DAIL: Beyond Task Ambiguity for Language-Conditioned Reinforcement Learning

Runpeng Xie[*1], **Quanwei Wang**[*2], **Hao Hu**[3], **Zherui Zhou**[4], **Ni Mu**[2], **Xiyun Li**[5], **Yiqin Yang**[†1], **Shuang Xu**[1], **Qianchuan Zhao**[2], **Bo Xu**[†1]

[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]Department of Automation, Tsinghua University [3]Moonshot AI
[4]Department of Computer Science and Engineering, Washington University [5]Tecent AI Lab
xierunpeng2021@ia.ac.cn, wqw21@mails.tsinghua.edu.cn

Abstract

Comprehending natural language and following human instructions are critical capabilities for intelligent agents. However, the flexibility of linguistic instructions induces substantial ambiguity across language-conditioned tasks, severely degrading algorithmic performance. To address these limitations, we present a novel method named DAIL (Distributional Aligned Learning), featuring two key components: distributional policy and semantic alignment. Specifically, we provide theoretical results that the value distribution estimation mechanism enhances task differentiability. Meanwhile, the semantic alignment module captures the correspondence between trajectories and linguistic instructions. Extensive experimental results on both structured and visual observation benchmarks demonstrate that DAIL effectively resolves instruction ambiguities, achieving superior performance to baseline methods. Our implementation is available at https://github.com/RunpengXie/Distributional-Aligned-Learning.

1 Introduction

Artificial agents are anticipated to master diverse skills while effectively interpreting human instructions and generalizing across various tasks. Therefore, comprehension and following of natural language emerges as critical capabilities for agents in this context. For example, language-conditioned agents have achieved remarkable success in robotic manipulation [36, 6], text-based environments [35, 7], visual navigation [62, 22], and autonomous driving [14, 51]. The fundamental requirement has propelled language-conditioned reinforcement learning (RL) to the forefront of research, which focuses on enabling agents to interpret and execute natural language instructions through RL frameworks.

Recent advancements in the language-conditioned RL domain have focused on bridging the gap between linguistic understanding and decision-making processes, aiming to create agents capable of executing complex instructions with human-like adaptability. For example, some works [12, 4] integrate language-conditioned policy with trial-and-error learning, significantly improving the performance and sample efficiency in robot task acquisition. Meanwhile, some studies [18, 19] leverage expert demonstrations to map language instructions to reward signals directly to address the issue of sparse rewards in language-conditioned RL. However, linguistic instructions exhibit high flexibility, which induces exponential growth in task space. In this case, identical tasks may have

[*]Equal contribution.
[†]Correspondence to Yiqin Yang and Bo Xu.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

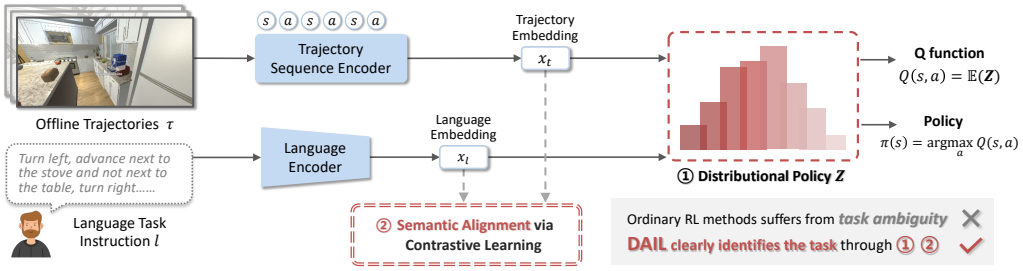

Figure 1: The framework of our method. The key ideas are: (1) use the distributional language-guided policy to aid task discrimination, (2) use the trajectory-wise semantic alignment module to extract task representation.

divergent expressions, while distinct tasks share overlapping language instructions. This variability makes language-conditioned RL methods face the significant challenge of task ambiguity, which hinders the agent from discerning the connection between rewards and task objectives, thereby significantly impairing learning efficiency.

To solve this issue, we propose a novel method, DAIL, which consists of two main components: a distributional language-guided policy and a trajectory-wise semantic alignment module. Specifically, the distributional policy module [3] estimates the value distribution, preserving more information to aid task discrimination. Theoretically, we analyze the sample complexity required to guarantee task disambiguation and establish that distributional estimation methods are sample-efficient in offline settings. On the other hand, the semantic alignment module constrains the language instruction representations by maximizing the mutual information between trajectories and the instructions, thereby achieving better differentiation across instructions. With theoretical guarantees, the two modules enable precise disambiguation and execution of linguistic instructions, thereby improving learning efficiency.

We conduct extensive experiments on both structured observation [10] and visual observation [54] benchmarks. This design is to validate the external validity of the DAIL agent, progressing from less complex structured inputs to more complex and expressive visual observations. The experimental results show that DAIL outperforms the state-of-the-art language-conditioned RL methods in both benchmarks. Further, the visualization analysis demonstrates that DAIL can learn a non-ambiguous task representation compared with baselines. Our main contributions are summarized as follows:

- First, we highlight the critical issue of task ambiguity and empirically analyze the limitations of current mainstream methods. We define the task distinction in our setting and analyze the sample complexity to avoid task ambiguity theoretically.
- Second, we propose DAIL, a simple yet efficient language-conditioned learning framework, which addresses the task ambiguity issue based on distributional policy and semantic alignment.
- Lastly, we conduct extensive experiments to show that DAIL significantly outperforms conventional language-conditioned methods. The results indicate that by improving task discrimination, we can effectively mitigate the task ambiguity issue, thereby broadening the application of language-conditioned RL.

2 Preliminaries

Language-conditioned RL Based on contextual markov decision process (CMDP) [20], we consider Language-conditioned Markov Decision Process (LCMDP) as a model consisting of a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mathcal{L}, p_0, p_l)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $\mathcal{L}$ is the language instruction space, $P(s'|s, a, l)$ represents the probabilistic transition model, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{L} \to \mathbb{R}$ is the reward function conditioned on language instructions and γ is the discount factor. p_0 represents the probability distribution of the initial state, and p_l denotes the probability distribution of language instruction. We establish language instructions l as task descriptors, and each instruction uniquely specifies a task.

Language-conditioned RL aims to obtain a policy $\pi(\cdot|s,l)$ that maximizes the cumulative discounted returns under a specific distribution of language instructions:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, l) \right], \tag{1}$$

where $s_0 \sim p_0(\cdot), l \sim p_l(\cdot), a_t \sim \pi(\cdot|s_t, l)$ and $s_{t+1} \sim P(\cdot|s_t, a_t, l)$. The temporal difference loss in language-conditioned RL is adapted as follows:

$$L_{\text{TD}}(\theta) = \mathbb{E}_{(s,a,s',l)\sim\mathcal{D}}[(r(s,a,l) + \gamma \max_{a'} Q_{\hat{\theta}}(s', a', l) - Q_\theta(s,a,l))^2] \tag{2}$$

where $Q_\theta(s,a,l)$ is the parameterized Q-function conditioned language instruction l, and $Q_{\hat{\theta}}(s,a,l)$ is the target Q-network.

Offline RL Due to the high cost of real-time interaction with the environment, we consider the offline learning setting, in which we learns a policy π without interacting with an environment. Rather, the learning is based on a dataset $\mathcal{D}$ generated by a behavior policy π_β. One of the major challenges in offline RL is the issue of distributional shift [17], where the learned policy is different from the behavioral policy. Existing offline RL methods apply various forms of regularization to limit the deviation of the current learned policy:

$$\pi^* = \arg\max_\pi \left[J_\mathcal{D}(\pi) - \alpha D(\pi, \pi_\beta) \right], \tag{3}$$

where $J_\mathcal{D}(\pi)$ is the cumulative discounted return of policy π on the empirical MDP induced by the dataset $\mathcal{D}$, and $D(\pi, \pi_\beta)$ is a divergence measure between π and π_β. As for the language-conditioned task, we will provide the language instruction in the evaluation. To make our writing more concise, let τ be a full trajectory, and τ_t be the trajectory ending at time-step t. Let $p_\mathcal{D}(\tau, l)$ represents the joint distribution of trajectory-instruction pairs in the dataset $\mathcal{D}$.

3 Ambiguity on Language-Conditioned Tasks

In practical tasks, language instructions have high flexibility. For example, similar tasks may employ divergent expressions, while distinct tasks share overlapping language instructions. The variability induces exponential growth of the task space. Therefore, when the number of language instructions increases, the agent is required to accurately identify the tasks; otherwise, it will significantly affect the agent's performance. We name this issue the ambiguity in language-conditioned tasks. We give a formal definition of semantics distinction from instructions as follows.

Definition 1 (Semantics Instructions Distinction)**.** *In a multi-task RL setting with known task instruction space $\mathcal{L}$ and unknown semantics space $\mathcal{G}$, for a task distinction threshold δ and sub-optimality gap ϵ, two task instructions $l_i, l_j \in \mathcal{L}$ are considered with **different underlying semantics**, $g_i \not\leftrightarrow g_j$, if the expected Q-values under any shared ϵ-optimal policy π satisfy:*

$$\mathbb{E}_\pi \left[|Q_\pi(s,a,l_i) - Q_\pi(s,a,l_j)| \right] \geq \delta.$$

Conversely, if

$$\mathbb{E}_\pi \left[|Q_\pi(s,a,l_i) - Q_\pi(s,a,l_j)| \right] \leq \delta/2,$$

*then l_i and l_j are considered with the **same underlying semantics**, $g_i \leftrightarrow g_j$, where $s_0 \sim p_0(\cdot), a \sim \pi(\cdot|s), s' \sim p(\cdot|s,a)$. $V_\pi(s) \geq V^*(s) - \epsilon, \forall s \in \mathcal{S}$, V^* is optimal value function.*

To make this point more straightforward, we introduce a toy experiment based on the Minigrid environment [11]. The setup consists of 10 accessible goal positions $\mathcal{G} = \{g_0, g_1, ..., g_9\}$, where $\mathcal{G}$ represents the set of all possible goal positions. The agent (red triangle-shaped) must follow a given instruction $l \in \mathcal{L}$ to navigate to a specific goal position (Left of Figure 2). We simulate instructions using numerical task IDs, making $\mathcal{L} \subset \mathbb{N}$. We employ a random mapping $F : \mathcal{L} \to \mathcal{G}$ to assign each l to goal position g, which simulates semantics of instructions and is hidden from the agent (Middle of Figure 2). With these settings, we can control the number of instructions $|\mathcal{L}|$ by simply adjusting the set of valid task IDs and the mappings. We conduct 10 experiments, varying the number of instructions from 1 to 2^9. For each experiment, we generate a new mapping F and collect 1024 random trajectories as the offline dataset.

We evaluate the standard offline RL algorithm, CQL [30], on the above settings. In addition, to test how model size affects the task ambiguity, we create an enhanced version called CQL-double by

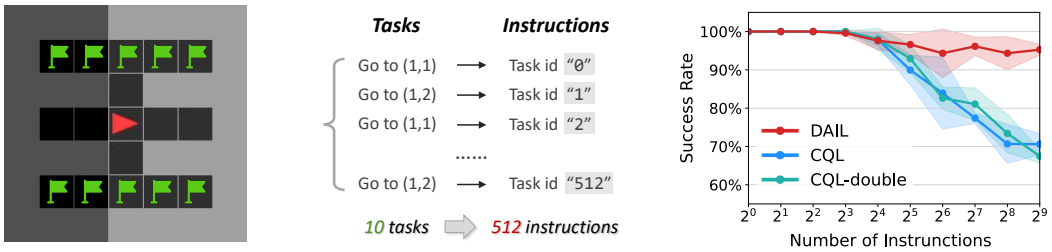

Figure 2: **Left:** The green flags in the map denote the accessible goals. **Middle:** An illustration of the mapping between goal positions and instructions. **Right:** Average success rates over 100 evaluations for each number of instructions and 3 seeds.

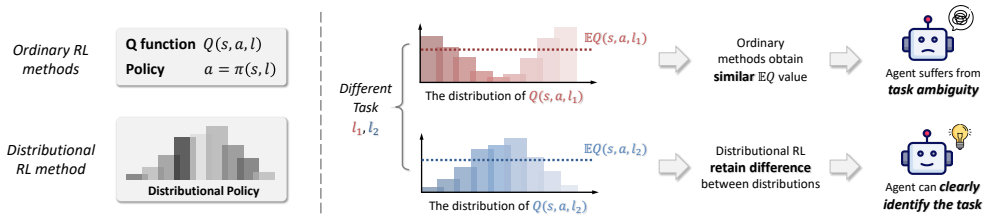

Figure 3: An illustrative case where $Q(s,a)$ functions in two tasks share the same expectations but have different distributions. In this case, traditional RL cannot discriminate between the tasks while distributional RL can. Please refer to Appendix D for more details.

doubling its parameters. The experimental results in Figure 2 show that if the number of instructions is small (e.g., under 16), agents can understand instructions and reach the goal position with high success. However, as instructions multiply, the number of demonstrations for each instruction declines, which increases task ambiguity and consequently makes CQL and CQL-double performance drop significantly. These observations reveal fundamental limitations in language-conditioned RL regarding task ambiguity, highlighting that model scaling alone remains insufficient to resolve this challenge. For detailed information on the toy experiment, please refer to Appendix E.

4 Method

To address the issue of task ambiguity, we hope to improve the algorithm's ability to distinguish tasks. In this section, we propose the Distributional Aligned Learning algorithm (DAIL), which enhances the task differentiability from policy and representation. Specifically, DAIL adopts the distributional language-guided policy to estimate the value distribution, preserving more information to aid task discrimination. On the other hand, DAIL uses the trajectory-wise semantic alignment module to extract task representations and help discriminate different instructions by maximizing the mutual information between trajectories and instructions. We show the overall framework of DAIL in Figure 1 and Algorithm 1 in Appendix B.

4.1 Distributional Language-Guided Policy

Current RL methods aim to estimate the expectation of the cumulative discounted reward. However, as shown in Section 3, when the number of instructions increases while keeping the number of actually-distinct tasks constant, the estimated expectation for different task instructions becomes similar. As a result, it is difficult for the agent to complete the task instructions accurately. Differently, the distributional technique addresses this issue by calculating the distribution of the cumulative discounted reward, which is more distinguishable than expectation, as illustrated through an extreme yet straightforward example in Figure 3. For a fixed policy π, let the random variable Z^π represent

the cumulative discounted reward obtained along the policy π, and it has the following relationships:

$$V^\pi(s,l) := \mathbb{E}[Z^\pi(s,l)] = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t,l)|s_0=s,\pi\right]$$
$$Q^\pi(s,a,l) := \mathbb{E}[Z^\pi(s,a,l)] = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t,l)|s_0=s,a_0=a,\pi\right], \tag{4}$$

Let $\mathcal{Z}$ denote the space of value distributions. In the following, for simplicity of notation, we denote $Z^\pi \in \mathcal{Z}$ as Z. Instead of estimating the expectation, we calculate the probability distribution of the random variable Z:

$$\mathcal{T}^\pi Z(s,a,l) :\overset{D}{=} R(s,a,l) + \gamma Z(s',a',l), \tag{5}$$

where $\mathcal{T}$ is the distributional Bellman operator, and $A :\overset{D}{=} B$ denotes that A equals B by probability laws. $R \in \mathcal{Z}$ is a function depicting the reward distributions. Since modeling continuous distributions is challenging, we can discretize the value function distribution Z and train it by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{Dist}}(\theta) = \mathbb{E}_{(\tau,l)\sim p_{\mathcal{D}}(\cdot,\cdot)}\frac{1}{T}\sum_{t=0}^{T-1} D_{\text{KL}}\left(\Phi\hat{\mathcal{T}}Z_{\hat{\theta}}(s_t,a_t,l)\|Z_\theta(s_t,a_t,l)\right), \tag{6}$$

where T is the trajectory length, $\Phi\hat{\mathcal{T}}$ is the sample Bellman update, which projects the value function distribution onto the parametric discrete distribution. Z_θ and $Z_{\hat{\theta}}$ are estimated value distributions parameterized by θ and $\hat{\theta}$, respectively. By optimizing Equation 6, we obtain the distribution Z_θ that exhibits strong discriminative power across different instructions l. Please refer to Appendix C for the details.

In addition, we conduct the following theoretical analysis. Let $n_{\text{value}}, n_{\text{dist}}$ denote the number of samples needed to avoid task ambiguity for value-based and distributional settings, respectively. As shown in Theorem 1 and Corollary 2, distributional RL achieves better sample efficiency compared with estimating the expectation when the number of tasks is sufficiently large, $n_{\text{value}} \geq n_{\text{dist}}$. Proofs and further details can be found in Appendix D.

Theorem 1 (Sample Complexity for Task Instruction Disambiguation)**.** *Consider an offline multi-task RL setting with M distinct tasks (with different semantics). In direct Q-value estimate setting, suppose $Q(s,a,l) \in [0, Q_{\max}],\ \forall(s,a,l) \sim \mathcal{D}$ with finite $Q_{\max}$, and the task distinction threshold is $\delta > 0$. When the number of training samples n_{value} satisfies:*

$$n_{\text{value}} \geq \frac{C_{\text{value}}\log(3M^2/\eta)}{\delta^2}.$$

The mean value estimate algorithm achieves task-level semantic disambiguation with confidence at least $1-\eta$. In the distributional RL setting, let $Z(s,a,l)$ denote the learned return distribution, and suppose the task distinction threshold is given by a 1-Wasserstein distance $d > 0$. Then, to ensure semantic disambiguation of task instructions with confidence at least $1-\eta$, it suffices that:

$$n_{\text{dist}} \geq \frac{C_{\text{dist}}\log(3M^2/\eta)}{d^2},$$

where $C_{\text{value}}, C_{\text{dist}} > 0$ are universal constants depending on certain attributes of Q-value distribution.

4.2 Trajectory-Wise Semantic Alignment

In this subsection, we focus on learning trajectory embedding that enhances the correspondence between instructions and trajectories, thereby reducing task ambiguity at the representational level. To achieve this, we attempt to maximize the mutual information between the language instructions and trajectory:

$$w = \max_w I(X_\tau(w); X_l(w)), \tag{7}$$

where w is the parameter of the representation module, X_τ, X_l are the random variables of trajectory embedding and language instruction, respectively. For the trajectory embedding x_τ, we adopt the

sequence model. Specifically, we first use $u_w(\cdot,\cdot)$ to encode the state-action pairs, and then pass the sequence of embeddings through a sequence model $h_w(\cdot)$:

$$x_\tau = h_w(u_w(s_1, a_1), u_w(s_2, a_2), ..., u_w(s_T, a_T)). \tag{8}$$

As for the language instruction representation, we employ a language encoder to tokenize and encode the instructions into language embeddings x_l. Please refer to Appendix F for the detailed model architecture. Let $f_w(\tau, l) = \frac{x_\tau^T x_l}{||x_\tau||\ ||x_l||}$ measure the similarity between the trajectory embedding and language instruction representation. Since minimizing InfoNCE is equivalent to maximizing the lower bound of the mutual information [45], we can maximize the mutual information in Equation 7 by minimizing the following NCE loss:

$$\mathcal{L}_c(w) = \mathbb{E}_{\substack{(\tau,l^+)\sim p_\mathcal{D}(\cdot,\cdot) \\ l^-\sim p_l(\cdot)}}[\log\sigma(f_w(\tau, l^+)) + \log(1 - \sigma(f_w(\tau, l^-)))], \tag{9}$$

where l^+ denotes the positive samples, which are sampled from the distribution of trajectory-instruction pair $p_\mathcal{D}(\cdot,\cdot)$. l^- denotes the negative samples, generated by uniform sampling over the language instructions from the offline datasets.

4.3 Practical Implementation

To address the partial observability issue in practical implementation, the trajectory encoding x_t defined in Equation 8 is incorporated as an additional input (s_t, a_t, x_{t-1}, l). The Equation 5 is transformed into correspondingly:

$$\mathcal{T}^\pi Z(s_t, a_t, x_{t-1}, l) :\overset{D}{=} R(s_t, a_t, l) + \gamma Z(s_{t+1}, a_{t+1}, x_t, l). \tag{10}$$

In practice, we estimate the value distribution Z_θ with discrete distribution, using a set of atoms $\{z_i = V_{\text{MIN}} + i\Delta z\}_{i=1}^{M-1}, \Delta z = \frac{V_{\text{MAX}} - V_{\text{MIN}}}{M-1}$. $V_{\text{MIN}}, V_{\text{MAX}} \in \mathbb{R}$ are the lower and upper bounds of the distributions with support, respectively, and $M \in \mathbb{N}$ is the number of atoms. Then the discrete value distribution is modeled as:

$$Z_\theta(s_t, a_t, x_{t-1}, l) = z_i, \text{ with probability } p_i(s_t, a_t, x_{t-1}, l) := \frac{e^{\theta_i(s_t, a_t, x_{t-1}, l)}}{\sum_j e^{\theta_j(s_t, a_t, x_{t-1}, l)}} \tag{11}$$

where $\theta_i : \mathcal{S} \times \mathcal{A} \times \mathcal{X} \times \mathcal{L} \to \mathbb{R}$ is a parametric model employed to approximate Z_θ and updated by Equation 6. Based on the analysis in Section 4.1, we compute Q-function with distributional mechanism by $Q_\theta(s_t, a_t, x_{t-1}, l) = \mathbb{E}[Z_\theta(s_t, a_t, x_{t-1}, l)] = \sum_{i=0}^{M-1} p_i(s_t, a_t, x_{t-1}, l) z_i$. Please refer to Appendix C for the details.

In addition, we consider the offline learning setting in this work, which learns a policy without interacting with the environment. For this reason, we adopt the standard offline learning term, CQL($\mathcal{H}$) [30], to address the distribution shift issue in offline RL learning [17]:

$$\mathcal{L}_{\text{CQL}}(\theta) = \mathbb{E}_{(\tau,l)\sim p_\mathcal{D}(\cdot,\cdot)} \frac{1}{T} \sum_{t=0}^{T-1} \log \sum_a \exp(Q_\theta(s_t, a, x_{t-1}, l)) - \mathbb{E}_{a\sim\pi_\beta(a|s)}[Q_\theta(s_t, a, x_{t-1}, l)], \tag{12}$$

Combining all the above loss functions, the total loss function is:

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{Dist}} + \lambda\mathcal{L}_c + \alpha\mathcal{L}_{\text{CQL}} \tag{13}$$

where λ, α are weights of the trajectory-wise semantic alignment module and the offline learning term, respectively. Finally, we select the action with the highest Q-value:

$$a_t^* = \underset{a}{\text{argmax}} Q_\theta(s_t, a, x_{t-1}, l) \tag{14}$$

The complete process of our method is shown in Algorithm 1 in Appendix B and Figure 14 in Appendix F.

5 Experiments

We designed our experiments to answer the following questions: *Q1*: How does DAIL compare to other state-of-the-art methods on offline language-conditioned tasks? *Q2*: How does DAIL perform as the number of tasks explodes? *Q3*: Can DAIL learn a meaningful alignment between trajectories and instructions? *Q4*: What is the contribution of each of the proposed techniques in DAIL?

Figure 4: **Left:** An example of the `SynthLoc` task in the BabyAI environment: "put the yellow key next to a green ball". **Right:** Two example scenes with instructions from ALFRED. Agents are asked to finish specific tasks in the 3D household environment according to received instructions.

Table 1: Success rate of out-of-distribution BabyAI tasks. Each score is evaluated over 3 seeds.

Tasks	GCBC	BC-Z	GRIF	IQL	CQL	DAIL (ours)
Open	94.4±2.5	93.7±1.0	95.9±1.7	98.0±0.4	98.8±0.5	**99.0±0.2**
Goto	90.3±1.6	76.9±3.0	88.8±2.6	86.1±1.2	88.9±2.1	**91.3±1.0**
PickUp	78.4±2.1	45.4±1.5	75.6±3.9	70.4±3.6	71.9±2.2	**87.6±2.0**
PutNext	27.4±1.6	11.2±3.3	22.5±2.7	21.4±3.1	27.6±0.8	**49.1±1.8**
All	74.1±0.7	57.9±1.8	71.2±2.6	69.7±2.3	72.6±0.4	**81.7±1.3**

5.1 Experimental Setting

We evaluate various methods in language-conditioned tasks, with detailed introductions as follows:

BabyAI [10] is a language learning research platform with different levels of tasks, shown on the Left of Figure 4. We choose level `SynthLoc` for evaluation, which contains four major groups of tasks `Open`, `Goto`, `PickUp`, and `PutNext`. The agent is asked to operate with an assigned object, like "open a red door" or "put the gray box in front of you next to the blue key". Tasks vary as the colors, types, or locations of objectives change, making around 6000 different tasks in total. To evaluate the algorithms' generalizations, we divide the task space into in-distribution tasks and out-of-distribution tasks. In-distribution tasks account for approximately 60% of the total number of tasks, with a total of 3325. For the offline learning, we construct an offline dataset with 50k expert trajectories, 50k imitation learning agent trajectories, and 25k random trajectories.

ALFRED [54] benchmarks sequential decision-making tasks involving household activities (e.g, cleaning, heating food) through language instructions and first-person vision, shown on the Right of Figure 4. The dataset provides 8055 expert demonstrations with 25k human-annotated language instructions detailing both high-level goals and sub-goal step-by-step guidance. As our work primarily focuses on low-level policy learning rather than high-level planning, we specifically concentrate on the `GOTO` sub-goal setting for our evaluation. In this task set, the agent must go to specific locations according to instructions like "Move to other side of couch on the right side of the table before the door". To simulate the presence of noisy data in real-world applications, we augment the training set with 30k random-agent trajectories, resulting in 97896 total trajectories with 53442 unique instructions across 108 household scenes.

Baselines We choose three state-of-the-art offline RL algorithms, CQL [30], IQL [28], and adapt them into a language-conditioned manner as our RL baselines. In addition, for imitation baselines, we include GCBC [15], BC-Z [25] and GRIF [42], which are also adapted into a language-conditioned manner by the benchmarks [10, 54]. Further details about baselines are shown in Appendix E.

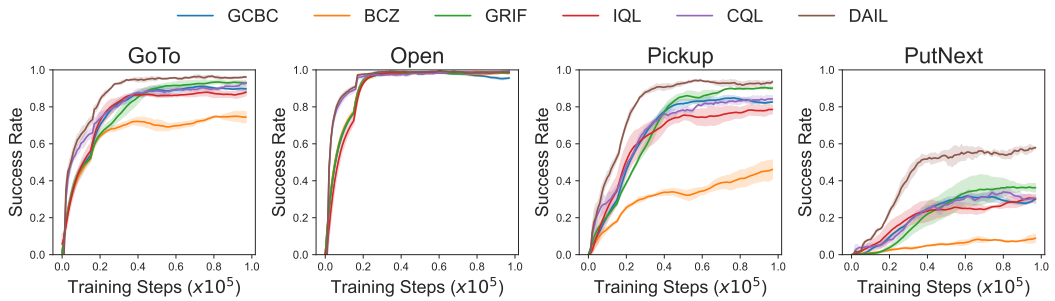

Figure 5: Training curves on in-distribution BabyAI tasks. Success rates are evaluated over 3 seeds.

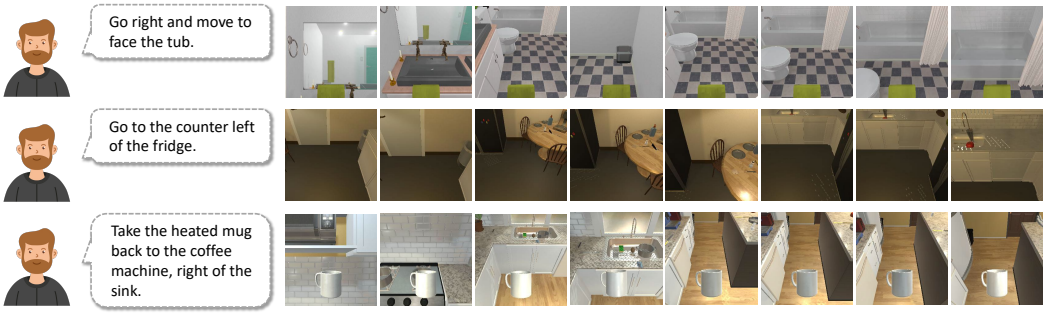

Figure 6: Example trajectories of DAIL in ALFRED validation set instructions with varied instructions and scenes.

5.2 Main Results

BabyAI experimental results. We provide training curves of in-distribution tasks in Figure 5 and out-of-distribution experimental results in Table 1. The complete results are shown in Table 5 in Appendix H. It shows that our method achieves superior performance compared with other baselines, especially in the `PutNext` task category. Vanilla offline RL algorithms like CQL and IQL underperform compared to imitation learning methods like GCBC, which we attribute to the adverse impact of task ambiguity on RL-based approaches as discussed in Theorem 1. On the other hand, modified algorithms designed for language-conditioned IL (BC-Z and GRIF) perform poorly under our setting. This is primarily because their contrastive learning objectives are not robust in the presence of noisy or suboptimal data. In contrast, our alignment-based approach, built on an offline RL framework, maintains strong performance. We further evaluate various algorithms on a more challenging dataset with fewer expert trajectories (Table 6 in Appendix H). Our method still achieves the optimal results. Further details about the experiment are shown in Appendix E.

ALFRED experimental results. The complexity and variety of instructions in ALFRED challenge the agent's ability to discern instructions. The experimental results in Table 2 show that our approach demonstrates the highest success rate (SR), validating its positive impact on instruction recognition capability compared to other baselines. GCBC exhibits poorer resistance to suboptimal data, resulting in performance comparable to other RL baseline models. We also report path-length weighted success rates (PLW SR), which considers the length of expert demonstration and demonstrates the effectiveness of the behavior policy following [54]. We further illustrate the observation trajectories generated by DAIL under the validation set instructions of ALFRED with varied instructions and scenes in Figure 6. This visualization demonstrates DAIL's robust task execution in complex scenes under diverse instructions, with additional trajectory demonstrations provided in Appendix I.

5.3 Visualization

To better understand how our proposed method enhances the performance, we show the t-SNE [57] results of the language instructions internal embedding on BabyAI `SynthLoc`. Each point repre-

Table 2: Success rate (SR) and path-length weighted success scores (PLW SR) in the ALFRED tasks. The results are shown on the training set and validation set respectively. Each score is evaluated over 3 seeds.

Tasks	GCBC	BC-Z	GRIF	IQL	CQL	DAIL (ours)
SR (Training)	87.9±2.4	86.5±0.8	87.8±1.6	88.4±0.6	87.2±1.2	**92.3±2.2**
PLW SR (Training)	84.3±2.6	82.3±2.7	82.3±2.6	84.4±2.7	83.0±2.9	**90.2±3.1**
SR (Validation)	47.1±2.4	43.0±2.5	48.6±1.0	52.0±3.0	50.4±1.7	**56.8±2.1**
PLW SR (Validation)	40.5±3.1	39.5±2.2	43.2±2.5	47.0±3.2	44.4±1.3	**50.3±2.4**

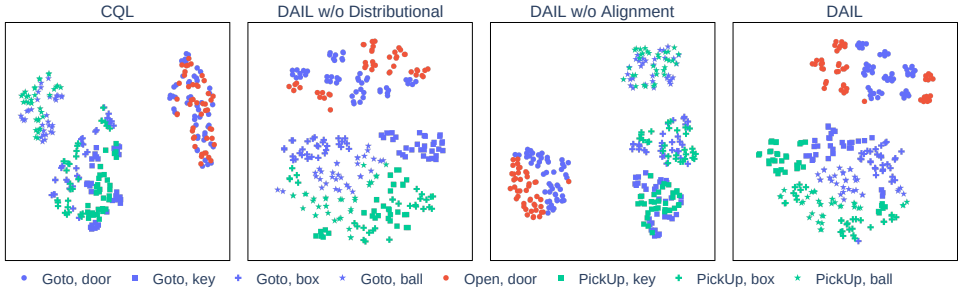

Figure 7: T-SNE visualization of instructions in BabyAI tasks. The figure distinguishes between different task categories (e.g., Open) and target object types (e.g., box), using marker colors and shapes to represent each separately. For example, "pick up a red box" corresponds to ■, "go to a green box" corresponds to +.

sents the internal representation of a unique language instruction within the policy network. The experimental results in Figure 7 show that vanilla RL can only marginally separate some broad task categories, but fails at distinguishing `PickUp` and `Goto`. Moreover, it is completely confused between `Open` (door) and `Goto` (door). Alignment and distributional methods help discriminate tasks between and within categories. Our method substantially enhances task representation by clearly differentiating between task categories and target object types (for example, separating ● and ●, ⋆ and ⋆).

For more precise visualization, we use the same method to visualize the task representations of algorithms on the same task category, illustrated in Figure 17 in Appendix G. Our method effectively distinguishes all tasks in these two categories without overlapping confusion and group similar tasks into smaller clusters (■, ●, +, ⋆ and so on). We conduct the visualization experiments under 3 training seeds, all of which yield consistent experimental conclusions. Moreover, we quantitatively measure the clustering quality of our proposed components with the Silhouette score[52] using the learned language embeddings. The experimental results in Table 8 in Appendix H demonstrate that our method effectively enhances clustering performance, which aligns with the experimental findings from the visualization. Details of the quantitative evaluation and more visualizations of task representations can be referred in Appendix H and G, respectively.

5.4 Ablation Studies

Ablation of components. To study the contribution of each component in our learning framework, we conduct the following ablation study. We compare the performance of algorithms that only apply trajectory-wise alignment or distributional language-guided policy alone with our method on `SynthLoc`. The experimental results in Table 3 show that both modules significantly improve the performance over vanilla CQL on in-distribution and out-of-distribution tasks. Further, combining both components can achieve the best performance compared to other approaches.

Ablation of alignment weight λ. In Equation 13, λ is the weight of the alignment loss. For this reason, we evaluate the choice of λ in BabyAI tasks with various λ. The experimental results in Figure 8 show that there is no noticeable performance difference between $\lambda = 0.2$ and $\lambda = 1$. The

Table 3: Ablation results for components of our method. Each score is evaluated over 3 seeds.

Algorithm	In Distribution		Out of Distribution	
	PutNext	All	PutNext	All
CQL	25.6±2.5	78.1±1.6	27.6±0.8	72.6±0.4
DAIL w/o Distributional	39.6±0.6	83.3±0.2	39.3±1.7	77.3±0.8
DAIL w/o Alignment	39.1±1.0	82.2±1.3	32.0±0.9	75.1±1.6
DAIL	**57.9±0.9**	**89.2±0.5**	**49.1±1.8**	**81.7±1.3**

performance begins to degrade when the influence of the loss is either too small ($\lambda = 0.01$) or too large ($\lambda = 2$). Therefore, we recommend choosing a value between 0.2 and 1.

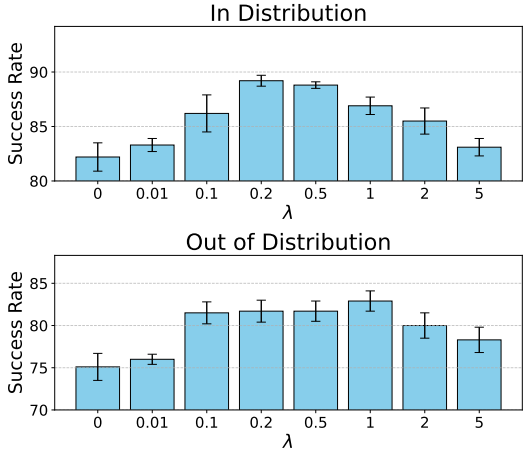

Figure 8: Percent difference of the performance of an ablation over λ, compared to the average results of all λs evaluated.

6 Conclusion and Future Work

In this work, we aim to address the task discrimination and comprehension challenges in language-conditioned RL. To achieve this, we propose a novel method called DAIL, which incorporates a distributional language-guided policy and a trajectory-wise semantic alignment module. We first theoretically demonstrate that distributional RL methods are more sample-efficient than traditional RL methods for learning in language-conditioned tasks. We then perform extensive experiments on both structured and visual observation benchmarks. The experimental results and visualization analysis show that our method learns language instruction representations with clearer semantics. The simplicity and robustness of DAIL make it easily adaptable as a plug-in for other methods tackling language-conditioned problems. While our method is theoretically and empirically validated, demonstrating its significant advantages in language-conditioned tasks, several limitations remain. First, due to experimental constraints, we are unable to test our method in real-world scenes to further validate the method's effectiveness and robustness, which we will consider in future work. Second, our theoretical analysis of distributional RL's advantages relies on the assumptions of offline RL. Although we believe that this approach remains effective in online settings, we defer the theoretical analysis to future work.

Acknowledgments and Disclosure of Funding

This work is supported by Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA27040200)

References

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International conference on machine learning*, pages 166–175. PMLR, 2017.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[3] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[4] Zhenshan Bing, Alexander Koch, Xiangtong Yao, Kai Huang, and Alois Knoll. Meta-reinforcement learning via language instructions. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5985–5991. IEEE, 2023.

[5] Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Summer school on machine learning*, pages 208–240. Springer, 2003.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

[8] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*, pages 1430–1440. PMLR, 2021.

[9] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

[11] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

[12] John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, Jacob Andreas, John DeNero, Pieter Abbeel, and Sergey Levine. Guiding policies with language via meta-learning. *arXiv preprint arXiv:1811.07882*, 2018.

[13] Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Language-conditioned goal generation: a new approach to language grounding for rl. *arXiv preprint arXiv:2006.07043*, 2020.

[14] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, and Ziran Wang. Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 902–909, 2024.

[15] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[16] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

[17] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[18] Prasoon Goyal, Scott Niekum, and Raymond Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. In *Conference on Robot Learning*, pages 485–497. PMLR, 2021.

[19] Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*, 2019.

[20] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

[23] David Yu-Tung Hui, Maxime Chevalier-Boisvert, Dzmitry Bahdanau, and Yoshua Bengio. Babyai 1.1. *arXiv preprint arXiv:2007.12770*, 2020.

[24] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[25] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[26] Russell Kaplan, Christopher Sauer, and Alexander Sosa. Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539*, 2017.

[27] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[29] Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022.

[30] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[31] Jing Lei. Convergence and concentration of empirical measures under wasserstein distance in unbounded functional spaces. 2020.

[32] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

[33] Alexander Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

[34] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

[35] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[36] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

[37] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[38] Jason Yecheng Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. Offline goal-conditioned reinforcement learning via f-advantage regression. *Advances in neural information processing systems*, 35:310–323, 2022.

[39] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

[40] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781, 2013.

[41] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. *arXiv preprint arXiv:1704.08795*, 2017.

[42] 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. Goal representations for instruction following: A semi-supervised language interface to control. In *Conference on Robot Learning*, pages 3894–3908. PMLR, 2023.

[43] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[44] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[45] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[46] Jing-Cheng Pang, Xin-Yu Yang, Si-Hang Yang, and Yang Yu. Natural language-conditioned reinforcement learning with inside-out task language development and translation. *arXiv preprint arXiv:2302.09368*, 2023.

[47] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

[48] Shaohui Peng, Xing Hu, Rui Zhang, Jiaming Guo, Qi Yi, Ruizhi Chen, Zidong Du, Ling Li, Qi Guo, and Yunji Chen. Conceptual reinforcement learning for language-conditioned tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9426–9434, 2023.

[49] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[51] Junha Roh, Chris Paxton, Andrzej Pronobis, Ali Farhadi, and Dieter Fox. Conditional driving from natural language instructions. In *Conference on Robot Learning*, pages 540–551. PMLR, 2020.

[52] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[53] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[54] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[55] Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *Robotics: Science and Systems*, 2015.

[56] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[57] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[58] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[59] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478*, 2022.

[60] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11501–11516, 2021.

[61] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21579–21590, 2020.

[62] Hongkuan Zhou, Xiangtong Yao, Yuan Meng, Siming Sun, Zhenshan BIng, Kai Huang, and Alois Knoll. Language-conditioned learning for robotic manipulation: A survey. *arXiv preprint arXiv:2312.10807*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately summarize the paper's contributions, particularly in highlighting the proposed method's effectiveness in reducing task ambiguity and its robustness under noisy, offline settings.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Section 6, including experimental settings and theoretical assumptions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In the main text, we mainly conduct theoretical analysis in Section 3 and 4.1. The paper provides the full set of assumptions and a complete proof in Appendix D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We discuss the implementation of our method in Section 4.3 and details of experiments in Section 5 and Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide data and code in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We discuss the experimental details of experiments in Section 5 and Appendix E. We provide the hyperparameters and network architectures in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the standard deviation in all the experimental results in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information or computer resources are provided in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our research conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work focuses on addressing task ambiguity in offline reinforcement learning; this work does not present any foreseeable societal consequences.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit the existing assets and respect the terms of use in our research.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The model and code we propose are well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Related Work

Language-conditioned Agents. Two primary approaches for enabling an agent to follow human instructions are reinforcement learning (RL) and imitation learning (IL). Some approaches in language-conditioned RL focus on aligning language with policies or performing feature extraction to integrate language information into the learning process [9, 26, 1, 41, 53, 4], while others prioritize reward shaping [55, 16, 19, 18, 53] to formulate language-conditioned reward functions. However, most methods rely on simulators and are limited in scalability when applied in the offline setting. On the other hand, IL is designed to learn from large datasets but is heavily dependent on the quality of the data [24]. To mitigate this dependency and enhance data efficiency, recent works have leveraged crowd-sourced annotations [43], multimodal alignment [25, 42, 44], or carefully designed model architectures [39, 56]. These methods enhance IL by supplying richer supervisory signals or building more robust model structures. Despite the success of IL methods in solving complex tasks, the scale and quality of the dataset remain significant constraints, particularly without external annotations or auxiliary datasets.

Offline Goal-conditioned RL. Learning a task-specific policy from demonstrations with different goals presents a significant challenge in offline goal-conditioned reinforcement learning (GCRL). By relabeling trajectories and treating intermediate states as additional goal states [2, 32, 33, 59, 60, 38], offline GCRL has achieved notable improvements in sample efficiency. However, this approach is challenging to replicate in language-conditioned settings. First, replicating a specific goal state is particularly challenging in environments characterized by randomness or partial observability. Second, directly using goal states that lack semantic context as labels proves ineffective for learning. While some works use language instructions to guide planning [13, 48, 46], the ambiguity of language can complicate this process. Several studies have applied hierarchical RL to guide policy through subgoals [32, 61, 8, 47, 34] and demonstrate strong performance. However, they rely heavily on a high-level policy that accurately decomposes language instructions into subgoals.

B Algorithm

Algorithm 1 Distributional Aligned Learning

Require: Offline dataset $\mathcal{D} = \{(\tau, l)\}$, target network update frequency K_{update} and support atoms Z_{atoms}.

Initialize policy parameters θ and target policy parameters $\hat{\theta}$.

for each gradient step **do**

  Sample batch $\mathcal{B} = \{(\tau, l)\}_{i=1}^{N} \sim \mathcal{D}$

  Encode instructions: $\{x_l\}_{i=1}^{N}$

  Compute the history information $\{x_0, x_1, .., x_T\}_{i=1}^{N}$ using (8)

  // Distributional Language-Guide Policy

  for each transition $(s_t, a_t, r_t, s_{t+1}, l)$ in batch $\mathcal{B}$ **do**

    Compute estimated value distribution $Z_\theta(s_t, a_t, x_{t-1}, l)$ using (15)

    Compute projected update $\Phi\hat{\mathcal{T}}Z_\theta(s, a, x, l)$ using (18)

    $l_{\text{Dist}}(\theta) \leftarrow D_{KL}(\Phi\hat{\mathcal{T}}Z_{\hat{\theta}}(s_t, a_t, x_{t-1}, l)||Z_\theta(s_t, a_t, x_{t-1}, l))$

  end for

  // Trajectory-Wise Semantic Alignment

  for each trajectory-instruction pair (τ, l) **do**

    for each pair (τ', l') other than (τ, l) **do**

      View l' as negative instruction l^-, and compute contrastive loss l_c by using (9)

    end for

  end for

  Add up the losses above to get $\mathcal{L}_{\text{Dist}} \leftarrow \sum l_{\text{Dist}}$, and $\mathcal{L}_c \leftarrow \sum_{(\tau,l)} l_c$

  Compute conservative Q-learning loss $\mathcal{L}_{\text{CQL}}$ using (12)

  $\mathcal{L}_{\text{tot}} \leftarrow \mathcal{L}_{\text{Dist}} + \lambda\mathcal{L}_c + \alpha\mathcal{L}_{\text{CQL}}$

  Update $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_{\text{tot}}(\theta)$

  if step % K_{update}=0 **then**

    Update target network: $\hat{\theta} \leftarrow \theta$

  end if

end for

Extract policy: $\pi(s, x, l) \leftarrow \arg\max_a \sum_{i=1}^{M} p_i(s, a, x, l)z_i$

return $\pi(s, x, l)$

C Details on Distributional Language-Guided Policy

In this section, we provide a more detailed introduction to the specific computational workflow of our proposed Distributional Language-Guided Policy. We follow the approach in [3], employing discrete atoms to model and approximate the original value distribution. To facilitate understanding, we have included the calculation methodology for this section of the work here.

Specifically, we model the discrete value distribution with discrete units called atoms $\{z_i = V_{\text{MIN}} + i\Delta z\}_{i=0}^{M-1}$, which are uniformly spaced support points that discretize the range of possible returns into $[V_{MIN}, V_{MAX}]$. $M \in \mathbb{N}$ is the number of atoms. $V_{\text{MAX}}, V_{\text{MIN}}, M$ are pre-defined parameters, which together decide the step between the atoms of the categorical value distribution: $\Delta z := \frac{V_{\text{MAX}} - V_{\text{MIN}}}{M-1}$.

To represent the discrete value distribution, we need to estimate the probability p_i, which means the probability for the discrete value equaling z_i. In practical implementation, we approximate the probabilities p_i by a parametric model $\theta : \mathcal{S} \times A \times \mathcal{X} \times \mathcal{L} \to \mathbb{R}^M$. We use $\theta_i(\cdot, \cdot, \cdot, \cdot)$ to denote the i-th dimension of this model's output. Therefore, the discrete value distribution can be written as:

$$\begin{aligned} Z_\theta(s, a, x, l) &= z_i \quad \text{w.p.} \quad p_i(s, a, x, l) := \frac{e^{\theta_i(s,a,x,l)}}{\sum_j e^{\theta_j(s,a,x,l)}} \\ Q_\theta(s, a, x, l) &= \mathbb{E}[Z_\theta(s, a, x, l)] = \sum_{i=0}^{M-1} p_i(s, a, x, l) z_i \end{aligned} \tag{15}$$

where $\sum_{i=0}^{M-1} p_i = 1$. We now explain how to learn θ through RL. Given a transition (s, a, r, s', l), the Bellman update for each atom z_i is computed as:

$$\hat{\mathcal{T}} z_i = r + \gamma z_i \tag{16}$$

The Bellman update $\hat{\mathcal{T}} z_i$ maps the original support points to new locations that do not align with predefined discrete atoms $\{z_i\}_{i=0}^{M-1}$, making it impossible to represent as a valid discrete distribution over the fixed atoms. Therefore, we need to project these values back onto the fixed support $\{z_0, ..., z_{M-1}\}$. To do so, we distribute probability mass $p_i(s', a', x', l)$ to the nearest two atoms in $[V_{\text{MIN}}, V_{\text{MAX}}]$, where $a' = \pi(s', x', l)$ is the output of the greedy policy $\pi(\cdot, \cdot, \cdot)$. x' is computed as $x' = h_w(x, (s, a))$. Ultimately, we can compute the projected probability for $\hat{\mathcal{T}} z_i$ via a local interpolation mechanism:

$$\text{Projected probability} = \begin{cases} \frac{z_{j+1} - \hat{\mathcal{T}} z_i}{\Delta z} \cdot p_i(s', a', x', l), & \text{assigned to } z_j \\ \frac{\hat{\mathcal{T}} z_i - z_j}{\Delta z} \cdot p_i(s', a', x', l), & \text{assigned to } z_{j+1} \end{cases} \tag{17}$$

Sum all the projected probabilities from all $\hat{\mathcal{T}} z_j$, we can compute the projected update probabilities by:

$$(\Phi \hat{\mathcal{T}} Z_\theta(s, a, x, l))_i = \sum_{j=0}^{M-1} [1 - \frac{|[\hat{\mathcal{T}} z_j]_{V_{MIN}}^{V_{MAX}} - z_i|}{\Delta z}]_0^1 \, p_j(s', a', x', l) \tag{18}$$

where $[\cdot]_a^b$ bounds the argument in the range $[a, b]$. Given the project update $\Phi \hat{\mathcal{T}} Z_{\hat{\theta}}$ and the current estimates of the discrete value distributions Z_θ, we can update θ by minimizing the KL divergence:

$$D_{KL}(\Phi \hat{\mathcal{T}} Z_{\hat{\theta}}(s, a, x, l) || Z_\theta(s, a, x, l)) \tag{19}$$

where $\hat{\theta}$ is the target network.

D Theoretical Analysis

D.1 Insights Using Distributional RL

The key insight of applying distributional RL to language-conditioned tasks lies in its capacity to capture value distributions, which provides fine-grained task differentiation across various tasks. Traditional RL methods, however, rely on learning scalar value expectations, discarding critical distributional information, making it require more samples to discriminate between tasks properly. We first demonstrate this key insight through an extreme yet illustrative example as illustrated in Figure 9.

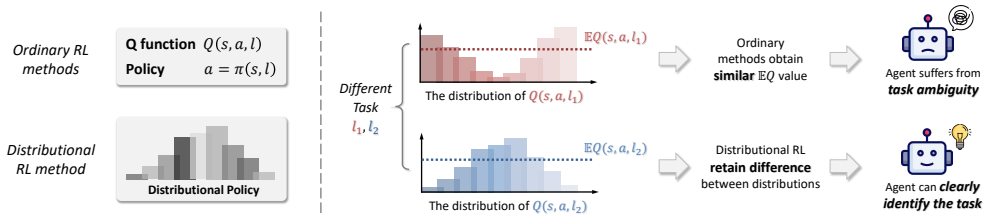

Figure 9: An illustrative case where $Q(s,a)$ functions in two tasks share the same expectations but have different distributions. In this case, traditional RL cannot discriminate between the tasks, while distributional RL can.

Consider two distinct tasks l_1 and l_2 that share identical expected returns $Q_\pi(s,a,l_1) = Q_\pi(s,a,l_2)$ for a specific state-action pair (s,a), but exhibit fundamentally divergent Q value distributions $Z(s,a,l_1) \neq Z(s,a,l_2)$. In this case, traditional RL methods relying on value estimation would inevitably conflate tasks l_1, l_2 at (s,a), regardless of sample size, while distributional RL resolves this ambiguity through approximating the full distributions of values.

To formally validate this insight, we present the following theorem, accompanied by a rigorous proof.

D.2 Theoretical Proof

Definition 1 (Restatement of Definition 1)**.** *In a multi-task RL setting with known task instruction space $\mathcal{L}$ and unknown semantics space $\mathcal{G}$, for a task distinction threshold δ and sub-optimality gap ϵ, two task instructions $l_i, l_j \in \mathcal{L}$ are considered with **different underlying semantics**, $g_i \not\leftrightarrow g_j$, if the expected Q-values under any shared ϵ-optimal policy π satisfy:*

$$\mathbb{E}_\pi \left[|Q_\pi(s,a,l_i) - Q_\pi(s,a,l_j)| \right] \geq \delta.$$

Conversely, if

$$\mathbb{E}_\pi \left[|Q_\pi(s,a,l_i) - Q_\pi(s,a,l_j)| \right] \leq \delta/2,$$

*then l_i and l_j are considered with the **same underlying semantics**, $g_i \leftrightarrow g_j$, where $s_0 \sim p_0(\cdot), a \sim \pi(\cdot|s), s' \sim p(\cdot|s,a)$. $V_\pi(s) \geq V^*(s) - \epsilon, \forall s \in \mathcal{S}$, V^* is optimal value function.*

*In the case of distributional reinforcement learning, the Wasserstein distance W_1 between the return distributions $Z(s,a)$ is used as the criterion. Specifically, l_i and l_j are considered to represent **different semantics** if*

$$\mathbb{E}_\pi \left[W_1 \left(Z_\pi(s,a,l_i), Z_\pi(s,a,l_j) \right) \right] \geq d,$$

*and the **same semantics** if this expectation is less than or equal $d/2$.*

Theorem 1 (Sample Complexity for Task Instruction Disambiguation)**.** *Consider an offline multi-task RL setting with M distinct tasks (with different semantics). In direct Q-value estimate setting, suppose $Q(s,a,l) \in [0, Q_{\max}]$, $\forall (s,a,l) \sim \mathcal{D}$ with finite $Q_{\max}$, and the task distinction threshold is $\delta > 0$. When the number of training samples n_{value} satisfies:*

$$n_{\text{value}} \geq \frac{C_{\text{value}} \log(3M^2/\eta)}{\delta^2}.$$

The mean value estimate algorithm achieves task-level semantic disambiguation with confidence at least $1-\eta$. In the distributional RL setting, let $Z(s,a,l)$ denote the learned return distribution, and

suppose the task distinction threshold is given by a 1-Wasserstein distance $d > 0$. Then, to ensure semantic disambiguation of task instructions with confidence at least $1 - \eta$, it suffices that:

$$n_{\text{dist}} \geq \frac{C_{\text{dist}} \log(3M^2/\eta)}{d^2},$$

where $C_{\text{value}}, C_{\text{dist}} > 0$ are universal constants depending on certain attributes of Q-value distribution.

Proof. We denote $Q_i(s,a)$ as a shorthand for $Q_\pi(s,a,l_i)$ and $Z_i(s,a)$ as a shorthand for $Z_\pi(s,a,l_i)$ in the following discussion. For direct offline Q-learning, when the task distinction threshold is $\delta > 0$, we define the semantic ambiguity event $\mathcal{S}_a$ as:

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[|\hat{Q}_i(s,a) - \hat{Q}_j(s,a)|\right] \leq \frac{\delta}{2}, \tag{20}$$

$$\mathbb{E}_\pi\left[|Q_i(s,a) - Q_j(s,a)|\right] \geq \delta, g_i \not\Leftrightarrow g_j, \tag{21}$$

where $\mathcal{D}$ is the offline dataset, Q is the optimal Q function, π is the optimal policy along with Q, and $\hat{Q}$ is the learned Q function.

Following the triangle inequality, we have:

$$\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|Q_i - Q_j|]\right| \leq \tag{22}$$

$$\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\mathcal{D}\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|]\right| + \left|\mathbb{E}_\mathcal{D}\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|Q_i - Q_j|]\right| \tag{23}$$

In general, we assume that the offline RL dataset is collected by ϵ-optimal policy, i.e., there exists a ϵ-optimal policy π that $\mathcal{D}$ satisfies

$$\mathbb{E}_\mathcal{D}\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] = \mathbb{E}_\pi[|\hat{Q}_i - \hat{Q}_j|], \mathbb{E}_\mathcal{D}(\hat{Q}) = Q_\pi = Q \tag{24}$$

Besides, we have:

$$\left|\mathbb{E}_\pi[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|Q_i - Q_j|]\right| = \left|\mathbb{E}_\pi\left(|\hat{Q}_i - \hat{Q}_j| - |Q_i - Q_j|\right)\right| \tag{25}$$

$$\leq \mathbb{E}_\pi\left|\,|\hat{Q}_i - \hat{Q}_j| - |Q_i - Q_j|\,\right| \tag{26}$$

$$\leq \mathbb{E}_\pi\left|(\hat{Q}_i - \hat{Q}_j) - (Q_i - Q_j)\right| \tag{27}$$

$$\leq \mathbb{E}_\pi|\hat{Q}_i - Q_i| + \mathbb{E}_\pi|\hat{Q}_j - Q_j| \tag{28}$$

Therefore,

$$\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|Q_i - Q_j|]\right| \geq \frac{\delta}{2}\right) \tag{29}$$

$$\leq \Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|\hat{Q}_i - \hat{Q}_j|]\right| + \mathbb{E}_\pi|\hat{Q}_i - Q_i| + \mathbb{E}_\pi|\hat{Q}_j - Q_j| \geq \frac{\delta}{2}\right) \tag{30}$$

$$\leq \Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|\hat{Q}_i - \hat{Q}_j|]\right| \geq \frac{\delta}{2}\right) \tag{31}$$

$$+ \Pr\left(\mathbb{E}_\pi|\hat{Q}_i - Q_i| \geq \frac{\delta}{2}\right) + \Pr\left(\mathbb{E}_\pi|\hat{Q}_j - Q_j| \geq \frac{\delta}{2}\right) \tag{32}$$

Since $\hat{Q}, Q \in [0, Q_{\max}]$, $|\hat{Q}_i - \hat{Q}_j|, |\hat{Q}_i - Q_i|, |\hat{Q}_j - Q_j| \in [0, Q_{\max}]$, together with Equation 24, we can apply Theorem 1 (Hoeffding's inequality) in [5], and obtain that for $\forall l_i, l_j \in \mathcal{L}$,

$$\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|\hat{Q}_i - \hat{Q}_j|]\right| \geq \delta/2\right) \leq 2\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{33}$$

$$\Pr\left(\mathbb{E}_\pi|\hat{Q}_i - Q_i| \geq \delta/2\right) \leq 2\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{34}$$

$$\Pr\left(\mathbb{E}_\pi|\hat{Q}_j - Q_j| \geq \delta/2\right) \leq 2\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{35}$$

where $n_{\text{value}} = |\mathcal{D}|$ is the size of dataset, $C_{\text{value}} \in (0, \mathcal{O}(Q_{\max}^2)]$ is a constant value.

Combining with Equation 32, we have

$$\Pr\left(\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] - \mathbb{E}_\pi[|Q_i - Q_j|] \leq -\delta/2\right) \leq 6\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{36}$$

Consider the extreme case in task semantics distinction, when $\exists l_i, l_j \in \mathcal{L}, g_i \not\leftrightarrow g_j, \mathbb{E}_\pi[|Q_i - Q_j|] = \delta$ that can be exactly partitioned under the threshold. We have

$$\Pr\left(\mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] \leq \delta/2\right) \leq 6\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{37}$$

Consider all task pairs with different real semantics $\begin{pmatrix} M \\ 2 \end{pmatrix} \approx \frac{M^2}{2}$:

$$\Pr\left(\exists i, j, g_i \not\leftrightarrow g_j, \mathbb{E}_{(s,a)\sim\mathcal{D}}[|\hat{Q}_i - \hat{Q}_j|] \leq \delta/2\right) \leq 3M^2\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \tag{38}$$

To ensure a confidence level of at least $1-\eta$ for any task ambiguity, we require that the probability of the event $\mathcal{S}_a$ satisfies:

$$\Pr(\mathcal{S}_a) \leq \eta,$$

Thus, we need to let:

$$3M^2\exp(-n_{\text{value}}\delta^2/C_{\text{value}}) \leq \eta.$$

Then,

$$n_{\text{value}} \geq \frac{C_{\text{value}}}{\delta^2}\log\frac{3M^2}{\eta} \tag{39}$$

For the distributional RL setting, we have the estimated return distributions $\hat{Z}_i, \hat{Z}_j$ and real distributions Z_i, Z_j, with the threshold $d > 0$. Similar to Equation 21, we have the semantic ambiguity event $\mathcal{S}_a$ as:

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i(s,a), \hat{Z}_j(s,a))\right] \leq \frac{d}{2}, \tag{40}$$

$$\mathbb{E}_\pi\left[W_1\left(Z_i(s,a), Z_j(s,a)\right)\right] \geq d, g_i \not\leftrightarrow g_j, \tag{41}$$

where Z is the optimal distribution of Q-value, π is the optimal policy along with Z, and $\hat{Z}$ is the learned distribution.

Similarly, we have:

$$\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i, \hat{Z}_j)\right] - \mathbb{E}_\pi\left[W_1\left(Z_i, Z_j\right)\right]\right| \tag{42}$$

$$\leq \left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i, \hat{Z}_j)\right] - \mathbb{E}_\mathcal{D}\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i, \hat{Z}_j)\right]\right| \tag{43}$$

$$+ \left|\mathbb{E}_\mathcal{D}\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i, \hat{Z}_j)\right] - \mathbb{E}_\pi\left[W_1(Z_i, Z_j)\right]\right| \tag{44}$$

$$= \left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i, \hat{Z}_j)\right] - \mathbb{E}_\pi\left[W_1(\hat{Z}_i, \hat{Z}_j)\right]\right| + \left|\mathbb{E}_\pi\left[W_1(\hat{Z}_i, \hat{Z}_j) - W_1(Z_i, Z_j)\right]\right| \tag{45}$$

Following the triangle inequality of Wasserstein distance, we have:

$$W_1(\hat{Z}_i, \hat{Z}_j) - W_1(Z_i, Z_j) \leq W_1(\hat{Z}_i, Z_i) + W_1(Z_j, \hat{Z}_j) \tag{46}$$

Then,

$$\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]-\mathbb{E}_\pi\left[W_1\left(Z_i,Z_j\right)\right]\right|\geq\frac{d}{2}\right) \tag{47}$$

$$\leq\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]-\mathbb{E}_\pi\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]\right|+\left|\mathbb{E}_\pi\left[W_1(\hat{Z}_i,\hat{Z}_j)-W_1(Z_i,Z_j)\right]\right|\geq\frac{d}{2}\right) \tag{48}$$

$$\leq\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]-\mathbb{E}_\pi\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]\right|+\left|\mathbb{E}_\pi\left[W_1(\hat{Z}_i,Z_i)+W_1(\hat{Z}_j,Z_j)\right]\right|\geq\frac{d}{2}\right) \tag{49}$$

$$\leq\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]-\mathbb{E}_\pi\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]\right|\geq\frac{d}{2}\right) \tag{50}$$

$$+\Pr\left(\left|\mathbb{E}_\pi\left[W_1(\hat{Z}_i,Z_i)\right]\right|\geq\frac{d}{2}\right)+\Pr\left(\left|\mathbb{E}_\pi\left[W_1(\hat{Z}_j,Z_j)\right]\right|\geq\frac{d}{2}\right) \tag{51}$$

Since $\hat{Z}$ is the empirical measure of Z and the estimated return is 1-dimensional, $Q\in[0,Q_{\max}], W_1(Z_i,Z_j)\in[0,Q_{\max}]$. Following Theorem 1 (Hoeffding's inequality) in [5] , Corollary 5.2 and Remark 5 in [31], we have the following equation where $C_{\text{dist}_1}, C_{\text{dist}_2}\in(0,\mathcal{O}(Q_{\max}^2)]$ are constant values.

$$\Pr\left(\left|\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]-\mathbb{E}_\pi\left[W_1(\hat{Z}_i,\hat{Z}_j)\right]\right|\geq d/2\right)\leq 2\exp(-n_{\text{dist}}d^2/C_{\text{dist}_1}) \tag{52}$$

$$\Pr\left(\mathbb{E}_\pi\left[W_1(\hat{Z}_i,Z_i)\right]\geq d/2\right)\leq 2\exp(-n_{\text{dist}}d^2/C_{\text{dist}_2}) \tag{53}$$

$$\Pr\left(\mathbb{E}_\pi\left[W_1(\hat{Z}_j,Z_j)\right]\geq d/2\right)\leq 2\exp(-n_{\text{dist}}d^2/C_{\text{dist}_2}) \tag{54}$$

Combining with Equation 51, we can obtain the following equation where $C_{\text{dist}}=\max(C_{\text{dist}_1},C_{\text{dist}_2})>0$

$$\Pr\left(\mathbb{E}_{(s,a)\sim D}\left(W_1(\hat{Z}_i,\hat{Z}_j)\right)-\mathbb{E}_\pi\left(W_1(Z_i,Z_j)\right)\leq -d/2\right)\leq 6\exp(-n_{\text{dist}}d^2/C_{\text{dist}}) \tag{55}$$

In the same way, consider the extreme case when $\exists l_i,l_j\in\mathcal{L}, g_i\not\leftrightarrow g_j, \mathbb{E}_\pi\left(W_1(Z_i,Z_j)\right)=d$. We have

$$\Pr\left(\mathbb{E}_{(s,a)\sim D}\left(W_1(\hat{Z}_i,\hat{Z}_j)\right)\leq d/2\right)\leq 6\exp(-n_{\text{dist}}d^2/C_{\text{dist}}) \tag{56}$$

Consider all task pairs with different real semantics $\binom{M}{2}\approx\frac{M^2}{2}$:

$$\Pr\left(\exists i,j,g_i\not\leftrightarrow g_j,\mathbb{E}_{(s,a)\sim D}\left(W_1(\hat{Z}_i,\hat{Z}_j)\right)\leq d/2\right)\leq 3M^2\exp(-n_{\text{dist}}d^2/C_{\text{dist}}) \tag{57}$$

Similarly, to ensure a confidence level of at least $1-\eta$ for avoiding task ambiguity, we need

$$3M^2\exp(-n_{\text{dist}}d^2/C_{\text{dist}})\leq\eta.$$

Then,

$$n_{\text{dist}}\geq\frac{C_{\text{dist}}}{d^2}\log\frac{3M^2}{\eta} \tag{58}$$

□

Corollary 2. *In a multi-task RL setting, to avoid task ambiguity with confidence level $1-\eta$, learning the distribution over Q-values requires fewer samples than learning point estimates of Q-values when the number of tasks M is sufficiently large. Formally, $n_{\text{value}}\geq n_{\text{dist}}$, where $n_{\text{value}}, n_{\text{dist}}$ denote the samples needed to avoid task ambiguity for value-based and distributional settings.*

Proof. From Theorem 1, we have

$$n_{\text{value}}\geq\frac{C_{\text{value}}}{\delta^2}\log\frac{3M^2}{\eta} \tag{59}$$

$$n_{\text{dist}}\geq\frac{C_{\text{dist}}}{d^2}\log\frac{3M^2}{\eta} \tag{60}$$

First, $Q \in [0, Q_{\max}], W_1(Z_i, Z_j) \in [0, Q_{max}]$, following Theorem 1 (Hoeffding's inequality) in [5] , Corollary 5.2 and Remark 5 in [31] we obtain that the constant C satisfies:

$$C_{\text{value}} \leq \mathcal{O}(Q_{\max}^2), C_{\text{dist}} \leq \mathcal{O}(Q_{\max}^2). \tag{61}$$

Then, we can prove that for any Q-value distribution Z_i, Z_j,

$$W_1(Z_i, Z_j) \geq |\mathbb{E}_{Z_i}(Q) - \mathbb{E}_{Z_j}(Q)| \tag{62}$$

The definition of Wasserstein-1 distance is

$$W_1(Z_i, Z_j) = \inf_{\pi \in \Pi(Z_i, Z_j)} \mathbb{E}_{(X,Y)\sim\pi}|X - Y|$$

Here, $\Pi(Z_i, Z_j)$ denotes the set of all joint distributions with marginals Z_i, Z_j.

For $(Z_i, Z_j) \sim \pi$, we have

$$\mathbb{E}[|X - Y|] \geq |\mathbb{E}(X - Y)| = |\mathbb{E}_{Z_i}(X) - \mathbb{E}_{Z_j}(Y)| = |\mathbb{E}_{Z_i}(X) - \mathbb{E}_{Z_j}(X)|$$

Thus, for any joint π,

$$\mathbb{E}_{(X,Y)\sim\pi}|X - Y| \geq |\mathbb{E}_{Z_i}(X) - \mathbb{E}_{Z_j}(X)|$$

For the optimal joint π,

$$W_1(Z_i, Z_j) \geq |\mathbb{E}_{Z_i}(Q) - \mathbb{E}_{Z_j}(Q)| \tag{63}$$

Thus, we must choose a much smaller threshold δ for point estimates than that d for distribution learning, $\delta \leq d$. In some scenarios, when the mean difference of the Q function is small but the distribution difference is large, $\delta << d$.

Combining Equation 61, 63, we can obtain that

$$\frac{C_{\text{value}}}{\delta^2} \log \frac{3M^2}{\eta} \geq \frac{C_{\text{dist}}}{d^2} \log \frac{3M^2}{\eta} \tag{64}$$

we prove that $n_{\text{value}} \geq n_{\text{dist}}$, even $n_{\text{value}} >> n_{\text{dist}}$ in some scenarios.

□

E Experimental Details

E.1 Toy Experiment

In this toy experiment based on the Minigrid environment [11], we demonstrate that vanilla offline RL fails to establish the relationship between the tasks and their underlying reward functions as the number of tasks explodes.

The setup consists of 10 accessible goal positions $\mathcal{G} = \{g_0, g_1, ..., g_9\}$, where $\mathcal{G}$ represents the set of all possible goal positions. The agent (red triangle-shaped) must follow a given instruction $l \in \mathcal{L}$ to navigate to a specific goal position. We simulate instructions using numerical task IDs, making $\mathcal{L} \subset \mathbb{N}$. We employ a random mapping $F : \mathcal{L} \to \mathcal{G}$ to assign each l to goal position g, which simulates semantics of instructions and is hidden from the agent. With these settings, we can control the number of instructions $|\mathcal{L}|$ by simply adjusting the set of valid task IDs and the mappings. We conduct 10 experiments, varying the number of instructions from 1 to 2^9. For each experiment, we generate a new mapping F and collect 1024 random trajectories as the offline dataset.

The agent always starts from the center of the map. For each step, the agent receives the task ID and current state as input, and the agent can choose to move forward, turn left, or turn right. A reward of $1 - 0.9 \times (\texttt{STEP\_COUNT}/\texttt{MAX\_STEP})$ is given for success and 0 for failure. `MAX_STEP` is fixed to 12 in our toy experiment. For offline datasets, a random policy with $\texttt{MAX\_STEP} = 12$ is used to generate

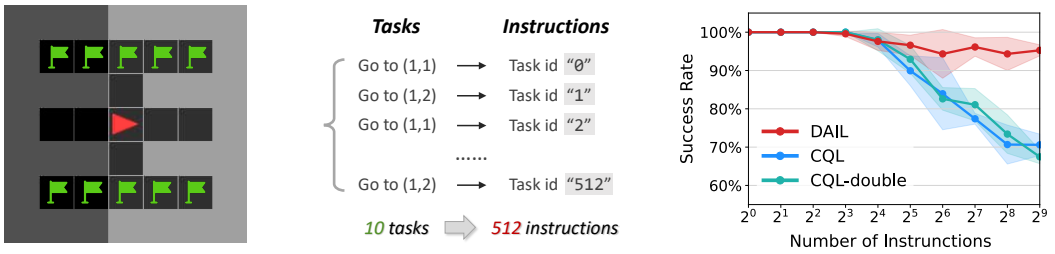

Figure 10: **Left:** The green flags in the map denote the accessible goals. **Middle:** An illustration of the mapping between goal positions and instructions. **Right:** Average success rates over 100 evaluations for each number of instructions and 3 seeds.

$64 \times n$ trajectories for 8 or fewer tasks, where n is the number of tasks, and 1024 trajectories for 16 or more tasks, respectively. For all these datasets, the success rate is fixed to be 0.5.

For the observation signals in this toy experiment, we use a simple architecture, which combines Bag-of-Words encoding [40] and a two-layer CNN. We use 1 fully connected layer to embed task IDs into task representations and a two-layer MLP as the output network. We use 64 as the feature size for CQL and our method, 128 for CQL-double separately. For each number of tasks, each agent is trained with $\alpha = \{0, 0.01, 0.1, 0.5, 1, 2\}$ over 3 seeds. We calculate the average of the results from these 3 seeds and record the best performance among them as the performance for this number of instructions, as illustrated in the Right of Figure 10.

E.2 Offline Datasets

E.2.1 BabyAI

BabyAI [10] is a language-conditioned research platform built on MiniGrid [11], which provides different levels of tasks equipped with varied language instructions. We choose level `SynthLoc` as the benchmark, which is the union of all instructions from `PutNext`, `Open`, `Goto`, and `PickUp`. The agent needs to deal with synthetic Baby Language and interact with the specified objects at the goal position. Some examples of language instructions are "put the green key behind you next to a box", "go to the red ball behind you", and "pick up a green box".

We follow the default map configuration of BabyAI, where each room has a size of 7×7, arranged in a 3×3 grid with a total of 9 rooms. Each room may be connected to others via a door, as illustrated in Figure 11. The agent has a field of view of 7×7 in front of it, with the observation size being $(7 \times 7 \times 3)$. Each grid cell in the observation contains the values (`OBJECT_IDX`, `COLOR_IDX`, `STATE`),

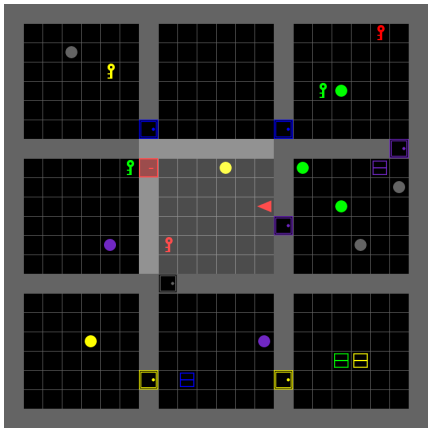

Figure 11: Level `SynthLoc` in BabyAI

all represented in a structured format. A reward of '1 - 0.9 * (step_count / max_steps)' is only given for success, and '0' in all other cases. Agents are permitted to take up to 300 steps before truncation in our setting. In `SynthLoc`, the total number of tasks is large, and their distribution is sparse. We

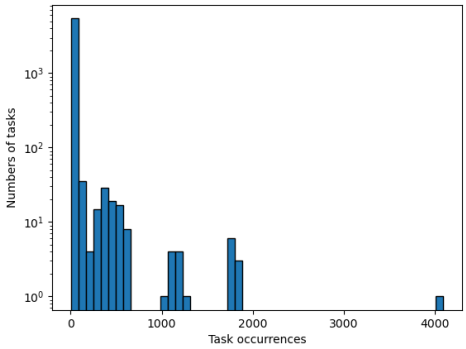

Figure 12: The histogram shows the frequency distribution of tasks in 10^6 samples. The x-axis represents the total frequency of each task, while the y-axis indicates the number of tasks that fall within each frequency interval.

reset the environment 10^6 times and obtain over 5500 unique instructions. From the histogram of task count (Figure 12), it can be observed that most tasks only appear less than 500 times in 10^6, while some tasks appear over 4000 times. The distribution of tasks highlights the highly uneven distribution of tasks. We divide the task set into two subsets, designating approximately 60% of the tasks as in-distribution tasks. All trajectories in the offline dataset are collected under in-distribution instructions, while tasks encountered during testing outside this set are considered out-of-distribution tasks.

To construct the offline dataset, we collect three types of data: expert data, gathered by a pre-designed bot within the environment; medium data, collected by a well-trained agent; and random data. The built-in bot has access to global information to accomplish every possible task with a near-optimal solution. We train an IL agent following BabyAI 1.1 [23], the state-of-the-art model proposed by the original environmental authors. Trained on a dataset of 100k expert trajectories, it achieved approximately 87.9% success rate across all tasks. Random agent achieves a 10.5% success rate during data collection. We conduct a high-quality dataset with 50k expert trajectories, 50k IL agent trajectories, and 25k random trajectories; a medium-quality dataset with 12.5k expert trajectories, 25k IL agent trajectories, and 40k random trajectories. All the trajectories in the dataset are generated under in-distribution instructions.

E.2.2 ALFRED

ALFRED [54] benchmarks sequential decision-making tasks involving household activities (e.g., cleaning, heating food) through language instructions and first-person vision, shown in Figure 13 and Table 4. The dataset provides 8055 expert demonstrations with 25k human-annotated language instructions detailing both high-level goals and sub-goal step-by-step guidance. As our work primarily focuses on low-level policy learning rather than high-level planning, we specifically concentrate on the `GOTO` sub-goal setting for our evaluation. In this task set, the agent must go to specific locations according to instructions like "Move to other side of couch on the right side of the table before the door". To simulate the presence of noisy data in real-world applications, we augment the training set with 30k random-agent trajectories, resulting in 97896 total trajectories with 53442 unique instructions across 108 household scenes.

As for the experiment, we use the Modeling Quickstart dataset, which is recommended [54], including trajectory JSONs and pre-generated ResNet features. The ResNet features are obtained using a pre-trained ResNet-18 [21] to extract $512 \times 7 \times 7$ features from the `conv5` layer, which are used as observation input during training and evaluation.

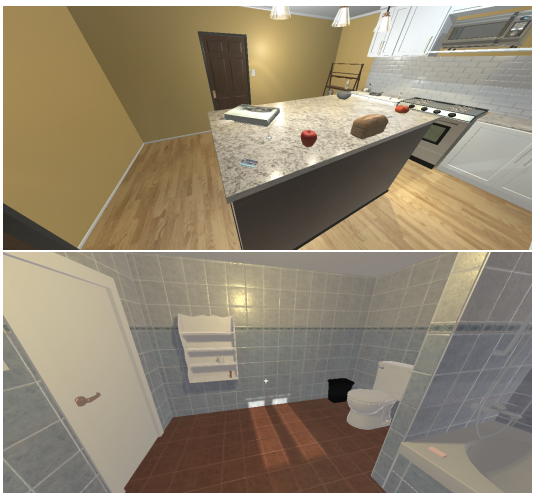

Figure 13: Two example scenes from ALFRED.

Table 4: Instruction examples in ALFRED `GOTO` task set.

#	Instructions
1	Go left and turn to the right to face the couch.
2	Turn around and make a left immediately after the toilet turn a quick left to face the side of the toilet.
3	Move to other side of couch on the right side of the table before the door.
4	go back to your right to the fridge and open the door
5	Turn around and walk to the white stove on the right.
6	Turn to the right and go to the sink across from you.
7	Turn around and walk towards the toilet, then turn right and walk towards the door, turn left to face the counter.
8	Walk forward, then hang a right and walk across the room, turn left and walk up to the chair.

For sub-goal evaluation, we follow [54] to use the expert trajectory to move the agent until the start state of the tested sub-goal, and the agent takes over to operate based on the instructions and observations. Episodes with 5 or more failed actions or exceeding the `MAX_STEP` $= 32$ are counted as failures immediately. A reward of 1 is given for success and 0 for failure. Failed actions refer to actions that cannot be successfully executed in the current state (for example, moving against the walls or other obstacles in the room).

F Architecture, Training, and Evaluation Details

F.1 Details of Encoders

This section describes the network architecture of we used for language and observation encoding in BabyAI and ALFRED environments. In all the experiments, all models share the same encoders, detailed as follows, unless otherwise specified.

Language Encoder For the language signals, we use the Transformer model [58] from the pre-trained CLIP network [50] for language encoding in BabyAI and ALFRED experiments, freezing the entire model during training and adding an additional fully connected layer at the end for fine-tuning.

Observation Encoder Given the differing input structures of BabyAI and ALFRED, we employ separate observation encoders.

For **BabyAI**, we adopt the original visual encoding framework from BabyAI [23], which integrates a Bag-of-Words embedding layer [40], a convolution backbone, and a linear layer. The BOW module first turns the structural inputs with size $7 \times 7 \times 3$ into $7 \times 7 \times 256$ embeddings. The subsequent convolutional backbone processes these features through two sequential blocks and a max-pooling layer: each block contains a 3×3 convolutional layer, followed by batch normalization and ReLU activation. The output is then processed with a 7×7 max-pooling layer. The features are then flattened and projected to 256 dimensions through a linear layer, producing a 256-dimensional vector as the observation encoder's final output.

For **ALFRED**, we use the original encoding framework in ALFRED [54] for all implemented methods, which contains two sequential blocks: each block contains a 1×1 convolutional layer, followed by batch normalization and ReLU activation. The features are then flattened and projected to 512 dimensions through a linear layer, producing a 512-dimensional vector as the observation encoder's final output.

FiLM and Sequence Encoder We follow [23] to use FiLM [49] to fuse language and observation encodings through feature-wise affine transformations. For history encoding, all baseline methods employ a two-layer unidirectional LSTM to model temporal dependencies.

F.2 Architecture Details of DAIL

The overview architecture of DAIL is shown in Figure 14. We adopt the language and observation encoders described above to encode instructions and observations separately. To use the same sequence encoder for both trajectory-wise semantic alignment and history encoding, we modify the sequence encoder to process observation-action trajectories jointly and output history information x_t.

The computation process of state-action value $q(s_t, a_t, x_{t-1}, l)$ is shown on the Left of Figure 14, given instruction l, observation s_t. The outputs of the FiLM network are concatenated with the outputs of the sequence encoder, then processed through a Multi-Layer Perceptron (MLP) and a Softmax layer to generate the final value distribution with dimension M. For Trajectory-Wise Semantic Alignment as shown on the Right of Figure 14, we derive embeddings from instruction l and trajectory τ. The trajectory embedding is represented by the sequence model's final output x_τ.

F.3 Training Details

All models are implemented with PyTorch, and trained with a batch size of 64, using the Adam optimizer [27] at a learning rate of $3e-4$. All layers in the networks utilize PyTorch's default weight initialization, and the network outputs fixed-dimensional embeddings suitable for downstream tasks. In BabyAI experiments, all methods were trained for 50 epochs over 3 seeds. And in the ALFRED experiments, all methods were trained for 20 epochs over 3 seeds following [54].

As for DAIL, we fix $\alpha = 2$ and $\lambda = 0.2$ except for the toy experiment and ablation experiment of λ. We use $V_{MAX} = -V_{MIN} = 20, M = 51$ in all our experiments following [29].

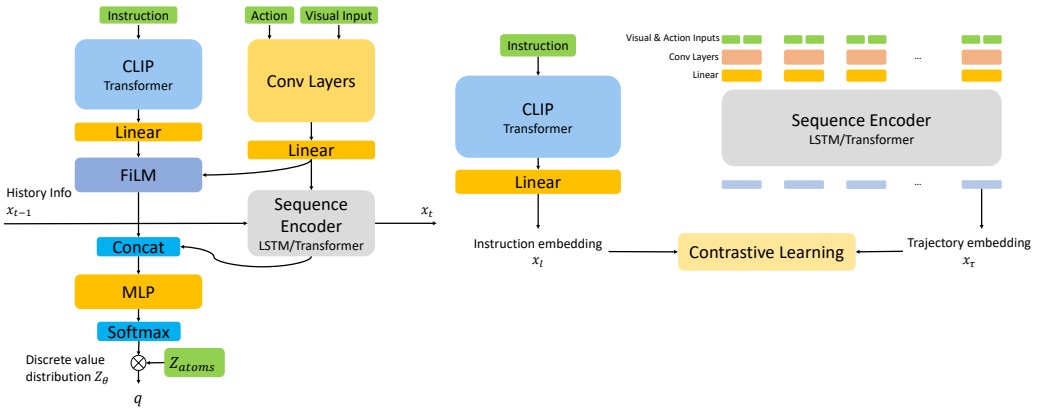

Figure 14: Overview of our algorithm. **Left:** Computation process of state-action value. **Right:** trajectory-wise semantic alignment.

F.4 Baseline Details

In this work, all the baselines share the same language encoder and similar observation encoder, and only differ in the decision module. In the following part, we introduce the decision modules of several baselines used in our paper:

GCBC: language-conditioned behavior cloning with all data to maximize

$$\mathcal{J}_{\mathrm{BC}}(\pi_\theta) = \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[\log \pi_\theta(a_t|x_t,l)] \tag{65}$$

BC-Z: learning two task encodings from language and trajectory (video), and aligning them through similarity. We use cosine distance $D_{cos}(p,q) = \frac{p^T q}{|p||q|}$ to measure the similarity. We denote (τ, l) as a pair of trajectory and instruction.

$$\mathcal{J}_{\mathrm{BC-Z}}(\pi_\theta) = \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[\log \pi_\theta(a_t|x_t, f_\varphi(l))] - \mathbb{E}_{(\tau,l)\sim\mathcal{D}}[D_{\cos}(q_\phi(\tau), f_\varphi(l))], \tag{66}$$

where $q_\phi(\cdot)$ is the video encoder proposed by BC-Z to encode trajectory information, and $f_\varphi(\cdot)$ is the instruction encoder.

GRIF: explicitly aligning the representations of language-conditioned tasks through contrastive learning with similarity measure $\mathcal{C}(s,g,l) = D_{cos}(f_\varphi(l), h_\psi(s_0,g))$, where φ, ψ are learnable parameters of the language encoder and goal encoder respectively and g is the last state of a trajectory. Positive data $(\tau^+, l^+) \sim p_\mathcal{D}(\cdot,\cdot)$ are uniformly sampled from the dataset, with s^+, g^+ being the start state and end state of τ^+ respectively. Negative examples s^-, g^- are the start state and end state of a randomly sampled trajectory from the dataset. Negative instruction $l^- \sim p_l(\cdot)$ is the instruction of another random trajectory. For each positive example, k negative examples are sampled noted as $\{s_i^-, g_i^-\}_{i=1}^k$ and $\{l_i^-\}_{i=1}^k$.

$$\begin{aligned}
\mathcal{L}_{\mathrm{lang}\to\mathrm{goal}}(\varphi,\psi) &= -\log \frac{\exp(\mathcal{C}(s^+,g^+,l^+)/\tau)}{\exp(\mathcal{C}(c^+,g^+,l^+)/\tau) + \sum_{i=1}^k \exp(\mathcal{C}(c_i^-,g_i^-,l^+)/\tau)} \\
\mathcal{L}_{\mathrm{goal}\to\mathrm{lang}}(\varphi,\psi) &= -\log \frac{\exp(\mathcal{C}(s^+,g^+,l^+)/\tau)}{\exp(\mathcal{C}(c^+,g^+,l^+)/\tau) + \sum_{i=1}^k \exp(\mathcal{C}(c_i^+,g_i^+,l^-)/\tau)}
\end{aligned} \tag{67}$$

where τ is the temperature parameter. We employ the last state s_T in the trajectory as the goal state: $h_\psi(s_0,g) = h_\psi(s_0,s_T)$. Then the policy network is trained with behavior cloning by maximizing the likelihood of the actions:

$$\mathcal{J}_{\mathrm{GRIF}}(\pi_\theta) = \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[\log \pi_\theta(a_t|x_t, f_\varphi(l))] \tag{68}$$

We use the **GRIF(Joint)** setting to train the model [42].

CQL: we implement CQL (w/o distributional) based on DDPG,

$$\mathcal{J}_{CQL}(\pi_\theta) = \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[Q(x_t, \pi_\theta(x_t,l),l)] \tag{69}$$

where Q function is learned by minimizing:

$$\begin{aligned}\mathcal{L}_{CQL(\mathcal{H})}(\theta) =& \mathbb{E}_{(s_t,a_t,r_t,s_{t+1},l)\sim\mathcal{D},(x_t,x_{t+1})\sim h_w}[(Q_\theta(x_t,a_t,l) - \mathcal{B}^\pi Q_\theta(x_t,a_t,l))^2] + \\ & \alpha\mathbb{E}_{(s_t,l)\sim\mathcal{D},x_t\sim h_w}[\log\sum_a \exp(Q_\theta(x_t,a,l)) - \mathbb{E}_{a\sim\hat{\pi}_\beta(a|s_t,l)}[Q_\theta(x_t,a,l)]]\end{aligned} \tag{70}$$

where $\hat{\pi}_\beta$ is the behavior policy, $\mathcal{B}^\pi$ is the Bellman operator, and the balance ratio $\alpha = 2$ in our experiment.

IQL: training an additional value network and extracting policy through advantage weighted regression.

$$\begin{aligned}\mathcal{L}_V(\psi) &= \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[L_2^\tau(Q_{\hat{\theta}}(x_t,a_t,l) - V_\psi(x_t,l)] \\ \mathcal{L}_Q(\phi) &= \mathbb{E}_{(s_t,a_t,r_t,s_{t+1},l)\sim\mathcal{D},(x_t,x_{t+1})\sim h_w}[(r_t + \gamma V_\psi(x_{t+1},l) - Q_\phi(x_t,a_t,l)^2] \\ \mathcal{J}(\pi_\theta) &= \mathbb{E}_{(s_t,a_t,l)\sim\mathcal{D},x_t\sim h_w}[\exp(\beta Q_\phi(x_t,a_t,l) - V_\psi(x_t,l))\log\pi_\theta(a|x_t,l)]\end{aligned} \tag{71}$$

The expectile $\tau = 0.7$, $\beta = 5$. We follow the authors' suggestions and subtract 1 from the reward if it equals 0.

To follow the original IL setting and simplicity, we merely use observation without action information in the history state encoding in BC and IQL. Our primary experiments show that it has little impact on the results. We also investigate recent approaches of language-conditioned IL such as LLfP [37] and R3M [44], but they perform poorly in prior experiments, so only BC-Z is chosen as the representative in the final baselines.

G Visualization Supplementary Results

We apply t-SNE visualization in different algorithms and task types to show that our method substantially improves task representation on offline language-conditioned RL. Beyond the main text, here are some supplementary visualization results.

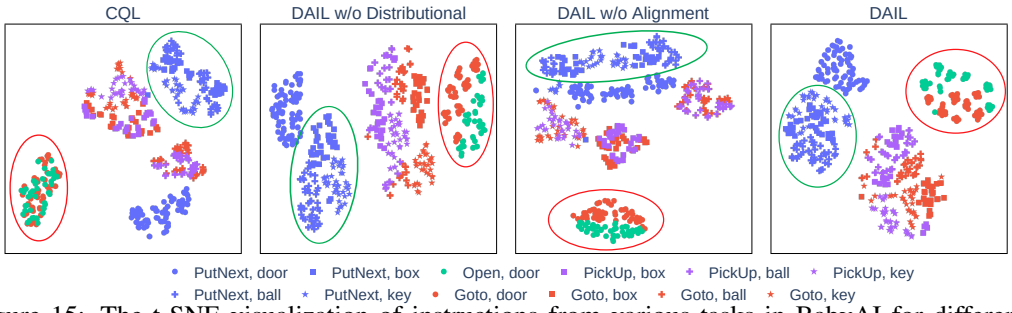

Figure 15: The t-SNE visualization of instructions from various tasks in BabyAI for different algorithms. The figure distinguishes between different **task categories** (e.g., PutNext) and **target object types** (e.g., box), using marker colors and shapes to represent each separately.

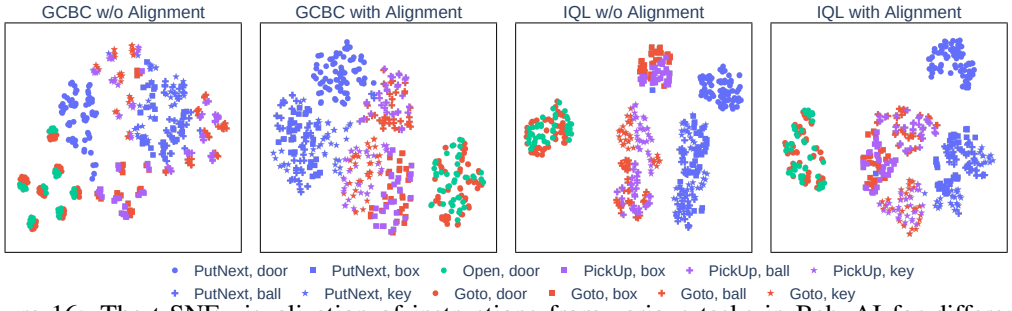

Figure 16: The t-SNE visualization of instructions from various tasks in BabyAI for different algorithms. The figure distinguishes between different **task categories** (e.g., PutNext) and **target object types** (e.g., box), using marker colors and shapes to represent each separately.

Overall task representation As introduced in Section 5, in BabyAI `SynthLoc` tasks, there are four main categories of tasks: `Goto`, `PickUp`, `PutNext`, and `Open`. We sample tasks from all four categories to visualize in Figure 15. Our method demonstrates superior task embedding capabilities compared to other approaches in the `Goto`, `PickUp`, and `Open`—effectively reducing confusion as highlighted by the red circles. Some degree of confusion remains inevitable in the `PutNext` tasks, due to their complexity and variability (as indicated by the green circles).

We also visualize representations from GCBC and IQL (with and w/o alignment) in Figure 16. GCBC shows reliable task representation, but confuses tasks from `Goto` and `PickUp`, "Open, door" ●and "Goto, door" ●, where their embeddings are tightly gathered into small clusters. Our further results in Figure 18 show that each cluster represents a specific color. Similarly, alignment significantly eases this issue by separating each instruction. IQL shows similar results to CQL, while alignment has a more significant influence on CQL. This result explains why simple GCBC shows comparable performance in our settings while vanilla offline RL fails due to confusion in task encoding.

Task `PickUp` and `Goto` We take a closer look at tasks in `GoTo` and `PickUp` that only have one target object. The task representations of DAIL and ablation algorithms are shown in Figure 17, and GCBC and IQL (with and w/o alignment) are shown in Figure 18. The color of the markers indicates that of the target objects other than black markers. As illustrated in Figure 7, our method further subdivides tasks (for example, ●and ●) into smaller clusters. Upon closer observation, these smaller clusters correspond to different target object colors. Similarly, CQL performs poorly in object color recognition, while distributional representation and semantic alignment substantially help task discrimination. As illustrated by the red circles, when the target object type is key, only our method

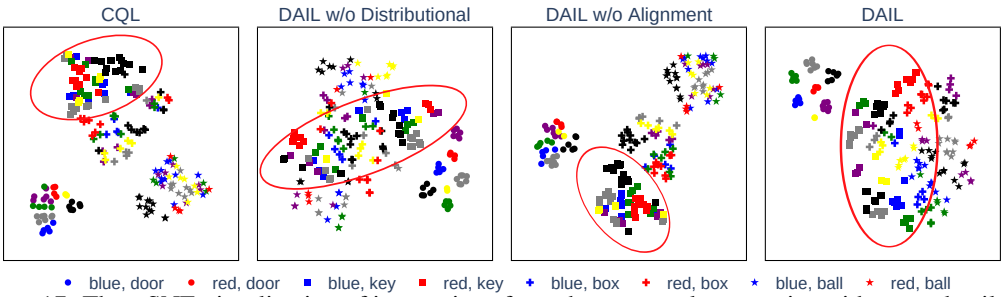

Figure 17: The t-SNE visualization of instructions from the same task categories with more detailed distinction. The figure distinguishes between different **target object types** (e.g., door) and **target object colors** (e.g., blue), using marker colors and shapes to represent each separately. For example, "go to the red door" corresponds to ●;. "go to a red ball behind you" corresponds to ⋆, "pick up the ball in front of you" corresponds to ⋆.

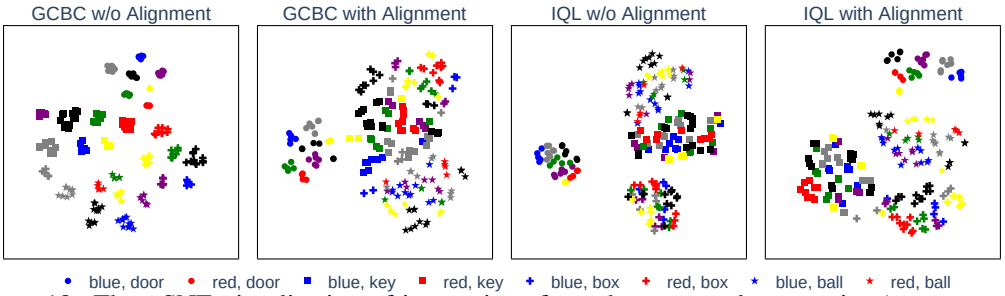

Figure 18: The t-SNE visualization of instructions from the same task categories (`PickUp` and `Goto`) with more detailed distinction. The figure distinguishes between different **target object types** (e.g., door) and **target object colors** (e.g., blue), using marker colors and shapes to represent each separately. The legend has the same meaning as in Figure 17.

succeeds in separating embeddings of different target colors while forming clusters for targets of the same color. In contrast, other methods tend to produce entangled representations, leading to less distinguishable task embeddings.

GCBC yields strong results in this scenario, whereas IQL relatively performs poorly. However, excessive overlap in GCBC suggests a confusion between the `PickUp` and `Goto` tasks. IQL can distinguish between different types of targets (different shapes of markers) but fails in differentiating target colors (for example, ■ ■ ■ . . .). After our alignment method was applied, its performance significantly improved.

Representation with instruction texts We add some instruction texts to the representation map to better demonstrate the language instructions' representation results. We draw around 300 instructions from BabyAI `SynthLoc` level to visualize and sample around 20 tasks and include their original text in the figure, displayed to the right of the corresponding marker. Figure 19 shows the detailed representation result of vanilla CQL, and Figure 20 shows that of our method DAIL. The red dashed circles highlight the "Goto, door ●" and "Open, door +" tasks. Our method DAIL successfully separates the two task categories and further organizes them into clusters (based on colors, see Figure 17) while CQL fails. Similarly, the green dashed circles denote the "PickUp, key ●" and "Goto, key ■" tasks. While CQL fails to disentangle the `PickUp` and `Goto` task types, our method achieves a clear separation between them.

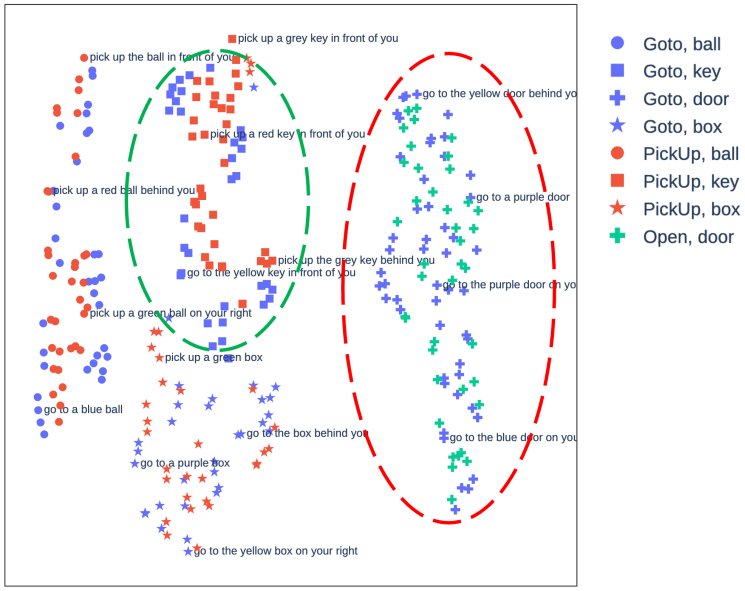

Figure 19: The t-SNE visualization of instructions with text annotations from `Open`, `Goto`, `PickUp` in BabyAI for CQL. The figure distinguishes between different **task categories** (e.g., PickUp) and **target object types** (e.g., box), using marker colors and shapes to represent each separately.

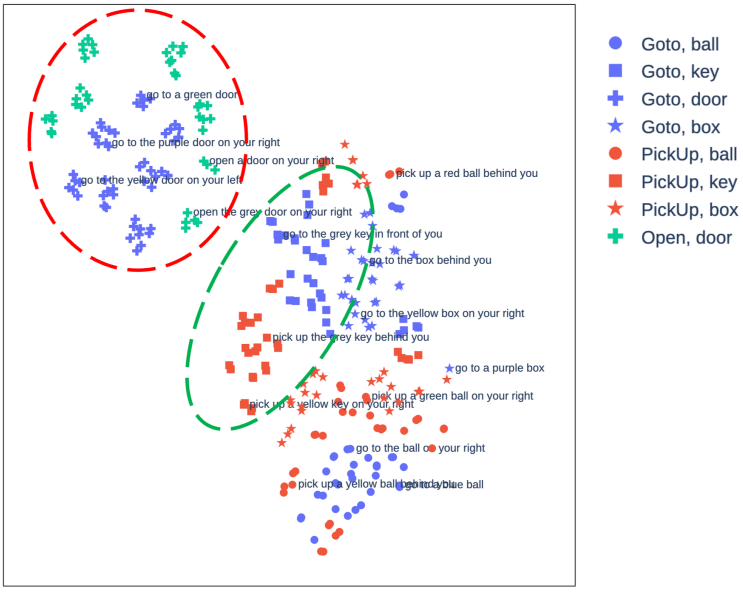

Figure 20: The t-SNE visualization of instructions with text annotations from `Open`, `Goto`, `PickUp` in BabyAI for our method. The figure distinguishes between different **task categories** (e.g., PickUp) and **target object types** (e.g., box), using marker colors and shapes to represent each separately.

H Additional Results

Results on the hight-quality dataset of BabyAI The success rates of in-distribution and out-of-distribution tasks on the high-quality dataset are shown in Table 5. in both in-distribution and out-of-distribution tasks. Our method demonstrates significant advantages over other approaches on out-of-distribution tasks, particularly achieving substantial performance improvements on complex tasks such as `PutNext` compared to the baseline RL method CQL (49.1% vs. 27.6%). Vanilla offline RL algorithms like CQL and IQL underperform compared to imitation learning methods like GCBC, which we attribute to the adverse impact of task ambiguity on RL-based approaches as discussed in Theorem 1. On the other hand, modified algorithms designed for language-conditioned IL (BC-Z and GRIF) perform poorly under our setting. This is primarily because their contrastive learning objectives are not robust in the presence of noisy or suboptimal data. In contrast, our alignment-based approach, built on an offline RL framework, maintains strong performance.

Table 5: Success rate of in-distribution tasks and out-of-distribution BabyAI tasks. Each score is evaluated over 3 seeds.

Algorithm	Open	Goto	PickUp	PutNext	All
In Distribution					
GCBC	96.9±0.8	91.8±1.1	85.6±0.0	27.6±3.3	79.1±1.3
BC-Z	96.3±0.7	77.5±1.0	49.9±4.4	14.2±0.7	64.0±1.8
GRIF	96.6±0.8	89.4±2.5	87.6±0.1	27.7±3.7	78.6±2.5
IQL	**98.2**±0.4	87.9±1.4	73.7±1.1	26.2±3.5	75.2±0.7
CQL	**98.7**±0.3	92.2±0.9	83.8±1.7	25.6±2.5	78.1±1.6
DAIL (ours)	97.2±0.2	**96.5**±1.4	**94.9**±1.4	**57.9**±0.9	**89.2**±0.5
Out of Distribution					
GCBC	94.4±2.5	90.3±1.6	78.4±2.1	27.4±1.6	74.1±0.7
BC-Z	93.7±1.0	76.9±3.0	45.4±1.5	11.2±3.3	57.9±1.8
GRIF	95.9±1.7	88.8±2.6	75.6±3.9	22.5±2.7	71.2±2.6
IQL	98.0±0.4	86.1±1.2	70.4±3.6	21.4±3.1	69.7±2.3
CQL	98.8±0.5	88.9±2.1	71.9±2.2	27.6±0.8	72.6±0.4
DAIL (ours)	**99.0**±0.2	**91.3**±1.0	**87.6**±2.0	**49.1**±1.8	**81.7**±1.3

Results on the medium-quality dataset of BabyAI The success rates of in-distribution and out-of-distribution tasks on the medium-quality dataset are shown in Table 6. Due to the lower proportion of successful trajectories, learning in this dataset is more challenging. As a result, all methods show a significant decline in performance compared to results on the high-quality dataset (Table 1). However, our method still achieves optimal results, especially in the `PutNext` task category.

Table 6: Success rate of on the medium-quality dataset. Each score is evaluated over 3 seeds.

Algorithm	In Distribution		Out of Distribution	
	PutNext	All	PutNext	All
GCBC	15.6±1.5	58.0±2.2	10.5±1.3	52.6±2.3
BC-Z	4.4±1.0	54.2±0.2	5.0±1.6	49.7±1.4
GRIF	7.0±1.5	61.4±0.2	4.9±2.5	54.7±0.6
IQL	17.7±2.8	67.3±0.5	12.3±0.9	61.4±0.0
CQL	13.5±1.8	69.2±0.7	14.5±1.9	63.4±1.3
Ours	**32.8**±2.7	**81.3**±0.7	**26.4**±0.6	**73.7**±0.6

Ablation of components To study the contribution of each component in our learning framework, we conduct the following ablation study. We compare the performance of algorithms that only

Table 7: Ablation results of AUC on the high-quality dataset. Each score is evaluated over 3 seeds.

Algorithm	10	20
CQL	38,385.3±853.5	81,876.0±848.0
DAIL w/o Alignment	40,374.7±601.3	85,247.1±540.2
DAIL w/o Distributional	40,752.9±248.7	85,416.5±543.6
DAIL	**42,370.6±404.8**	**88,450.5±291.6**

apply trajectory-wise alignment or distributional language-guided policy alone with our method on `SynthLoc`. The experimental results in Figure 21 show that both modules significantly improve the performance over vanilla CQL on in-distribution and out-of-distribution tasks. Further, combining both components can achieve the best performance compared to other approaches.

We further evaluate the sample efficiency of each method by calculating the Area Under the Curve (AUC) for the success rates of their learning curves, and present the AUC comparisons of in-distribution learning curves at the key milestones of 10 and 20 epochs in Table 7. Statistical analysis confirms that at 20 epochs, DAIL holds a significant lead over the other three ablation models, as demonstrated by Kolmogorov-Smirnov tests on the AUC results (p = 0.004).

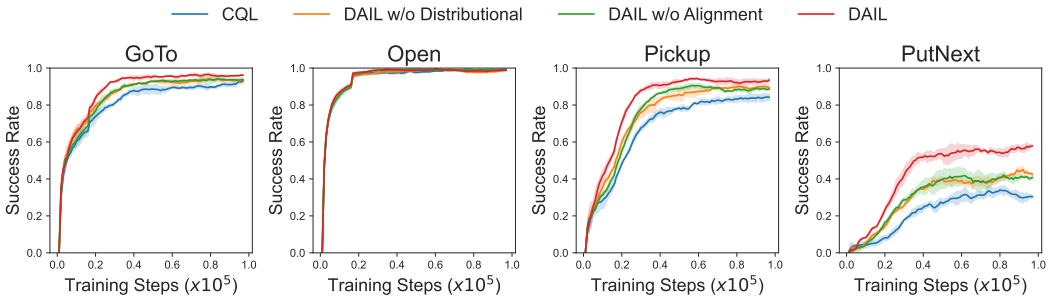

Figure 21: Ablation experiments on BabyAI tasks. The success rates are evaluated over 3 seeds.

Quantitative measure of clustering. To quantitatively demonstrate the impact of different components in DAIL on representation clustering, we measure the clustering quality with the Silhouette score[52], and present the result in Table 8. The labels required to calculate the score are defined as follows: two instructions share the same label if they require the agent to perform the same action on the same kind of object with the same color as the final target object. The distance between language embeddings is measured using cosine distance.

The results demonstrate that in the All-task setting, the clustering metric of CQL even exhibits negative values, indicating pronounced task confusion. In contrast, both proposed methods evidently improve clustering performance.

Table 8: Silhouette score of in-distribution BabyAI tasks.

Algorithm	All	PutNext
CQL	-0.030±0.005	0.004±0.007
DAIL w/o Alignment	0.024±0.016	0.048±0.008
DAIL w/o Distributional	0.110±0.019	0.088±0.020
DAIL	**0.127±0.013**	**0.107±0.012**

Ablation of α In Equation 13, α is the weight of the CQL loss. The ablation is done on the high-quality dataset in BabyAI tasks with various α. The experimental results in Table 9 indicates that a wide range of α values (from 0.5 to 2) yield comparable performance, which drops off at the extremes ($\alpha = 0.2$ and $\alpha = 5$). We therefore recommend setting α between 0.5 and 2.

Table 9: Ablation experimental results on α. Each score is evaluated over 3 seeds.

α	In Distribution		Out of Distribution	
	PutNext	All	PutNext	All
0.2	44.0±3.6	83.7±0.7	40.8±5.1	78.4±1.7
0.5	57.4±2.2	88.1±0.3	50.7±2.7	83.1±0.8
1	56.2±3.6	87.7±1.3	**50.8±3.7**	**84.1±0.7**
2	**57.9±0.9**	**89.2±0.5**	49.1±1.8	81.7±1.3
5	44.8±3.2	85.2±0.2	37.1±5.1	77.4±1.7
10	35.7±2.3	82.8±1.1	39.1±4.4	77.2±0.7

I Demonstration Trajectories in ALFRED

We present extended trajectory visualizations of DAIL's task execution in the ALFRED benchmark, illustrating its semantic comprehension and generalization capabilities.

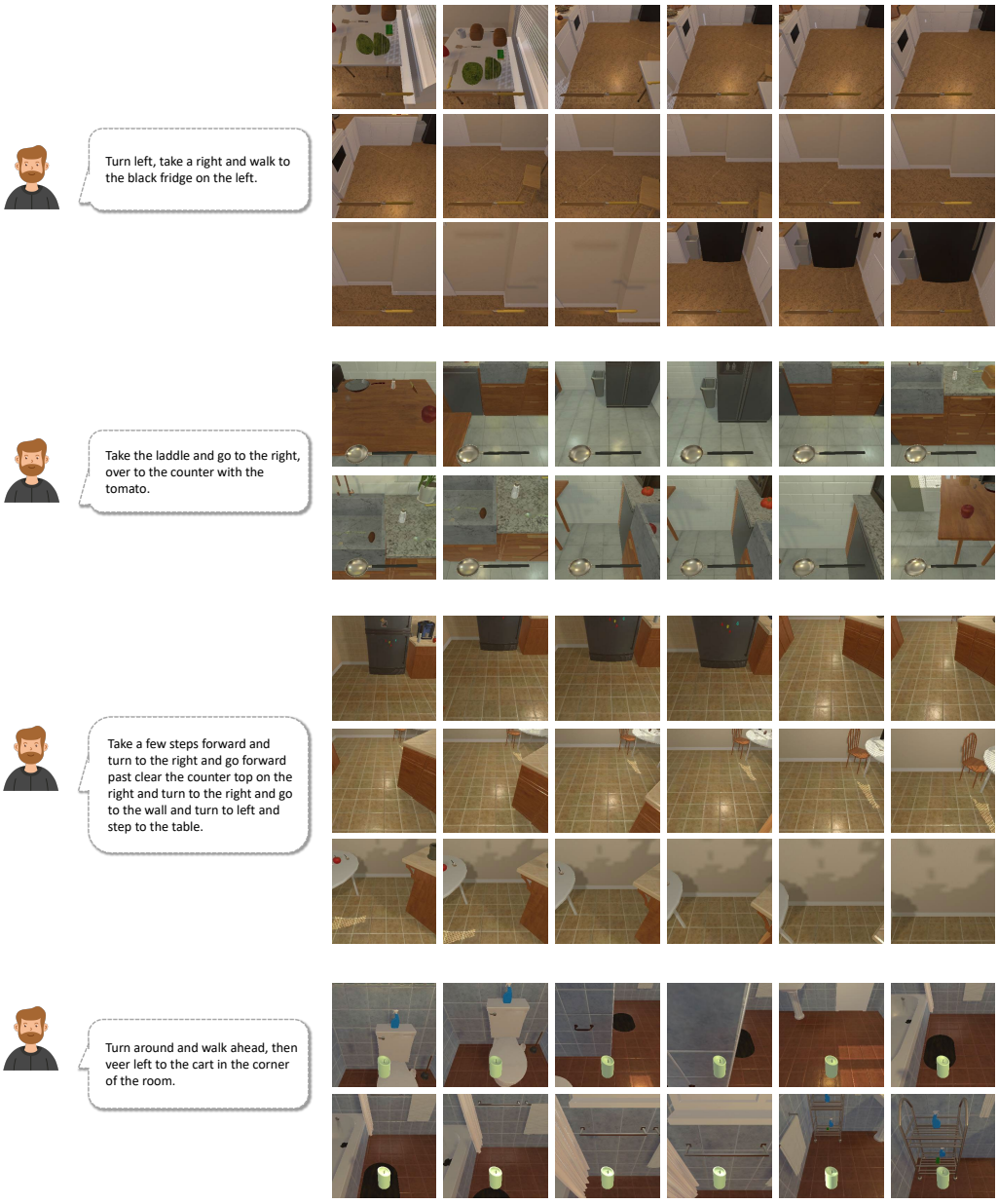

Figure 22: Extended trajectories of DAIL in ALFRED validation tasks.