InCLET: Large Language Model In-context Learning can Improve Embodied Instruction-following

Peng-Yuan Wang*
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
Polixir.ai, Nanjing, China
wangpy@lamda.nju.edu.cn

Jing-Cheng Pang*
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
Polixir.ai, Nanjing, China
pangjc@lamda.nju.edu.cn

Chen-Yang Wang*
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
221300004@smail.nju.edu.cn

Xuhui Liu
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
liuxh@lamda.nju.edu.cn

Tian-Shuo Liu
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
Polixir.ai, Nanjing, China
liuts@lamda.nju.edu.cn

Si-Hang Yang
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
Polixir.ai, Nanjing, China
yangsh@lamda.nju.edu.cn

Hong Qian
Shanghai Institute of AI for Education and School of Computer Science and Technology,
East China Normal University,
Shanghai, China
hqian@cs.ecnu.edu.cn

Yang Yu†
National Key Laboratory for Novel Software Technology
School of Artificial Intelligence
Nanjing University, Nanjing, China
Polixir.ai, Nanjing, China
yuy@nju.edu.cn

ABSTRACT

Natural language-conditioned reinforcement learning (NLC-RL) empowers embodied agent to complete various tasks following human instruction. However, the unbounded natural language examples still introduce much complexity for the agent that solves concrete RL tasks, which can distract policy learning from completing the task. Consequently, extracting effective task representation from human instruction emerges as the critical component of NLC-RL. While previous methods have attempted to address this issue by learning task-related representation using large language models (LLMs), they highly rely on pre-collected task data and require extra training procedure. In this study, we uncover the inherent capability of LLMs to generate task representations and present a novel method, in-context learning embedding as task representation (InCLET). InCLET is grounded on a foundational finding that LLM in-context learning using trajectories can greatly help represent tasks. We thus firstly employ LLM to imagine task trajectories following the natural language instruction, then use in-context learning of LLM to generate task representations, and finally aggregate and project into a compact low-dimensional task representation. This representation is then used to train a human instruction-following agent. We conduct experiments on various embodied control environments and results show that InCLET creates effective task representations. Furthermore, this representation can significantly improve the RL training efficiency, compared to the baseline methods.

KEYWORDS

Reinforcement Learning; In-context Learning; Embodiment Agent

ACM Reference Format:
Peng-Yuan Wang*, Jing-Cheng Pang*, Chen-Yang Wang*, Xuhui Liu, Tian-Shuo Liu, Si-Hang Yang, Hong Qian, and Yang Yu†. 2025. InCLET: Large Language Model In-context Learning can Improve Embodied Instruction-following. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Developing robots capable of executing tasks following human instructions is a highly attractive area of research [4, 16, 25, 26]. Natural language-conditioned reinforcement learning (NLC-RL) has emerged as a powerful approach in this field, as it focuses on training agents to perform tasks specified by natural language instructions [23, 28, 29]. The key of NLC-RL lies in processing the complex and diverse natural language (NL) into a *task representation*, which is then exposed to agent to complete the control task.

*Equal Contribution †Corresponding Author.

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Traditional methods typically employ pre-trained large language models (LLMs) such as BERT [19] and Llama [9] to extract semantic information directly from natural language. They convert the NL instruction into LLM embeddings as the task representations, and then train a policy conditioned on these embeddings. However, these embeddings generated by LLMs are typically learned independently of the RL task, making it challenging to capture the task-related information in the natural language instruction. An alternative method is to train a natural language translator that converts natural language instructions into a low-dimensional machine-readable representations, leveraging pre-collected data from environmental interactions [29, 40]. Despite their potential, such methods are heavily dependent on high-quality interaction data and necessitate a separate training process for translator, which in turn increases the cost and reduces the flexibility of the approach. This highlights the pressing need to *develop efficient task representations that require minimal additional efforts.*

In this work, we introduce a novel method, in-context learning embedding as task representation (InCLET), which produces effective task representation for NLC-RL without relying on pre-collecting environment data or additional training. Instead of directly utilizing LLM's embedding or additionally training a translator, InCLET leverages the in-context learning paradigm to extract task representations, capturing more intrinsic information, which is inspired by [12]. InCLET consists of three components: (1) an in-context generator that generates few task-related trajectories; (2) a task representation extractor that extracts task representation from generated few trajectories; (3) a policy that solves concrete RL tasks given task representation from human instructions. Specifically, in the in-context generator, we leverage the extensive knowledge of LLMs to automatically generate (initial state, terminal state) trajectory pairs about task, which are used for task representation extraction. In the second module, we format these as *"[Initial state]→[Terminal state]"*, extracting the hidden state at the '→' position. These hidden states are then fused to form the task representation. In the third module, we combine the state and task representations for policy training in reinforcement learning. To address the high dimensionality of the task representation space [46], we employ random projection to map it onto a much lower-dimensional subspace [35], making the representation more efficient for training. In contrast to previous methods, InCLET obtains effective task representations without requiring any task-specific information or additional training.

We justify the effectiveness of InCLET method through theoretical analysis, which shows that our method achieves tighter error bound compared to traditional methods. Besides, through extensive experiments in two embodied control environments: FrankaKitchen [10] and CLEVR-Robot [17], we demonstrate that the policy learned by InCLET outperforms previous methods, in terms of the ability to follow different natural language instructions and adapt to previously unseen NL instructions. Furthermore, we verify the source of InCLET's effectiveness, by performing t-SNE [39] dimensionality reduction on the task representations for visualization. We observe that InCLET effectively separates the representations of different tasks, indicating the feasibility of the way to leverage in-context learning to extract task representations. Finally, we conducted an ablation study to analyze the impact of each module in InCLET on the overall performance.

We highlight the main contributions of our work as follows:

- We successfully verify that the in-context learning capabilities of LLMs can be harnessed to guide the embodied agents to follow human instructions.
- We introduce a novel approach, InCLET that creates task representations using a pre-trained LLM, without the need for pre-collected datasets or additional training procedures.
- Theoretical analysis and empirical results demonstrate that InCLET effectively generates task-related representations, enabling agents to understand the task, complete the instruction successfully, and outperform the baseline NLC-RL methods.

2 BACKGROUND

2.1 RL and NLC-RL

We consider environments represented as a finite Markov decision process (MDP) [30, 32, 36], which is described by a tuple $(\mathcal{S}, \mathcal{A}, P, \rho, \mathbf{r}, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space. P represents the probability of transition while ρ represents the initial state distribution. r is a reward function while γ represents the discount factor determining the weights of future rewards. In NLC-RL, the agent receives an NL instruction (L_N) that reflects the human's instruction. The policy $\pi(\cdot|s_t, L_N)$ which is used for decision making is trained conditioned on the state $s_t \in \mathcal{S}$ and language instruction L_N which describes the task (e.g. 'Can you open the light?'). It is crucial to highlight that in our work, we assume no prior knowledge of the exact number of tasks, the relevant task details, or the task instruction descriptions. The overall objective of NLC-RL is to maximize the expected return under different NL instructions:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t, L_N\right) \mid s_0 \sim \rho, a_t \sim \pi\left(\cdot \mid s_t, L_N\right)\right]. \tag{1}$$

2.2 ICL and task representations

ICL is a paradigm that enables language models to learn tasks by providing only a few examples as demonstrations [6, 8]. Formally, given a query input prompt p, a pre-trained language model $\mathcal{M}$ takes the candidate answer conditioned on a demonstration S. S contains k demonstrations, thus $S = \{(x_1, y_1), \cdots (x_k, y_k)\}$. To perform specified tasks for a given query p, the model is asked to predict y based on the demonstrations namely,

$$y = \arg\min \mathcal{M}([S, p]), \tag{2}$$

where $[S, p]$ represents a concatenation of the demonstrations S and input prompt p.

There has been work [12] demonstrating that the mechanism behind in-context learning involves using demonstrations to extract a task representation θ. The θ is used to identify the task and generate the output y for the current query p.

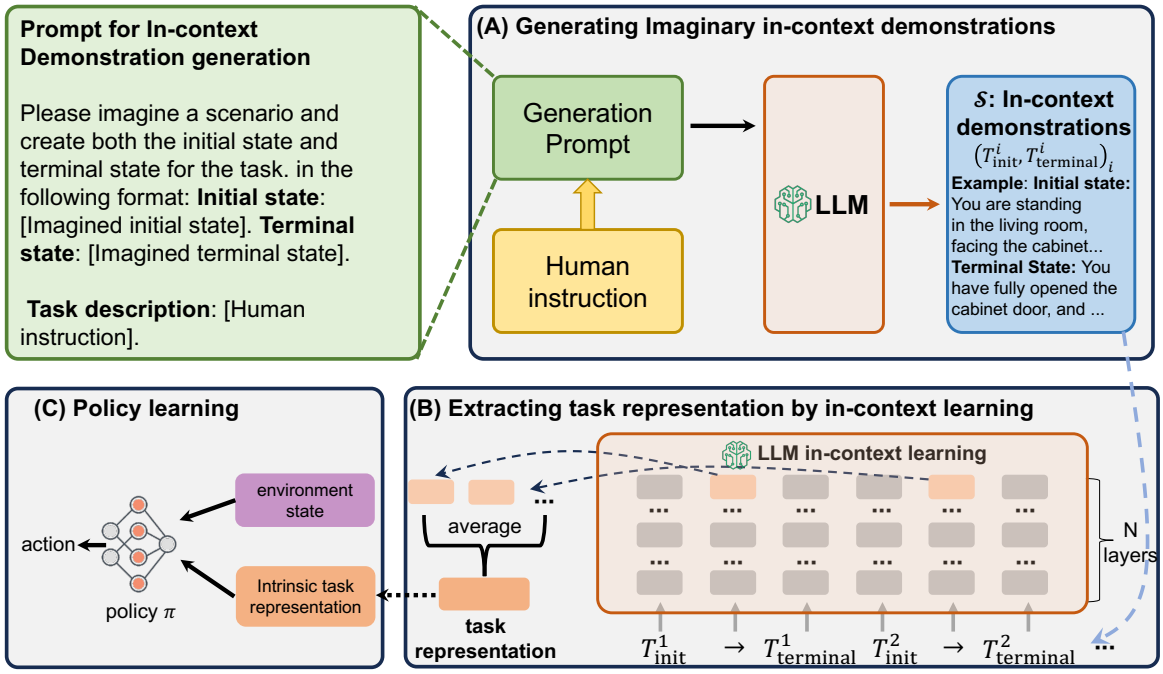


Figure 1: Overall framework of InCLET method.

2.3 Random Projection Technique for Dimension Reduction

In real-world applications, the task of optimizing black-box functions that operate within high-dimensional spaces (often surpassing 1000 dimensions) presents significant challenges, making direct black-box optimization in high-dimensional space difficult [31]. One effective approach is to use random projection, which transforms the data into a intrinsic dimensional space. The intrinsic dimensionality of an objective function refers to the minimum number of parameters required to achieve satisfactory solutions [20]. As demonstrated by [2], large-scale pre-training empirically compresses the intrinsic dimensionality of downstream tasks. [20] proposes a method to measure this intrinsic dimensionality in neural networks by identifying the minimal subspace dimensionality derived through random projections. This approach to random mapping is rooted in the Johnson-Lindenstrauss lemma [18], which posits that when points in a vector space are projected onto a randomly chosen subspace of sufficiently high dimension, the distances between points are approximately preserved.

3 METHOD

Given a task description, we aim to extract a corresponding task representation to guide the agent in completing instruction-following task. The method leverages LLM's internal ability to generate in-context demonstrations S. Then we construct task pairs with S and extract a task-specific vector to serve as the conditioning input for policy training.

Our framework, as illustrated in Fig. 1, consists of three stages Specifically. In the first stage, given a task description L_N, we utilize LLM to generate multiple task-relevant <initial state, terminal state> trajectory pair. In the second stage, we concatenate the generated pairs with L_N in the form of *"initial state → terminal state,"* using these as input to the LLM to extract the corresponding hidden states. These hidden states are then fused to form the task representation θ. In the third stage, the high-dimensional task representation is mapped to a lower-dimensional space and used as the condition for policy learning.

3.1 Generating Imaginary In-context Demonstrations by Prompting LLM

To leverage the in-context learning ability of LLMs, the first step is to obtain a set of in-context learning trajectories and construct the demonstration set, which we refer to as *imaginary in-context demonstrations*. InCLET generates imaginary in-context demonstrations by prompting LLMs to imagine specific trajectories (initial/terminal states) corresponding to the human instructions. Given a human instruction x, we utilize the the following prompt p for the input of LLM $\mathcal{M}$: 'Please help me imagine a scenario and create both the initial state and terminal state for the task $\cdots$'. During generation, we create one trajectory at a time, repeating the process n times until the entire trajectory set is generated. We extract the initial state and terminal state from the trajectory to construct the set of imaginary in-context demonstrations S:

$$S = \{(T^1_{\text{init}}, T^1_{\text{terminal}}), \cdots, (T^n_{\text{init}}, T^n_{\text{terminal}})\},$$

where $(T^i_{\text{init}}, T^i_{\text{terminal}}) = \mathcal{M}(x, p)$ is the imaginary in-context demonstrations extracted from the output by the LLM.

The demonstration is structured by presenting the initial state as T_{init} and the terminal state as T_{terminal}. This format is chosen because, when given the task and the initial state T_{init}, the output T_{terminal} from the LLM represents task completion. Namely, $T_{\text{init}} \rightarrow$

Algorithm 1 Train Procedure of In-context Learning Embedding as Task Representation (InCLET)

Require: LLM $\mathcal{M}$; imaginary in-context demonstrations num n; random projection matrix $A \in \mathbb{R}^{k \times d}$; interaction steps M.
Output: instruction following policy π_ϕ;

Initialize parameters of the policy network π_ϕ, and value network V_ψ.
for *timestep* $k = 0$ to M **do**
  Sample a natural language task instruction L_N from the environment.
  // Generate Imaginary In-context Demonstrations
  for $i = 1$ to n **do**
    Query LLM $\mathcal{M}$ to generate imaginary in-context demonstrations $(T^i_{\text{init}}, T^i_{\text{terminal}})$ with NL L_N.
  end for
  // Extract Task Representation
  Construct mapping $T^i_{\text{init}} \rightarrow T^i_{\text{terminal}}$ with demonstrations.
  Extract hidden state with Eq. (3). And fuse it with Eq. (4)
  // Train Policy with RL
  Use random projection metric to transfer into intrisic task representation Z_{L_N} with Eq. (5).
  while episode not terminal **do**
    Observe current state s_t.
    Execute action $a_t \sim \pi_\phi(\cdot|s_t, Z_{L_N})$, and receive a reward r_t from the environment.
  end while
  Update the policy π_ϕ and value functions V_ψ based on the samples collected from the environment.
end for

T_{terminal} implicitly signifies the successful completion of the current task, making it easier for us to extract task-related representations.

3.2 Extracting Task Representations via In-context Learning from LLM

In the previous subsection, we introduce how InCLET generates a set of imaginary in-context demonstrations. Now we elaborate on the process for obtaining the task representation that indicates the tasks corresponding to the human instruction. Specifically, we leverage the imaginary in-context demonstrations generated in section 3.1 to perform the task of mapping $T^i_{\text{init}} \rightarrow T^i_{\text{terminal}}$. In particular, the latent state $h^i \in \mathbb{R}^d$ for each $T^i_{\text{init}} \rightarrow T^i_{\text{terminal}}$ is obtained by inputting them into the LLM. Specifically, we extract the hidden state corresponding to the arrow position token and take the hidden state from the final layer of the attention block as the task representation. All the demonstrations are input into LLM total, namely, $\{h_i\}_{i=1}^n \leftarrow \mathcal{M}(\{T^i_{\text{init}} \rightarrow T^i_{\text{terminal}}\}_{i=1}^n)$. However, directly inputting it into the LLM can lead to the following problem: If L_N were placed after the demonstration, the $\rightarrow$ for the first demonstration would only have seen the initial state without being exposed to the task description, leading to an inaccurate hidden state. Therefore, we concatenate L_N before the demonstration S. This method is finally formalized as:

$$\{h_i\}_{i=1}^n \leftarrow \mathcal{M}([L_N, \{T^i_{\text{init}} \rightarrow T^i_{\text{terminal}}\}_{i=1}^n]). \tag{3}$$

For a total of n demonstrations, the extracted hidden states form the set $H := \{h_1, h_2, \cdots, h_n\}$. Finally, we average the hidden states vectors as the final task representation θ:

$$\theta = \frac{1}{n} \sum_{i=1}^{n} h_i. \tag{4}$$

Due to the complexity and diversity of language, directly inputs the task description L_N into the LLM $\mathcal{M}$ to obtain the hidden state embedding as task representation struggles to get the exact task representation. In contrast, our in-context learning approach extracts a more specific task representation by leveraging provided demonstrations [22], enabling better task representation.

3.3 Policy Learning with Task Representation

InCLET uses RL to train an instruction-following policy (IFP). When the agent collects samples in the environment, the environment randomly generates natural language instruction based on the current task. We utilize the method introduced in section 3.1 and section 3.2 to extract task representation. Next, the policy makes decisions for the entire episode based on the current observation and task representation until either the task is completed or the maximum timestep is reached. However, typically the task representation $\theta \in \mathbb{R}^d$ from the LLM is a high-dimensional vector, which is challenging when directly inputting it to the policy. We address this by *random projection* technique, which converts the task representation into a low intrinsic dimensionality, easing the burden on the policy. Specifically, InCLET uses a random matrix $\mathbf{A} \in \mathbb{R}^{k \times d}$ to project the original d-dimensional data, θ, onto a k-dimensional subspace ($k \ll d$) , as expressed in the equation below:

$$Z_k = \mathbf{A}_{k \times d} \theta_d, \tag{5}$$

where Z_k is defined as intrinsic task representation and k represents the dimension of subspace. By sampling from a uniform distribution, we set the values of the random matrix $\mathbf{A}$, similar to Sun et al. [35]. Therefore, we utilize the intrinsic task representation Z as the policy condition input rather than task representation θ. The IFP can be optimized with an arbitrary RL algorithm using the samples collected from the environments. In our implementation, we use PPO [34] for both InCLET and all baselines. During IFP training, the LLM's parameters are frozen. We provide the pseudo-code shown in Algorithm 1 for InCLET to further clarify the process.

4 THEORETICAL JUSTIFICATION

Let $\mathcal{Z}$ denote the embedding space, and $\mathcal{T}$ denote the task. We assume that the value function can be represented as:

$$V(s) = \phi(s)^T w,$$

where $\phi(s) \in \mathbb{R}^d$ is the representation of the state s in some feature space. This assumption is not too restrictive. When using a neural network to represent the value function, $\phi(s)$ corresponds to the output of the penultimate layer (i.e., the second-to-last layer).

Without loss of generality, we assume:

$$\|\phi(s)\| \leq 1 \quad \text{and} \quad \|w\| \leq W_{\max}.$$

Based on this setup, we aim to minimize the regularized expected risk function:

$$R_{\mathcal{T},\phi}(w) = \mathbb{E}_{(s_i)\sim\mathcal{T}}\left[\left(\phi(s_i)^T w - y_i\right)^2\right] + \frac{1}{2}\|w\|^2,$$

where $y_i = \sum_{t=0}^{\infty} \gamma^t r_t$ is the cumulative discounted return.

Since computing the full expectation is often infeasible, we instead minimize the regularized empirical risk function:

$$R_D(w) = \frac{1}{n}\sum_{i=1}^{n}\left(\phi(s_i)^T w - y_i\right)^2 + \frac{1}{2}\|w\|^2,$$

where $D = \{(s_i, y_i)\}_{i=1}^n$ is dataset sampled from the distribution $\mathcal{T}$.

Now, when we concatenate the task representation $z \in \mathcal{Z}$ with the state representation, the feature vector becomes $\phi(s, z)$. Thus, the new regularized empirical risk function becomes:

$$R_{D,\mathcal{Z}}(w) = \frac{1}{n}\sum_{i=1}^{n}\left(\phi(s_i, z_i)^T w - y_i\right)^2 + \frac{1}{2}\|w\|^2.$$

Here, $z_i \in \mathcal{Z}$ corresponds to the embedding associated with the state s_i.

In our algorithm, the goal is to solve the following problem:

$$\min_w \mathcal{E}(\mathcal{Z}), \quad \text{where} \quad \mathcal{E}(\mathcal{Z}) = \mathbb{E}_{(\mathcal{T}_i, z_i)\sim p(\mathcal{T},\mathcal{Z})}\mathbb{E}_{D\sim\mathcal{T}_i}\left[R_{D,\mathcal{Z}}(w)\right].$$

The objective is to find the weights w that minimize the expected risk over all possible tasks $\mathcal{T}$ and task representations $z \in \mathcal{Z}$.

Finally, let $\mathcal{E}^*$ denote the optimal minimum risk, expressed as:

$$\mathcal{E}^* = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[R_{\mathcal{T},\phi}(w_{\mathcal{T}})\right], \quad \text{where} \quad w_{\mathcal{T}} = \arg\min_w R_{\mathcal{T},\phi}(w).$$

Then we derive the error bound of value function of NLC-RL.

Theorem 4.1. *Let $(e_i)_{i=1}^S \subset \mathbb{R}^S$ be the standard basis and*

$$P \triangleq \mathbb{E}_{i\sim\nu}[e_i e_i^T].$$

We use Φ_z to represent the feature matrix when task representation exists. We assume further that $\Phi_z\Phi_z^\dagger$ is diagonal, then with probability at least $1-\eta$

$$\mathcal{E}(\mathcal{Z}) - \mathcal{E}^* \le \frac{4R_{\max}}{\sqrt{n(1-\gamma)}}\left(W_{\max} + \frac{R_{\max}}{1-\gamma}E_{z\sim p(z)}\operatorname{Tr}(M(z)N(z))\right)^{\frac{1}{2}} + O\left(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_{\phi(s,z)}}{n}\log\left(\frac{n}{d_{\phi(s,z)}}\right)}\right),$$

where $W_{\max}$, $R_{\max}$, and γ are constants defined in the context, $M(z) = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T}|z)}P$, $N(z) = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T}|z)}P^\dagger$, and d_ϕ is VC dimension of function space of $\phi(s, z)$.

The proof of this theorem is in Appendix B. The bound presented in Theorem 4.1 consists of two terms. The first term highlights the performance improvement achieved through the additional information provided by task representations. In contrast, the second term accounts for the performance degradation arising from the increasing complexity of the representation function. Therefore, NLC-RL establishes a trade-off between these two terms.

For one-hot method, task representation z only corresponds to one task $\mathcal{T}$. Therefore, $p(\mathcal{T}|z)$ is a deterministic distribution, and the results of one-hot method can be derived as follows:

Corollary 4.2. *Under the condition of Theorem 4.1, with probability at least $1-\eta$, the error bound of one-hot method is*

$$\mathcal{E}(\mathcal{Z}) - \mathcal{E}^* \le \frac{4R_{\max}}{\sqrt{n(1-\gamma)}}\left(W_{\max} + \frac{R_{\max}}{1-\gamma}\right)^{\frac{1}{2}} + O\left(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_{\phi(s,z_{\text{one-hot}})}}{n}\log\left(\frac{n}{d_{\phi(s,z_{\text{one-hot}})}}\right)}\right).$$

Similarly, for naive method without NL instruction, the result can be derived by setting $p(\mathcal{T}|z) = p(\mathcal{T})$.

Corollary 4.3. *Under the condition of Theorem 4.1, with probability at least $1-\eta$, the error bound of naive method is*

$$\mathcal{E} - \mathcal{E}^* \le \frac{4R_{\max}}{\sqrt{n(1-\gamma)}}\left(W_{\max} + \frac{R_{\max}}{1-\gamma}E_{z\sim p(z)}\operatorname{Tr}(MN)\right)^{\frac{1}{2}} + O\left(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_{\phi(s)}}{n}\log\left(\frac{n}{d_{\phi(s)}}\right)}\right),$$

where $M = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}P$ and $N = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}P^\dagger$.

Compared to Theorem 4.1, one-hot encoding method yields the finest task representation, resulting in the smallest first component but the largest second component. Conversely, the naive method has the largest first component and the smallest second component. NLC-RL, on the other hand, offers a balanced approach. Additionally, Theorem 4.1 outlines a method for comparing different NLC-RL approaches. Specifically, the less random the distribution $p(\mathcal{T}|z)$ is, the better the resulting error bound. This finding elucidates the characteristics of an effective task representation. In Figure 6, we visualize the results from our method alongside several baseline methods, illustrating why our approach outperforms the others.

5 EXPERIMENTS

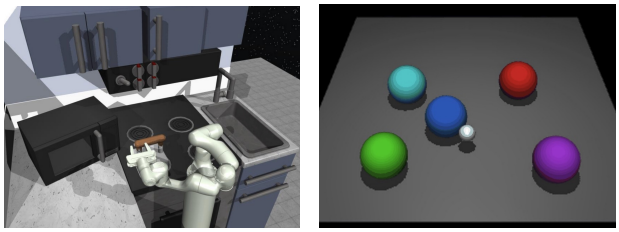
We conduct extensive experiments to evaluate the effectiveness of the proposed InCLET method. Our experiments aim to answer the following important questions:

- How does InCLET perform compared to previous methods on NLC-RL tasks? (Section 5.2)
- How are the task representations generated by InCLET and how do they compare to the task representations from baselines (Section 5.3)?
- Can InCLET be compatible with different large language models? (Section 5.4)
- What is the impact of each component on the overall performance of InCLET? (Section 5.5)

5.1 Experiment Setup

Environments. We conduct experiments on FrankaKitchen [10] and CLEVR-Robot environments [17] as shown in Fig. 2. FrankaKitchen is a multitask environment in which a 9-DoF Franka robot is placed in a kitchen. The goal is to control the robot to interact with the items in order to reach a desired goal configuration. In the environment, we choose four sub-tasks: activate the bottom burner, move the kettle to the top left burner, turn on the light switch, and open the slide cabinet. We treat each sub-task as a different

goal configuration. In each trajectory, the environment randomly generates an NL instruction that describes a goal configuration. The language instruction is generated by ChatGPT [1]. The CLEVR-Robot environment is designed for agent manipulation in object interaction tasks and is built on the MuJoCo physics engine [37]. It contains five different colored balls and an agent (silver point). In each task, the agent moves a ball to achieve a specific position relative to another ball (including four possible positions: front, behind, left, or right) to complete the goal.



(a) FrankaKitchen (b) CLEVR-Robot

Figure 2: Visualization of the environments in our experiments. (a) FrankaKitchen. The agent controls a 9-DoF Franka robot to manipulate various objects in a kitchen. (b) CLEVR-Robot. The agent (silverpoint) manipulates five different colored balls to reach a specific goal position

Details of natural language instructions. For FrankaKitchen environment, we used ChatGPT to generate 15 different task descriptions for each goal configuration, yielding 60 different NL descriptions. For example, in the task "open the sliding cabinet," one of the descriptions is "Can you please open the sliding cabinet door for me?". In the CLEVR-Robot task, the variation between tasks comes from the different colored balls selected and their relative positions. Therefore, we used 18 natural language sentence patterns to generate different tasks. We fixed the combinations of balls and relative positions, resulting in 8 distinct tasks and a total of 144 distinct NL instructions. An example of NL instruction is "Push the red ball behind the blue ball". We split the total NL instructions into two sets: training and testing set. The training set comprises 10 and 9 different NL instructions for each goal configuration in FrankaKitchen and CLEVR-Robot, respectively. The testing set consists of the remaining NL instructions. During policy training, the agent can only interact with the NL instructions in the training set. The full set of task descriptions can be found in Appendix A.1.

Baselines for comparison. We consider multiple representative methods in NLC-RL that do not train the LLM as baselines: (1). **One-hot** method encodes all natural language instructions (including both training and testing set) into a one-hot vector. (2). **BERT** method processes all language instructions using a pre-trained BERT model to generate task representations. These representations are then mapped into a continuous space using a fully connected neural network. (3). **Llama3-8B** method is similar to BERT, except that it uses Llama-3.1-8B-Instruct [9] as the pre-trained LLM instead of the BERT model.

Implementation Details. In our experiments, we use Llama-3.1-8B-Instruct [9] to generate imaginary in-context demonstrations and extract task representation, with temperature factor $T = 1$. We conducted all experiments with three random seeds, and the shaded area in the figures represents the standard deviation across all three trials. we use the open-sourced RL repository, stable-baselines3 [33] for PPO training. For detailed implementation and hyper-parameters, please refer to Appendix A.1.

5.2 Main Results

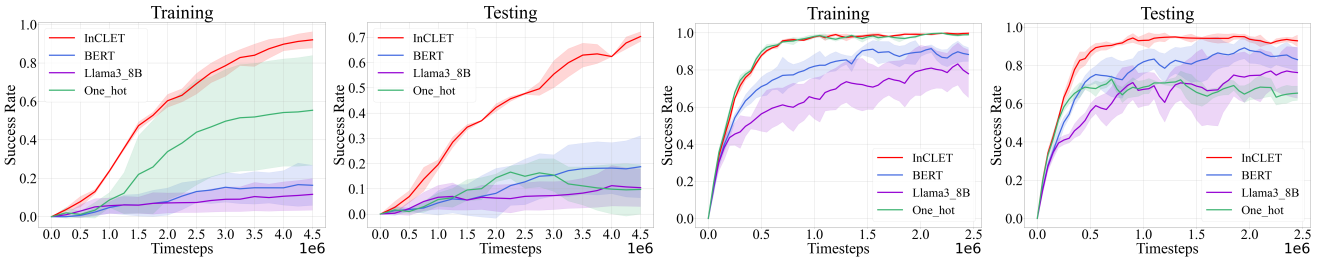
Fig. 3 shows the performance of InCLET and baselines on FrankaKitchen and Ball environments. Overall, InCLET outperforms all baselines on both training and testing NL instructions. In the Kitchen environment, baselines that directly employ LLM embedding such as BERT and Llama3-8B struggle to improve the policy performance. This is due to the complexity of diverse NL instructions and the robotic control. The experimental results in Fig. 6 further justify this conclusion. Though one-hot performs well on training NL instructions, it suffers from a significant performance degradation on the test instructions. This can be attributed to that one-hot baseline can only deal with these seen NL instructions, and fail to generalize to unseen instructions. Notably, on these training NL instructions that are unseen during the RL training process, InCLET also obtains a high score. These results highlight InCLET's potential to develop an agent capable of dealing with diverse NL instructions from different scenarios.

5.3 Effectiveness of the InCLET

Previous experiments show InCLET can effectively improve the policy performance compared to the baseline methods. This section investigates the source of such improvement. First of all, we utilize t-SNE projection technique to convert the generated task representation into two-dimensional vectors, as shown in Fig. 6. The points with the same color represent the task representations of the same task, but with different natural language descriptions. We observe that the task representations generated by InCLET get together closer. In contrast, baselines' task representations are more dispersed and scattered. These results justify that InCLET policy can learn more efficiently, as the task representations more accurately capture the meaning of the tasks. Then, we reviewed the examples generated by the LLM, as shown in Fig. 4, which depicts the imaginary demonstrations in the FrankaKitchen environment. We found that the LLM generates demonstrations are highly related to the task description, and these demonstrations cover various scenarios which are helpful to extract task representations.

5.4 Compatibility with Different LLMs

We further verify the effectiveness of InCLET method by conducting experiments with different LLMs of various sizes and structures. We conducted experiments with three models in the FrankaKitchen environment and reported their test performance. Specifically, we implement InCLET with Llama3-8B [9], Gemma-7B [3] and Mistral-7B [15], respectively. Additionally, we implemented baseline methods that directly use LLM embeddings for policy training with the models mentioned above. Fig. 5 presents the experimental results. We find that InCLET can effectively improve the NLC-RL performance compared to directly using LLM embeddings as the task representations. Notably, our method delivers consistent performance across different models.



(a) Kitchen Training (b) Kitchen Testing (c) Ball Training (d) Ball Testing

Figure 3: Performance of different methods in two environments on training and testing NL instructions. The shaded area stands for the standard deviation across three random trials.

Imaginary Trajectories Example

Task: Activate the bottom burner
Task Description: Can you turn the knob on the oven that activates the bottom right burner?
Imaginary In-context Demonstrations:
(1). Initial State: The oven is turned off and the kitchen is quiet, with no one around. the knob on the oven that activates the bottom right burner is in the \"off\" position. **Terminal State:** The knob on the oven that activates the bottom right burner is in the \"on\" position, and the burner is heating up with a gentle blue flame. the kitchen is now bustling with activity, with the aroma of baking bread filling the air.
(2). Initial State: The oven door is open, the knob is not turned, and the bottom burner is off. **Terminal State**: The oven door is open, the knob is turned, and the bottom right burner is on.
(3). ···

Figure 4: Examples of the imaginary in-context trajectories on FrankaKitchen. The generated language trajectories highly relevant to the instruction are highlighted with green shading, demonstrating that the generated initial and terminal states are strongly aligned with the given instruction.

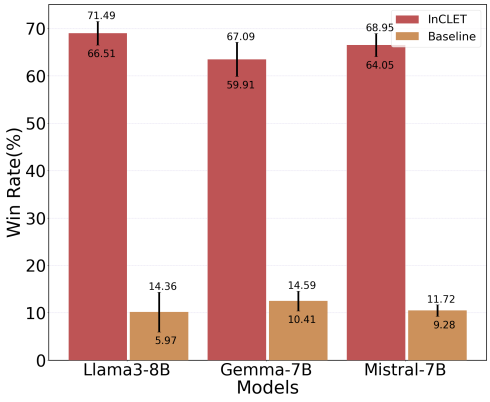
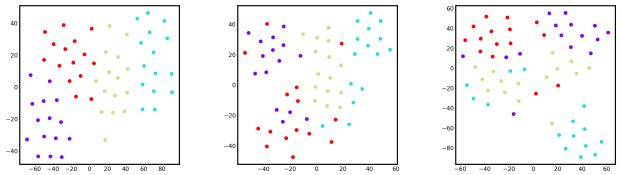


Figure 5: Performance of InCLET with different LLMs on FrankaKitchen environment.

5.5 Ablation Study

We conduct ablation study to evaluate the impact of each component in InCLET on the overall performance. We consider the following components:in-context learning (**ICL**) that generates task vectors, random projection (**RP**) and different fusion methods.



(a) InCLET (b) Baseline (Bert) (c) Baseline (Llama)

Figure 6: T-SNE projection of the task representations generated by different methods.

Ablation study on ICL and RP. We conduct ablation studies on the ICL and RP techniques of InCLET method. To implement InCLET w/o ICL, we directly utilize Llama to convert NL instructions into embeddings, and random projection to reduce the dimension. Fig. 7 presents the results of ablation study. The results indicate that both ICL and RP play an important role in InCLET's effectiveness, as InCLET gets the highest scores and InCLET w/o RP & ICL gets the lowest scores. Notably, InCLET w/o RP suffers from a large deviation across different random trials. This could be attributed to that the origin embedding from the LLM could be high-dimensional and complex, highlighting the importance of the dimension reduction by random projection techniques.

Comparing different fusion method. We conducted a fusion-specific ablation to evaluate the impact of the fusion method in InCLET. We conducted a comparative experiment, directly using the hidden state corresponding to $\rightarrow$ in the last demonstration, rather than utilizing the fusion of the hidden states from all demonstrations S. The t-SNE visualization of the representations is shown in the Fig. 8. Experimental results indicate that using only the last hidden state results in less efficient representations. Upon examining the responses shown in Fig. 4, we think this is because the demonstrations imagined by the LLM contain some scene-specific information. By fusing multiple hidden states, we can improve the representation of task-related information.

6 RELATED WORK

Natural Language Conditioned Reinforcement Learning. Natural language conditioned reinforcement learning (NLC-RL) requires agents to complete tasks based on natural language instructions. Previous approaches have mainly focused on two areas. One

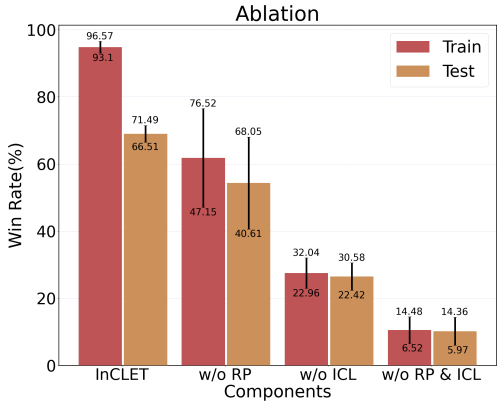
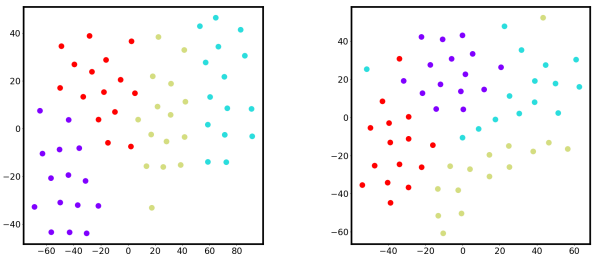


Figure 7: Ablation study on FrankaKitchen environment. The error bar stands for the half standard deviation over three random trials.

involves directly using natural language representations as input for the RL policy and training the policy. For example, [13] uses a pre-trained language model to encode natural language, and provides the encoded NL as input to the policy. [7] combines language instruction with observation via Gated-Attention mechanism and then learns policy to execute the instruction. The other involves training an additional encoder, where natural language is first input into the encoder to obtain its representation, which is then used as input for the policy to train the RL model. For example, [17] learns a high-level policy that generates abstract language representations to guide hierarchical RL. [40] learns a task manager module to handle the language representation for reinforcement learning. Our approach is similar to the first one. The key difference is that we use in-context learning to extract representations that are better suited for policy learning.

Large Language Model for Reinforcement Learning. Large language models store a wealth of knowledge [1, 9, 38, 45], which can be effectively used to guide policy training [5] by aligning the prior knowledge with the knowledge learned by RL. LLMs are leveraged to decompose complex, high-level tasks into multiple low-level, executable step-by-step plans [5, 14, 40, 42]. These decomposed plans are then used as conditions for downstream reinforcement learning tasks. However, due to the limited reasoning capabilities of LLMs, they often struggle to deliver precise, effective reasoning. In addition, some works directly use large language models for extracting useful embed information [24], generating reward functions [27, 44], or generating imaginary rollouts for world modeling and policy learning [28]. Compared with the above methods, we use large language models as both a generator and task information extractor module. The model automatically generates in-context learning trajectories and uses these to construct demonstrations and extract relevant task information.

In-context Learning. In-context learning (ICL) enables language models to learn tasks given a few demonstrations [8] which is similar to the decision process of human beings by learning from analogy [43]. The language model is expected to learn the hidden task pattern and accordingly make the right prediction. Therefore, "How to effectively generate high-quality demonstrations" is an



(a) InCLET w/ $\{h_i\}$ fusion **(b) InCLET w/ only h_n**

Figure 8: T-SNE projection of the task representations generated by different fusion method.

important area of research in ICL. Traditional methods heavily rely on human-written demonstrations, which are often limited in quantity, diversity, and creativity. Some work leverages LLM to generate demonstrations [11, 41] which is similar to InCLET's trajectories generation (see details in section 3.1). Furthermore, some studies focus on extracting the latent representations from demonstrations [21] and find that these can provide good task information [12, 22]. Compared to previous methods, our approach does not rely on human-written demonstrations. Instead, InCLETleverages the internal knowledge of LLMs to generate demonstrations and extract task-specific information. In contrast to [21, 22], we validate our method on more complex RL control tasks.

7 CONCLUSION

In this paper, we study the integration of LLMs for embodied instruction-following control. We propose a simple and effective approach, InCLET, that processes natural language instruction into task representation. This is the first study investigating the usage of ICL for embodied instruction-following control. Compared to previous NLC-RL methods, InCLET removes the requirements for pre-collecting data from the environment and improves the performance over LLM embeddings. However, there are still some limitations. First, it is challenging to utilize ICL from LLMs to produce effective task representations, when natural language instructions become too complex. When NL instructions are long and detailed, e.g., 'place the apple on the table, but only after the orange is placed next to the cup', it is hard to generate the trajectories that capture the conditional or sequential nature of these tasks, leading to failed task representations. Fortunately, the development of LLMs has the potential to mitigate this issue. Second, our method relies on LLMs to generate multiple demonstrations through imagination. If the model is too small, the quality of the generated demonstrations may be sub-optimal. In the future, we will explore the effectiveness of using closed-source models to generate demonstrations in resource-constrained environments, while using smaller models to generate task representations.

ACKNOWLEDGMENTS

This work is supported by NSFC (62495093) and Jiangsu SF (BK20243039). The authors thank anonymous reviewers for their helpful discussions and suggestions for improving the article.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *CoRR* abs/2303.08774 (2023).

[2] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *ACL/IJCNLP*.

[3] Jeanine Banks and Tris Warkentin. 2024. Gemma: Introducing new state-of-the-art open models. *Google. Available online at: https://blog.google/technology/developers/gemma-open-models/(accessed 6 April, 2024)* (2024).

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *CoRR* abs/2212.06817 (2022).

[5] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *CoRL*. PMLR.

[6] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[7] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding. In *AAAI*, Vol. 32.

[8] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *CoRR* abs/2301.00234 (2022).

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *CoRR* abs/2407.21783 (2024).

[10] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. 2020. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *CoRL*.

[11] Wei He, Shichun Liu, Jun Zhao, Yiwen Ding, Yi Lu, Zhiheng Xi, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Self-Demos: Eliciting Out-of-Demonstration Generalizability in Large Language Models. *CoRR* abs/2404.00884 (2024).

[12] Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *CoRR* abs/2310.15916 (2023).

[13] Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. 2020. Human instruction-following with deep reinforcement learning via transfer-learning from text. *CoRR* abs/2005.09382 (2020).

[14] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*.

[15] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *CoRR* abs/2310.06825 (2023).

[16] Shengyi Jiang, Jing-Cheng Pang, and Yang Yu. 2020. Offline Imitation Learning with a Misspecified Simulator. In *NeurIPS*.

[17] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[18] William B Johnson. 1984. Extensions of Lipshitz mapping into Hilbert space. In *CMAP*.

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

[20] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the Intrinsic Dimension of Objective Landscapes. In *ICLR*.

[21] Jiahao Li, Quan Wang, Licheng Zhang, Guoqing Jin, and Zhendong Mao. 2024. Feature-Adaptive and Data-Scalable In-Context Learning. *CoRR* abs/2405.10738 (2024).

[22] Sheng Liu, Haotian Ye, Lei Xing, and James Y Zou. 2024. In-context Vectors: Making In Context Learning More Effective and Controllable Through Latent Space Steering. In *ICML*.

[23] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob N. Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In *IJCAI*.

[24] Vivek Myers, Andre Wang He, Kuan Fang, Homer Rich Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. 2023. Goal representations for instruction following: A semi-supervised language interface to control. In *CoRL*. PMLR.

[25] Jing-Cheng Pang, Si-Hang Yang, Xiong-Hui Chen, Xinyu Yang, Yang Yu, Mas Ma, Ziqi Guo, Howard Yang, and Bill Huang. 2023. Object-Oriented Option Framework for Robotics Manipulation in Clutter. In *IROS*.

[26] Jing-Cheng Pang, Nan Tang, Kaiyuan Li, Yuting Tang, Xin-Qiang Cai, Zhen-Yu Zhang, Gang Niu, Masashi Sugiyama, and Yang Yu. 2025. Learning View-invariant World Models for Visual Robotic Manipulation. In *ICLR*.

[27] Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2024. Language Model Self-improvement by Reinforcement Learning Contemplation. In *ICLR*.

[28] Jing-Cheng Pang, Si-Hang Yang, Kaiyuan Li, Jiaji Zhang, Xiong-Hui Chen, Nan Tang, and Yang Yu. 2024. Knowledgeable Agents by Offline Reinforcement Learning from Large Language Model Rollouts. In *NeurIPS*.

[29] Jing-Cheng Pang, Xin-Yu Yang, Si-Hang Yang, Xiong-Hui Chen, and Yang Yu. 2024. Natural language instruction-following with task-related language development and translation. *NeurIPS* (2024).

[30] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[31] Hong Qian, Yi-Qi Hu, and Yang Yu. 2016. Derivative-Free Optimization of High-Dimensional Non-Convex Functions by Sequential Random Embeddings. In *IJCAI*. 1946–1952.

[32] Hong Qian and Yang Yu. 2021. Derivative-free reinforcement learning: a review. *Frontiers of Computer Science* 15, 6 (2021), 156336.

[33] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research* 22, 268 (2021).

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR* abs/1707.06347 (2017).

[35] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-Box Tuning for Language-Model-as-a-Service. In *ICML*, Vol. 162. 20841–20855.

[36] Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book* (2018).

[37] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *IROS*.

[38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR* abs/2307.09288 (2023).

[39] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9 (2008).

[40] Zoya Volovikova, Alexey Skrynnik, Petr Kuderov, and Aleksandr I Panov. 2024. Instruction Following with Goal-Conditioned Reinforcement Learning in Virtual Environments. *CoRR* abs/2407.09287 (2024).

[41] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *ACL*.

[42] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, and Team CraftJarvis. 2023. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In *NeurIPS*.

[43] Patrick H Winston. 1980. Learning and reasoning by analogy. *Commun. ACM* 23, 12 (1980).

[44] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2024. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning. In *ICLR*.

[45] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *CoRR* abs/2407.10671 (2024).

[46] Yangwenhui Zhang, Hong Qian, Xiang Shu, and Aimin Zhou. 2023. High-Dimensional Dueling Optimization with Preference Embedding. In *AAAI*.