Andreas, K., Tai, J. J., Lee, S. R., and Terry, J. Gymnasium robotics, 2023.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. Guiding pretraining in reinforcement learning with large language models, 2023.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Erez, T., Tassa, Y., and Todorov, E. Infinite horizon model predictive control for nonlinear periodic tasks. *Manuscript under review*, 4, 2011.

Farahmand, A.-m., Szepesvári, C., and Munos, R. Error propagation for approximate policy and value iteration. *Advances in Neural Information Processing Systems*, 23, 2010.

Farahmand, A.-m., Barreto, A., and Nikovski, D. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Feng, Y., Wang, Y., Liu, J., Zheng, S., and Lu, Z. Llama rider: Spurring large language models to explore the open world. *arXiv preprint arXiv:2310.08922*, 2023.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Jones, D. R., Perttunen, C. D., and Stuckman, B. E. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79:157–181, 1993.

Kim, W., Son, B., and Kim, I. Vilt: Vision-and-language transformer without convolution or region supervision. In *International conference on machine learning*, pp. 5583–5594. PMLR, 2021.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Tu, Z., and Shi, S. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

Liu, B., Pu, Z., Pan, Y., Yi, J., Liang, Y., and Zhang, D. Lazy agents: a new perspective on solving sparse reward problem in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 21937–21950. PMLR, 2023.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Maei, H., Szepesvari, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. Convergent temporal-difference learning with arbitrary smooth function approximation. *Advances in neural information processing systems*, 22, 2009.

Mao, Y., Zhang, H., Chen, C., Xu, Y., and Ji, X. Supported trust region optimization for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 23829–23851. PMLR, 2023.

Mao, Y., Zhang, H., Chen, C., Xu, Y., and Ji, X. Supported value regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Merckling, A., Coninx, A., Cressot, L., Doncieux, S., and Perrin, N. State representation learning from demonstration. In *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part II 6*, pp. 304–315. Springer, 2020.

Merckling, A., Perrin-Gilbert, N., Coninx, A., and Doncieux, S. Exploratory state representation learning. *Frontiers in Robotics and AI*, 9:762051, 2022.

Miao, N., Teh, Y. W., and Rainforth, T. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Niv, Y. Learning task-state representations. *Nature neuroscience*, 22(10):1544–1553, 2019.

Oberman, A. M. and Calder, J. Lipschitz regularized deep neural networks converge and generalize. *arXiv preprint arXiv:1808.09540*, 2018.

Ohnishi, S., Uchibe, E., Yamaguchi, Y., Nakanishi, K., Yasui, Y., and Ishii, S. Constrained deep q-learning gradually approaching ordinary q-learning. *Frontiers in neurorobotics*, 13:103, 2019.

OpenAI. Gpt-4 technical report, 2023.

Ota, K., Oiki, T., Jha, D., Mariyama, T., and Nikovski, D. Can increasing input dimensionality improve deep reinforcement learning? In *International conference on machine learning*, pp. 7424–7433. PMLR, 2020.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Qiu, J., Xu, M., Han, W., Moon, S., and Zhao, D. Embodied executable policy learning with language-based scene summarization. *arXiv preprint arXiv:2306.05696*, 2023.

Qu, Y., Wang, B., Shao, J., Jiang, Y., Chen, C., Ye, Z., Linc, L., Feng, Y., Lai, L., Qin, H., et al. Hokoff: Real game dataset from honor of kings and its offline reinforcement learning benchmarks. *Advances in Neural Information Processing Systems*, 36, 2024.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. Vision-language models are zero-shot reward models for reinforcement learning, 2023.

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Schuck, N. W., Wilson, R., and Niv, Y. A state representation for reinforcement learning and decision-making in the orbitofrontal cortex. In *Goal-directed decision making*, pp. 259–278. Elsevier, 2018.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, J., Qu, Y., Chen, C., Zhang, H., and Ji, X. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Shek, C. L., Wu, X., Manocha, D., Tokekar, P., and Bedi, A. S. Lancar: Leveraging language for context-aware robot locomotion in unstructured environments. *arXiv preprint arXiv:2310.00481*, 2023.

Shen, B., Zhang, J., Chen, T., Zan, D., Geng, B., Fu, A., Zeng, M., Yu, A., Ji, J., Zhao, J., et al. Pangu-coder2: Boosting large language models for code with ranking feedback. *arXiv preprint arXiv:2307.14936*, 2023.

Shi, R., Liu, Y., Ze, Y., Du, S. S., and Xu, H. Unleashing the power of pre-trained language models for offline reinforcement learning, 2023.

Shukla, Y., Gao, W., Sarathy, V., Velasquez, A., Wright, R., and Sinapov, J. Lgts: Dynamic task sampling using llm-generated sub-goals for reinforcement learning agents. *arXiv preprint arXiv:2310.09454*, 2023.

Sodhani, S., Zhang, A., and Pineau, J. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pp. 9767–9779. PMLR, 2021.

Song, J., Zhou, Z., Liu, J., Fang, C., Shu, Z., and Ma, L. Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics. *arXiv preprint arXiv:2309.06687*, 2023.

Sontakke, S. A., Zhang, J., Arnold, S. M., Pertsch, K., Bıyık, E., Sadigh, D., Finn, C., and Itti, L. Roboclip: one demonstration is enough to learn robot policies. *arXiv preprint arXiv:2310.07899*, 2023.

Stremmel, J., Saeedi, A., Hassanzadeh, H., Batra, S., Hertzberg, J., Murillo, J., and Halperin, E. Xaiqa: Explainer-based data augmentation for extractive question answering, 2023.

Sun, J., Zhang, Q., Duan, Y., Jiang, X., Cheng, C., and Xu, R. Prompt, plan, perform: Llm-based humanoid control via quantized imitation learning. *arXiv preprint arXiv:2309.11359*, 2023.

Tesauro, G. et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Triantafyllidis, E., Christianos, F., and Li, Z. Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks. *arXiv preprint arXiv:2309.16347*, 2023.

Trott, A., Zheng, S., Xiong, C., and Socher, R. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

Wawrzyński, P. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pp. 380–390. Springer, 2009.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Wu, Y., Min, S. Y., Prabhumoye, S., Bisk, Y., Salakhutdinov, R., Azaria, A., Mitchell, T., and Li, Y. Spring: Gpt-4 outperforms rl algorithms by studying papers and reasoning. *arXiv preprint arXiv:2305.15486*, 2023.

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

Yan, X., Song, Y., Cui, X., Christianos, F., Zhang, H., Mguni, D. H., and Wang, J. Ask more, know better: Reinforce-learned prompt questions for decision making with large language models, 2023.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Yoo, M., Cho, S., and Woo, H. Skills regularized task decomposition for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 37432–37444, 2022.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., and Lu, Z. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.

Zhang, J., Zhang, J., Pertsch, K., Liu, Z., Ren, X., Chang, M., Sun, S.-H., and Lim, J. J. Bootstrap your own skills: Learning to solve new tasks with large language model guidance, 2023.

Zhang, W. and Lu, Z. Rladapter: Bridging large language models to reinforcement learning in open worlds. *arXiv preprint arXiv:2309.17176*, 2023.

Zheng, R., Wang, X., Xu, H., and Huang, F. Is model ensemble necessary? model-based rl via a single model with lipschitz regularized value function. *arXiv preprint arXiv:2302.01244*, 2023.

A. Theoretical Analysis

Definition A.1. We firstly define the identity projection and closet point projection map $\text{Id}: \mathcal{X} \to \mathcal{X}, \sigma_{\mathcal{X}_k}: \mathcal{X} \to \mathcal{X}_k = \{x_1, \ldots, x_n\}$ that satisfies

$$\begin{aligned} \forall x \in \mathcal{X}, \text{Id}(x) = x \\ \forall x \in \mathcal{X}, \|x - \sigma_{\mathcal{X}_k}\|_2 = \min_{1 \le i \le n} \{\|x - x_i\|_2\} \end{aligned} \tag{8}$$

Now we investigate the convergence under the mapping f in Theorem 3.4:

Theorem A.2 (Convergence)**.** *If $u_1 \in \mathcal{U}_1$ is any minimizer of $\mathcal{L}(u, \mathcal{X}_1)$, where $\mathcal{U}_1$ denotes the solution set of $\mathcal{L}(u, \mathcal{X}_1)$ in Definition 3.2, then for any $t > 0$, there exists $C \ge 1$ and $0 < c < 1$ satisfies the following with probability at least $1 - Ct^{-1}N^{-(ct-1)}$:*

$$\mathbb{E}_{x \sim \rho_f} \|u_1^* - u_1\|_2 \le C \Big[\text{Lip}(u_1^*) + \text{Lip}(u_1) \Big] \left(\frac{t \log N}{N} \right)^{1/d} \tag{9}$$

Proof of Theorem A.2. Since $u_1 \in \mathcal{U}_1$ is a minimizer of $\mathcal{L}(u, \mathcal{X}_1)$, we must have $u_1(x_i) = u_1^*(x_i)$ for all $1 \le i \le N$. Then for any $x \in X$ we have

$$\begin{aligned} \|u_1^*(x) - u_1(x)\|_2 &= \|u_1^*(x) - u_1^*(\sigma_{\mathcal{X}_1}(x)) + \underbrace{u_1^*(\sigma_{\mathcal{X}_1}(x)) - u_1(\sigma_{\mathcal{X}_1}(x))}_{=0} + u_1(\sigma_{\mathcal{X}_1}(x)) - u_1(x)\|_2 \\ &\le \|u_1^*(x) - u_1^*(\sigma_{\mathcal{X}_1}(x))\|_2 + \|u_1(\sigma_{\mathcal{X}_1}(x)) - u_1(x)\|_2 \\ &\le \Big[\text{Lip}(u_1^*) + \text{Lip}(u_1) \Big] \|x - \sigma_{\mathcal{X}_1}(x)\|_2 \end{aligned} \tag{10}$$

Now we provide a bound between the identity projection and closet point projection:

Lemma A.3 (Lemma 2.9. in (Oberman & Calder, 2018))**.** *For any $t > 0$, the following holds with probability at least $1 - Ct^{-1}N^{-(ct-1)}$:*

$$\mathbb{E}_{x \sim \rho_f} \|x - \sigma_{\mathcal{X}_1}(x)\|_2 \le C \left(\frac{t \log N}{N} \right)^{1/d} \tag{11}$$

The proof is completed by combining Lemma A.3 into Eq 10. □

Next, we can prove that the generalization loss converges based on Theorem A.2:

Theorem A.4. *Assume that for some $q \ge 1$ the loss ℓ in Definition 3.2 satisfies $\ell(y_i, y_k) \le C\|y_i - y_k\|_2^q$ for all $y_i, y_k \in \mathcal{Y}$. Then under Theorem A.2, the following bound of the loss $\mathcal{L}[u_1, \mathcal{X}]$ in Definition 3.2 holds with probability at least $1 - Ct^{-1}N^{-(ct-1)}$:*

$$\mathcal{L}[u_1, \mathcal{X}] \le C \Big[\text{Lip}(u_1^*) + \text{Lip}(u_1) \Big]^q \left(\frac{t \log N}{N} \right)^{q/d} \tag{12}$$

Proof of Theorem A.4. We can bound the loss as:

$$\mathcal{L}[u_1, \mathcal{X}] = \int_{x \in \mathcal{X}} \rho_f(x) \ell\Big(u_1^*(x), u_1(x)\Big) dx \le C \mathbb{E}_{x \sim \rho_f} \|u_1^* - u_1\|_2^q \tag{13}$$

The proof is completed by combining Theorem A.2 into Eq 13. □

Now we turn to the proof of Theorem 3.4, we start with the following lemma:

Lemma A.5. *Under Assumption 3.3, ρ and ρ_f denote the source probability distribution on $\mathcal{X}$ and probability distribution on $f(\mathcal{X})$:*

$$\mathbb{E}_{x \sim \rho_f} \|x - \sigma_{\mathcal{X}_1}(x)\|_2 = \mathbb{E}_{x \sim \rho} \|x - \sigma_{\mathcal{X}_0}(x)\|_2 \tag{14}$$

Proof of Lemma A.5. Let $\mathcal{X}_2 = \{x | x \in \mathcal{X}, x \notin \mathcal{X}_0\}$, from Definition A.1, when $x \in \mathcal{X}_0$, $\sigma_{\mathcal{X}_0}(x) = x$, then we have:

$$\begin{aligned}\int_{x\in\mathcal{X}} \rho(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx &= \int_{x\in\mathcal{X}_2} \rho(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx + \underbrace{\int_{x\in\mathcal{X}_0} \rho(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx}_{=0} \\ &= \int_{x\in\mathcal{X}_2} \rho(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx\end{aligned} \tag{15}$$

Consider the mapping $f : \mathcal{X} \to \mathcal{X}$ only swaps the order of x in the dataset $\mathcal{X}_0$, which means:

$$\begin{gathered} f(x) = \begin{cases} x' \,\big|\, x, x' \in \mathcal{X}_0 \\ x \,\big|\, x \in \mathcal{X}_2 \end{cases} \\ \mathcal{X}_1 = \{f(x_i) | f(x_i) \in \mathcal{X}_0, i = 1, \ldots, N\} \text{ and } \mathcal{X}_0 = \mathcal{X}_1 \end{gathered} \tag{16}$$

Therefore we get $\forall x \in \mathcal{X}_2, \rho_f(x) = \rho(x)$. Hence:

$$\begin{aligned}\mathbb{E}_{x\sim\rho}\|x - \sigma_{\mathcal{X}_0}(x)\|_2 &= \int_{x\in\mathcal{X}_2} \rho(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx \\ &= \int_{x\in\mathcal{X}_2} \rho_f(x)\|x - \sigma_{\mathcal{X}_0}(x)\|_2\, dx = \mathbb{E}_{x\sim\rho_f}\|x - \sigma_{\mathcal{X}_1}(x)\|_2\end{aligned} \tag{17}$$

□

Proof of Theorem 3.4. Since $u_0 \in \mathcal{U}_0$ is a minimizer of $\mathcal{L}(u, \mathcal{X}_0)$, we must have $u_0(x_i) = u_0^*(x_i)$ for all $1 \le i \le n$. Then for any $x \in X$ we have

$$\begin{aligned}\|u_0^*(x) - u_0(x)\|_2 &= \|u_0^*(x) - u_0^*(\sigma_{\mathcal{X}_0}(x)) + u_0^*(\sigma_{\mathcal{X}_0}(x)) - u_0(\sigma_{\mathcal{X}_0}(x)) + u_0(\sigma_{\mathcal{X}_0}(x)) - u_0(x)\|_2 \\ &\le \|u_0^*(x) - u_0^*(\sigma_{\mathcal{X}_0}(x))\|_2 + \|u_0(\sigma_{\mathcal{X}_0}(x)) - u_0(x)\|_2 \\ &\le \Big[\mathrm{Lip}(u_0^*) + \mathrm{Lip}(u_0)\Big]\|x - \sigma_{\mathcal{X}_0}(x)\|_2\end{aligned} \tag{18}$$

Therefore, combined with Theorem A.2, we have:

$$\begin{aligned}\sup_{u_1\in\mathcal{U}_1} \mathbb{E}_{x\sim\rho_f}\|u_1^* - u_1\|_2 &= \Big[\mathrm{Lip}(u_1^*) + \mathrm{Lip}(u_1)\Big]\mathbb{E}_{x\sim\rho}\|x - \sigma_{\mathcal{X}_1}(x)\|_2 \\ &\le \Big[\mathrm{Lip}(u_0^*) + \mathrm{Lip}(u_0)\Big]\mathbb{E}_{x\sim\rho_f}\|x - \sigma_{\mathcal{X}_1}(x)\|_2 \\ &= \sup_{u_0\in\mathcal{U}_0} \mathbb{E}_{x\sim\rho}\|u_0^* - u_0\|_2 \;\; \text{(invoking Assumption 3.3, Lemma A.5)}\end{aligned} \tag{19}$$

□

B. Why Lipschitz constant is crucial for RL

In this section we main focus on the relationship between the Lipschitz constant of the reward function and the continuity of the value function in RL. Firstly we make some assumptions.

Definition B.1. In reinforcement learning, given a policy π, γ denotes the discounted factor, r ($r : s \to \mathbb{R}$) denotes the reward function, H denotes the length of trajectory, the definition of the value function $V(s)$ is:

$$V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{H} \gamma^t r | s_0 = s, a_t \sim \pi(s_t)\Big]. \tag{20}$$

Similar to Farahmand et al. (2017), we make the deterministic assumption of the environment and RL policy. Besides, the policy of RL algorithm TD3 utilized in our method is also deterministic.

Assumption B.2. The environment transition and policy are deterministic.

We also make the assumptions of the Lipschitz constant of reward function and environment dynamic transition which is similar to previous work (Asadi et al., 2018).

Assumption B.3. There exists constants K_1, K_2 such that the lipschitz constant of reward function r ($r : s \to \mathbb{R}$) is K_1, where $s \in \mathcal{S}$ is the state. In other words, $\mathrm{Lip}(r) = K_1$, and $\mathcal{P}(s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the environment dynamic transition function, K_2 that satisfies: $\mathrm{Lip}(\mathcal{P}) = K_2$.

• **Relationship between** $\mathrm{Lip}(r; \mathcal{S})$ **and** $\mathrm{Lip}(V; \mathcal{S})$

Drawing upon the aforementioned definitions and assumptions, let us commence our analysis:

Theorem B.4. *Under Assumption B.2, B.3, given a RL policy π, the value function of π, namely V^π, satisfies:*

$$\forall s_1, s_2 \in \mathcal{S}, \quad \|V^\pi(s_1) - V^\pi(s_2)\| \leq \frac{[1 - (\gamma K_2)^H] K_1}{1 - \gamma K_2} \|s_1 - s_2\|. \tag{21}$$

Proof of Theorem B.4. Because the policy π and environment dynamic transition function $\mathcal{P}$ are deterministic, for $\forall s_1, s_2 \in \mathcal{S}$, two trajectories τ_1, τ_2 can be generated:

$$\begin{aligned} \tau_1 &= \{s_{1,i} | t = 0, \ldots, H \;\; s_{1,0} = s_1, \;\; s_{1,t+1} = \mathcal{P}(s_{1,t}, \pi(s_{1,t}))\}, \\ \tau_2 &= \{s_{2,i} | t = 0, \ldots, H \;\; s_{2,0} = s_2, \;\; s_{2,t+1} = \mathcal{P}(s_{2,t}, \pi(s_{2,t}))\}. \end{aligned} \tag{22}$$

Under Definition B.1:

$$\begin{aligned} \left\| V^\pi(s_1) - V^\pi(s_2) \right\| &= \left\| \sum_{t=0}^{H} \gamma^t r(s_{1,t}) - \sum_{t=0}^{H} \gamma^t r(s_{2,t}) \right\| \\ &= \left\| \sum_{t=0}^{H} \gamma^t \big[r(s_{1,t}) - r(s_{2,t}) \big] \right\| \\ &\leq \sum_{t=0}^{H} \gamma^t \left\| \big[r(s_{1,t}) - r(s_{2,t}) \big] \right\| \quad \text{\# invoking Assumption B.3} \\ &\leq \sum_{t=0}^{H} \gamma^t K_1 \big\| s_{1,t} - s_{2,t} \big\|. \end{aligned} \tag{23}$$

The state dynamic distances $\big\| s_{1,t} - s_{2,t} \big\|$ can be estimated as follows:

$$\big\| s_{1,t} - s_{2,t} \big\| \leq K_2 \big\| s_{1,t-1} - s_{2,t-1} \big\| \leq K_2^2 \big\| s_{1,t-2} - s_{2,t-2} \big\| \cdots \leq K_2^t \big\| s_1 - s_2 \big\|. \tag{24}$$

Combined Equation 24 with 23:

$$\left\| V^\pi(s_1) - V^\pi(s_2) \right\| \leq \sum_{t=0}^{H} \big(\gamma K_2\big)^t K_1 \big\| s_1 - s_2 \big\| = \frac{[1 - (\gamma K_2)^H] K_1}{1 - \gamma K_2} \|s_1 - s_2\|. \tag{25}$$

□

Proof of Theorem B.4 is completed. A direct corollary from Theorem B.4 is:

Corollary B.5.

$$\mathrm{Lip}(V^\pi; \mathcal{S}) \leq \frac{[1 - (\gamma K_2)^H] K_1}{1 - \gamma K_2}. \tag{26}$$

Corollary B.5 demonstrates that reducing the Lipschitz constant K_1 of the reward function directly decreases the upper bound of the Lipschitz constant of value functions in RL. This confirms that our approach effectively improves the continuity of the value functions.

• **How does $\mathrm{Lip}(V;\mathcal{S})$ work for RL algorithms?**

We provide additional definitions to aid in understanding the subsequent theorems. Inspired by Farahmand et al. (2017); Zheng et al. (2023); Farahmand et al. (2010); Asadi et al. (2018):

Definition B.6. Denote environment dynamic transition function $\mathcal{P}$ defined in B.4. For a given value function V, the Bellman operator as $\mathcal{T}$, policy transition matrix as $P^\pi : \mathcal{S}\times\mathcal{S}\to\mathbb{R}$, the greedy policy π_g of V, and the optimal policy π^* and optimal value funtion V^* are defined as follows:

$$\begin{aligned}
&\mathcal{T}V(s) = r(s) + \max_a \mathcal{P}(s,a)V(\mathcal{P}(s,a)),\\
&\mathcal{T}^\pi V = r + \gamma P^\pi V,\\
&P^\pi(s_1,s_2) = \begin{cases} 0, & s_2 \neq \mathcal{P}(s_1,\pi(s_1)) \\ 1, & s_2 = \mathcal{P}(s_1,\pi(s_1)) \end{cases},\\
&\forall s\in\mathcal{S},\ \ \pi_g = \arg\max_a \Big[r(s) + \mathcal{P}(s,a)V(\mathcal{P}(s,a))\Big],\\
&\forall s\in\mathcal{S},\ \ V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s).
\end{aligned} \tag{27}$$

Definition B.7. Denote $\pi_1,\pi_2,\ldots,\pi_m$ as any m policies arranged in any order, $\forall m\geq 1$, u and v are two state distributions, define $C(u)\in\mathbb{R}^+\cup+\infty$, $c(m)\in\mathbb{R}^+\cup+\infty$ and $c(0)=1$:

$$\begin{aligned}
c(m) &= \max_{\pi_1,\pi_2,\ldots\pi_m,s\in\mathcal{S}} \frac{\Big(vP^{\pi_1}P^{\pi_2}\ldots P^{\pi_m}\Big)(s)}{u(s)},\\
C_1(v,u) &= (1-\gamma)\sum_{m\geq 0}\gamma^m c(m).
\end{aligned} \tag{28}$$

$c(m)$, as defined in Definition B.7, quantifies the discrepancy between transitions of m policies from an initial state distribution v over m time steps and a target distribution u, while $C_1(v,u)$ quantifies the discounted discrepancy.

Theorem B.8. *Denote π_g as the greedy policy of a given value function V, V is the value function of some policy π_0. u and v are two state distributions. The norm $\|V\|_{q,u}$ is the q-norm of V weighted by u which is defined as $\|V\|_{q,u} = \big[\sum_s u(s)\big(V(s)\big)^q\big]^{\frac{1}{q}}$. Suppose $\forall s_1,s_2\in\mathcal{S}$, there exists a constant D_1 that satisfies $\|s_1-s_2\|\leq D_1$ and under assumption B.2, B.3:*

$$||V^* - V^{\pi_g}||_{q,v} \leq \frac{2\gamma K_1 D_1\big(1-(\gamma K_2)^H\big)}{(1-\gamma)(1-\gamma K_2)}\big[C_1(v,u)\big]^{\frac{1}{q}}. \tag{29}$$

Proof of Theorem B.8. We first present two lemmas that aid in the subsequent proof.

Lemma B.9. *Denote matrix I as the identity matrix(here $V_1\preceq V_2$ means $\forall s\in\mathcal{S}, V_1(s)\leq V_2(s)$):*

$$V^* - V^{\pi_g} \preceq \Big[(I-\gamma P^{\pi^*})^{-1} + (I-\gamma P^{\pi_g})^{-1}\Big]|\mathcal{T}V - V|. \tag{30}$$

Proof of Lemma B.9. It can be derived that $\mathcal{T}^{\pi_g}V = \mathcal{T}V$, $\mathcal{T}^{\pi_g}V^{\pi_g} = V^{\pi_g}$:

$$\begin{aligned}
V^* - V^{\pi_g} &= \mathcal{T}^{\pi^*}V^* - \mathcal{T}^{\pi^*}V + \underbrace{\mathcal{T}^{\pi^*}V - \mathcal{T}^{\pi_g}V}_{\preceq\mathbf{0}} + \mathcal{T}^{\pi_g}V - \mathcal{T}^{\pi_g}V^{\pi_g}\\
&\preceq \mathcal{T}^{\pi^*}V^* - \mathcal{T}^{\pi^*}V + \mathcal{T}^{\pi_g}V - \mathcal{T}^{\pi_g}V^{\pi_g}\\
&= \gamma P^{\pi^*}(V^* - V^{\pi_g} + V^{\pi_g} - V) + \gamma P^{\pi_g}(V - V^{\pi_g}).\\
&\Rightarrow (I-\gamma P^{\pi^*})(V^* - V^{\pi_g}) \preceq \gamma(P^{\pi^*} - P^{\pi_g})(V^{\pi_g} - V).\\
&\Rightarrow V^* - V^{\pi_g} \preceq (I-\gamma P^{\pi^*})^{-1}\gamma(P^{\pi^*} - P^{\pi_g})(V^{\pi_g} - V).
\end{aligned} \tag{31}$$

$$
\begin{aligned}
(I-\gamma P^{\pi_g})(V^{\pi_g}-V) &= V^{\pi_g}-V-\gamma P^{\pi_g}V^{\pi_g}+\gamma P^{\pi_g}V \\
&= r+\gamma P^{\pi_g}V-\Big(r+\gamma P^{\pi_g}V^{\pi_g}\Big)+V^{\pi_g}-V \\
&= \mathcal{T}^{\pi_g}V-\mathcal{T}^{\pi_g}V^{\pi_g}+V^{\pi_g}-V \\
&= \mathcal{T}V-V.
\end{aligned} \tag{32}
$$

Combine Inequation 31 and Equation 32:

$$
\begin{aligned}
V^*-V^{\pi_g} &\preceq (I-\gamma P^{\pi^*})^{-1}\gamma(P^{\pi^*}-P^{\pi_g})(V^{\pi_g}-V) \\
&= (I-\gamma P^{\pi^*})^{-1}\gamma(P^{\pi^*}-P^{\pi_g})(I-\gamma P^{\pi_g})^{-1}(\mathcal{T}V-V) \\
&= (I-\gamma P^{\pi^*})^{-1}\Big[(I-\gamma P^{\pi_g})-(I-\gamma P^{\pi^*})\Big](I-\gamma P^{\pi_g})^{-1}(\mathcal{T}V-V) \\
&= \Big[(I-\gamma P^{\pi^*})^{-1}-(I-\gamma P^{\pi_g})^{-1}\Big](\mathcal{T}V-V) \\
&\preceq \Big[(I-\gamma P^{\pi^*})^{-1}+(I-\gamma P^{\pi_g})^{-1}\Big]|\mathcal{T}V-V|.
\end{aligned} \tag{33}
$$

Proof of Lemma B.9 is completed. □

Lemma B.10. *For a given policy π:*

$$
(I-\gamma P^{\pi})^{-1}=\sum_{k=0}^{\infty}\Big(\gamma P^{\pi}\Big)^k. \tag{34}
$$

Proof of Lemma B.10.

$$
\begin{aligned}
(I-\gamma P^{\pi})\sum_{k=0}^{\infty}\Big(\gamma P^{\pi}\Big)^k &= \sum_{k=0}^{\infty}\Big[\Big(\gamma P^{\pi}\Big)^k-\Big(\gamma P^{\pi}\Big)^{k+1}\Big] \\
&= I-\lim_{k\to\infty}\Big(\gamma P^{\pi}\Big)^{k+1}=I.
\end{aligned} \tag{35}
$$

Proof of Lemma B.10 is completed. □

Now we turn to the proof of Theorem B.8, we rewrite Lemma B.9 as follows:

$$
\begin{aligned}
V^*-V^{\pi_g} &\preceq \frac{2}{1-\gamma}A|\mathcal{T}V-V| \\
A &= \frac{1-\gamma}{2}\Big[(I-\gamma P^{\pi^*})^{-1}+(I-\gamma P^{\pi_g})^{-1}\Big].
\end{aligned} \tag{36}
$$

Under the q-norm weighted by v:

$$
\begin{aligned}
\|V^*-V^{\pi_g}\|_{q,v}^q &\le [\frac{2}{1-\gamma}]^q\sum_{s\in\mathcal{S}}v(s)\big[A(s)|\mathcal{T}V(s)-V(s)|\big]^q \\
&\le [\frac{2}{1-\gamma}]^q\sum_{s\in\mathcal{S}}v(s)A(s)|\mathcal{T}V(s)-V(s)|^q.
\end{aligned} \tag{37}
$$

Equation 37 stems from Jensen's inequality, exploiting the convexity of the function $f(x)=x^q$ for $q\ge 1$. Furthermore, when combined with Lemma B.10 and Equation 36, each row of matrix A attains a sum of $\frac{1-\gamma}{2}\left[\frac{1}{1-\gamma}+\frac{1}{1-\gamma}\right]=1$, meeting the prerequisites of Jensen's inequality.

Next we focus on the $v(s)A(s)$ in Equation 37:

$$\begin{aligned} vA &= \frac{1-\gamma}{2} v\Big[(I-\gamma P^{\pi^*})^{-1} + (I-\gamma P^{\pi_g})^{-1}\Big] \\ &= \frac{1-\gamma}{2}\sum_{m=0}^{\infty}\gamma^m\Big(vP^{\pi^*} + vP^{\pi_g}\Big) \text{ \# invoking Lemma B.10} \\ &\preceq \frac{1-\gamma}{2}\Big(\sum_{m\geq 0}\gamma^m c(m)u + \sum_{m\geq 0}\gamma^m c(m)u\Big) \text{ \# under Definition B.7} \\ &= C_1(v,u)u. \end{aligned} \tag{38}$$

Now we return to Equation 37, combined with Equation 38:

$$\|V^* - V^{\pi_g}\|_{q,v}^q \leq [\frac{2}{1-\gamma}]^q C_1(v,u)\sum_{s\in\mathcal{S}} u(s)|\mathcal{T}V(s) - V(s)|^q. \tag{39}$$

In the description of Theorem B.8, Because π_g is denoted as the greedy policy of a given value function V, V is the value function of some policy π_0. Therefore under assumption B.2: $\mathcal{T}V(s) - V(s) = r(s) + \gamma V(s_1) - r(s) - \gamma V(s_2) = \gamma(V(s_1) - V(s_2))$, $s_1 = \mathcal{P}(s, \pi_g(s)), s_2 = \mathcal{P}(s, \pi_0(s))$.

Continuing Equation 39:

$$\begin{aligned} \|V^* - V^{\pi_g}\|_{q,v}^q &\leq [\frac{2}{1-\gamma}]^q C_1(v,u)\sum_{s\in\mathcal{S}} u(s)|\mathcal{T}V(s) - V(s)|^q \\ &= [\frac{2}{1-\gamma}]^q C_1(v,u)\sum_{s\in\mathcal{S}} u(s)|\gamma(V(s_1) - V(s_2)|^q \text{ \# } (s_1 = \mathcal{P}(s,\pi_g(s)), s_2 = \mathcal{P}(s,\pi_0(s))) \\ &\leq [\frac{2}{1-\gamma}]^q C_1(v,u)\sum_{s\in\mathcal{S}} u(s)\Big[\gamma\,\mathrm{Lip}(V;\mathcal{S})\|s_1 - s_2\|\Big]^q \text{ \# invoking Corollary B.5} \\ &\leq \Big[\frac{2\gamma}{1-\gamma}\frac{\big(1-(\gamma K_2)^H\big)K_1 D_1}{1-\gamma K_2}\Big]^q C_1(v,u). \end{aligned} \tag{40}$$

Proof of Theorem B.8 is completed. □

A direct corollary from Theorem B.8 is:

Corollary B.11. *Under the setting of Theorem B.8:*

$$||V^* - V^{\pi_g}||_\infty \leq \frac{2\gamma K_1 D_1\big(1-(\gamma K_2)^H\big)}{(1-\gamma)(1-\gamma K_2)}. \tag{41}$$

C. All of Our Prompts

There are three prompt templates in total.

The initial prompt for the first iteration. This prompt is designed to use at the commencement of the first iteration, which is the letter 'p' in Algorithm 1. LLM is required to output the state representation and intrinsic reward functions in python code format. Notably, the 'task_description' and 'detail_content of each dimensions' in the prompt are derived from the official document of Mujoco[2] and Gym-Robotics[3].

Prompt for Chain-of-thought Feedback Analysis. This prompt is formulated to facilitate the examination by LLM of the outcomes from all training experiments during each iteration in a chain-of-thought process(Wei et al., 2022). This prompt corresponds to the variable $p_{feedback}$ in Algorithm 1. LLM is expected to provide suggestions about how to enhance

[2]https://www.gymlibrary.dev/environments/mujoco/

[3]https://robotics.farama.org/

the performance of the state representation function codes. It is noteworthy that the 'iteration_results' referred to in the prompt encompasses both policy performance and correlation coefficients, as elaborated in Section 3.2.

Subsequent Prompt for Later Iterations. This prompt is similar to 'The initial prompt for the first iteration'. However, the difference is that it contains the information of history iterations, as well as LLM's suggestions about how to enhance the performance of the state representation and intrinsic reward functions.

Here are the prompt templates:

Initial Prompt for the First Iteration

Revise the state representation for a reinforcement learning agent.
==========
The agent's task description is:
{task_description}
==========

The current state is represented by a {total_dim}-dimensional Python NumPy array, denoted as 's'.

Details of each dimension in the state 's' are as follows:
{detail_content}
You should design a task-related state representation based on the source {total_dim} dim to better for reinforcement training, using the detailed information mentioned above to do some caculations, and feel free to do complex caculations, and then concat them to the source state.

Besides, we want you to design an intrinsic reward function based on the revise_state python function.

That is to say, we will:
1. use your revise_state python function to get an updated state: updated_s = revise_state(s)
2. use your intrinsic reward function to get an intrinsic reward for the task: r = intrinsic_reward(updated_s)
3. to better design the intrinsic_reward, we recommond you use some source dim in the updated_s, which is between updated_s[0] and updated_s[{total_dim - 1}]
4. however, you must use the extra dim in your given revise_state python function, which is between updated_s[{total_dim}] and the end of updated_s

Your task is to create two Python functions, named 'revise_state', which takes the current state 's' as input and returns an updated state representation, and named 'intrinsic_reward', which takes the updated state 'updated_s' as input and returns an intrinsic reward. The functions should be executable and ready for integration into a reinforcement learning environment.

The goal is to better for reinforcement training. Lets think step by step. Below is an illustrative example of the expected output:

```python
import numpy as np
def revise_state(s):
# Your state revision implementation goes here
return updated_s
def intrinsic_reward(updated_s):
# Your intrinsic reward code implementation goes here
return intrinsic_reward
```

Prompt for Chain-of-thought Feedback Analysis

We have successfully trained Reinforcement Learning policy using {args.sample_count} different state revision codes and intrinsic reward function codes sampled by you, and each pair of code is associated with the training of a policy relatively.

Throughout every state revision code's training process, we monitored:
1. The final policy performance(accumulated reward).
2. Most importantly, every state revise dim's Lipschitz constant with the reward. That is to say, you can see which state revise dim is more related to the reward and which dim can contribute to enhancing the continuity of the reward function mapping.

Here are the results:
{iteration_results(performance and Lipschitz constants)}

You should analyze the results mentioned above and give suggestions about how to imporve the performace of the "state revision code".

Here are some tips for how to analyze the results:
(a) if you find a state revision code's performance is very low, then you should analyze to figure out why it fail
(b) if you find some dims' Lipschitz constant very large, you should analyze to figure out what makes it fail
(c) you should also analyze how to imporve the performace of the "state revision code" and "intrinsic reward code" later

Lets think step by step. Your solution should aim to improve the overall performance of the RL policy.

Subsequent Prompt for Later Iterations

Revise the state representation for a reinforcement learning agent.
==========
The agent's task description is:
{task_description}
==========

The current state is represented by a {total_dim}-dimensional Python NumPy array, denoted as 's'.

Details of each dimension in the state 's' are as follows:
{detail_content}
You should design a task-related state representation based on the source {total_dim} dim to better for reinforcement training, using the detailed information mentioned above to do some caculations, and feel free to do complex caculations, and then concat them to the source state.

For this problem, we have some history experience for you, here are some state revision codes we have tried in the former iterations:
{former_histoy}

Based on the former suggestions. We are seeking an improved state revision code and an improved intrinsic reward code that can enhance the model's performance on the task. The state revised code should incorporate calculations, and the results should be concatenated to the original state.

Besides, We are seeking an improved intrinsic reward code.

That is to say, we will:
1. use your revise_state python function to get an updated state: updated_s = revise_state(s)
2. use your intrinsic reward function to get an intrinsic reward for the task: r = intrinsic_reward(updated_s)
3. to better design the intrinsic_reward, we recommond you use some source dim in the updated_s, which is between updated_s[0] and updated_s[{total_dim - 1}]
4. however, you must use the extra dim in your given revise_state python function, which is between updated_s[{total_dim}] and the end of updated_s

Your task is to create two Python functions, named 'revise_state', which takes the current state 's' as input and returns an updated state representation, and named 'intrinsic_reward', which takes the updated state 'updated_s' as input and returns an intrinsic reward. The functions should be executable and ready for integration into a reinforcement learning environment.

The goal is to better for reinforcement training. Lets think step by step. Below is an illustrative example of the expected output:

```python
import numpy as np
def revise_state(s):
# Your state revision implementation goes here
return updated_s
def intrinsic_reward(updated_s):
# Your intrinsic reward code implementation goes here
return intrinsic_reward
```

D. Additional Environment Information

D.1. Mujoco Environment Tasks

HalfCheetah This environment is based on the work by P. Wawrzyński in (Wawrzyński, 2009). The HalfCheetah is a 2-dimensional robot consisting of 9 body parts and 8 joints connecting them (including two paws). The goal is to apply a torque on the joints to make the cheetah run forward (right) as fast as possible, with a positive reward allocated based on the distance moved forward and a negative reward allocated for moving backward. The torso and head of the cheetah are fixed, and the torque can only be applied on the other 6 joints over the front and back thighs (connecting to the torso), shins (connecting to the thighs) and feet (connecting to the shins).

Hopper This environment is based on the work done by Erez, Tassa, and Todorov in (Erez et al., 2011). The environment aims to increase the number of independent state and control variables as compared to the classic control environments. The hopper is a two-dimensional one-legged figure that consist of four main body parts - the torso at the top, the thigh in the middle, the leg in the bottom, and a single foot on which the entire body rests. The goal is to make hops that move in the forward (right) direction by applying torques on the three hinges connecting the four body parts.

Walker This environment builds on the **Hopper** environment by adding another set of legs making it possible for the robot to walk forward instead of hop. Like other Mujoco environments, this environment aims to increase the number of independent state and control variables as compared to the classic control environments. The walker is a two-dimensional two-legged figure that consist of seven main body parts - a single torso at the top (with the two legs splitting after the torso), two thighs in the middle below the torso, two legs in the bottom below the thighs, and two feet attached to the legs on which the entire body rests. The goal is to walk in the in the forward (right) direction by applying torques on the six hinges connecting the seven body parts.

Ant This environment is based on the environment introduced by Schulman, Moritz, Levine, Jordan and Abbeel in (Schulman et al., 2015). The ant is a 3D robot consisting of one torso (free rotational body) with four legs attached to it with

each leg having two body parts. The goal is to coordinate the four legs to move in the forward (right) direction by applying torques on the eight hinges connecting the two body parts of each leg and the torso (nine body parts and eight hinges).

Swimmer This environment corresponds to the Swimmer environment described in Rémi Coulom's PhD thesis (Coulom, 2002). The environment aims to increase the number of independent state and control variables as compared to the classic control environments. The swimmers consist of three or more segments ('links') and one less articulation joints ('rotors') - one rotor joint connecting exactly two links to form a linear chain. The swimmer is suspended in a two dimensional pool and always starts in the same position (subject to some deviation drawn from an uniform distribution), and the goal is to move as fast as possible towards the right by applying torque on the rotors and using the fluids friction.

D.2. Gym-Robotics Environment Tasks

AntMaze This environment was refactored from the D4RL repository, introduced by Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine in (Fu et al., 2020). The task in the environment is for an ant-agent to reach a target goal in a closed maze. The ant is a 3D robot consisting of one torso (free rotational body) with four legs attached to it with each leg having two body parts. The goal is to reach a target goal in a closed maze by applying torques on the eight hinges connecting the two body parts of each leg and the torso (nine body parts and eight hinges).

Fetch This environment was introduced in (Plappert et al., 2018). The robot is a 7-DoF Fetch Mobile Manipulator with a two-fingered parallel gripper. The robot is controlled by small displacements of the gripper in Cartesian coordinates and the inverse kinematics are computed internally by the MuJoCo framework. The gripper is locked in a closed configuration in order to perform the push task. The task is also continuing which means that the robot has to maintain the block in the target position for an indefinite period of time. Notably, in "FetchPush" or "FetchSlide" when the absolute value of the reward is smaller than 0.05, namely $|r| < 0.05$, we consider the task to be terminated. This is because the returned dense reward is the negative Euclidean distance between the achieved goal position and the desired goal.

Adroit Hand This environment was introduced in (Rajeswaran et al., 2017). The environment is based on the Adroit manipulation platform, a 28 degree of freedom system which consists of a 24 degrees of freedom ShadowHand and a 4 degree of freedom arm. Notably, in "AdroitDoor" when the absolute value of the reward is greater than 20, namely $|r| > 20$, we consider the task to be terminated. This is because a total positive reward of 20 is added if the door hinge is opened more than 1.35 radians. In "AdroitHammer" when the absolute value of the reward is greater than 25, namely $|r| > 25$, we consider the task to be terminated. This is because a total positive reward of 25 is added if the euclidean distance between both body frames is less than 0.02 meters.

For more information about the tasks, please refer to the official document and the links are presented in Appendix C.

D.3. Toy Example Settings

This subsection introduces the settings of the toy example in Figure 1.

Task Description We employ the tailored interface of the `PointMaze` environment within Gym-Robotics(de Lazcano et al., 2023). As illustrated in Figure 6, the maze has dimensions of 10×10, and the agent commences its navigation from the bottom-left corner, aiming to reach the top-right corner.

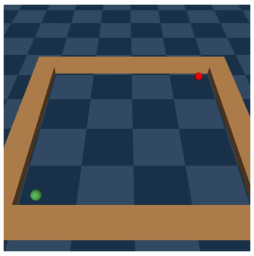


Figure 6. Demonstration of the customized maze in 'Human' mode render.

Observation Space The source observation only contains an array of shape (4,). The entire observation space is continuous, where obs[0] and obs[1] denotes the x, y coordinates of the agent's current position, while obs[2] and obs[3]

represent the x, y coordinates of the target location.

Action Space The source action only contains an array of shape (2,). The entire action space is continuous, where action[0] and action[1] denote the linear force in the x, y direction to the agent.

Rewards We use dense rewards. The returned reward is the negative Euclidean distance between the current agent's location and the target location.

Training Parameters We use TD3 (Fujimoto et al., 2018) as the RL algorithm. The values of hyperparameters for TD3(Fujimoto et al., 2018) are derived from their original implementation[4]. In Figure 1a '**1** Iteration', the 'Episode Return' is normalized to 0-100.

E. Additional Experiments

E.1. Could LESR work if it uses the LLM-coded reward function as the only reward signal?

We have conducted several experiments in Mujoco tasks. **Directly Intrinsic** means directly using the best state representation and intrinsic reward codes $\mathcal{F}_{best}, \mathcal{G}_{best}$ to train without extrinsic reward. **LESR w/o ER** means throughout the entire iterative process of LESR, extrinsic rewards are entirely removed. **LESR w/o LC** means throughout the entire iterative process of LESR, Lipschitz constant is entirely removed. Results under five random seeds are demonstrated in the following Table 5.

Table 5. Experiments of using the LLM-coded reward function as the only reward signal, while dropping the external reward information.

Environments \ Algorithm	TD3	Directly Intrinsic	LESR w/o ER	LESR w/o LC	LESR(Ours)
HalfCheetah	9680.2±1555.8	8919.9±1761.9	10252.8±277.1	10442.9±304.4	**10614.2±510.8**
Hopper	3193.9±507.8	3358.4±76.6	**3408.4±144.0**	3362.3±109.4	**3424.8±143.7**
Walker2d	3952.1±445.7	1924.3±969.3	3865.8±142.5	**4356.2±335.3**	**4433.0±435.3**
Ant	3532.6±1265.3	3518.0±147.6	**4779.1±30.4**	3242.3±459.9	4343.4±1171.4
Swimmer-v3	84.9±34.0	25.2±4.1	51.9±0.1	116.9±6.3	**164.2±7.6**

Directly Intrinsic: When we directly use $\mathcal{F}_{\lfloor\rceil\int\sqcup}, \mathcal{G}_{\lfloor\rceil\int\sqcup}$ to train without extrinsic reward. It's observed that the performance of our method is subject to perturbations but remains relatively stable, achieving comparable results to TD3 in environments such as HalfCheetah, Hopper, and Ant.

LESR w/o ER: Particularly, when we iterate without extrinsic rewards throughout the process, LESR demonstrates superior final performance over TD3 in most tasks.

LESR w/o LC: In experiments where the Lipschitz constant is removed, there is a performance decrease compared to LESR, yet still outperforming TD3, further indicating the effectiveness of utilizing the Lipschitz constant for feedback in our approach.

E.2. Two New Task Decriptions

To further validate the generalization capability of LESR, we conduct experiments on two entirely new task descriptions: (1) requiring the walker agent to learn to jump in place, and (2) requiring the walker agent to learn to stand with its legs apart. We abstain from any iterations or feedback, training solely on the $\mathcal{F}, \mathcal{G}$ produced in the first round of LESR, and only for 300k time steps. We generate the final GIFs in the github repository of LESR[5], and from the GIFs on the webpage, it is observed that the walker can perform the jumping and standing with legs apart actions as per the task descriptions, which further highlights the generalization significance of LESR.

E.3. Different Choices of Lipschitz constant

As mentioned in Section 3.3, apart from estimating the Lipschitz constant computed independently for each state dimension concerning the extrinsic reward, we can also employ the discouted return (**LESR(DR)**) or spectral norm (**LESR(SN)**) instead of the extrinsic reward. We have conducted experimental evaluations on these various feedback signals in Mujoco tasks in Table 6. Note: In this setting the sample count K is set to 3.

[4]https://github.com/sfujim/TD3
[5]https://github.com/thu-rllab/LESR

Table 6. Various choices of Lipschitz constant estimation as feedback signals to LLM. **LESR(DR)**: LESR with Discounted Return. **LESR(SN)**: LESR with Spectral Norm.

Environments \ Algorithm	TD3	LESR(DR)	LESR(SN)	LESR(Ours)
Ant	3532.6±1265.3	4451.0±1532.7	3649.2±1709.3	**5156.4±267.3**
HalfCheetah	9680.2±1555.8	8926.1±1487.2	**10830.8±340.6**	9740.1±761.0
Hopper	3193.9±507.8	**3405.6±482.3**	3179.4±592.5	3370.9±135.8
Swimmer	84.9±34.0	60.1±10.5	49.7±1.9	**142.0±7.6**

It's illustrated that both **LESR(DR)** and **LESR(SN)** are feasible for estimating the Lipschitz constant. However, LESR demonstrates comparatively more stable performance with lower variance in the context of dense reward environments. Nonetheless, this does not discount the utility of employing discounted return and spectral norm estimation methods. Specifically, in scenarios involving discontinuous rewards, the necessity of employing LESR(DR) and LESR(SN) becomes apparent.

E.4. More Training Details of LESR

We have presented the performance data for a limited scope of 300k environment interaction training steps in Table 7. The results demonstrate that LESR exhibits superior performance within significantly fewer training steps, excelling comprehensively across all tasks in comparison to the RL baseline algorithm TD3, thereby manifesting heightened sample efficiency.

We have also presented the final evaluating curves in Figure 7 using all tasks' best state representation and intrinsic reward functions $\mathcal{F}_{best}, \mathcal{G}_{best}$. This further substantiates that our approach outperforms the baseline in terms of both sample efficiency and ultimate performances.

In Figure 8 we have presented performance improments compared with the baseline using every iteration's best state representation and intrinsic reward functions $\mathcal{F}_{best}, \mathcal{G}_{best}$. It is demonstrated that the feedback part plays a critical role in our method. The final performance demonstrated a gradual amelioration across successive iterations.

E.5. Semantic Analysis

Here are the state representation functions for the *Swimmer* task across four random seeds. The state representations in this task can be categorized into five groups, whose names are derived from the comments provided by the LLM: *cosine or sine of the angles (c1)*, *relative angles between adjacent links (c2)*, *kinetic energy (c3)*, *distance moved (c4)*, and *sum of torques applied (c5)*.

State representation functions for the *Swimmer* task

$seed_1$:

```
def revise_state(s):
    x_velocity_squared = s[3]**2
    total_angular_velocity = np.sum(s[5:]**2)
    cos_angles = np.cos(s[:3])
    sin_angles = np.sin(s[:3])
    relative_angles = np.diff(s[1:3])
    updated_s = np.concatenate((s, [x_velocity_squared,
        total_angular_velocity], cos_angles, sin_angles, relative_angles))
    return updated_s
```

$seed_2$:

```
def revise_state(s):
```

```
    if len(s) > 3:
        relative_angles = [s[i+1] - s[i] for i in range(1, len(s)-3)]
    else:
        relative_angles = []
    kinetic_energy = 0.5 * (s[3]**2 + s[4]**2) + 0.5 * sum(s[5:]**2)
    distance_moved = s[3]
    sum_torques = sum(abs(s[5:]))
    updated_s = np.concatenate((s, relative_angles, [kinetic_energy,
        distance_moved, sum_torques]))
    return updated_s
```

$seed_3$:

```
def revise_state(s):
    relative_angle_1_2 = s[1] - s[2]
    total_kinetic_energy = 0.5 * (s[5]**2 + s[6]**2 + s[7]**2)
    mass = 1.0
    length = 1.0
    gravity = 9.81
    potential_energy = mass * gravity * length * np.sin(s[0])
    dt = 1.0
    distance_x = s[3] * dt
    updated_s = np.concatenate((s, [relative_angle_1_2, total_kinetic_energy,
        potential_energy, distance_x]))
    return updated_s
```

$seed_4$:

```
def revise_state(s):
    cos_angles = np.cos(s[:3])
    sin_angles = np.sin(s[:3])
    relative_angles = np.diff(s[:3])
    kinetic_energy = 0.5 * (s[3]**2 + s[4]**2) + 0.5 * np.sum(s[5:]**2)
    distance_moved = s[3]
    updated_s = np.concatenate((s, cos_angles, sin_angles, relative_angles, [
        kinetic_energy, distance_moved]))
    return updated_s
```

F. Future Work

Methodology framework viability Our methodology framework in LESR remains viable for image tasks, where Vision-Language Models (VLMs)(Radford et al., 2021; Kim et al., 2021) can be employed to extract semantic features from images, followed by further processing under the LESR framework. We anticipate utilizing VLMs for future research.

General applicability beyond symbolic environments While our primary focus lies on symbolic environments, our method extends beyond them. LESR serves as a general approach for leveraging large models to generate Empowered State Representations, offering potential applicability across various environments.

Offline reinforcement learning LESR is also feasible for offline reinforcement learning scenarios(Qu et al., 2024; Shao et al., 2024; Mao et al., 2023; 2024). The LESR framework is versatile and not limited to online RL. In the future, we aim to explore various applications and possibilities.

Within LESR, we utilize LLMs to generate Empowered State Representations, showcasing their effectiveness in enhancing the Lipschitz continuity of value networks in reinforcement learning. Our experimental results, along with supplementary theorems, validate these advantages. We believe that LESR holds promise in inspiring future research endeavors.

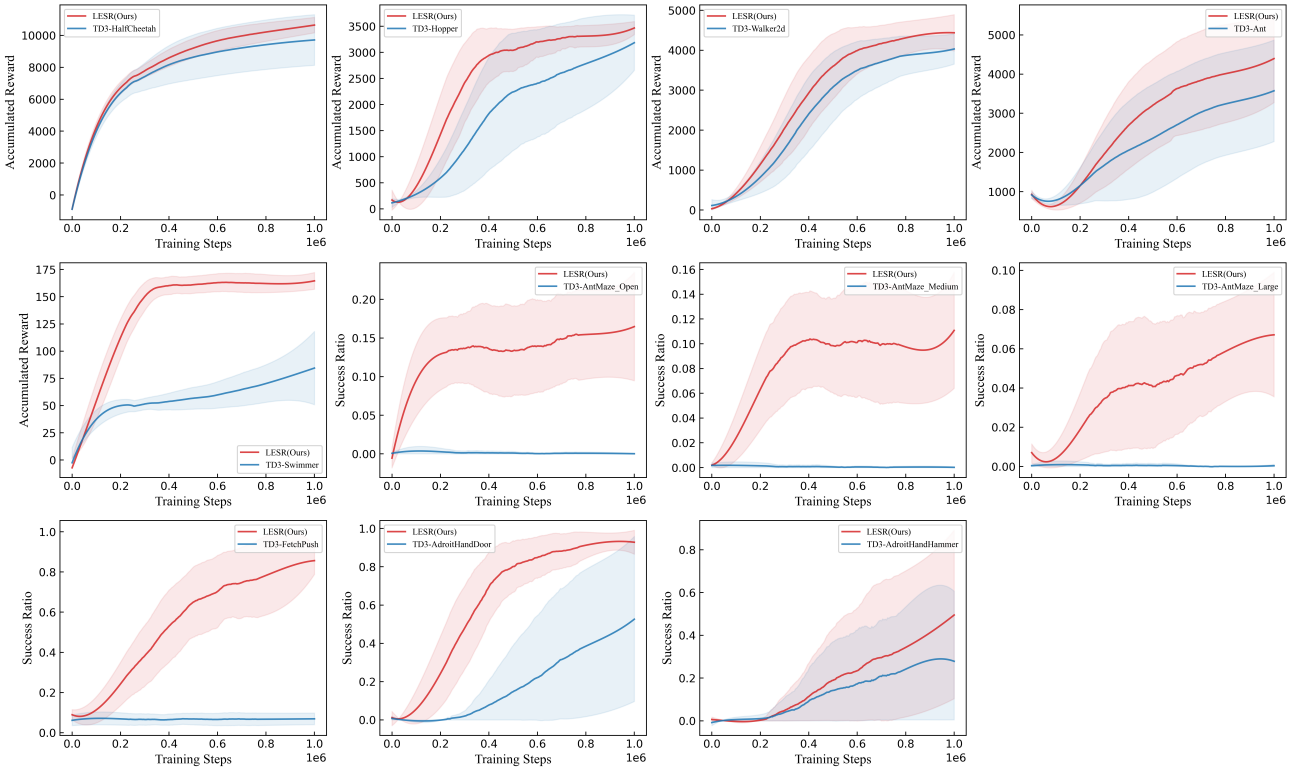
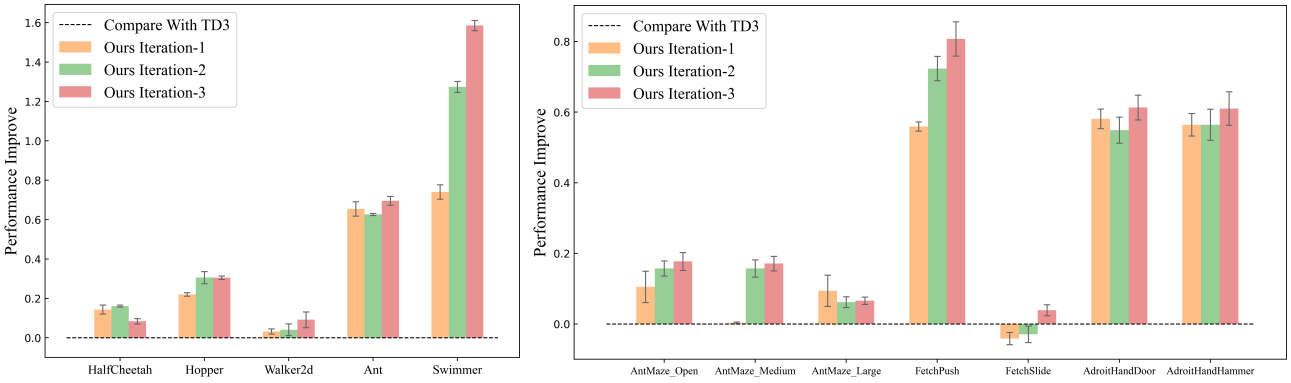


Figure 7. All tasks' final evaluating curves using the best state representation and intrinsic reward functions $\mathcal{F}_{best}, \mathcal{G}_{best}$ over 5 random seeds. We have smoothed the curves using avgol_filter in 'scipy'.



(a) Mujoco Tasks

(b) Gym-Robotics Tasks

Figure 8. Performance improments compared with the baseline TD3 during every iteration. We use every iteration's best state representation and intrinsic reward functions $\mathcal{F}_{best}, \mathcal{G}_{best}$. Performance improments are calculated by $\frac{accumulated_rewards - baseline}{baseline}$, while in Gym-Robotics, it is determined as $success_rate - baseline$.

Table 7. Final Performance on Mujoco and Gym-robotics environments over only **300k** environment interaction training steps. **Ours w/o IR:** without intrinsic reward. **Ours w/o SR:** without state representation. **Ours w/o FB:** without feedback. More details about the ablation study are elucidated in Section 4.4. **RPI =** `(accumulated_rewards - baseline)/baseline`: relative performance improvement. **PI =** `success_rate - baseline`: success rate improvement. This distinction arises because the performance is represented by the accumulative rewards in Mujoco, whereas in Gym-Robotics, it is measured by the success rate. The "mean±std" denotes values computed across five random seeds.

Environments \ Algorithm	TD3	EUREKA	RPI	Ours w/o IR	RPI	Ours w/o SR	RPI	Ours w/o FB	RPI	LESR(Ours)	RPI
HalfCheetah	7288.3±842.4	7299.1±344.2	0%	6985.7±921.8	-4%	6924.5±483.9	-5%	5391.6±3221.2	-26%	**7639.4±464.1**	**5%**
Hopper	921.9±646.8	2132.1±0.0	131%	2509.0±619.8	172%	1638.0±1031.3	78%	1993.4±956.0	116%	**2705.0±623.2**	**193%**
Walker2d	1354.5±734.1	512.9±132.3	-62%	1805.5±948.1	33%	962.1±388.4	-29%	1270.2±659.5	-6%	**1874.8±718.2**	**38%**
Ant	1665.2±895.5	1138.2±0.0	-32%	**2409.9±575.2**	**45%**	1967.9±1090.3	18%	2025.0±910.5	22%	1915.3±885.5	15%
Swimmer	49.9±5.2	32.3±0.0	-35%	57.5±4.2	15%	65.8±24.0	32%	51.3±5.0	3%	**150.9±9.5**	**203%**
Mujoco Relative Improve Mean	-	-	1%	-	52%	-	19%	-	22%	-	**91%**
-	-	-	**PI**	-	**PI**	-	**PI**	-	**PI**	-	**PI**
AntMaze_Open	0.0±0.0	0.12±0.03	12%	0.0±0.0	0%	**0.19±0.05**	**19%**	0.15±0.06	15%	0.15±0.06	15%
AntMaze_Medium	0.0±0.0	0.01±0.0	1%	0.0±0.0	0%	0.06±0.03	6%	0.01±0.01	1%	**0.11±0.05**	**11%**
AntMaze_Large	0.0±0.0	0.0±0.0	0%	0.0±0.0	0%	0.03±0.02	3%	0.01±0.02	1%	0.04±0.03	4%
FetchPush	0.06±0.03	0.08±0.03	2%	0.29±0.17	23%	0.06±0.03	0%	0.26±0.14	19%	**0.41±0.15**	**35%**
AdroitHandDoor	0.0±0.01	0.10±0.07	10%	0.12±0.2	12%	0.02±0.03	2%	0.04±0.07	4%	**0.38±0.26**	**38%**
AdroitHandHammer	0.02±0.02	0.04±0.05	2%	0.0±0.01	-1%	0.0±0.01	-1%	**0.07±0.11**	**5%**	0.01±0.01	-1%
Gym-Robotics Improve Mean	-	-	5%	-	6%	-	5%	-	8%	-	**17%**

G. Hyperparameters

We have listed the hyperparameters of our algorithm in Table 8, which encompasses the parameters of the training process, algorithm and optimizer settings. Notably, for PPO and SAC, we adopt the code in `https://github.com/Lizhi-sjtu/DRL-code-pytorch`. For EUREKA, to ensure comparative fairness, we utilized the same hyperparameters as LESR, namely the sample count K and the iteration count I.

Table 8. Hyperparameters for our algorithm. The values of hyperparameters for TD3(Fujimoto et al., 2018) are derived from their original implementation.

Hyperparameters Name	Explaination	Value
K	How many sample to generate during one iteration	6
I	How many iterations	3
N_{small}	Total training timesteps for iteration	8e5
N	Final total training timesteps for $\mathcal{F}_{best}, \mathcal{G}_{best}$	1e6
w	weight of intrinsic reward	0.02 for Mujoco tasks, 0.2 for Gym-Robotics tasks