Goal-Guided Reinforcement Learning: Leveraging Large Language Models for Long-Horizon Task Decomposition

Ceng Zhang[1], Zhanhong Sun[1], Gregory S. Chirikjian[1,2]

***Abstract*— Reinforcement learning (RL) has long struggled with exploration in vast state-action spaces, particularly for intricate tasks that necessitate a series of well-coordinated actions. Meanwhile, large language models (LLMs) equipped with fundamental knowledge have been utilized for task planning across various domains. However, using them to plan for long-term objectives can be demanding, as they function independently from task environments where their knowledge might not be perfectly aligned, hence often overlooking possible physical limitations. To this end, we propose a goal-based RL framework that leverages prior knowledge of LLMs to benefit the training process. We introduce a hierarchical module that features a goal generator to segment a long-horizon task into reachable subgoals and a policy planner to generate action sequences based on the current goal. Subsequently, the policies derived from LLMs guide the RL to achieve each subgoal sequentially. We validate the effectiveness of the proposed framework across different simulation environments and long-horizon tasks with complex state and action spaces. The LLM prompts we use and more details can be found at `https://chirikjianlab.github.io/G2RL-LM/`.**

I. Introduction

Reinforcement learning algorithms are beneficial for addressing novel unseen tasks as they do not depend on handcrafted policies or labeled trajectories, instead, the agent learns from reward signals through iterative interactions with the task environment [1]. However, this characteristic is effective only for tasks with simple state spaces and brief action sequences. For complex long-term tasks, exploring the environment in the early stage of training becomes problematic, since most RL algorithms rely on random action selection such as ϵ-greedy [2] or policy noise addition [3], which fails to consistently produce successful trajectories, especially in environments with large or continuous state or action spaces, as shown in Fig. 1. Therefore, a key challenge in RL is reducing the searching space for efficient early-stage exploration and logically defining short-term goals that match the present states of complex tasks for step-by-step learning.

On the other hand, with extensive pretraining on vast amounts of open source knowledge, LLMs have shown reasoning abilities on par with humans across numerous domains, including robotics and automation. Using abstract task descriptions as input, they can apply common sense reasoning to devise policies that guide towards task objectives [4]–[6]. However, while these generated policies appear to be feasible, adhering strictly to them and performing actions may introduce potential hazards. This is primarily because LLMs are not connected to the task environment, which hinders their ability to accurately assess environmental information about current state. Although advanced foundation models are capable of processing multimodal information [7], [8], representations that fully depict the environmental state remains a challenge, rendering it partially observable for these large pretrained models. As a result, when the task goal is distant from the present state, the planning process might neglect possible restrictions within the environment, such as that shown in Fig. 2. Without an understanding of these constraints, it is unsafe to execute the generated plans reliably as the policies may not be applicable to the state.

Fig. 1. Overview. When presented with a complex long-horizon task, making decisions for the next step can be challenging from a large number of available options. Instead, it is intuitive to divide the task objective to several manageable subgoals and achieve them sequentially.

To bridge the gaps mentioned above, in this study we introduce an LLM-enhanced RL framework for efficient exploration and training. For complex novel tasks that require a long sequence of actions to reach the target, we utilize LLMs to break down distant task objectives into smaller, manageable goals and encourage RL agents to reach them sequentially. Rather than rely solely on LLM-generated policies, we integrate them with RL algorithms to facilitate guided exploration and subgoal achievement, thereby significantly accelerating the training process and enabling convergence to optimal policies that take environmental constraints into account.

Overall, the main contribution of this work includes:

- Investigation on suboptimality of LLM policies when neglecting environmental constraints and enhancement

This work was supported by NUS Startup grants A-0009059-02-00, A-0009059-03-00, CDE Board account E-465-00-0009-01, and National Research Foundation, Singapore, under its Medium Sized Centre Programme - Centre for Advanced Robotics Technology Innovation (CARTIN), sub award A-0009428-08-00.

[1] Ceng Zhang, Zhanhong Sun and Gregory S. Chirikjian are with the Department of Mechanical Engineering, National University of Singapore, Singapore.

[2] Gregory S. Chirikjian is with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA.

Address all correspondence to G. S. Chirikjian: mpegre@nus.edu.sg, gchirik@udel.edu

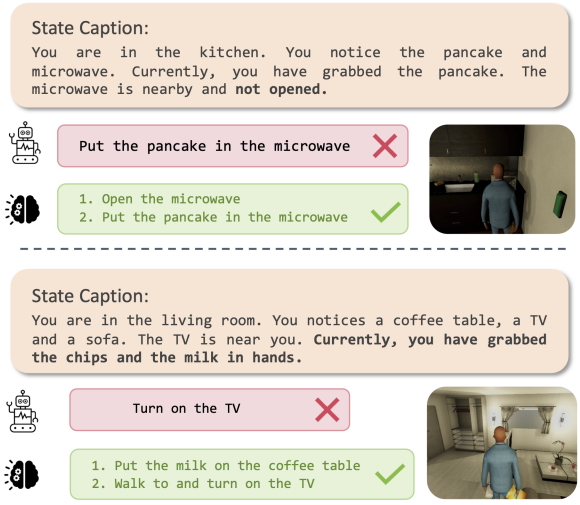

Fig. 2. Examples of Environment Constraints. Explicit (above) and implicit (below) types that might be overlooked when directly using LLMs for long-term goals planning.

through decomposition of distant objectives.

- An LLM-based hierarchical module with brief prompt to perform subgoal proposing and policy generation for complex long-horizon task planning.
- An adaptive goal-conditioned RL framework that incorporates LLMs to guide actions for enhanced exploration in early stages and automatically adjusts the loss weight as training proceeds.

II. Related Work

A. Exploration Issue in RL

The exploration-exploitation dilemma in reinforcement learning questions how agents should balance the act of exploring new possibilities against utilizing known profitable strategies [1]. Recent advances in exploration strategies include the Intrinsic Curiosity Module (ICM) [9], which generates intrinsic rewards based on the prediction error of the environment dynamics, encouraging exploration in sparse reward settings. Similarly, Random Network Distillation (RND) [10] enhances exploration by rewarding agents for encountering unpredictable states, using the novelty measured by a fixed random neural network as a benchmark. However, most of these methods introduce additional network models for online update during training, which greatly increases the computational complexity of RL algorithms and adds to the manual cost of network design. Instead, our method achieves efficient exploration by querying policies from a pretrained LLM to narrow down the searching range in the state-action space with prior knowledge.

B. Task Planning with Pretrained LLMs

Pretrained on diverse datasets to handle various scenarios [11], [12], LLMs offer notable benefits in handling queries across multiple disciplines, drastically reducing the time and data required for task-specific training [13]–[15]. However, a significant challenge is their propensity to adhere to the dataset on which they are trained, potentially leading to improper outcomes in novel task environments. This limitation can be particularly problematic in robotics, where environmental uncertainty is common. To this end, recent approaches, such as online learning, allow robots to update their strategies based on human feedback [16], [17], thus improving adaptability and long-term performance in changing conditions at the expense of introducing human intervention. However, their disconnection from the task environment remains unsolved. Despite the advancement on multimodal large models that can process information in different forms other than text [7], [18], [19], there is inevitable loss in the conversion that leads to inaccurate information about the environment state, thus ignoring potential physical constraints and adopting risky or unavailable actions. To this end, we use LLMs to break down long-horizon tasks and provide guidance in exploration for the RL agent, by which it learns optimal policies from grounded information obtained through iterative interaction with the task environment.

C. Combining LLMs with Reinforcement learning

Recent studies have highlighted the advantages of integrating LLMs and RL to accelerate the training process by providing accurate environment information to LLMs, thus, in return, facilitating effective exploration through their reasoning capabilities [20], [21]. However, this typically requires concurrent weight updates for these large models. Although there are new methods [22] that allow fine-tuning by modifying a small fraction of parameters, the complexity and task generalization have not been sufficiently mitigated. On the other hand, there are works that use pretrained LLMs to help with the RL training expedition [23]–[25], but most of them require complex prompt engineering and seldom focus on task simplification. In this study, we employ a hierarchical LLM architecture for goal-policy generation using concise prompts, which assists the RL agent in exploring efficiently with common sense and accelerates the training process for intricate long-horizon tasks.

III. Problem Formulation

For a long-horizon task within a complex environment, we model the solution process as a Partially Observable Markov Decision Process (POMDP) defined by the tuple (S, A, O, T, R, γ) [26]. Here, S and A refer to the state and action spaces, O indicates the observations from the task environment, and T captures the state transitions described by $T(s' \mid s, a)$, where s' is the subsequent state following action a in state s. The reward signal from the environment for each state-action pair is represented by R, with γ serving as the discount factor. The objective of RL is to learn the optimal policy π^* that maximizes the expectation of cumulative return at time t as

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid (s_t, a_t)\right]. \quad (1)$$

In this work, the applied RL algorithm Proximal Policy Optimization (PPO) [27], based on policy gradient, mini-

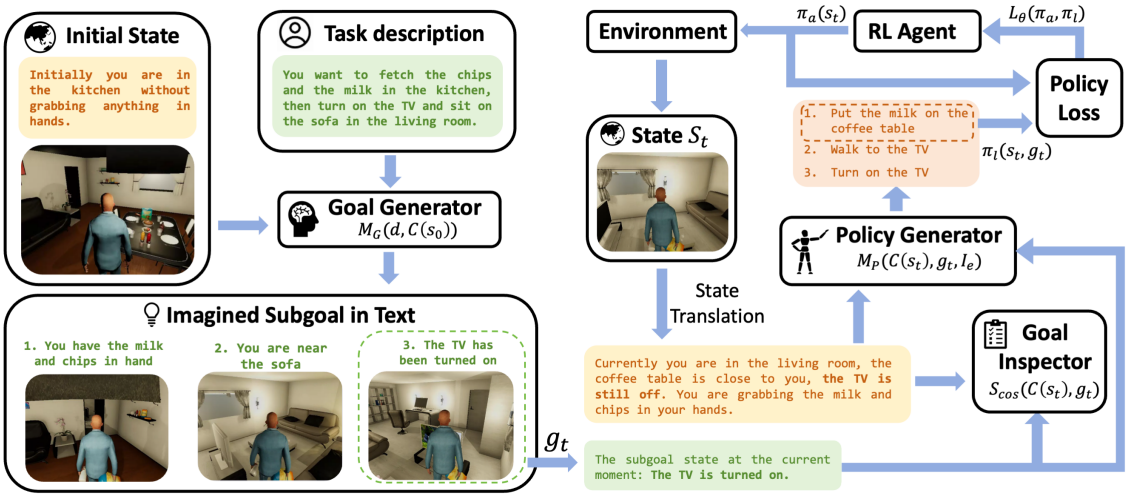

Fig. 3. Method Framework. As mentioned in Section III and IV, the goal generator breaks down the complex task into several subgoals in text using the task description and initial state of the environment as inputs. Subsequently, the policy generator produces actions based on the caption of a given state from the LLM policy π_l. The disparity between the agent's policy π_a and π_l is calculated as an additional policy loss for RL algorithms. In addition, a goal inspector measures the cosine similarity between the encoded embeddings of the subgoal and the state caption to check if the subgoal is reached.

mizes the loss $L(\theta)$ and updates the parameters θ in the actor-critic network.

On the other hand, autoregressive LLMs receive input natural language as prompt p and output content c^* from corpus $\mathcal{V}$ with maximum probability based on previous tokens:

$$c^* = \arg\max_{c \in \mathcal{V}} P(c \mid (p, c_{1:t})). \tag{2}$$

In our case, the prompt p_{task} for LLM includes a brief language description on the task and environmental information, the output is a sequence of generated subgoals $g_{1:t}$ or policies composed of action sequences to reach the current subgoal at state s_t.

IV. Goal-conditioned RL with LLMs

Our proposed LLM-assisted RL framework consists of two hierarchically connected modules, as shown in Fig. 3. This part sequentially introduces each module and elucidates their interaction in improving the performance of RL training.

A. Subgoal Generation with LLMs

Although pretrained LLMs exhibit strong reasoning skills in high-level task planning, they can often neglect potential environmental constraints when it comes to low-level decision making, resulting in risky policies or even invalid actions. When tasked with a complex long-term objective, planning from the start to completion of the task, LLMs are prone to encountering situations where the proposed policy appears conducive to task objective but is impractical in the current scenario; e.g., in the cases shown in Fig. 2, the agent must ensure that one hand is available before attempting to turn on the TV if both hands are currently occupied.

To this end, our approach utilizes an LLM module to segment a long-horizon task objective into multiple, more attainable subgoals. Rather than directly querying action sequences from LLMs, we define these goals as different stages of task completion that describe the state of the task environment. In this way, LLMs are employed to characterize the world model and comprehend state transitions instead of formulating action policies, as depicting the world state proves to be more effective when there are numerous action options, thereby potentially enhancing the reasoning performance of LLMs [28].

Given the brief task description, the subgoals generated can appear in countless forms as the distribution becomes as extensive as the entire language vocabulary [23]. To this end, we impose constraints on the generation by inserting in the prompt information about the initial state of the task environment. This serves two purposes: First, understanding the initial state helps LLMs avoid repeating subgoals that are already completed and focus on future objectives; second, the caption acts as a response template, assisting LLMs in generating subgoals in a uniform format and facilitating verification of goal achievement by comparing with captions of other states. Therefore, we denote the subgoals generated as

$$g_{1:k} = M_G(d, C(s_0)), \tag{3}$$

where d represents the task description and $C(s_0)$ denotes the textual caption of the initial environmental state. Through the subgoal generation module M_G, k subgoals are proposed by the LLM to achieve in order and the number of goals depends on the difficulty of the task.

Once subgoals are generated, the RL agent is anticipated to accomplish the subgoal for the current state. To assess whether a subgoal has been achieved, the current state caption and the subgoal are encoded using pretrained Paraphrase-MiniLM [29] and the cosine similarity between their embeddings is calculated as

$$S_{cos}(C(s_t), g_t) = \frac{E(C(s_t)) \cdot E(g_t)}{\|E(C(s_t))\| \|E(g_t)\|}, \tag{4}$$

and the subgoal is regarded accomplished if the outcome surpasses the threshold ϵ, prompting the agent to pursue the

next subgoal g_{t+1}.

B. Goal-based Policy Generation and Adaptive Improvement

By dividing the intricate task into multiple subgoals, the agent can focus on an immediate target at the moment, which allows environmental constraints to be considered within a restricted planning horizon, thereby greatly enhancing the precision of policy generation by LLMs. Building on this, for a specific state s_t and its related subgoal g_t, another LLM module M_P is employed to produce policies in the form of an action sequence aimed at achieving the goal as

$$a_{t:t+n} = M_P(C(s_t), g_t, I_e) \tag{5}$$

with I_e denoting information about the task environment, which is fixed throughout the training process. Following the chain-of-thought principle (CoT) [30], rather than performing the entire action sequence, we choose the first planned action a_t in the sequence as the policy generated from the LLM and repeat the procedure at the next step. This allows for real-time adjustment and re-planning, consequently improving the performance of the policy generator.

With the subgoals generated and the associated policies to achieve them, we aim to enhance the RL learning process through action guidance, leveraging prior knowledge from LLMs. One method is to introduce additional rewards by giving positive feedback when the agent reaches these subgoals. However, this can lead to instability in the learning process due to variations in goal distribution and reward scaling. To address this, we adopt the concept of minimizing divergence from the target policy [25], [31], [32]. We choose the policy generated by M_P as our target and incorporate an additional policy term into the loss function to minimize during each update. Specifically, we choose the policy loss as

$$L_\theta(\pi_a, \pi_l) = D_{KL}(\pi_a(s_t) \| \pi_l(s_t, g_t)) \tag{6}$$

in which $D_{KL}(\cdot)$ calculates the KL divergence between the agent policy π_a and LLM policy π_l, θ denotes the parameters in the actor-critic network. Because the probability distribution of LLM generation is unavailable, we approximate π_l by repeatedly querying the LLM for m times, with m an adjustable hyperparameter.

Then we add term (6) to the total loss in the RL algorithm by multiplying a constant coefficient λ and a weight w that decays with the training process. Instead of manually designed decaying pattern [25], which introduces additional hyperparameters to finetune, we find it is natural to define w's value to be the entropy $H(s_t)$ of the action probability distribution of the state s_t. This is intuitive since the entropy is large due to the significant uncertainty in states during the early stage of training, promoting the agent to follow the policy from M_G for effective exploration. As training progresses, states become more predictable and entropy diminishes. Therefore, reducing the loss weight aids in stabilizing learning, enabling the agent to adjust to the environment's inherent rewards and optimal pathways to reach the task objective.

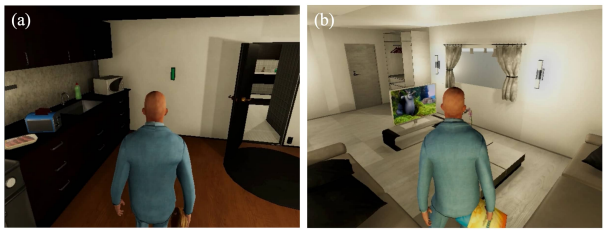

Fig. 4. VirtualHome task environments. (a) FoodPreparation. The task is to fetch the pancake from the kitchen and place it in the microwave for heating. (b) Entertainment. The task is to grab the milk and chips from the kitchen and then relax on the sofa in the living room to watch TV.

V. Experiment

We evaluate the effectiveness of our proposed LLM-assisted RL framework across two task environments, encompassing both discrete and continuous action spaces with extensive state spaces for exploration. We opt for challenging long-horizon tasks characterized by sparse rewards, where the agent only receives a reward of +1 upon completion of the entire task. In our framework, we use GPT-4 [12] for goal generation and GPT-4-turbo to generate policies for quick response during training. To demonstrate the advantages of our method, we compare it with different baselines and carry out ablation studies to examine the impact of specific hyperparameters and components within the framework on the training results.

A. Task Environments

VirtualHome [33] models a domestic setting with an agent figure to perform various everyday activities. This environment presents difficulties in decision making because the state space encompasses multiple rooms and a large number of household items that can be interacted with, considering physical restrictions. Building on settings established by TWOSOME [20], we choose two particular tasks, *Food Preparation* and *Entertainment*, depicted in Fig. 4, to evaluate the presented method. Besides the long action sequence required to complete the whole tasks, the environment constraints also impose challenges on the learning process, these factors pose significant challenges to conventional RL algorithms.

ROMAN [34] depicts a simulated scene in a laboratory where a robotic arm needs to manipulate and transfer objects in a clustered and constrained environment using of a set of available action primitives, which are pretrained through imitation learning combined with RL. In a given state, the observations of the environment are input into a master manipulation network and the action decision is output for controlling the end effector, for visualization purpose, we use Final IK [35] to perform inverse kinematics for joints controlling of the robot arm, shown in Fig. 5. And based on the original setting, we develop two task variations that feature distinct horizon lengths with discrete and continuous action spaces, respectively. After completing training, we evaluate the trained RL agents in this environment with and without a noise of 1 cm in state observations to test their robustness in environments with uncertainty.

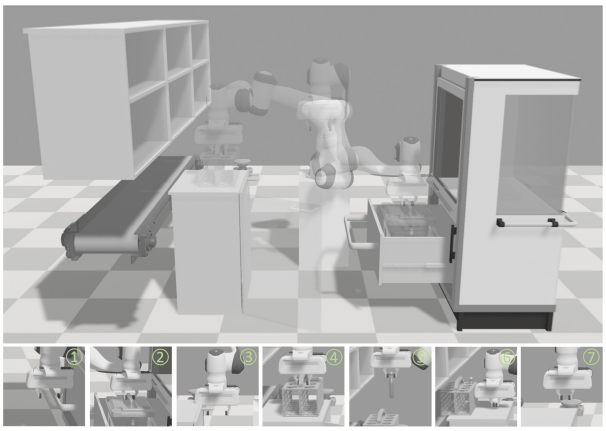

Fig. 5. ROMAN Task Environment. The complex task that requires a long action sequence: (1) pull the drawer where the box containing the vial is stored; (2) remove the box cover; (3) rotate the door of the cabinet where the rack is stored; (4) pick the rack and place it on the table; (5) insert the vial from the box into the rack; (6) push the rack onto the conveyor; (7) press the button to start the conveyor.

In the first task *ROMAN-Short*, we define a timestep as: the robot executes a deterministic action in a given state until the action is completed. The sequence of actions in this setup is relatively short; however, the task remains difficult due to the continuous and highly dimensional nature of the state space. Based on this, *ROMAN-Long* synchronizes each timestep with the real-time frequency of the robot controller. As a result, completing the entire task requires more than 1k timesteps. Moreover, in this scenario, the master manipulation network outputs the weight for each action primitive at every timestep, introducing substantial uncertainty and increasing the task's difficulty. Since learning from scratch in this case presents a substantial challenge for all tested learning-based methods, to mitigate early-stage exploration issues, we use the provided demonstration data for imitation learning as an initial warm-up by annealing Behavior Cloning (BC) for 1.5 million timesteps.

B. Baselines and Ablation Studies

In this study, we evaluate the effectiveness of our method against three baseline approaches and carry out ablation studies to examine the importance of specific elements within our proposed framework.

RL-Base: This refers to the standard RL approach without any alterations. All RL-based techniques discussed in this paper employ Proximal Policy Optimization (PPO) [27] for training, and our approach utilizes the same actor-critic network architectures.

RL-Mask: This method is built on conventional RL algorithms and filters invalid actions for the current state, thereby reducing the exploration space and increasing the likelihood of success in the initial phase. We implement this by applying an action mask to the actor network's output and re-normalizing the masked action probability distribution. In cases of continuous action space like *ROMAN-Long*, the invalid actions are assigned a weight of 0. To validate the effectiveness of our method, we do not mask invalid actions for it and retain the original action space when making state-based decisions.

LLM-CoT: This method aims to improve the reasoning abilities of LLMs by employing a step-by-step logical reasoning approach, resulting in more credible output. Instead of directly generating whole trajectories, the model produces the intermediate steps involved in problem solving, making complex issues understandable to LLMs. Our implementation queries LLMs the next action to take based on the current state and task objective, then repeating this for each subsequent state. For evaluation, we run it for 1k episodes for each task environment to calculate its success rate.

In addition to the baseline methods mentioned above, we also perform three studies to investigate the importance of certain components in our proposed framework. First, we experiment with three weight coefficients $\lambda_1 = 0.1$, $\lambda_2 = 1$, and $\lambda_3 = 10$ to determine whether our method can work with robustness on different scales of the extra policy loss. Besides, to investigate the impact of different modules in the framework, we assess two ablated variations, one is termed **Ours-NoSub** that skips goal generation and directly instructs LLMs to devise policies targeting the final task objective, while **Ours-NoTrain** omits the RL module, retaining the LLM components to explore the role of the applied learning techniques.

VI. Results

In this section, we analyze the experimental results and compare the performance of different methods. First, we focus on the learning curves during training to illustrate the different learning speeds of the algorithms, and we examine how the parameter settings impact the performance of our proposed framework. Subsequently, we evaluate the trained agents in the ROMAN task environment under perception noise by running 1,000 episodes and calculating the success rates to assess the robustness of the algorithms.

A. Training Result Analysis

We train agents in various task environments, as illustrated by the learning curves of the approaches tested in Fig. 6. For tasks with huge action-state spaces, the standard RL-Base algorithm fails to achieve satisfactory performance, unable to obtain any rewards during training in complex scenarios such as *Entertainment* and *ROMAN-Long*. By filtering out invalid actions, RL-Mask demonstrates enhanced performance due to the reduced action space, though it introduces instability during training and shows limited improvement in intricate environments. LLM-CoT excels in simple task planning, achieving high performance in *FoodPreparation* and *ROMAN-Short*. However, its performance drops when longer action sequences are required, due to ignored environmental constraints during decision-making.

Compared to other methods, our method consistently demonstrates superior performance in all evaluated tasks. Not only learns quickly during the initial training phase, it also maintains stable progress, eventually achieving almost 100% of the reward for each episode in *VirtualHome* tasks.

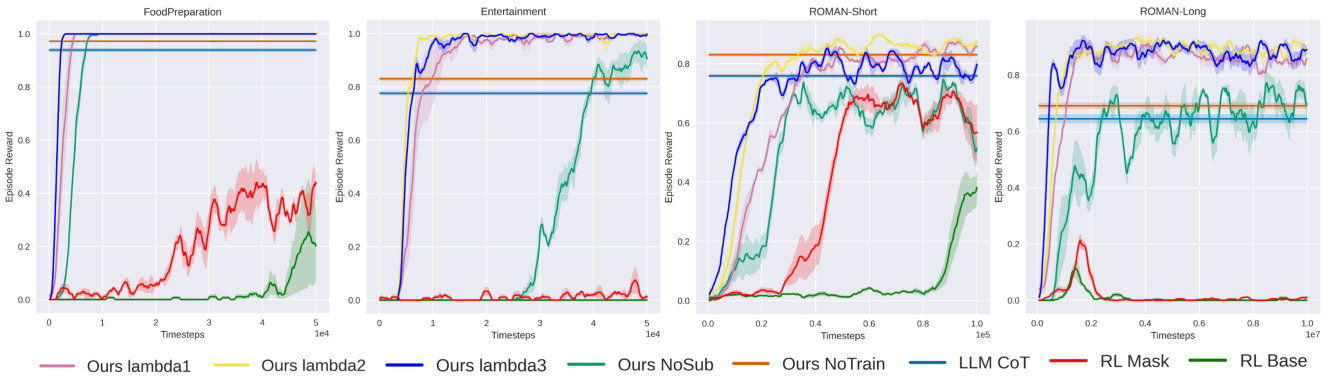

Fig. 6. Training results. Each method runs for n = 3 seeds, with each seed LLM-CoT and Ours-NoTrain are evaluated for 1000 episodes.

Analyzing the results for different coefficients λ, we observe that a higher loss weight improves early stage learning speed by prompting the agent policy to quickly follow the LLM policy. However, this greater loss induces performance fluctuations in complex tasks. In contrast, selecting a smaller weight stabilizes the training but extends the convergence time. Consequently, we find that a weight of $\lambda = 1$ is reasonable, balancing learning speed with training stability and achieving high performances. When the goal generation module is removed, Ours-NoSub directly targets at the task objective. Despite this removal, it still performs well in *FoodPreparation*, though it struggles with tasks requiring lengthy action sequences due to inaccurate LLM policies. The significance of subgoals is also highlighted by comparing the two learning-free approaches, which shows that Ours-NoTrain achieves better performance compared with LLM-CoT across all tasks. However, in contrast to the combined method, its reduced effectiveness indicates that, due to lack of the learning module, it is hindered by suboptimal policies from LLMs that are not connected with task environments.

B. Evaluation on Robustness

In addition to the aforementioned results, we also evaluate our trained agent ($\lambda = 1$) against other methods in terms of task success rates in *ROMAN* task environment, as the results shown in Tab. I. RL-Base and RL-Mask struggle with these intricate tasks, leading to the result that the agent robot fails in achieving the objective, especially in the presence of noises. On the other hand, LLM-CoT exhibits a high task success rate in the simple *ROMAN-Short* scenario, but its reasoning performance is hindered in *ROMAN-Long* due to a higher propensity for errors when dealing with a target far from the initial state and requires long action sequences.

In contrast, our approach achieves the highest success rates in both tasks. Despite the presence of observation noise in the environment, it demonstrates robustness and is slightly affected by uncertainty. On the other hand, Ours-NoSub without subgoal generation shows performance similar to LLM-CoT in *ROMAN-Short* but a significant improvement in *ROMAN-Long*. We propose the reason is that the agent learns the constraints of the environment by interacting with the task world, which leads to better performances than relying on the LLM policy. This is also confirmed by Ours-NoTrain, which sorely depends on the output of LLMs and results in a reduced success rate. We suggest that this is because the agent falls into suboptimal policies generated by LLMs. These policies are not aligned with the physical task environment and may cause missed observations.

TABLE I
EVALUATION ON TASK SUCCESSFUL RATE

Method	ROMAN-Short		ROMAN-Long	
	w/o Noise	w/ Noise	w/o Noise	w/ Noise
Ours (λ=1)	**0.958**	**0.884**	**0.952**	**0.864**
Ours-NoSub	0.781	0.697	0.764	0.705
Ours-NoTrain	0.824	0.751	0.697	0.602
LLM-CoT	0.794	0.743	0.635	0.593
RL-Mask	0.536	0.422	0.023	0.002
RL-Base	0.382	0.224	0.015	0

VII. CONCLUSION AND FUTURE WORK

In this work, we introduce an automated LLM-assisted RL framework designed to accelerate learning and enhance exploration. By incorporating a subgoal generation module, intricate tasks are divided into more manageable phases, thereby enhancing the decision-making with LLMs. The goal-based policy generation module integrates the discrepancy between agent and LLM policies as an additional loss to minimize, allowing the agent to assimilate distilled prior knowledge while considering potential environmental constraints. We evaluated our framework in various task environments and the results indicate that it facilitates efficient training with a high success rate and remains resilient to state observation noises. For further studies, we consider there are spaces to enhance some modules in the framework. For instance, the hardcoded state captioner could be substituted with autoregressive generation using vision-based foundation models. Additionally, since subgoals for a task are predetermined, feedback and online regeneration could be explored to ensure relevant and appropriately distributed goals throughout task completion.

REFERENCES

[1] R. S. Sutton, "Reinforcement learning: an introduction," *A Bradford Book*, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, IEEE, 2023.

[5] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, IEEE, 2023.

[6] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," in *7th Annual Conference on Robot Learning*, 2023.

[7] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference on Machine Learning*, pp. 12888–12900, PMLR, 2022.

[8] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[9] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*, pp. 2778–2787, PMLR, 2017.

[10] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[13] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.

[14] J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, *et al.*, "Large language models for robotics: Opportunities, challenges, and perspectives," *arXiv preprint arXiv:2401.04334*, 2024.

[15] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," *arXiv preprint arXiv:2303.04129*, 2023.

[16] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.

[17] Z. Wu, Y. Hu, W. Shi, N. Dziri, A. Suhr, P. Ammanabrolu, N. A. Smith, M. Ostendorf, and H. Hajishirzi, "Fine-grained human feedback gives better rewards for language model training," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[18] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Z. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal, S. Kirmani, *et al.*, "How to prompt your robot: A promptbook for manipulation skills with code as policies," in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[19] Z. Zhang, L. Zhou, C. Wang, S. Chen, Y. Wu, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, *et al.*, "Speak foreign languages with your own voice: Cross-lingual neural codec language modeling," *arXiv preprint arXiv:2303.03926*, 2023.

[20] W. Tan, W. Zhang, S. Liu, L. Zheng, X. Wang, and B. An, "True knowledge comes from practice: Aligning llms with embodied environments via reinforcement learning," *arXiv preprint arXiv:2401.14151*, 2024.

[21] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," in *International Conference on Machine Learning*, pp. 3676–3713, PMLR, 2023.

[22] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[23] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas, "Guiding pretraining in reinforcement learning with large language models," in *International Conference on Machine Learning*, pp. 8657–8677, PMLR, 2023.

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

[25] Z. Zhou, B. Hu, C. Zhao, P. Zhang, and B. Liu, "Large language model as a policy teacher for training reinforcement learning agents," in *33rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[26] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

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

[28] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[29] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.

[31] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[32] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[33] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8494–8502, 2018.

[34] E. Triantafyllidis, F. Acero, Z. Liu, and Z. Li, "Hybrid hierarchical learning for solving complex sequential tasks using the robotic manipulation network roman," *Nature Machine Intelligence*, vol. 5, no. 9, pp. 991–1005, 2023.

[35] RootMotion, "Final ik plugin for unity." Unity Asset Store, 2024.