RevealIt: REinforcement learning with Visibility of Evolving Agent poLicy for InTerpretability

Anonymous authors
Paper under double-blind review

Abstract

Understanding the agent's learning process, particularly the factors that contribute to its success or failure post-training, is crucial for comprehending the rationale behind the agent's decision-making process. Prior methods clarify the learning process by creating a structural causal model (SCM) or visually representing the distribution of value functions. Nevertheless, these approaches have constraints as they exclusively function in 2D environments or with simple transition dynamics. Understanding the agent's learning process in complex environments or tasks is more challenging. In this paper, we propose RevealIt, a novel framework for explaining the learning process of an agent in complex environments. Initially, we visualize the policy structure and the agent's learning process for various training tasks. By visualizing these findings, we can understand how much a particular training task or stage affects the agent's test performance. Then, a GNN-based explainer learns to highlight the most important section of the policy, providing a clearer and robust explanation of the agent's learning process. The experiments demonstrate that explanations derived from this framework can effectively help optimize the training tasks, resulting in improved learning efficiency and final performance.

1 Introduction

Reinforcement learning (RL) involves an agent acquiring the ability to make decisions within an environment to maximize the total reward obtained over a series of attempts. The recent achievements in resolving decision-making challenges demonstrate the effectiveness of this paradigm, e.g., video games (1; 2), robotic controlling (3). Despite numerous remarkable achievements over the past few decades, applying RL methods in the real world remains challenging. One of the main obstacles is that RL agents lack a fundamental understanding of the world and must, therefore, learn from scratch through numerous trial-and-error interactions. Nevertheless, the challenge of verifying and predicting the actions of RL agents frequently impedes their use in real-world scenarios.

This difficulty is exacerbated when RL is paired with deep neural networks' generalization and symbolic power. Insufficient comprehension of the agent's functioning impedes the ability to intervene when needed or have confidence in the agent's rational and secure behavior. Explainability in RL refers to the ability to understand and interpret the decisions made by an RL agent. Explanations reflect the knowledge learned by the agent, facilitating in-depth understanding and they also allow researchers to participate efficiently in the design and continual optimization of an algorithm (4). Recent work (5; 6) in explaining RL models to gain insights into the agent's decision-making process. Some (7; 8) propose to understand the agent's action by visualizing the distribution of the value function or state value, which is straightforward but limited by the environments, i.e., the distribution of the learned value function is easy to understand in 2D environments but unfeasible in 3D environments.

A line of recent work in explanation and causal RL (9; 10; 11) proposes to understand agent's behavior based on a single moment or a single action. Intuitively speaking, a one-step explanation is acceptable, but is it the best approach for understanding the agent's behavior? For example, (12) proposes learning a causal model to understand what causes an agent to succeed or fail in a given task. However, establishing a causal model

with precise causal assumptions in complex environments is challenging and inefficient (13; 14; 15). On the other hand, this becomes more challenging in a long-horizon task as the agent must do hundreds of steps before accomplishing the task. Hence, we debate **the definition of a good interpretability framework for providing explanations in RL**. Firstly, to answer the question, "Why can an agent succeed or fail in a task?". Indeed, we may deduce the pertinent factors from the agent's training process. Similar to how we may make a general assessment of a person's test performance based on their learning process, an agent also requires specific abilities to accomplish a task. The agent's mastery of these abilities during the training process will directly impact its ability to effectively complete the task. Conversely, a proficient explanation can enhance the comprehension of the agent's actions and facilitate the agent's improvement of its performance (11; 16). Finally, an effective interpretability framework necessitates providing intuitive and comprehensible explanations.

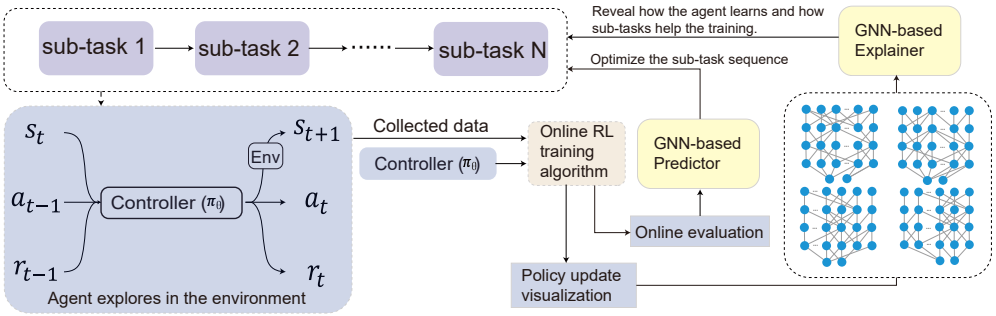

Figure 1: **The main structure of RevealIt.** In this setting, we aim to train an RL agent within an environment to accomplish complex tasks. Drectly on such tasks can be inefficient; therefore, we introduce a sequence of predefined subtasks (sub-task 1→N) to structure the training process. The agent interacts with the environment to explore and collect data, after which the control policy (π_θ) is updated using a chosen RL algorithm. We then visualize the policy updates using a node-link diagram, which illustrates the policy structure and the corresponding changes in weights throughout training. A GNN-based explainer is employed to identify the most significant policy updates, while a GNN-based predictor estimates the agent's learning progress on each subtask. Notably, the RevealIt framework is compatible with any online RL algorithm.

Based on the discussion about a good interpretability mechanism, we propose understanding the agent's performance after training from a more fundamental perspective, i.e., **the learning process of the policy and the sequences of training tasks**. By focusing on the policy and tasks, we may effectively mitigate this problem. They possess a minimum of three evident benefits: **(1)** Both the training tasks and the policy provide well-organized and generalizable information, allowing it to work in diverse environments. **(2)** The information presented in the policy is aligned with the agent's capacities. The limitation of the policy rests in our ability to extract intuitive explanations from the high-dimensional information it provides to understand the actions and behaviors of the agent. **(3)** When compared to SCM or counterfactual methods, which can not deal with the big and complex problems that can not be seen in the real world, using structured training tasks as the goal can solve this problem. This can be further enhanced by extracting relevant information from the policy. At this location, the agent's learning progress can serve as a temporal framework to enhance our comprehension of the agent's behavior.

Considering the preceding discussion, we propose a novel explanation framework, RevealIt. This framework can visually represent updating the policy and the agent's learning process. In addition, it can offer explanations for the training tasks. This paradigm facilitates a more intuitive comprehension of how agents acquire knowledge in various tasks, identifies which tasks are more efficient in training agents, and explains the reasons for their effectiveness. Gleichzeitig, this architecture can offer a reliable foundation for optimizing task sequences. In RevealIt, we first adopt a node-link diagram to visualize the structure of the policy. This design enables humans to directly see the updates made by the RL agent during training. By doing so, we can gain insights into the reasons behind the agent's success or failure during testing. Simultaneously, we can also perceptibly observe and compare the components of the policy that exist autonomously or are common across several tasks in diverse training scenarios. This offers a visual foundation for aiding the design

of training tasks in intricate contexts. Then, We adopt a GNN-based explainer that learns from the policy update process and subsequently analyzes its correlation with the policy's functionality during evaluation. This analysis aims to comprehend the connection between the behaviors performed during training and test. Note that while this explainer will not directly view the data collected from the environment, it is trained in sync with the RL agent. Essentially, we may consider the training of this explainer as a POMDP problem. In the experiments, we test RevealIt in various environments, and the results demonstrate the ability to effectively elucidate the agent's learning process in the training task, optimize the sequence of tasks, and enhance the effectiveness of reinforcement learning.

2 Related Works

Explainability for RL. (17; 18) categorize the explainable RL methods into two groups. The first group is "Post hoc", which aims to provide explanations after the model execution, while the other group is "intrinsic approaches", which inherently possess transparency. Many post hoc explanations, like the saliency map approach (19; 20), are based on correlations. This means that conclusions drawn from correlations may not be accurate and do not answer questions about what caused what. However, these algorithms may not be able to model complex behaviors (17) fully. Our method proposes a new mechanism in the problem of explainability for RL from the perspective of the agent's learning process, i.e., it can help to understand how the agent is learning in a given task or environment.

Explanation in real world. (21) provides a general discussion on the challenges posed by reality in real-world scenarios. The non-stationary and constant evolution in the real world requires the RL algorithm to be more robust in handling these changes. (22; 23) found nine significant problems with RL that make it hard to use in real life. These problems include small sample sizes, unknown or long delays, complex inputs, safety concerns, limited observability, ambiguous or multiple reward functions, short delays, offline learning, and explainability. In this paper, we will address the limitations of generating explanations for RL in a high-dimensional environment from the perspective of the training tasks and the learning progress.

Curriculum RL. Curriculum RL focuses on improving the efficiency and performance of RL agents by organizing training tasks in a meaningful sequence, progressing from simpler to more complex tasks. Previous works (24; 25) have explored the role of task sequencing in enhancing learning by adapting the difficulty or order of tasks based on the agent's capabilities. These approaches often rely on heuristics or predefined strategies to design task curricula. In contrast, RevealIt employs a data-driven approach, leveraging a GNN predictor to optimize task sequences based on the agent's real-time learning progress. By aligning task sequences with the agent's evolving policy, RevealIt enhances learning efficiency and provides interpretable insights into how tasks contribute to overall agent performance.

3 Preliminaries on Reinforcement Learning and Visualization

Our framework comprises two parts. One is the RL policy visualization module that shows the policy update in training and policy activation in the test. This module is based on fully connected neural networks' well-established node-link diagram representation. The other is a GNN-based explainer, trained to analyze the visualized results in the first module and underly the critical update in the training phase. We can further explore the relationship between different training tasks and the contribution of a specific training task/phase to the final performance. Now, we provide more details on these two parts:

Reinforcement Learning. The problem of controlling an agent can be modeled as a Markov decision process (MDP), denoted by the tuple$\{\mathcal{S}, \mathcal{A}, P, R, \mu_0, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ represent the set of state space and action space respectively, $P : \mathcal{S} \times \mathcal{A} \times S \to [0, 1]$ is the transition probability function that yields the probability of transitioning into the next state s_{t+1} after taking an action a_t at state s_t, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that assigns the immediate reward for taking an action a_t at state s_t. $\mu_0 : \mathcal{S} \to [0, 1]$ is the probability distribution that specifies the generation of the initial state, and $\gamma \in [0, 1]$ denotes the discount factor. A is a mapping from states to actions: $\pi : \mathcal{S} \to \mathcal{A}$. Every episode starts with sampling an initial state s_0. At every timestep t the agent produces an action based on the current state: $a_t = \pi(s_t)$, A discounted sum of future rewards is called a return: $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$. The agent's goal is to maximize

its expected return $\mathbb{E}_{s_0}[R_0 \mid s_0]$. In RevealIt, We can use a deep RL algorithm to train the agent (e.g., DQN (2), A2C (26), SAC (27), PPO (28), etc.). In this paper's experiments, we use DQN and PPO as the foundational RL methods. Note that any online RL algorithm can be accepted.

Strucutral visualization of the policy. The visualization should meet the following goals: summarize the unique properties of MLP networks in RL and reflect lessons learned from prior tasks. We choose to visualize the policy with a node-link diagram, and the only issue with a straightforward implementation of a node-link diagram is that large networks yield a dense mass of edges between layers. We base the visualization model on (29), which can choose to display some or all of the weights connected to a specific node, and record and correspond to the changes in the value of a specific part of the weights during the training process.

4 RevealIt

In RevealIt, assume that we first have a randomly drafted subtask sequence for the RL policy training. We start from the conventional RL training process, i.e., the agent will explore and interact with the environment, trying to complete each sub-task. After the agent collects data, we can apply any online RL algorithm to update the control policy. We can visualize the policy update with a node-link graph G_O by comparing which part of the policy weights are updated. Subsequently, for every fixed number of environmental steps the agent takes, we can compute the true learning progress of the RL agent by evaluating the current policy. We then construct a dataset for training the GNN explainer training based on the learning progress and graphs of policy updates G_O. We formulate the training scheme of RevealIt as in Alg. 1 and introduce more details in RevealIt as follows.

4.1 Background on GNN-based Explainer

This section establishes the notation and provides an overview of GNN's structured explainer. Assume that we have visualized the learning progress of the RL policy with node-link diagrams following Alg. 1. Let G_O denote the observed diagram on edges E (corresponding to the weights) and nodes V (corresponding to the node in the policy network) associated with $D-$dimensional node features $\mathcal{X} = \{x_1, \cdots, x_n\}$, $x_i \in \mathbb{R}^D$, and v_i represents the i-th node in V. We use the training data of the RL policy to construct the training set for this GNN explainer. In this set, we will record the update information of the policy π_θ, i.e., for a node v_T^i in the policy at time T, we denote the weights information connected with this node as $\mathcal{X}_T^i$, and denote the updated weights connected to this node v_T^i as $\mathcal{X}_{T+1}^i$. Thus, we can construct the dataset for training GNN explainer by calculating the updated information by $|\mathcal{X}_{T+1}^i - \mathcal{X}_T^i|$.

Algorithm 1 RevealIt

Initialize: control policy π_0, RL replay buffer $\mathcal{B} \leftarrow \emptyset$, policy updated dataset $\mathcal{D}_p \leftarrow \emptyset$, a set of N training tasks $\mathcal{D}_{task}$, randomly sampled sequence of training tasks Seq_0, the predicted learning progress $\{\mathcal{P}(task_n, \pi_0) = 0 | n = 1, 2, \cdots, N\}$;
for t in iterations **do**
  $p =$ `random()`;
  if $p < \epsilon$ **then**
    Randomly sample task sequence Seq_t from $\mathcal{D}_{task}$;
  else
    Sample task sequence Seq_t in terms of $\{\mathcal{P}(task_n, \pi_t)\}$;
  end if
  for $task_i$ in Seq_t **do**
    RL agent collects MDP data to $\mathcal{B}$ to complete $task_i$;
    Use RL algorithm to update π_0 using samples from $\mathcal{B}$;
    Evaluate the current policy π_t and compute $\mathcal{P}(task_i, \pi_t)$ following the Eq. 1;
    Visualize policy update information with graph $G_{O,t}$;
    Save $(G_{O,t}, \mathcal{P}(task_i, \pi_t))$ into $\mathcal{D}_p$;
  end for
  Train GNN predictor to minimize $|\hat{\mathcal{P}} - \mathcal{P}|^2$ based on $\mathcal{D}_p$;
  Train GNN explainer to partition $G_{O,t}$ into the optimal subgraph $G_{X,t}^m$;
end for

The l-th layer of the GNN explainer performs three essential computations: MESSAGE, AGGREGATE, and UPDATE. The model first calculates the messages between every pair of nodes by $m_{ij}^l = MESSAGE(h_i^{l-1}, h_j^{l-1}, r_{ij})$, where h_i^{l-1} and h_j^{l-1} denote the node representations of v_i and v_j obtained from the previous layer $l-1$, respectively, and r_{ij} represents the relation between them. Next GNN aggregates the messages for each node v_i by $m_i^l = AGGREGATE(m_{ij}^l | v_j \in \mathcal{N}(v_i))$, where $\mathcal{N}(v_i)$ refers to the neighborhood of v_i. Finally, the GNN explainer updates the node representation from the previous layer by $h_i^l = UPDATE(m_i^l, h_i^{l-1})$. The final embedding for node v_i after L layers of computation is $z_i = h_i^L$, which is then used for node classification, i.e., the understanding the critical nodes in the RL agent's evaluation is a crucial pre-requisite for determining the significance of weights updating.

4.2 RevealIt

The learning objective of GNN predictor. The overall goal of the GNN explainer is to learn to optimize the sequences of training tasks to improve the learning efficiency of the RL agent. We take inspiration from the learning progress in earlier work on curriculum learning (24; 30), i.e., a higher learning progress signifies that the agent can acquire more knowledge regarding a specific training task, reflecting a higher learning efficiency. Therefore, in RevealIt, we train the GNN predictor to learn to predict how much improvement the RL agent can have in a given task. Based on the discussion in Sec. 4.1, we denote the prediction task of our GNN explainer as Φ: $G_O \rightarrow \hat{\mathcal{P}}$, where $\hat{\mathcal{P}}$ is the predicted improvement of the RL policy. Thus, in a given task n, we train the GNN predictor to minimize the error between $\hat{\mathcal{P}}$ and the true learning progress $\mathcal{P}$ defined as:

$$\mathcal{P}(task_n, \pi_t) = \mathcal{R}(task_n, \pi_t) - \mathcal{R}(task_n, \pi_{t-1}), \tag{1}$$

where $\mathcal{R}(task_n, \pi_t)$ denotes the expected return that the RL agent achieves with policy π_t.

Train GNN explainer with RL agent simultaneously. In RevealIt, we separately train a GNN explainer that helps to understand how an RL agent learns to complete tasks, i.e., the structural visualization of the policy allows us to observe which part of the policy weights is updated in one iteration, and the GNN explainer learns to highlight the most crucial updates related to the agent's final success. One thing to note is that the training data used for the GNN explainer is based on the RL, which allows the GNN explainer to adapt to the learning process of the RL agent and can provide guidance during the training. More specifically, as the RL agent acquires MDP data from the interactions with the environment and learns from it, we can use any online RL algorithm to update the RL policy. After each policy update, we can obtain a structured update visualization of the policy by comparing the changes in the policy weight parameters and constructing the dataset for the GNN explainer based on the updated information. We can regard the visualized policy as a node-link graph, i.e., the updated weights can be viewed as the edges that connect with nodes in the graph. Thus, the learning objective of the GNN explainer can be described as "learn to find which part of the edges (updated weights) is more important for the success of the RL agent". Now, we introduce more details of the GNN explainer to help the understanding.

Step 1. Understanding the visualized policy. We first train the GNN explainer on the learning data collected from the RL. During the RL evaluation, the activated nodes in the policy will be tagged and utilized as the ground truth for the GNN explanation.

Note that the active nodes during evaluation will vary as the RL policy training progresses. This variability may raise issues regarding the stability of GNN's training. However, this problem is non-existent. This is because we require the GNN explainer to evaluate the significance of the training job by considering the RL agent's present level of learning progress. The active nodes in the evaluation and the collected environment reward reflect the capabilities of the current policy. When the policy is updated, especially when the agent's ability is increased, the different activated nodes and the increase in the environment reward can give the GNN predictor the information it needs. This enables the predictor to acquire knowledge on predicting the agent's improvement based on the current learning process. Consequently, the GNN predictor can more precisely assess the value of the current training task.

Step 2. Highlight the important updates for explanations. As mentioned in Sec. 3, the control policy can be too complex to understand humans, even with the visualized diagrams of the policy. Therefore, we also train the GNN explainer to highlight these important nodes and their related updates to humans to

achieve a feasible explanation. The GNN explainer aims to partition G_O into two subgraphs:

$$G_O = G_X + \Delta G, \tag{2}$$

where G_X represents the explanatory subgraph reflecting the important edges (corresponding to policy updates), and ΔG is the remaining graph containing unimportant edges. When explaining a graph of visualized policy updates, we denote the optimal subgraph by G_X^m. When G_X^m is used as input to Φ, it can retain the original prediction. G_X^m is optimal, as removing the features from it would result in a different prediction.

The learning objective of GNN explainer. In RevealIt, We use the loss in the PGEexplainer (31) to train the GNN explainer, and we use an MSE loss for the GNN predicter. On the other hand, in the traditional GNN explainer method, the learning objectives of the predictor (evaluator) GNN and the explainer are often consistent, i.e., maximizing the mutual information between the label and the selected subgraph, and we can express this process as:

$$\max_{G_s} = MI(Y, G_S) = H(Y) - H(Y|G = G_S), \tag{3}$$

where G_S denotes the masked subgraph, the GNN explainer optimizes this function locally by training an NN to provide a customized explanation for each instance. However, RevealIt optimizes this by adopting an NN to learn the crucial edges on multiple instances (each evaluation result from the RL agent).

The different roles of GNN explainer and predictor in RevealIt. The GNN predictor and explainer serve distinct roles in RevealIt. The former assesses the improvement of the training task on the RL policy performance, thereby increasing the learning efficiency, whereas the latter evaluates whether the GNN predictor comprehends the learning process of the RL agent by analyzing the correlation between "nodes linked to significant updates" and "the activated nodes during the test". Simultaneously, by analyzing the graphs of the policy update by the GNN explainer, we can ascertain the ability learned by the RL agent in a specific task and comprehend the significance of this capability in the RL agent's success in the final task. RevealIt is distinguished from other explainable RL methods by its independence from external data and structure. It learns to explain the learning process of the RL agent within the training task without imposing constraints on the environment or specific RL algorithms.

5 Experiments

We design our experiments to answer the following questions: (1) Can RevealIt show the learning process of an RL agent in a given environment? (2) Can RevealIt improve the learning efficiency of the agent based on the explanations? Our project can be viewed by: RevealIt.

5.1 Experiments Setup

Environments. We base our experiments on two types of benchmarks. The first one is **ALFWorld benchmark** (32), a cross-modality simulation platform encompassing a wide range of embodied household tasks. ALFWorld combines a textual environment with a visual environment that the Ai2Thor simulator (33) renders for each task. This textual part uses the Planning Domain Definition Language (PDDL) (34) to turn each pixel observation from the simulator into a text-based observation that is equal to it. It then uses the TextWorld engine (35) to create an interactive environment. The tasks within the ALFWorld benchmark are categorized into six types: Pick &Place, Clean & Place, Heat & Place, Cool & Place, Look in Light, and Pick Two Objects & Place. Each task requires an agent to execute a series of text-based actions, such as "go to safe 1", "open safe 1", or "heat egg 1 with microwave 1", following a predefined instruction. These actions involve navigating and interacting with the environment. The other benchmark involves six OpenAI RL environments (36), which are commonly adopted in RL tasks.

The complex tasks in ALFWorld. In ALFWorld, accomplishing a given task requires the agent to complete a sequential arrangement of sub-tasks step by step (as shown in Fig. 2). A task may entail interactions with more than 10 items and necessitate over 30 steps for a human expert to solve, thereby

testing an agent's ability in long-term planning, following instructions, and using common knowledge. To facilitate a thorough comprehension, we have included an instance of each task category in the appendix in Fig. 4. Using the original tasks directly to train an RL agent is inefficient. Therefore, it is more suitable to use subtask training intuitively. However, this raises the question of how to structure the subtask sequence, specifically if the subtask sequence should remain consistent throughout the training process. What is the criteria for adjusting the subtask sequence? The crucial factor is determining the optimal subtask sequence that aligns with the agent's current training progress.

Baselines. The fundamental inquiry of RevealIt is whether it can operate effectively in complex tasks and environments, i.e., does it have the ability to enhance the training efficiency of RL significantly? Can it be modified to suit various reinforcement learning techniques? Hence, we will focus on answering the first question in the ALFWorld. The baseline methods include MiniGPT-4 (37), BLIP-2 (38), LLaMA-Adapter (39) and InstrctBLIP (40). They share state-of-the-art and competitive performance in ALFWorld and solely interact with the visual world, aligning with the approach of RevealIt. Our primary emphasis in the OpenAI RL benchmark was on the third question. We use PPO (28), SAC (41), DQN (2) as the fundamental RL algorithms to assess if RevealIt can enhance training efficiency and performance across various approaches.

5.2 Main Results

Visualization of the learning process in ALFWorld. In Figure. 2, we visualize the learning process of the RL policy in ALFworld. The RL policy we used in ALFworld is designed as an actor-critic structure consisting of two multi-layer prediction (MLP) networks as "actor" and "critic". Considering that when we evaluate or test the RL agents, the "actor" is responsible for making decisions, we primarily focus on showcasing the learning process of this aspect of the policy. The actor-network consists of 4 fully connected layers, each including 64 nodes. We implemented ReLU function on the first and third layers to visually identify the activated nodes throughout the evaluation process.

Agent	**Success Rate**						
	Avg.	Pick	Clean	Heat	Cool	Look	Pick2
ResNet- 18*	0.06	-	-	-	-	-	-
MCNN-FPN*	0.05	-	-	-	-	-	-
MiniGPT-4	0.16	0.04	0.00	0.19	0.17	0.67	0.06
BLIP-2	0.04	0.00	0.06	0.04	0.11	0.06	0.00
LLaMA-Adapter	0.13	0.17	0.10	0.27	0.22	0.00	0.00
InstructBLIP	0.22	0.50	0.26	0.23	0.06	0.17	0.00
PPO	0.04	0.01	0.05	0.04	0.07	0.03	0.00
RevealIt	0.76	**0.62**	**0.86**	**0.74**	**0.76**	**0.81**	**0.62**
Human Performance*	**0.91**	-	-	-	-	-	-

Table 1: **Comparison with the SOTA agents in ALFWorld.** Results marked with an asterisk are reported from prior work. The highest success rates for each task type are shown in bold. RevealIt significantly outperforms other state-of-the-art agents that interact with the visual environment, highlighting its potential to achieve near-human performance in ALFWorld.

RevealIt brings improvement. We report the performance of RevealIt on the ALFWorld benchmark in Tab. 1 and compare it to the performance of 7 other representative agents. We report the performance on the OpenAI benchmark in Tab. 2. Given that RevealIt exclusively engages with the visual engine within ALFWorld, the baseline agents similarly only interact with the visual environment, aligning with their configurations as stated in the original paper. We evaluate the primary metric: the success rate, defined as the proportion of completed trials. RevealIt demonstrates significantly better performance than other VLM agents, establishing its superiority. Notably, RevealIt does not rely on LLM agents, unlike other baseline models that utilize pre-trained LLM agents to generate planning steps using ALFWorld's text engine during the training phase. This demonstrates that the GNN-based explanation efficiently optimizes the sequence of tasks and significantly enhances the agent's training.

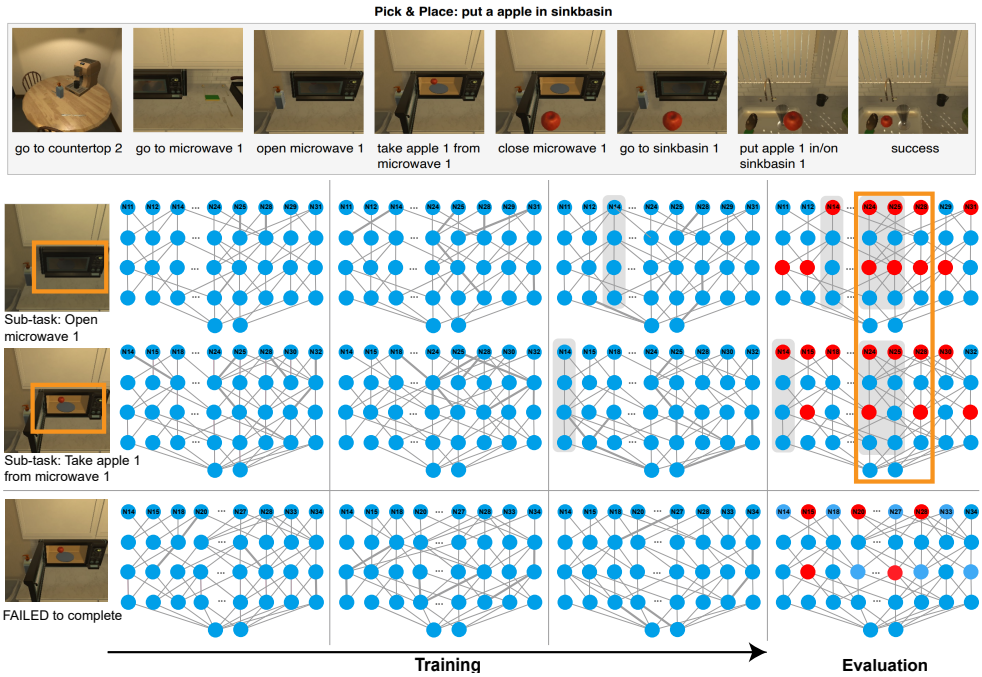

Figure 2: **Visualized important policy updates by GNN explainer.** The first row illustrates the sequence of sub-tasks that must be completed step-by-step to accomplish the overall task. The left side of the second row highlights one such sub-task. The tree diagrams in columns one through three visualize the corresponding policy update process: blue circles represent nodes in the policy network, and gray lines represent weight updates. Thicker lines indicate larger weight changes, as determined by the GNN explainer. The diagram in the fourth column shows red circles indicating the nodes activated during evaluation (i.e., important nodes). The third row displays the policy update when the agent fails to complete the sub-task.

The optimization of the training task sequences. We report the distribution of the sub-tasks in Fig. 3. As the training progresses, it becomes evident that the GNN-based explainer modifies the distribution of sub-tasks. Fig. 3 (left) shows the distribution of randomly sampled subtasks at the start of the training process. It is evident that the subtask "put" occurs most frequently. During the training, while examining the distribution of subtasks in Fig. 3 (middle left), task "put" remains the most common subtask; the frequencies of "look", "pick", and "find" have experienced a substantial increase. This demonstrates that these three qualities are important for the agent in ALFWorld. This aligns with our inherent comprehension of the environment, wherein the agent must initially acquire the ability to locate and retrieve the relevant item before it can proceed with future activities. As the training advances, the occurrence rate of these three task types declines proportionally (Fig. 3 (middle right)), suggesting that the agent already possesses the necessary skills. Training the agent with these tasks will not significantly enhance its performance. Fig. 3 (right) demonstrates that as the training advances, the training tasks increasingly emphasize the agent's additional skills, which align with the original work's final one or two stages. This indicates that the agent's training is almost over, and the alterations in the distribution of the training task align with our comprehension of the agent's learning process. It is evident that the explainer has comprehended and become proficient in the agent's learning process, enabling them to modify the training tasks effectively.

Comparison with other interpretability frameworks In the experiments on ALFworld, we mainly focus on how RevealIt can help in the RL training. However, it is necessary to evaluate the performance of our explanation mechanism, i.e., can other interpretability methods have better improvement for RL? Therefore, We add the comparison with other interpretability methods (GNNexplainer (42), MixupExplainer (43)) and report the results in the Tab.ref 3. Due to the limited time, we can only compare with GNNexplainer and MixupExplainer in ALFworld, and we keep the hyperparameters the same as those in the released code.

Agent	Environment					
	HalfCheetah	Hopper	InvertedPendulum	Reacher	Swimmer	Walker
PPO	1846.25 (1.00)	2250.46 (1.00)	986.85 (1.00)	-10.34 (1.00)	108.81 (1.00)	2954.01 (1.00)
PPO+RevealIt	**1921.08 (0.90)** ↑	2104.88 (0.90)	**1004.92 (0.90)** ↑	-11.27 (0.90)	**112.51 (0.90)** ↑	**3148.64 (0.90)** ↑
A2C	1014.02 (1.00)	630.51 (1.00)	1002.45 (1.00)	-27.02 (1.00)	25.28 (1.00)	676.52 (1.00)
A2C+RevealIt	**1147.89 (0.80)** ↑	**742.17 (0.80)** ↑	966.20 (0.80)	-28.54 (0.80)	17.63 (0.80)	**810.26 (0.80)** ↑
PG	602.27 (1.00)	2489.07 (1.00)	1028.33 (1.00)	-15.65 (1.00)	62.88 (1.00)	1280.29 (1.00)
PG+RevealIt	**742.33 (0.90)** ↑	2253.70 (0.90)	975.04 (0.90)	**-13.21 (0.90)** ↑	**70.66 (0.90)** ↑	**1546.38 (0.90)** ↑

Table 2: **Effect of RevealIt on learning efficiency in OpenAI Gym benchmarks.** This table presents the evaluation performance of three reinforcement learning algorithms, both with and without integration of RevealIt. The values in parentheses denote the number of environment steps (in millions) used during training. The results indicate that RevealIt enhances training efficiency across multiple environments.

Methods	Avg.	Pick	Clean	Heat	Cool	Look	Pick2
RevealIt	**0.80**	**0.66**	**0.90**	**0.81**	**0.80**	**0.85**	**0.70**
RevealIt (with GNNexplainer)	0.64	0.47	0.70	0.61	0.69	0.70	0.55
RevealIt (with MixupExplainer)	0.52	0.33	0.59	0.50	0.55	0.60	0.42

Table 3: **Comparison with alternative interpretability methods within the RevealIt framework.** In this ablation study, the GNN explainer component was replaced with other explanation models, while all reinforcement learning algorithms and training parameters were kept constant.

5.3 Ablation and Interpretability Evaluation

While the primary evidence of RevealIt's interpretability comes from qualitative visualizations of the learned internal policy graph, we complement these results with quantitative metrics that assess the *alignment* between the GNN explainer's importance scores and the actual learning dynamics of the policy network. Unlike state-centric explanation methods, our explainer operates over the policy's hidden units (nodes) and their connecting weights (edges). Therefore, an informative evaluation must measure whether the parameters identified as important by the explainer are indeed those that the learning process updates most and that most influence the policy's output.

To this end, we introduce four metrics:

Update Direction Alignment (UDA@k) measures the cosine similarity between the full policy update vector and the update restricted to the top-k% most important parameters according to the explainer. A high UDA@k indicates that the explainer's selection captures the main direction of learning.

Gradient Mass Recall (GMR@k) quantifies the fraction of the total update magnitude (in ℓ_2 norm) that lies within the top-k% selected parameters. This measures the extent to which the explainer covers where most of the "learning signal" flows.

Top-k Overlap (Prec@k) computes the proportion of the explainer's top-k% parameters that are also in the top-k% parameters ranked by the absolute magnitude of the actual update. This captures discrete agreement on which parameters change most.

Rank Correlation (ρ) uses Spearman's correlation between the explainer's importance ranking and the ranking by absolute update magnitude. This measures overall monotonic agreement, even beyond a fixed cutoff k.

Together, these metrics provide a multi-faceted view of explanation quality: UDA@k and GMR@k assess *coverage of update direction and magnitude*, while Prec@k and ρ evaluate *ranking consistency*. The results are reported in Table. 4-5. By grounding evaluation in the policy's own parameter updates, these metrics

directly test whether the explainer reveals the parts of the internal computation graph most responsible for the agent's learning behaviour.

Environment	k(%)	Prec@10%	GMR@10%	ρ
ALFWorld	10%	0.86	0.81	0.83
Hopper-v3	10%	0.84	0.78	0.81
HalfCheetah-v3	10%	0.88	0.82	0.85

Table 4: The results show that the explainer reliably recovers the edges with the largest structural changes in the policy during training, confirming its structural fidelity.

Environment	**Full REVEAL-IT**		**w/o Explainer (control)**	
	Acc (%)	**UDA@k**	**Acc** (%)	**UDA@K**
ALFWorld (Cool)	81.2 ± 3.1	0.431 ± 0.018	54.7 ± 1.9	0.071 ± 0.012
ALFWorld (Pick)	66.1 ± 2.6	0.347 ± 0.015	50.2 ± 1.8	0.049 ± 0.011
ALFWorld (Pick2)	70.4 ± 1.9	0.405 ± 0.017	52.8 ± 2.2	0.060 ± 0.013

Table 5: Recovery of edges with the largest parameter changes during training. Numbers are mean ± std over 5 seeds and 4 checkpoints per run.

To further analyze the contribution of each component in RevealIt and validate its interpretability, we conduct ablation studies and quantitative interpretability evaluations and report the results in Table. 6.

Variant	ALFWorld Avg. SR	ALFWorld SR Δ vs. Full	MuJoCo Efficiency
Full RevealIt	**0.80**	-	-
w/o PV	0.61	-0.09	-8.5%
w/o Pred	0.65	-0.15	-11.2%
w/o Exp	0.72	-0.08	-7.4%
Baseline RL	0.54	-0.26	-18.6%

Table 6: **Comparison of each component contributes to RevealItperformance**. This table compares the contribution of each component in RevealIt. "w/o PV" replaces structured node-link graph with raw parameter difference vector as GNN input. "w/o Pred" removes GNN Predictor; sub-task sequence sampled randomly. "w/o Exp" removes GNN Explainer and uses GNN Predictor to select sub-tasks but without highlighting important policy updates.

5.4 Understanding the Relationship between Sub-tasks from Policy

This section will explore how the agent acquires knowledge in various sub-tasks from the perspective of policy. Given that the different nodes in the policy will be activated in the evaluation, how will the weights linked to these nodes be modified? Can GNN-explainer learn to differentiate between the sections that have more significant updates in terms of weight? Fig. 2 illustrates updating the policy during training. Firstly, it is imperative to focus on the modifications in policy updates in every individual row. As training advances, we will observe that the sections with more significant policy updates will undergo modifications (shown by the thick lines). Furthermore, our findings indicate that as the training progresses toward the latter stage, there is a greater overlap between the region with a larger update amplitude and the region triggered by evaluation. This illustrates that GNN explainer comprehends the connection between policy training and evaluation. By carefully considering each vertical direction, we have identified and emphasized the sections of policies that are common to multiple activities (shown by gray boxes). By analyzing the neural network nodes within the gray box, we discovered that certain tasks, which necessitate agents to possess comparable abilities based on common knowledge, exhibit increasingly evident overlaps in their policies. For instance, the subtasks "open microwave 1" and "take apple 1 from microwave 1" both require the agent to comprehend

the spatial location of the microwave (highlighted by the orange box). However, the second subtask also demands the agent identify the object "apple" and its specific location. There is less collaboration across distinct subtasks. Upon reviewing the prior distance, we observe that the subtask "put apple 1 on sinkbasin 1" shares some similarities with the previous subtask. Both subtasks are interconnected with "apple 1".

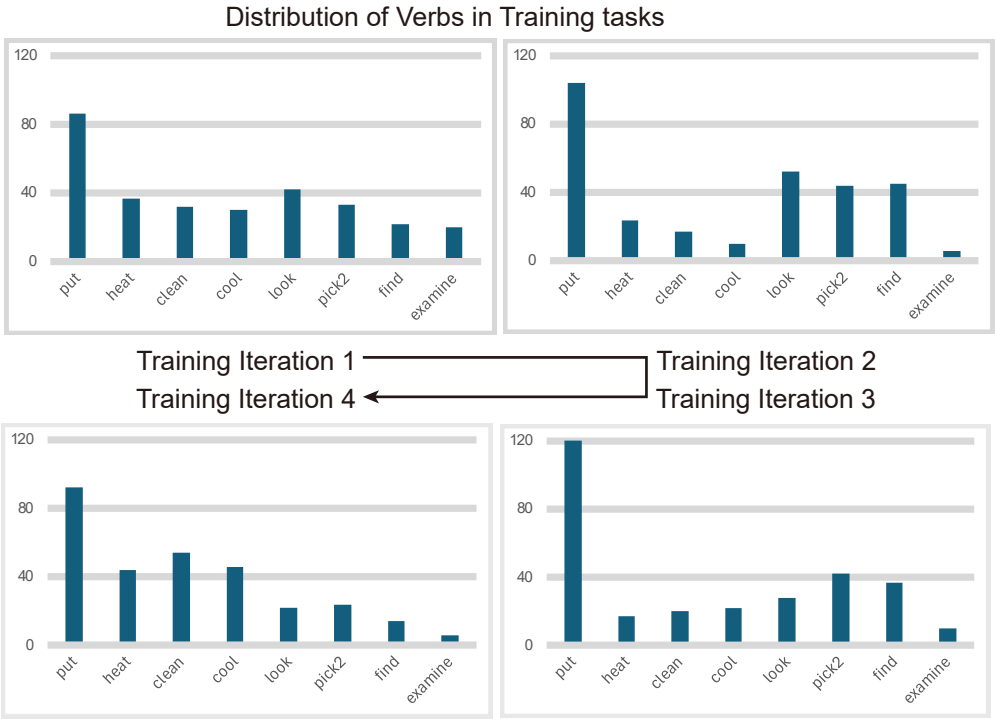

Figure 3: **The distribution of verbs in training tasks.** This figure demonstrates how RevealIt optimizes the task sequences. We employ various verbs in tasks to differentiate, as each type necessitates distinct capabilities from the agent. The agent training process reflects the task sequence change from left to right. Analyzing the changes in task distribution shows that the "put" type of task is the most prevalent. As the training advances, the initial focus of training tasks is on teaching the agent to locate and retrieve objects in the environment ("look", "pick"). Subse,"ntly, the agent is trained on tasks that require it to acquire other skills, e.g., "clean", "heat", and "examine".

This finding demonstrates why our GNN-based explainer is capable of optimizing subtask sequences. The agent's learning process consists of both shared and distinct components. Once the agent becomes proficient at the intersecting components, decreasing the frequency of these specific training tasks is advisable. This is because the agent cannot acquire new knowledge from them, resulting in an improvement in the agent's training efficiency.

6 Conclusion and Future Work

We present RevealIt, a novel framework designed to explain the learning process of RL agents in complex environments and tasks while also serving as a foundation for optimizing task sequences. We demonstrate the mechanisms and functionalities of RevealIt in complex environments. In the ALFWorld, we exclusively implemented the RevealIt within the visual environment. It is worth investigating if RevealIt can effectively transform the knowledge acquired from policies and training problems into natural language, enabling its broader and more direct application in other reinforcement learning domains.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[3] L. Kaelbling, "Learning to achieve goals," in *IJCAI*, 1993.

[4] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2017.

[5] H. Chen and C. Liu, "Safe and sample-efficient reinforcement learning for clustered dynamic environments," *IEEE Control Systems Letters*, vol. 6, pp. 1928–1933, 2023.

[6] U. H. Nielsen, J.-P. Pellet, and A. Elisseeff, "Explanation trees for causal bayesian networks," *ArXiv*, vol. abs/1206.3276, 2008.

[7] S. Deshpande, B. Eysenbach, and J. G. Schneider, "Interactive visualization for debugging rl," *ArXiv*, vol. abs/2008.07331, 2020.

[8] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, 2017.

[9] J. Pearl, "Probabilistic reasoning in intelligent systems - networks of plausible inference," in *Morgan Kaufmann series in representation and reasoning*, 1989.

[10] D. B. Rubin, "Estimating causal effects of treatments in randomized and nonrandomized studies." *Journal of Educational Psychology*, vol. 66, pp. 688–701, 1974.

[11] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward causal representation learning," *Proceedings of the IEEE*, vol. 109, pp. 612–634, 2021.

[12] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, pp. 96–146, 2009.

[13] E. P. Martens, W. R. Pestman, A. de Boer, S. V. Belitser, and O. H. Klungel, "Instrumental variables: Application and limitations," *Epidemiology*, vol. 17, pp. 260–267, 2006.

[14] K. L. Sainani, "Instrumental variables: Uses and limitations," *PM&R*, vol. 10, 2018.

[15] Y. Cui, H. Pu, X. Shi, W. Miao, and E. J. T. Tchetgen, "Semiparametric proximal causal inference," *Journal of the American Statistical Association*, 2020.

[16] S. A. Sontakke, A. Mehrjou, L. Itti, and B. Scholkopf, "Causal curiosity: Rl agents discovering self-supervised experiments for causal representation learning," in *International Conference on Machine Learning*, 2020.

[17] E. Puiutta and E. M. S. P. Veith, "Explainable reinforcement learning: A survey," *ArXiv*, vol. abs/2005.06247, 2020.

[18] A. Heuillet, F. Couthouis, and N. D. Rodríguez, "Explainability in deep reinforcement learning," *Knowl. Based Syst.*, vol. 214, p. 106685, 2020.

[19] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding atari agents," *ArXiv*, vol. abs/1711.00138, 2017.

[20] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, "Towards interpretable reinforcement learning using attention augmented agents," *ArXiv*, vol. abs/1906.02500, 2019.

[21] Z. Deng, J. Jiang, G. Long, and C. Zhang, "Causal reinforcement learning: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2307.01452

[22] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "An empirical investigation of the challenges of real-world reinforcement learning," *ArXiv*, vol. abs/2003.11881, 2020.

[23] C. Paduraru, D. J. Mankowitz, G. Dulac-Arnold, J. Li, N. Levine, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, pp. 2419 – 2468, 2021.

[24] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *ArXiv*, vol. abs/2003.04960, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:212657666

[25] D. Held, X. Geng, C. Florensa, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International Conference on Machine Learning*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:22729745

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *ArXiv*, vol. abs/1801.01290, 2018.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[29] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *ISVC*, 2015, pp. 867–877.

[30] S. Ao, T. Zhou, G. Long, X. Song, and J. Jiang, "Curriculum reinforcement learning via morphology-environment co-evolution," *ArXiv*, vol. abs/2309.12529, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:262217133

[31] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *ArXiv*, vol. abs/2011.04573, 2020.

[32] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht, "ALFWorld: Aligning Text and Embodied Environments for Interactive Learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[33] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. K. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *ArXiv*, vol. abs/1712.05474, 2017.

[34] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, "Pddl - the planning domain definition language," *ArXiv*, 08 1998.

[35] M.-A. Côté, Á. Kádár, X. Yuan, B. A. Kybartas, T. Barnes, E. Fine, J. Moore, M. J. Hausknecht, L. E. Asri, M. Adada, W. Tay, and A. Trischler, "Textworld: A learning environment for text-based games," *ArXiv*, vol. abs/1806.11532, 2018.

[36] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[37] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *ArXiv*, vol. abs/2304.10592, 2023.

[38] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International Conference on Machine Learning*, 2023.

[39] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue, H. Li, and Y. J. Qiao, "Llama-adapter v2: Parameter-efficient visual instruction model," *ArXiv*, vol. abs/2304.15010, 2023.

[40] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. A. Li, P. Fung, and S. C. H. Hoi, "Instructblip: Towards general-purpose vision-language models with instruction tuning," *ArXiv*, vol. abs/2305.06500, 2023.

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *ArXiv*, vol. abs/1801.01290, 2018.

[42] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, pp. 9240–9251, 2019.

[43] J. Zhang, D. Luo, and H. Wei, "Mixupexplainer: Generalizing explanations for graph neural networks with data augmentation," *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[44] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *ArXiv*, vol. abs/2210.03629, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252762395

[45] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *ArXiv*, vol. abs/2308.08155, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260925901

[46] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: language agents with verbal reinforcement learning," in *Neural Information Processing Systems*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258833055

A Visualized task examples in ALFWorld

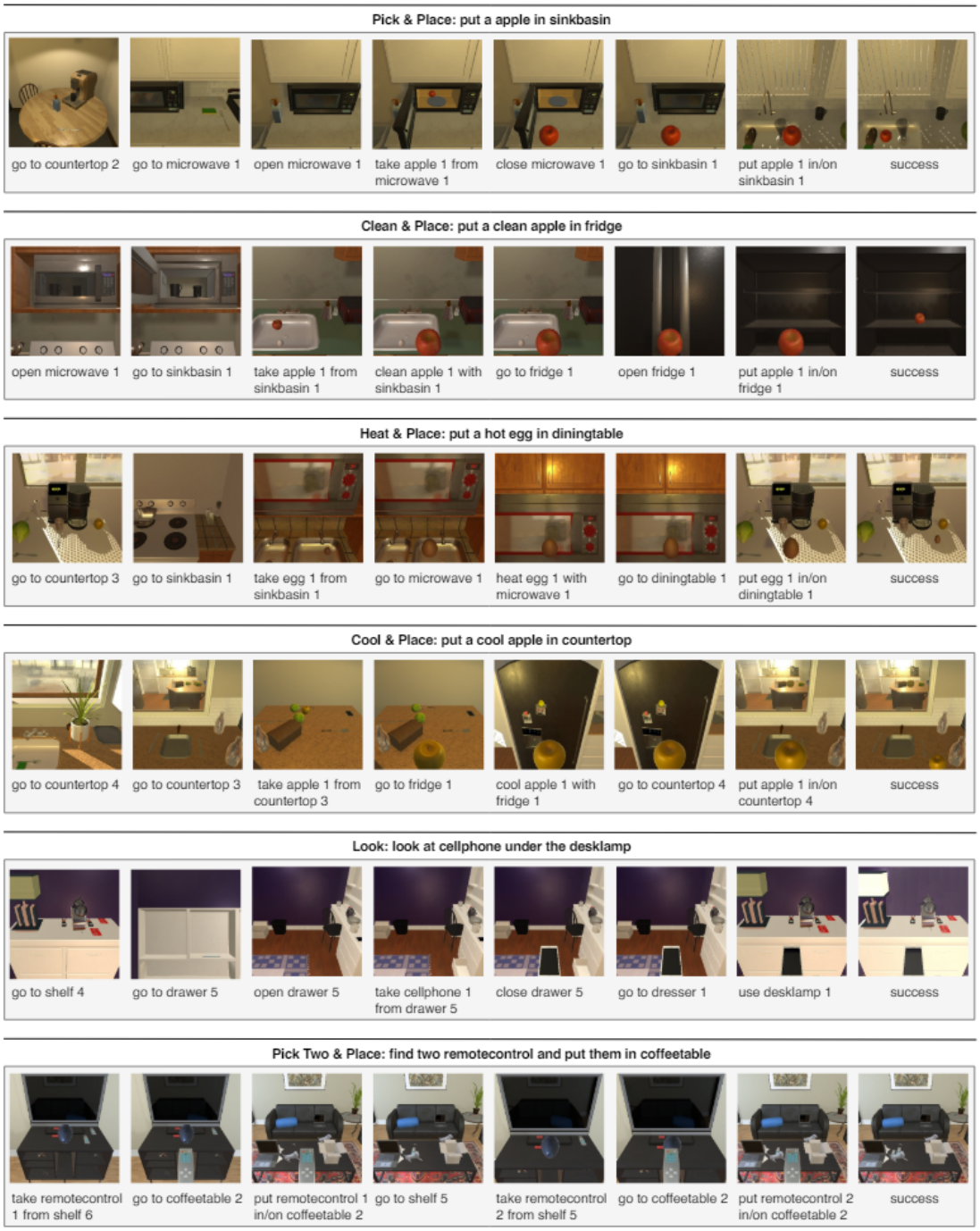

Figure 4: **Visualized task examples of ALFWorld (32).** This benchmark utilizes various household scenarios created within the Ai2Thor environment. All objects can be moved to different locations based on available surfaces and class restrictions in this environment. This allows for the generation of a wide range of new tasks by combining different objects and goal positions in a procedural manner.

B The difference between RevealIt and other explanation methods

In this section, We want to discuss the novelty of RevealIt with other explanation methods in more detail, and we hope this discussion can help clarify the contribution of RevealIt.

An interactive node-link visualization of convolutional neural networks. Note that utilizing the visualization to show the structure of a CNN/GNN network or an MLP network is previously discussed in (29). Nevertheless, the use of visualization is uncommon in RL, this is because:

- Gaining a visual representation of the policy structure does not enhance our comprehension of the possibilities of RL, and the motivation for visualizing CNNs is essentially distinct from this, i.e., we lack knowledge regarding the specific capabilities of the agent that correspond to a particular section of the policy. RevealIt confirmed the accuracy of this mapping relationship for the first time (refer to Fig. 2).
- The process of updating policies necessitates millions of revisions. Without a proficient interpreter, meaningful conclusions cannot be derived from the visualization outcomes. The GNN explanation plays a crucial part in RevealIt.

The difference between Gnnexplainer/PGExplainer: Generating explanations for graph neural networks". The primary objective behind the GNN explainer in RevealIt is to identify the important policy updates that occur during the agent's learning process. This allows us to correlate these updates with the agent's abilities, thereby explaining how the agent learns in complex tasks. Simultaneously, the GNN explainer can assist us in determining if the predictor comprehends the inherent connection between the agent's abilities and the learning process, hence assessing the credibility of the task's significance. This differs radically from the conventional GNN explainer employed for identifying important edges.

How does RevealIt differ from other interpretable RL algorithms? It is important to understand that RevealIt stands apart from conventional interpretable RL algorithms in: (1) RevealIt does not necessitate the use of artificially constructed causal structures. (2) RevealIt does not require additional knowledge of SCM. These two points are the most significant limitations of classical explanation RL in complex and real environments. To overcome this flaw and restriction, we employ a "task-agent policy-GNN explainer" architecture, enabling RL use in complex environments and task scenarios. This architecture also comprehensively explains how RL agents learn in these tasks, explaining why an agent can succeed or fail after training.

C Additional experiments based on the text engine in ALFworld

C.1 Text world in ALFworld

To further evaluate the generalizability and adaptability of RevealIt, we conducted a new set of experiments in the ALFWorld environment, focusing exclusively on interactions with the text engine. Unlike prior experiments that utilized the visual environment, these experiments aim to demonstrate the framework's capability to handle tasks requiring textual representations. We report the results in Table. 7.

Agent	**Success Rate**						
	Avg.	Pick	Clean	Heat	Cool	Look	Pick2
ReAct (44)	0.54	0.71	0.65	0.62	0.44	0.28	0.35
AutoGen (45)	0.77	0.92	0.74	0.78	0.86	0.83	0.41
Reflextion (46)	**0.91**	**0.96**	**1.00**	0.81	0.83	0.94	**0.88**
RevealIt (Ours)	0.86	0.72	0.96	**0.87**	**0.82**	**0.95**	**0.90**

Table 7: Comparison with text agents in ALFworld.

C.2 Compare with other RL techniques.

In this section, we conduct additional comparison to illustrate our framework is able to provide explanations in complex environments where other XRL techniques fail (e.g., saliency maps, reward decomposition, causal models).

The main reason we didn't include the comparison with other XRL techniques is that conterfacutal-based/causal-based methods cannot work in a complex environment (e.g., ALFworld). More specifically, saliency mapping, causal RL and counterfacutal methods require to intervening on the RL environment to generate counterfactual states/intervening data for conterfacual learning. In Atari games, this would be fine to work since the environment is quite simple. But intervening in a complex and continuous environment can be challenging.

More specifically, we want to argue that we are not targeting the same problem as the saliency map. It is intuitive to use state-value to reflect and analyze the agent's behavior in a discrete environment. However, this is not the same thing in a complex and continuous environment, i.e., we have tried saliency mapping (the source code provided in https://arxiv.org/pdf/1912.05743) in the past several days per your request. We create a dataset collected from 134 AlfWorld environments across six different tasks. Refer to Fig. 5, note that we only collect 8 pixel figures for each task in each environment, and the dataset reaches 4.4 GB. We tried to train a generator to generate counterfactual states based on it but failed.

C.3 How can RevealIt help human to understand the agent's learning process?

The most direct way to understand a deep learning framework is to explain how the network works, like Grad-CAM highlighting the important parts of a figure for recognition. However, this is not easy to achieve for RL, especially in complex environments/tasks. This is because we cannot directly map a specific section of the policy to an agent's behavior. On the other hand, it is quite costly to compute the state value for every instance from the agent's view in a visual environment.

To understand how the policy learns to complete a whole task, we designed RevealIt to visualize the policy update information for humans to understand what an ability the agent learns in a sub-task. To make this visualization simpler and easier to read, we deploy a GNN explainer to highlight the important update. Based on this, by comparing the visualized results across different sub-tasks, humans can understand which part of the policy maps to a specific ability to complete a necessary step for the whole task and which part of the policy corresponds to a sharing ability for diverse tasks.

C.4 RevealIt in Mujoco

In this section, we provide the experiments in Mujoco and report the results in Tab. 8. First, we want to emphasize that the targeting problem of RevealIt is to provide explanations for the agent learning process in complex environments and tasks. The policy structure visualization can be easy for humans to understand based on the task description. We include the experiments in the OpenAI GYM environment to show that RevealIt can also work/bring improvement to the basic RL algorithms in general environments.

Agent	Ant-v3	Swimmer-v3	Hopper-v3	HalfCheetah-v3
RevealIt+PPO	2745.57 ± 564.23	340.58 ± 6.20	2167.90 ± 102.81	6047.82 ± 87.21
PPO	1480.47 ± 407.39	281.78 ± 11.86	2410.11 ± 9.86	5836.27 ± 171.68
A2C	-15.93 ± 6.74	199.91 ± 1.32	679.01 ± 302.76	3096.61 ± 82.49

Table 8: Comparison with agents in mujoco.

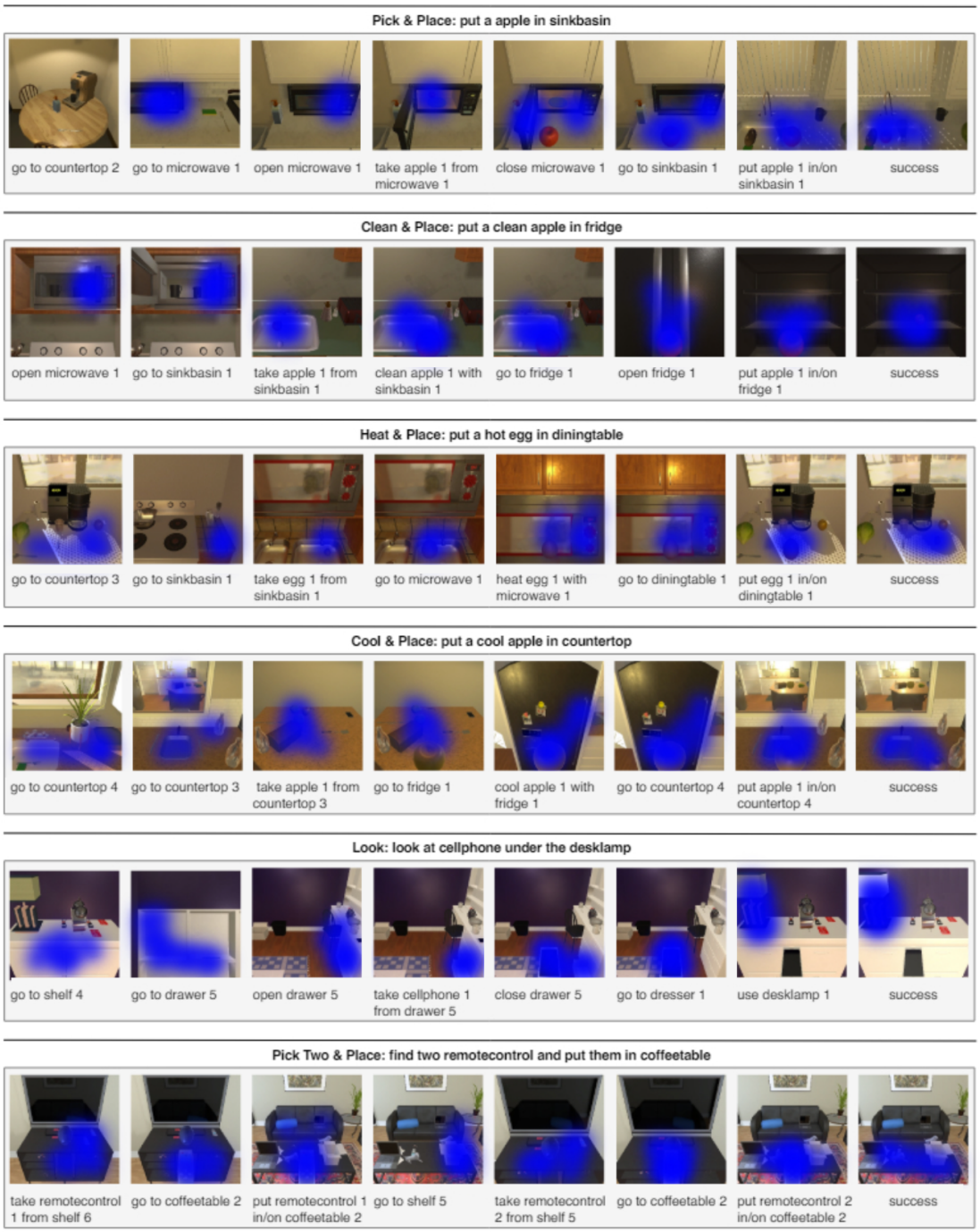

Figure 5: **Visualized important policy updates by GNN explainer.** In this figure, we visualize a value map based on the trained policy by RevealIt in AFLworld to show the ability that the agent learns.

D More Details about GNN explainer

D.1 The loss function for GNN explainer

In RevealIt, We use the loss in the PGEexplainer (31) to train the GNN explainer, and we use an MSE loss for the GNN predicter. On the other hand, in the traditional GNN explainer method, the learning objectives

of the evaluator GNN and the explainer are often consistent, i.e., maximizing the mutual information between the label and the selected subgraph, and we can express this process as:

$$\max_{G_s} = MI(Y, G_S) = H(Y) - H(Y|G = G_S), \tag{4}$$

where G_S denotes the masked subgraph, the GNN explainer optimizes this function locally by training an NN to provide a customized explanation for each instance. However, RevealIt optimizes this by adopting an NN to learn the crucial edges on multiple instances (each evaluation results of RL).

D.2 GNN explainer highlight the important update

To extract the explanation subgraph G_S, we first compute the importance weights on edges. A threshold is used to remove low-weight edges and identify the explanation subgraph G_S. The ground truth explanations of all datasets are connected subgraphs. Therefore, we identify the explanation as the connected component containing the explained node in GS. We identify the explanation for the generated graph classification by the maximum connected component of G_S. For all basic RL algorithms, we perform a search to find the maximum threshold such that the explanation is at least of size K_M. When multiple edges have tied importance weights, all of them are included in the explanation.

D.3 How to ensure the training process of GNN+RL is stable?

The fundamental principle of the GNN+RL training scheme is mutual learning, where the GNN predicts the value of the training task for the RL agent based on the RL training process, and the GNN predictor influences the training task of the RL agent. During the initial stages of training, the GNN predictor exhibits instability due to inadequate data. We employ a random sampling technique to select training tasks from the complete set. Once RL has accumulated a specific quantity of data, we systematically enhance the significance of the GNN predictor in choosing training tasks. Providing a precise theoretical guarantee for this procedure is challenging, yet it is a commonly used approach in curriculum RL to choose training tasks, e.g., (30; 25).

D.4 How does GNN explainer map the skills of the RL agent?

In RevealIt, the central concern revolves around the GNN explainer's capacity to comprehend the agent's learning prcess and ascertain whether the agent has achieved the certain abilities in accomplishing the task. The GNN explainer is designed to identify and emphasize important updates in the policy. Suppose the nodes associated with these highlighted updates correspond to the nodes engaged by the agent during the evaluation. In that case, it can be inferred that the explanation is cognizant of the specific sections of the policy that align with the agent's essential abilities. Hence, we ascertain the ratio of nodes identified by the GNN explainer and nodes stimulated by the agent in successful tasks and report it in Fig. 6. In this series of comparisons, we incorporated a pre-trained Graph Neural Network (GNN) explanation that utilizes data from RL training data from different sequences of training tasks in the same environment. An intriguing observation is that we discovered that the efficacy of the GNN-explainer, when trained on various tasks, was inferior than that of the GNN-explainer taught using a reinforcement learning agent. This discrepancy may arise from the varying demands that different tasks place on the training process of reinforcement learning (RL) agents. Additionally, pre-trained graph neural networks (GNNs) may struggle to comprehend the ongoing RL training progress accurately due to the incorporation of a priori bias.

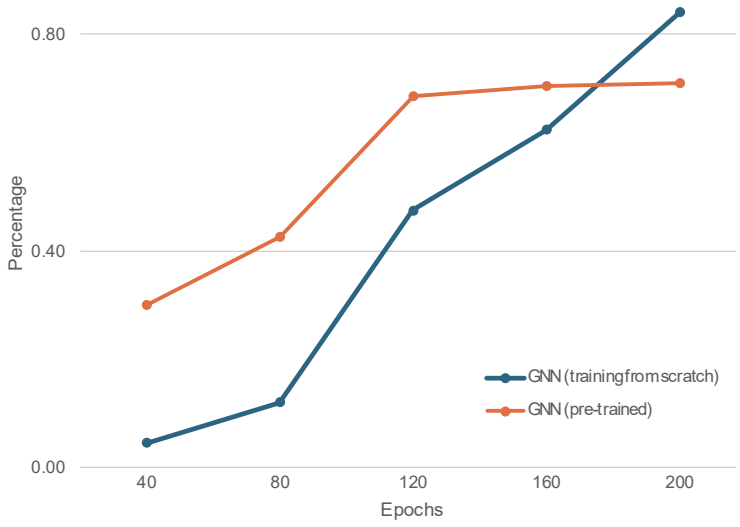

Figure 6: **The results of learning process of the GNN explainer.**