2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-6654-3161-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCNC49033.2022.9700600

RIRL: A Recurrent Imitation and Reinforcement Learning Method for Long-Horizon Robotic Tasks

†‡Zhitao Yu, §Jian Zhang, †Shiwen Mao, ‡Senthilkumar CG Periaswamy, and ‡Justin Patton
†Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201, USA
‡RFID Lab, Auburn University, Auburn, AL 36849, USA
§Department of Electrical and Computer Engineering, Kennesaw State University, Marietta, GA 30060, USA
Email: zzy0021@auburn.edu, {jianzhang, smao}@ieee.org, {szc0089, jbp0033}@auburn.edu

***Abstract*—The developments in reinforcement learning provide a powerful and efficient learning framework for autonomous robotic systems. However, prior works rarely embed historical observations due to the exponentially increasing complexity, which may not perform well for large-scale long-horizon tasks that might require hundreds and thousands of steps to complete. In this paper, we propose Recurrent Imitation and Reinforcement Learning (RIRL) to address the challenges and enable robots for such tasks. The proposed RIRL incorporates a long short-term memory (LSTM) network to retain long-term memories, which could be an effective and efficient method to tackle the long dependency problem raised in long-horizon robotic tasks. To assess the performance of the RIRL, we test it with an optimized path planning problem for a robot to perform a Radio-frequency identification (RFID) inventory in dynamic and previously unknown environments. We experimentally validate RIRL's feasibility and effectiveness in a visual game-based simulation platform, where the proposed RIRL model outperforms three baseline schemes with considerable gains.**

***Index Terms*—Long-horizon task, long short-term memory (LSTM), robotics, deep reinforcement learning (DRL).**

I. Introduction

With the considerable developments of deep learning methods in the past few years, reinforcement learning (RL), a method that has been proposed for over 20 years [1], is equipped with deep learning models and re-attracted the attention of academia and industry. It has been widely deployed to lead intelligent agents to interact with an environment to maximize the obtained cumulative rewards. Meanwhile, more and more robots are being deployed in various environments to accomplish tasks, such as inventory counting in retails and warehouses [2], [3]. In the past few years, researchers have shown a growing interest in applying deep reinforcement learning (DRL) methods to enable robotic systems to task in complex environments. For example, Kober et al. presented an automated meta-parameter acquirement for adjusting robots' movement by reinforcement learning [4]. The authors in [5] proposed a DRL-based motion planner that generates linear and angular velocities directly for navigation without an existing map. A model-based reinforcement learning framework for legged locomotion was introduced in [6], which allows the learned model to generalize to new tasks without any fine-tuning or using any extra collected data.

This work is supported in part by the NSF under Grant ECCS-1923163.

Currently, the RL methods applied to robotics could be roughly divided into two categories: traditional policy-based reinforcement learning and imitation learning methods. Some RL based algorithms only utilize the received rewards to gain an approximated optimal policy, which makes them heavily rely on the effectiveness of the reward function [7]. In [8], the author proposed two new dense reward functions to learn robust strategy in path planning tasks. The reward sketching method was proposed in [9] that extracts human preferences to learn a reward function for a new task. However, manually designed reward functions that gratify the desirable agent actions are extremely complicated and usually not feasible, especially in the scenarios of dynamic and real environments with only sparse rewards. Moreover, even an expert cannot accurately quantify the reward of every behavior for various agents. A practical solution for the problems above is imitation learning (IL). Instead of manually designing a reward function, a set of well-prepared demonstrations provided by experts are followed and imitated by agents to learn the optimal policy of given tasks. An imitation learning method was introduced by Abbeel et al. in [10] to predict the agent's actions from a set of demonstrations and sequential states for another demonstration with different initial conditions. However, the agent must consume over 100,000 demonstrations to learn a simple strategy for reacting to different situations and initializations for the same task. Ho and Ermon proposed Generative Adversarial Imitation Learning (GAIL) [11], which combined Generative Adversarial Network (GAN) [12] with imitation learning. First, it trains the discriminator with the current policy's sampled data and expert data. Then, the discriminator plays the role of reward function to lead the agent to learn the optimal policy.

Long-horizon tasks planning is a challenging and open problem in robotics. Its complexity grows exponentially with increased numbers of acting steps and sub-tasks. In many of these applications, the robotic agents need to execute a very large number of steps to reach the goal in unseen environments, considering an autonomous robot patrolling, searching, and retrieving objects in a giant unexplored building. In [13], the authors simplified the long-horizon policy learning problem to a hierarchical and goal-conditioned policy, where the low-level policy only requires a fixed, low number of steps to accomplish. Their simulation scenario was based

on a simple kitchen environment, where tasks can be executed with a short sequence of discrete actions. Pitis et al. designed a strategy that the agent pursues the maximization of the entropy of the historical achieved goal distribution instead of inaccessible goals when facing sparse extrinsic learning signals [14]. Similarly, in order to solve the sparse extrinsic reward problem of long-horizon tasks, the authors in [15] incorporated demonstrations into the RL method that is built on top of Deep Deterministic Policy Gradients (DDPG). They focused on teaching agents to stack blocks with a robot arm by continuous multi-steps control and generalize to varying goal states. However, the multi-step behavior needs far fewer steps for achieving the goal than in our target scenario, which requires thousands of operational steps for a robot to accomplish the task.

In addition, the training samples impose one more challenge for deploying RL methods in robotics. An RL based method requires a great number of training samples. It may take millions of steps of experience to learn a strategy for a simple task. It is a great challenge and even infeasible to collect such a large amount of operational data in many robotic applications. Although with IL methods, robots are able to learn a robust strategy with a small number of expert demonstrations in short-term tasks, they still suffer from compounding errors and the massive growth of expert demonstration demands when facing longer and complicated tasks. Some Pioneering works have exploited methods by combining the RL with demonstrations to overcome this challenge. For instance, demonstrations are used to initialize policy and accelerate training for reinforcement learning [16]. In addition to these challenges mentioned above, we find an additional difficulty of key interest: long-term dependency problem, which some robotic tasks not only depend on the current observation but also rely on previous observations.

In this paper, we present a novel method called Recurrent Imitation and Reinforcement Learning (RIRL) to remedy the above limitations, which enables agents to leverage historical observations as well as environment feedback for more accurate action prediction. Our method exploits the robustness of Long Short-Term Memory (LSTM) in representing sequential information and solving the long-range dependency problem in long-horizon tasks. The main contributions of our work are summarized as follows:

1) To the best of our knowledge, this is the first work to develop an LSTM-embedded imitation network to better reasoning the underlying relationship among a sequence of historical data from the demonstration. Combining with a reinforcement learning network, it enhances the action prediction ability of the agent by exploiting historical observations.
2) The proposed model allows the agent to achieve long-horizon goals with the capability to deal with the exponentially boosted complexity and uncertainty (e.g., to explore a large scale of the unknown environment in a continuous action space).
3) We experimentally validate the feasibility of RIRL in a visual game-based simulated environment and demonstrate that the proposed model enables the robot to perform inventory tasks in a dynamic environment.

The rest of this paper is organized as follows. In Section II, we present the proposed approach. Our experimental study is discussed in Section III. Section IV concludes this paper.

II. Proposed Approach

A. Problem Statement and Challenges

We are interested in large-scale, long-horizon robotic tasks that require an agent to take thousands of steps to achieve the goal in unseen and dynamic environments. The problem can be formulated as an Partially Observable Markov Decision Process (POMDP) denoted by $(\mathcal{S}, \mathcal{A}, R, T, \mathcal{O}, \Omega)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively; $T(s_{t+1}|s_t, a_t)$ is the state-transition probability for the agent to state s_{t+1} given it was at state s_t and takes action a_t; $R(s_t, a_t)$ is the reward function to provide an instant reward $r_t = R(s_t, a_t)$; Ω is a finite set of observations that the agent can experience of its world; $\mathcal{O}(o_t|s_t)$ is the probability for the agent to receive an observation o_t while it is at state s_t.

In a practical robotic application, the state s_t of the environment is hidden from the agent. Instead, it can only partially perceive the environment to attain an observation o_t by built-in sensors (e.g., cameras, sonar, LIDAR, etc.). Furthermore, to perform large-scale long-horizon tasks, the action a_t at step t not only depends on the current observation o_t, but also relies on the fixed length historical observations $o_0, o_1, ..., o_{t-1}$. We illustrate this long-range dependency problem in Fig. 1, where a robot is deployed for a patrolling task. It needs to patrol the area and cycle around the two objects (A and B) to check the target features, e.g., to inspect the corrosion of power poles or towers. As shown in Fig. 1(a), the robot at state s_t gains observation o_t to represent the surrounding environment (such as its distance to the objects). However, o_t lacks the information for a fully understanding of the environment, such as which area it has already scanned. Based on the information from o_t, it is difficult to correctly choose the subsequent actions that could lead to a better navigational path. Fig. 1(b) and Fig. 1(c) illustrate that if we know the historical information, the next actions could be easily made to enable the robot with a better strategy to complete the task. Therefore, in the long-horizon task scenario, historical observations provide valuable information for predicting the action a_t. However, most of the existing RL based methods predict action $a \sim \pi(a_t|o_t)$ only depending on the current observation, making them perform badly in the above described scenario. Thus, we formulate the objective policy as $\pi(a_t|\mathbf{O_t})$ to tackle the historical and instant observations, where $\mathbf{O_t} = (o_0, o_1, ..., o_t)$ represents a finite set including the historical and instant observations.

Our goal is to find such a policy π that maximizes the expected future discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $0 \leq \gamma < 1$ is a discount factor. We adopt a value-action function $Q_\pi(a_t, \mathbf{O_t}; \theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$ to represent the above reward. Therefore, the goal of RIRL is to learn a policy

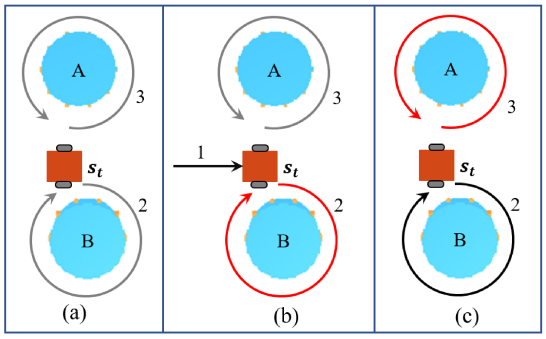

Fig. 1. A brief scenario illustrates the long-range dependency problem of long-horizon robotic tasks: (a) the robot at state s_t, while two potential paths 2 and 3 are available; (b) if it has moved from path 3 to state s_t, the next actions lead to path 2 is a better choice; (c) if it has moved from path 2 to state s_t, the next actions lead to path 3 is a better choice. Note that the completion state s_t is hidden from the root. Instead, it can only partially perceive s_t with limited historical data by built-in sensors, which, thus, makes it incapable of reasoning the covered paths.

π and Q_π that empowers the agent to achieve stable maximum overall rewards while performs various long-horizon tasks in dynamic environments. The objective function is given by

$$\underset{\mathbf{a}\sim\pi}{\operatorname{argmax}}\, Q_\pi(a_t, \mathbf{O_t}; \theta), \tag{1}$$

where θ serves as the parameter set of the value function Q_π; $a_t \sim \mathcal{A} \in \mathbb{R}^N$ denotes an action in a continuous space for the agent at time t; N is the dimension of the control space (usually, $N = 2$ for a ground robotic agent). The complexity of gaining such an optimized policy π grows exponentially with the increased amount of training data from historical observations. It is challenging and even infeasible for a traditional DRL method to converge quickly in the training processes for such tasks.

B. RIRL Network Architecture

The proposed RIRL is a model-free method that can learn a strategy from demonstrations to tackle the long-horizon task and subsequently fine-tune this strategy through the interaction with the environment for handling dynamics. As shown in Fig. 2, it deploys an architecture based on an Imitation Learning (IL) augmented Deep Reinforcement Learning (DRL) network that was introduced in [17]. The IL module adopts a Generative Adversarial Imitation Learning (GAIL) [11] network enhanced with an LSTM layer to enable it with the recurrent capability to retrain historical data and effectively learn the strategy from the demonstrations.

The recurrent enabled IL module, discussed in Section II-A, is the key to overcoming the complexity challenge by providing a seed policy through the demonstration data to reduce the searching space significantly for the DRL to learn an optimized policy π. The DRL module is based on the Proximal Policy Optimization (PPO) [18] and is also enhanced by an LSTM layer that enables it to tackle both historical and instant data.

The IL and the DRL modules are marked in pink and green in Fig. 2, respectively. The IL module helps to learn a seed

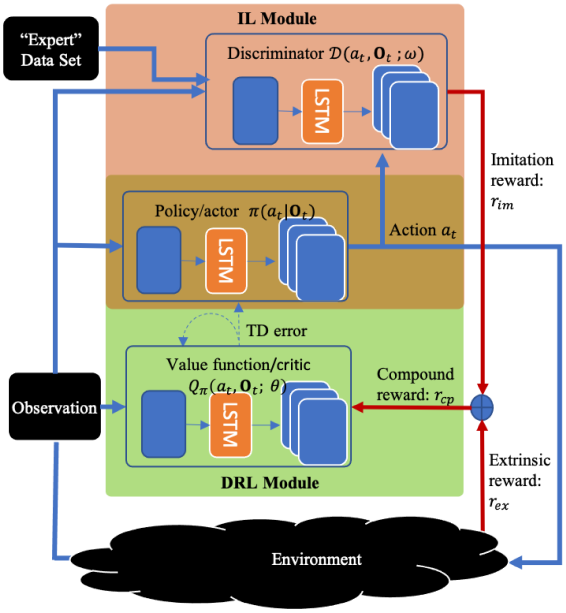

Fig. 2. The architecture of the proposed RIRL method.

strategy from demonstration data and subsequently augments the training of the DRL module to build an optimized policy π to solve the long-dependency problem in long-horizon tasks.

1) The LSTM Network: As mentioned before, the key to tackling the long-dependency problem is to incorporating memory at the agent to exploit historical data for predicting actions. To this end, we adopt an LSTM network [19] layer to both the IL and DRL module in our proposed RIRL. LSTM is a popular Recurrent Neural Network (RNN) for effectively solving the long-term dependency problem. The architecture of its basic cell unit is shown in Fig. 3. There are three control gates: forget gate, input gate, and output gate. The forget gate determines which information from the last cell state could still continue to pass through the current cell. The input gate controls the new data to flow into the memory and update the cell state. The output gate selects which part of the cell state can be exported as output. This structure helps to avoid the gradient exploding or vanishing problems of traditional RNN models. It exploits the temporal information of the observations by using the recursive hidden LSTM units. Important information over a long time horizon is stored by non-linear gate units of the built-in memory cell in each LSTM hidden unit. The last cell in the LSTM network outputs a vector h_t, which contains extracted features that not only include the information from the input data but also contain the important information from long-term historical observations. The memory built-in features h_t will be treated as the input of the subsequent network units to allow the proposed RIRL to obtain an optimized policy for tackling the long-dependency problem in long-horizon robotic tasks.

2) The Recurrent IL Module: The backbone of the proposed IL module is based on the GAIL framework [11] that adopts a network architecture developed from Generative Adversarial Network (GAN) [12]. It is composed of two basic parts,

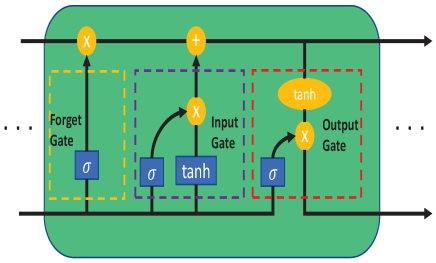

Fig. 3. The LSTM structure.

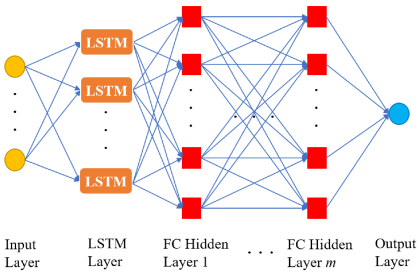

Fig. 4. Architecture of Discriminator $\mathcal{D}$.

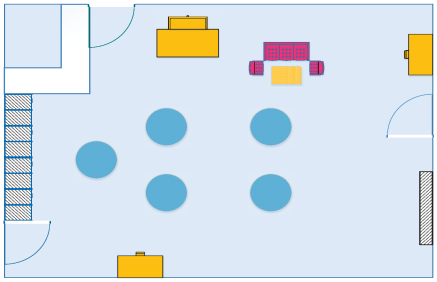

Fig. 5. Layout of the simulated apparel store.

a Discriminator $\mathcal{D}$ and a Generator $\mathcal{G}$, which work in an adversarial and cooperative manner of learning a strategy from a set of demonstration data. Discriminator $\mathcal{D}$ is responsible for distinguishing the data produced by $\mathcal{G}$ from demonstration data, while both $\mathcal{D}$ and $\mathcal{G}$ are simultaneously trained in an adversarial and competitive way. During the training process, Discriminator $\mathcal{D}$ will be wiser to tell the generated data while Generator $\mathcal{G}$ gains more expertise in counterfeiting data. Eventually, the IL module converges when the generated "fake" data from $\mathcal{G}$ could pretend as demonstration data and pass the detection of $\mathcal{D}$. In our proposed network architecture, Generator $\mathcal{G}$ is shared by the IL and the DRL module. It also serves as the policy π of the DRL module; the terms Generator $\mathcal{G}$ and policy π are interchangeably in this paper. To enable the IL module with recurrent capability, we deploy an LSTM layer to enhance both Discriminator $\mathcal{D}$ and Generator $\mathcal{G}$.

3) The Recurrent Discriminator D: The recurrent enabled Discriminator $D(a_t, \mathbf{O_t}; \omega)$ of the IL module evaluates the data based on instant and historical observations to augment the process of predicting the next action. The discriminator $\mathcal{D} : \mathbf{O} \times \mathbf{A} \rightarrow (0, 1)$ is a function with weight ω, where $\mathbf{O}$ and $\mathbf{A}$ are the observation and action space, respectively. The Discriminator $\mathcal{D}$ model is shown in Fig. 4, which is a fully connected LSTM layer followed by m hidden layers. The LSTM and each hidden layer have the same number of units. The size of the input layer is decided by the number of inputs. The LSTM layer maps the inputs to a feature vector h_t, which also carries the information from longer memories beyond the input vectors. Then the m fully connected hidden layer will convert the memory built-in features h_t to a score to measure the similarity of input data and "expert" data. By deploying an LSTM layer, any actions a_t will be evaluated on a large time scale to consider its performance for the long-horizon task. To train the Discriminator $\mathcal{D}$, we update and maximize the following value function, which is derived from the objective function of the GAIL network:

$$\mathcal{V}(\omega) = \mathbb{E}_{\pi}[\log(1 - \mathcal{D}(a_t, \mathbf{O_t}; \omega))] + \mathbb{E}_{\pi_e}[\log(\mathcal{D}(a_t, \mathbf{O_t}; \omega))] - \lambda H(\pi), \quad (2)$$

where π_e refers to the "expert" policy provided by a demonstration dataset, which provides seeds for subsequently training of the optimized policy π. Although referred as "expert," it is not a perfect policy the agent should take under any circumstance. In (2), π_e is collected by manipulating in several sample scenarios by manual settings with primitive strategies, which can only allow the agent to complete the long-horizon task in a non-optimized manner. $H(\pi)$ denotes the causal entropy of π and is defined as $H(\pi) \doteq \mathbb{E}_{\pi}[-\log \pi(a_t|\mathbf{O_t})]$. It serves as a policy regulator. It encourages the exploration behaviors and lets the learned strategy to be as random as possible while optimizing the objective, instead of quickly greedily converging to a local optimum. $\lambda \geqslant 0$ is the discount weight of H. The Discriminator $\mathcal{D}$ is updated to improve its ability to tell the similarity of a policy with expert data by increasing $\mathcal{V}(\omega)$. A well-trained $\mathcal{D}$ provides a higher score if the given data is more similar to the demonstrations. Therefore, by coordinating with Generator $\mathcal{G}$, the actions similar to the one provided in the demonstrations will get a higher selection rate. This way, the search space for the subsequent training can be narrowed to close to the seed behaviors and quickly converge to an optimized policy π.

4) The Recurrent DRL Module: In the proposed RIRL, the IL module could effectively learn a seed policy by imitating the behaviors from the demonstration dataset. This seed policy requires fine-tuning to allow the agent to engage with a dynamic environment. Therefore, we deployed a DRL module to learn an optimized policy π by interacting with the dynamic and complex environment.

The DRL module is based on a PPO network and has an actor-critic architecture to exploit the environment with a continuous action space. It consists of an actor representing policy π and a critic shown as the value function Q_{π}. The policy π is responsible for generating the action, a_t, based on given observations $\mathbf{o}$. The value function Q_{π} processes the received rewards and evaluates the current action prescribed by policy π. As shown in Fig. 2, we implement these two components with two neural networks embedded with an LSTM layer for referencing historical observations recurrently. The architectures of these two neural networks are similar to that shown in Fig. 4, except for the number of layers. Let N_{π} and N_Q be the numbers of hidden layers for the actor and critic network, respectively. Each hidden layer has the same number of units. The size of the input layer is decided by the input vector dimension. The size of the generated action a_t determines the final output layer size of the actor.

The goal of training the DRL network is to maximize the

value function Q_π for a given policy π, i.e.,

$$Q_\pi(a_t, \mathbf{O_t}; \theta) = \mathbb{E}[r_{cp}(o_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q_\pi(a_{t+1}, \mathbf{O_{t+1}})]], \quad (3)$$

where θ represents the parameter of the value function Q_π, γ is the discount factor for future reward, r_{cp} is a compound reward that consists of reward from the IL module and the extrinsic reward while interacting with the environment, i.e.,

$$r_{cp}(o_t, a_t) = \mu \cdot r_{ex}(o_t, a_t) + (1 - \mu) \cdot r_{im}(o_t, a_t), \quad (4)$$

where $\mu \in (0, 1)$ is a proportion parameter: a smaller μ will be implemented if the demonstration data is closer to the optimal policy. It helps the learning process to trade-off between demonstration data and engagement of the environment. In (4), $r_{im} = \log(\mathcal{D}(a_t, \mathbf{O_t}; \omega))$ is the reward that comes from the IL module, measuring how similar the action a_t is with the "expert policy" from demonstrations. The extrinsic rewards r_{ex} is provided as a sparse function to represent the basic constraints and rules for the agent to interact with the environment and guide it to the desired goals. For example, we can give a penalty if the agent collides with an object in the environment and a positive reward if it reaches a goal position.

During the training process, policy π will be updated to choose the actions a_t to increase Q_π by gaining a larger r_{ex} and r_{im}. From (2), (3), and (4), increasing r_{im} by updating policy π, which also acts as the generator of the IL module, will decrease the value of $\mathcal{V}(\omega)$. Together with the training of the discriminator in the IL module, which tends to maximize $\mathcal{V}(\omega)$, the competitive processes end up with a policy π that roots with the strategies provided in the demonstration data. Furthermore, increasing extrinsic rewards r_{ex} will eventually lead the trained policy π to react to the environment with a better strategy. The built-in LSTM layers in the components of policy π, value function Q_π, and Discriminator $\mathcal{D}$ enhance the agent with memories for action predicting and evaluation. Therefore, the proposed RIRL provides an effective means to train an optimized policy π for long-horizon tasks, especially for the scenarios that require long-term memories.

III. Experimental Study

A. Experiment Setup

We deploy the proposed RIRL system in a simulated environment using the Unity3D platform to simulate an application of deploying a mobile robot for RFID-based inventory in an apparel store, which is the same scenario as in [2]. Unity3D is a powerful game engine to render larger, detailed, 3D environments. It also simulates visually realistic worlds with sophisticated physics and complex interactions between agents with varying capacities. These characteristics enable it to be widely deployed as a simulation tool for the research of a variety of intelligent agents [20]. As shown in Fig. 5, we create a four-wall enclosed environment to simulate an apparel store of dimension $50 \times 50\text{m}^2$, which is crowded with racks (represented by blue circles) and obstacles (e.g., furniture). RFID tagged items are hosted on the racks. A simulated robotic agent is deployed to patrol the area to scan all the RFID tags. The positions of racks and the agent are randomly generated inside the room at the beginning of each episode. The agent simulates an RFID-enabled mobile robot controlled by action $a_t = (v_t, \phi_t)$, where $v_t \in [-v_{max}, v_{max}]$ represents the linear speed, while v_{max} is the maximum linear velocity, and $\phi_t \in [-\phi_{max}, \phi_{max}]$ denotes the rotation speed, while ϕ_{max} is the maximum rotation velocity.

The task of the agent is to scan all the RFID tags in the room with the shortest trajectory. Each training episode ends with all RFID tags scanned or maximum steps are reached. The number of agent collisions is also considered as one of the criteria for measuring the quality of training results. The robotic agent is equipped with two sensors, a ray-cast sensor and a simulated RFID reader to collect observations. Unity3D provides the ray-cast sensor to simulate a widely deployed Lidar sensor. It detects the surrounding environment by casting rays and outputs a vector with the detected objects and the corresponding distances. We design a simulated RFID reader for the agent to mimic the characteristics of practical RFID applications. For example, it can only scan the tags within a detectable range, and the probability of reading a tag decreases as the distance between the reader and a tag increases.

The RIRL network is implemented with Tensorflow on a computer with an Intel 9900K CPU and two Nvidia 2080 GPUs. We conducted all our experiments with the same network: Discriminator $\mathcal{D}$ is implemented with one LSTM layer and two fully connected hidden layers, where each layer has 128 units. For the DRL network module, both the value function Q_π and policy π consist of an LSTM layer with 128 units and three hidden layers each with 512 units. We set the proportion parameter μ in (4) as 0.1 for the remaining experiments. In the experiments, the robotic agent will be deployed in the simulated apparel store, while its initial position is randomly generated at each episode.

B. Results and Analysis

1) Training Results: We compare the proposed RIRL with three existing methods: the PPO network introduced in [18], the GAIL proposed by [11], and the IADRL scheme [17] that is an RL and IL combined method without the LSTM layers. We use the same basic reward settings and training parameters in the same environment set for these four approaches to guarantee a fair comparison. Fig. 6 plots the cumulative rewards acquired by the agent when it interacts with the environment. We only set several basic and sparse reward configurations: scanning a new RFID tag gains +10 rewards, collision results in a -1 punishment, and moving costs -0.001 for each step. Our approach, shown as the red solid line in Fig. 6, achieves the best reward in the training process, which is higher and more stable compared to the other three methods.

Fig. 7 presents the number of steps for finishing the tag scanning task in each episode. We set a maximum number of 20,000 steps for each training episode that aims to decrease the unnecessarily long training time. The red solid line depicts that our method result stabilizes after around 200 episodes. The agent implemented with RIRL could stably and consistently

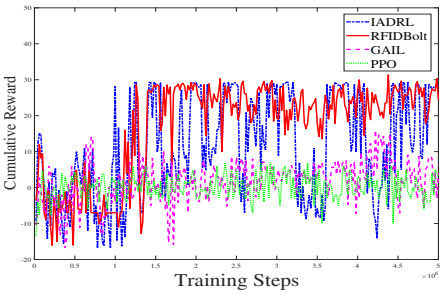

Fig. 6. Accumulated training rewards values.

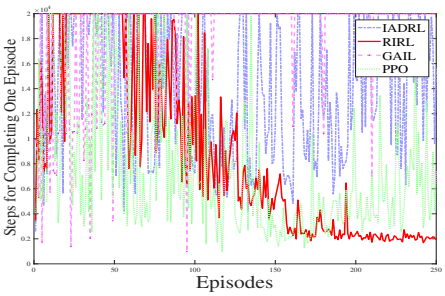

Fig. 7. Steps for finishing the tag scanning task.

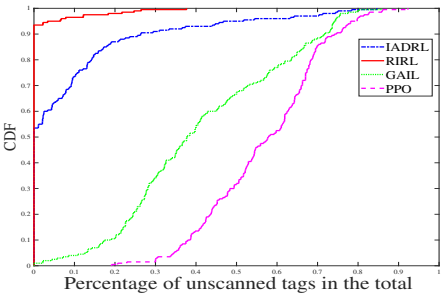

Fig. 8. CDF of percentage of total unscanned tags.

handle the given tasks within 2,000 steps. Apparently, as shown in Fig. 7, the other three methods cannot even converge with the same training process, indicating that the agent is unable to find a reliable strategy to accomplish the given task.

2) Testing Results: Fig. 8 presents the cumulative distribution function (CDF) of the percentage of unscanned tags in the testing stage. We test all the four trained models in 200 episodes within the required 20,000 steps. Fig. 8 shows that in about 96% of the episodes, the proposed RIRL model scans all the tags, while the IADRL scans all the tags in about 55% of the episodes. Obviously, the GAIL and PPO model cannot even scan all the tags in an episode. Moreover, RIRL attains a maximum unscanned tag percentage of 28.2%, which is much lower than the almost 90% missing rate achieved by the other three methods. Apparently, the proposed RIRL exhibits considerably higher effectiveness and robustness for such long-horizon tasks in dynamic environments.

The experiments validate that our proposed RIRL outperforms the three benchmark approaches. It help the agent to accomplish the long-horizon tag scanning task with high efficiency and robustness. Moreover, it proves the feasibility of utilizing the LSTM network to enhance the agent performance by leveraging historical observations.

IV. Conclusion

In this paper, we proposed RIRL, a deep recurrent imitation and reinforcement learning-based system, which enables agents to accomplish long-horizon tasks in dynamic and complicated environments. We also experimentally validated the feasibility of embedding an LSTM layer in the traditional IL and DRL methods. The outstanding result achieved by our method proved the effectiveness of leveraging history observations for enhancing RIRL to solve the long-range dependency problem in various long-horizon robotic tasks.

References

[1] C. J. Watkins and P. Dayan, "Q-learning," *Springer Machine Learning J.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[2] J. Zhang, Y. Lyu, T. Roppel, J. Patton, and C. Senthilkumar, "Mobile robot for retail inventory using RFID," in *Proc. IEEE ICIT'16*, Taipei, Taiwan, Mar. 2016, pp. 101–106.

[3] J. Zhang, Y. Lyu, J. Patton, S. C. Periaswamy, and T. Roppel, "BFVP: A probabilistic UHF RFID tag localization algorithm using Bayesian filter and a variable power RFID model," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8250–8259, Oct. 2018.

[4] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Robotics: Science and Systems*, Y. Matsuoka, H. Durrant-Whyte, and J. Neira, Eds. Cambridge, MA: The MIT Press, 2011, pp. 33–40.

[5] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE IROS'17*, Vancouver, Canada, Sept. 2017, pp. 31–36.

[6] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Proc. 3rd Conf. Robot Learning*, Osaka, Japan, Nov. 2020, pp. 1–10.

[7] X. Xia, T. Roppel, J. Y. Hung, J. Zhang, S. C. Periaswamy, and J. Patton, "Balanced map coverage using reinforcement learning in repeated obstacle environments," in *Proc. IEEE ISIE'20*, Delft, The Netherlands, June 2020, pp. 41–48.

[8] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105 669–105 679, July 2019.

[9] S. Cabi *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv preprint arXiv:1909.12200*, Sept. 2019. [Online]. Available: https://arxiv.org/abs/1909.12200

[10] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *arXiv preprint arXiv:1703.07326*, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.07326

[11] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. ACM NIPS'16*, Red Hook, NY, July 2016, pp. 4572–4580.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. ACM NIPS'14*, Montreal, Canada, Dec. 2014, pp. 2672–2680.

[13] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. CORL'19*, Osaka, Japan, Oct. 2020, pp. 1025–1037.

[14] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *Proc. PMLR ICML'20*, Virtual Conference, July 2020, pp. 7750–7761.

[15] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE ICRL'18*, Vancouver, Canada, Apr.-May 2018, pp. 6292–6299.

[16] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Elsevier Neural Networks*, vol. 21, no. 4, pp. 682–697, May 2008.

[17] J. Zhang, Z. Yu, S. Mao, S. Periaswamy, J. Patton, and X. Xia, "IADRL: Imitation augmented deep reinforcement learning enabled UGV-UAV coalition for tasking in complex environments," *IEEE Access*, vol. 8, no. 1, pp. 102 335–102 347, June 2020.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, July 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[20] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar *et al.*, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1809.02627