Hierarchical Adversarial Inverse Reinforcement Learning

Jiayu Chen, Graduate Student Member, IEEE, Tian Lan, Senior Member, IEEE, and Vaneet Aggarwal, Senior Member, IEEE

Abstract—Imitation learning (IL) has been proposed to recover the expert policy from demonstrations. However, it would be difficult to learn a single monolithic policy for highly complex long-horizon tasks of which the expert policy usually contains subtask hierarchies. Therefore, hierarchical IL (HIL) has been developed to learn a hierarchical policy from expert demonstrations through explicitly modeling the activity structure in a task with the option framework. Existing HIL methods either overlook the causal relationship between the subtask structure and the policy, or fail to learn the high-level and low-level policy in the hierarchical framework in conjuncture, which leads to suboptimality. In this work, we propose a novel HIL algorithm—hierarchical adversarial inverse reinforcement learning (H-AIRL), which extends a state-of-the-art (SOTA) IL algorithm—AIRL, with the one-steps option framework. Specifically, we redefine the AIRL objectives on the extended state and action spaces, and further introduce a directed information term to the objective function to enhance the causality between the low-level policy and its corresponding subtask. Moreover, we propose an expectation-maximization (EM) adaption of our algorithm so that it can be applied to expert demonstrations without the subtask annotations which are more accessible in practice. Theoretical justifications of our algorithm design and evaluations on challenging robotic control tasks are provided to show the superiority of our algorithm compared with SOTA HIL baselines. The codes are available at https://github.com/LucasCJYSDL/HierAIRL.

Index Terms—Inverse reinforcement learning (IRL), hierarchical imitation learning (HIL), robotic learning.

I. INTRODUCTION

REINFORCEMENT learning (RL) has achieved impressive performance in a variety of scenarios, such as transportation [1], [2], robotic control [3], [4], [5], and cybersecurity [6], [7]. However, most of its applications rely on carefully crafted, task-specific reward signals to drive exploration and learning, limiting its use in real-life scenarios. In this case, imitation learning (IL) methods have been developed to acquire a policy for a certain task based on the corresponding expert demonstrations (e.g., trajectories of state-action pairs) rather than reinforcement signals. However, complex long-horizon tasks can often be broken down and processed as a series of subtasks which can serve as basic skills for completing various compound tasks. In this case, learning a single monolithic policy with IL to represent a structured activity can be challenging. Therefore, hierarchical IL (HIL) has been proposed to recover a two-level policy for a long-horizon task from the demonstrations. Specifically, HIL trains low-level policies (i.e., skills) for accomplishing the specific control for each subtask, and a high-level policy for scheduling the switching of the skills. Such a hierarchical policy, which is usually formulated with the option framework [8], makes full use of the activity structure of the overall task and has the potential for improved performance.

The most state-of-the-art (SOTA) works on HIL [9], [10] are developed based on generative adversarial IL (GAIL) [11] which is a widely adopted IL algorithm. In [9], they additionally introduce a directed information [12] term to the GAIL objective function. In this way, their method can enhance the causal relationship between the skill choice and the corresponding state-action sequence to form low-level policies for each subtask, while encouraging the hierarchical policy to generate trajectories similar to the expert’s in distribution through the GAIL objectives. However, they update the high-level and low-level policy on two separate stages. Specifically, the high-level policy is learned with behavioral cloning (BC), which is a supervised IL algorithm and vulnerable to compounding errors [13], and remains fixed during the low-level policy learning with GAIL. Given that the two-level policies are coupled with each other, such a two-staged paradigm potentially leads to suboptimal solutions. In a separate line of work [10], Jing et al. propose to learn a hierarchical policy by option-occupancy measurement matching, that is, imitating the joint distribution of the options, states and actions of the expert demonstrations rather than only matching the state-action distributions like GAIL. However, they overlook the causal relationship between the subtask structure and the policy hierarchy, so the recovered policy may degenerate into a poorly performed monolithic policy for the whole task, especially when the option annotations of the expert demonstrations are not provided.
In this work, we propose a novel HIL algorithm—hierarchical adversarial inverse RL (H-AIRL), which integrates the SOTA IL algorithm AIRL [14] with the option framework, coupled with an objective based on the directed information for causality enhancement. To the best of our knowledge, this is the first work on the hierarchical extension of AIRL. Compared with GAIL, AIRL is able to recover the expert reward function along with the expert policy and has more robust and stable performance for challenging robotic tasks [14], [15], [16]. From the algorithm perspective, our contributions are as follows.

1) We propose a practical lower bound of the directed information between the option choice and the corresponding state-action sequences. This objective design enables our algorithm to update the high-level and low-level policy at the same time, which is an improvement compared with [9].

2) We redefine the AIRL objectives on the extended state and action space, in order to directly recover a hierarchical policy from the demonstrations.

3) We provide an expectation-maximization (EM) [17] adaption of our algorithm so that it can be applied to the expert demonstrations without the subtask annotations (i.e., unsegmented expert data) which are easier to obtain in practice.

4) We provide theoretical justification of the three folds mentioned above, and comparisons of our algorithm with SOTA HIL, and IL baselines on multiple Mujoco [18] continuous control tasks, where our algorithm significantly outperforms the others.

II. RELATED WORK

A. Learning From Demonstration

Learning from demonstration methods seek to learn to perform a task from expert demonstrations, where a learner is given only samples of trajectories from an expert and is not provided any reinforcement signals, such as the environmental rewards. There are two main branches for learning from demonstration: IL [19], [20] and inverse reinforcement learning (IRL) [21] (Piot et al. [22] deliver an insightful discussion on the interplay between IRL and IL). As a key algorithm in the field of IL, BC [23] learns a policy as a supervised learning problem over state-action pairs from expert trajectories. This method only tends to succeed with large amounts of data, due to the compounding error caused by covariate shift [13]. IRL aims to infer an expert’s reward function from demonstrations, based on which the policy of the expert can be recovered. Most implementations of IRL [24], [25], [26], [27], [28] can avoid the compounding error, but are expensive to solve and scale, since they require RL in each iteration for updating the reward function. GAIL [11] and AIRL [14] have been proposed to scale IRL for complex high-dimensional control tasks. They realize IRL through an adversarial learning framework, where they alternatively update a policy and discriminator network. Contrary to traditional IRL algorithms, AIRL and GAIL eliminate the need to execute a complete RL to derive the optimal policy under the corresponding reward function at each time step. Instead, for each time step (i.e., training episode), the discriminator and the policy undergo sequential updates, each for a certain number of epochs. The discriminator serves as the reward function and learns to differentiate between the expert demonstrations and state-action pairs from the learned policy while the policy is trained to generate trajectories that are difficult to be distinguished from the expert data by the discriminator. The mathematical details are provided in Section III-A. As GAIL and AIRL involve imitating the experts in the learning process, they can also be categorized as IL algorithms. AIRL explicitly recovers the reward function and provides more robust and stable performance among challenging tasks [14], [15], [16], which is chosen as our base algorithm for extension.

B. Hierarchical Learning From Demonstration

Given the nature of subtask decomposition in long-horizon tasks, hierarchical learning from demonstration can achieve better performance by forming micropolicies for accomplishing the specific control for each subtask and learning a macropolicy for scheduling among micropolicies. The micropolicies (also known as skills) in RL can be modeled with the option framework proposed in [8], which extends the usual notion of actions to include options, i.e., closed-loop policies for taking a sequence of actions over a period of time. We provide further details about the option framework in Section III-B. Through integration with options, hierarchical extensions of the learning from demonstration methods mentioned above have been developed, including hierarchical BC (HBC) and hierarchical IRL (HIRL). In HBC, they train a policy for each subtask through supervised learning with the corresponding state-action pairs, due to which the subtask annotations need to be provided or inferred. In particular, the methods proposed in [29] and [30] require segmented data with the subtask information. While, in [31] and [32], they infer the subtask information as the hidden variables in a hidden Markov model [33] and solve the HBC as an MLE problem with the EM algorithm [17]. Despite its theoretical completeness, HBC is also vulnerable to compounding errors in case of limited demonstrations. On the other hand, SOTA HIRL methods—DI-GAIL [9] and option-GAIL [10], have extended GAIL with the option framework to recover the hierarchical policy (i.e., the high-level and low-level policies mentioned above) from unsegmented expert data. Specifically, in DI-GAIL, they introduce a regularizer into the original GAIL objective function to maximize the directed information between generated trajectories and the subtask/option annotations. However, the high-level and low-level policies are trained in two separate stages in their approach, which will inevitably lead to convergence with a poor local optimum. As for Option-GAIL, which is claimed to outperform DI-GAIL and HBC, it replaces the occupancy measurement in GAIL, which measures the distribution of the state-action pairs, with option-occupancy measurement to encourage the hierarchical policy to generate state-action-option tuples with similar distribution to the expert demonstrations. However, they do not adopt the directed information objective to enhance the causal
relationship between the option choice and the corresponding state-action sequence. All the aforementioned algorithms fall under HIL umbrella. In this article, we introduce a novel AIRL algorithm grounded in AIRL. It leverages the directed information objective and concurrently updates both high-level and low-level policies. Further, we offer a theoretical validation of our algorithm and illustrate its superiority in complex robotic control tasks.

III. BACKGROUND

In this section, we introduce the background knowledge of our work, including AIRL, the one-step option framework and variational autoencoder. The definitions are based on the Markov decision process (MDP), denoted by $\mathcal{M} = (S, A, \mathcal{P}, \mu, R, \gamma)$, where $S$ is the state space, $A$ is the action space, $\mathcal{P} : S \times A \times S \rightarrow [0, 1]$ is the state transition function, $\mu : S \rightarrow [0, 1]$ is the distribution of the initial state, $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor. $\gamma$ is set as 1 in the following derivations.

A. Adversarial Inverse RL (AIRL)

IRL aims to infer an expert’s reward function from demonstrations, based on which the policy of the expert can be recovered. Maximum Entropy IRL [24] solves IRL as a maximum likelihood estimation (MLE) problem shown as

$$\max_{\theta} \mathbb{E}_{\tau_\theta} [\log P_\theta (\tau_E)]$$

$$P_\theta (\tau_E) = \tilde{P}_\theta (\tau_E)/Z_\theta$$

$$\tilde{P}_\theta (\tau_E) \approx \mu(S_0) \prod_{t=0}^{T-1} \mathcal{P}(S_{t+1}|S_t, A_t) \exp(\mathcal{R}_\theta(S_t, A_t)).$$

Here, $\tau_E \triangleq (S_0, A_0, \ldots, S_{T-1}, A_{T-1}, S_T)$ denotes an expert trajectory, i.e., a sequence of state-action pairs with horizon $T$; $Z_\theta$ is the partition function defined as $Z_\theta = \int \tilde{P}_\theta (\tau_E) d\tau_E$ (continuous $S$ and $A$) or $Z_\theta = \sum_{\tau_E} \tilde{P}_\theta (\tau_E)$ (discrete $S$ and $A$); $\mathcal{R}_\theta$ is the parametrized reward function. In deterministic environments, the approximation detailed in the second line of (1) is rendered precise. However, even in nondeterministic settings, this approximation remains applicable if the transition randomness has a limited impact on the probabilities of trajectories [24].

Since $Z_\theta$ is intractable for large-scale state-action space, Fu et al. [14] propose AIRL to solve this MLE problem in a sample-based manner. While AIRL’s key theoretical findings are tailored for deterministic environments and especially it assumes that the approximation in (1) applies [24], it has demonstrated success in standard RL benchmark environments that include nondeterministic settings. AIRL facilitates scalable maximum entropy IRL by alternately training a discriminator $D_\theta$ and policy network $\pi$ in an adversarial setting. Specifically, the discriminator is trained by minimizing the cross-entropy loss between the expert demonstrations $\tau_E$ and generated samples $\tau$ by $\pi$

$$\min_{\theta} - \sum_{t=0}^{T-1} \mathbb{E}_{\tau_E} [\log D_\theta (\tau)] - \mathbb{E}_\tau [\log (1 - D_\theta (\tau))]$$

$$D_\theta (\tau) = D_\theta (S, A_t) = \frac{\exp(f_\theta (S, A_t))}{\exp(f_\theta (S, A_t)) + \pi(A_t|S)}.$$ (2)

Here, $f_\theta$ denotes the output of the discriminator network. Contrary to GAIL [11], this output is not directly utilized as the discriminator value $D_\theta$, but rather, it undergoes an intermediary step as illustrated in (2). Meanwhile, the policy $\pi$ is trained with off-the-shelf RL algorithms using the reward function defined with the discriminator: $\log D_\theta (S, A_t) - \log (1 - D_\theta (S, A_t))$. Further, they justify that, at optimality, $f_\theta (S, A_t)$ can serve as the recovered reward function $\mathcal{R}_\theta(S, A)$ and $\pi$ is the recovered expert policy which maximizes the entropy-regularized objective: $\mathbb{E}_{\tau \sim \pi} [\sum_{t=0}^{T-1} \mathcal{R}_\pi(S_t, A_t) - \log \pi(A_t|S_t)]$.

B. One-Step Option Framework

As proposed in [8], an option $Z \in \mathcal{Z}$ can be described with three components: an initiation set $I_Z \subseteq S$, an intraoption policy $\pi_Z(A|S) : S \times A \rightarrow [0, 1]$, and a termination function $\beta_Z(S) : S \rightarrow [0, 1]$. An option $Z$ is available in state $S$ if and only if $S \in I_Z$. Once the option is taken, actions are selected according to $\pi_Z$ until it terminates stochastically according to $\beta_Z$, i.e., the termination probability at the current state. A new option will be activated in this call-and-return style by a high-level policy $\pi_Z(Z|S) : S \times Z \rightarrow [0, 1]$ once the previous option terminates. In this way, $\pi_Z(Z|S)$ and $\pi_Z(A|S)$ constitute a hierarchical policy for a certain task. Note that a two-level hierarchy assumption is commonly adopted in literature related to hierarchical learning that employs the option framework. This includes studies in Hierarchical RL [35], [36], [37] as well as HIL [9], [10], [38]. Also, hierarchical policies tend to have superior performance on complex long-horizon tasks which can be broken down into a series of subtasks [39], [40], [41].

However, it is inconvenient to deal with the initiation set $I_Z$ and termination function $\beta_Z$ while learning this hierarchical policy. Thus, in [10] and [36], they adopt the one-step option framework. It is assumed that each option is available in each state: $I_Z = S, \forall Z \in \mathcal{Z}$ and every option switch is effective: $P(Z_{t+1} = Z_t | \beta_Z(Z_{t-1})) = 0$. Also, the high-level and low-level (i.e., intraoption) policy are redefined as $\pi_\theta$ and $\pi_\phi$, respectively,

$$\pi_\theta(Z, S) = \beta_Z(S) \pi_Z(Z|S) + (1 - \beta_Z(S)) \delta_{Z = Z'}$$

$$\pi_\phi(A, S, Z) = \pi_Z(A|S)$$

where $Z'$ denotes the option in the previous time step and $\delta_{Z = Z'}$ is the indicator function. We can see that if the previous option terminates (with probability $\beta_Z(S)$), the agent will select a new option according to $\pi_Z(Z|S)$; otherwise, it will stick to $Z'$. With the new definition and assumption, we can optimize the hierarchical policy $\pi_\theta$ and $\pi_\phi$ without the extra need to justify the exact beginning and breaking condition of each option. As demonstrated in [10], with the assumption mentioned above, the one-step option framework aligns equivalently with the original option framework proposed in [8].
Nevertheless, \( \pi_\theta(Z|S, Z') \) still includes two separate parts, i.e., \( \beta_Z(S) \) and \( \pi_Z(Z|S) \), and due to the indicator function, the update gradients of \( \pi_Z \) will be blocked/gated by the termination function \( \beta_Z(S) \). In this case, Li et al. [37] propose to marginalize the termination function away, and instead implement \( \pi_\theta(Z|S, Z') \) as an end-to-end neural network (NN) with the multihead attention (MHA) mechanism [42] which enables their algorithm to temporally extend options in the absence of the termination function. We provide more details on MHA and the structure design of \( \pi_\theta \) and \( \pi_\phi \) in Supplement Appendix A. With the marginalized one-step option framework, we only need to train the two NN-based policy, i.e., \( \pi_\theta \) and \( \pi_\phi \). In particular, we adopt the SOTA HRL algorithm, i.e., SA, proposed in [37] to learn \( \pi_\theta \) and \( \pi_\phi \). Given that we do not have access to the environmental rewards, the return function in the HRL setting is instead calculated based on the directed information and AIRL objective terms which will be introduced in Section IV.

IV. PROPOSED APPROACH

Our research focuses on deriving a hierarchical policy from expert demonstrations by merging the one-step option framework with AIRL. This hierarchical policy includes low-level policies (skills) for individual subtasks and a high-level policy for scheduling among these skills. In Section IV-A, we put forth a novel objective term to strengthen the directed causal relation between the subtask embeddings and the trajectories, as the trajectory hierarchies should mirror the subtask structure. Subsequently, in Section III-A, we broaden the original AIRL, which could only reconstruct a monolithic policy from demonstrations, with the option framework and offers a corresponding EM adaptation, enabling us to learn a two-level policy directly from (unannotated) expert data. Finally, we construct the overall objective [i.e., (10)] by integrating the two aforementioned objective functions. The significance of each objective term is empirically validated in Section V.

A. Augmenting the Directed Causal Relationship

First, we present our intuition to introduce an objective related to the directed information between option choices and the trajectory. As mentioned in Section III-B, when observing a new state, the agent will first decide on its option choice \( Z \) using the high-level policy \( \pi_\theta \) and then select the primitive action based on the low-level policy \( \pi_\phi \) corresponding to \( Z \). In this case, the decision-making at each time step \( t \in \{0, \ldots, T\} \) should condition on the corresponding option choice \( Z_t \), so we view the option choices \( Z_{0:T} \) as the latent variables related to the trajectory in a probabilistic graphical model shown as Fig. 1. It can be observed from Fig. 1 that \( Z_{0:T} \) has a directed causal relationship with the trajectory \( X_{0:T} = (X_0, \ldots, X_T) = ((A_{-1}, S_0), \ldots, (A_T, S_T)) \), where \( A_{-1} \) is a dummy variable. As discussed in [9] and [12], this kind of connection can be established by maximizing the directed information (also known as casual information) flow from the trajectory to the latent factors of variation, i.e.,

\[
I(X_{0:T} \rightarrow Z_{0:T}), \text{ which is defined as (we use } X' \text{ to represent } X_{0:T} \text{ for simplicity, and so on)}
\]

\[
I(X_{0:T} \rightarrow Z_{0:T}) = \sum_{t=1}^{T} [H(Z_t|Z_{0:t-1}) - H(Z_t|X_{0:t}, Z_{0:t-1})]
\]

\[
= \sum_{t=1}^{T} \left[ H(Z_t|Z^{t-1}) + \sum_{X',Z'} P(X'|Z') \log P(Z_t|X', Z^{t-1}) \right].
\]  

(4)

Here, \( H(\cdot) \) denotes the entropy [44] and we assume the trajectory \( X_{0:T} \) to be discrete in order to simplify the notations, but it can be either discrete or continuous in use.

It is infeasible to directly optimize the above objective, since it is difficult to calculate the posterior distribution \( P(Z_t|X_{0:t}, Z_{0:t-1}) \) with only the hierarchical policy, i.e., \( \pi_\theta \) and \( \pi_\phi \). In this case, we instead maximize its variational lower bound as follows (please refer to Supplement Appendix B for derivations):

\[
L^{\text{DL}} \triangleq \sum_{t=1}^{T} \left[ H(Z_t|X^{t-1}, Z^{t-1}) \right. \\
+ \sum_{X',Z'} P_\theta,\phi(X', Z') \log P_\theta(Z_t|X', Z^{t-1}) \bigg].
\]  

(5)

Here, \( H(Z_t|X_{0:t-1}, Z_{0:t-1}) \) is the entropy of distribution of the option choice at time step \( t \), which can be computed from output of the high-level policy \( \pi_\theta; P_\theta(Z_t|X_{0:t}, Z_{0:t-1}) \) is a variational estimation of \( P(Z_t|X_{0:t}, Z_{0:t-1}) \), which is learned as a NN; \( P_\theta,\phi(X_{0:t}, Z_{0:t}) \) can be calculated with the hierarchical policy as follows (please refer to Supplement Appendix B for derivations):

\[
\mu(S_0) \prod_{i=1}^{t} P(Z_i|X^{i-1}, Z^{i-1})P(A_{i-1}|X^{i-1}, Z^{i-1})P_{S_{i-1}, A_{i-1}}^{S_i}
\]

\[
= \mu(S_0) \prod_{i=1}^{t} \pi_\theta(Z_i|S_{i-1}, Z_{i-1})\pi_\phi(A_{i-1}|S_{i-1}, Z_i)P_{S_{i-1}, A_{i-1}}^{S_i}
\]  

(6)

In this equation, \( \mu \) denotes the initial state distribution and \( P_{S_{i-1}, A_{i-1}}^{S_i} = P(S_i|S_{i-1}, A_{i-1}) \) is the transition dynamic. Note that (6) can be estimated through Monte Carlo sampling [45], even though \( \mu \) and \( P \) are unknown. As a result, we can update \( \theta, \phi, \omega \), i.e., parameters of the hierarchical policy and variational posterior, by maximizing \( L^{\text{DL}} \), and thus maximize \( I(X_{0:T} \rightarrow Z_{0:T}) \).

B. Hierarchical Adversarial Inverse RL (H-AIRL)

IL algorithms have been adopted to learn a monolithic policy based on expert demonstrations instead of reward signals which are usually difficult to acquire in real life. However, SOTA IL algorithms like AIRL [14] cannot be directly adopted to recover a hierarchical policy since they do not take the option choices of the expert \( Z^E_{0:T} \) into consideration. To this end, we propose a novel hierarchical extension of AIRL,
denoted as H-AIRL, as a solution. Further, we note that it is usually difficult to annotate the option choices \( Z^E_{0:T} \) for an expert trajectory \( X^E_{0:T} \), so we propose an EM adaption of H-AIRL as well to learn the hierarchical policy based on only the unsegmented expert trajectories, i.e., \( \{X^E_{0:T}\} \).

The underlying intuition of this hierarchical extension is drawn from comparing the definitions of the hierarchical policy and the monolithic policy [i.e., \( \pi(A|S) \)]. For the hierarchical agent, when observing a state \( S_t \) at time step \( t \in \{0,\ldots,T-1\} \), the agent needs first to decide on its option choice based on \( S_t \) and its previous option choice \( Z_t \) using the high-level policy \( \pi_0(Z_{t+1}|S_t, Z_t) \), and then decide on the action with the corresponding low-level policy \( \pi_\phi(A_t|S_t, Z_{t+1}) \). Thus, the definition of the hierarchical policy can be acquired through the chain rule

\[
\pi_\theta(Z_{t+1}|S_t, Z_t) \cdot \pi_\phi(A_t|S_t, Z_{t+1}) = \pi_\theta(Z_{t+1}|S_t, Z_t) \cdot \pi_\phi(A_t|S_t, Z_{t+1}, Z_{t+1}) \\
= \pi_\theta(Z_{t+1}|S_t, Z_t) \cdot \pi_\phi(A_t|S_t, Z_{t+1}, Z_{t+1}) \\
= \pi_\theta,\phi(\tilde{A}_t|S_t).
\]

(7)

Here, the first equality holds because of the one-step Markov assumption (i.e., \( A_t \) is independent of \( Z_t \) given \( Z_{t+1} \)). \( \tilde{S}_t \triangleq (S_t, Z_t) \) and \( \tilde{A}_t \triangleq (Z_{t+1}, A_t) \) denote the extended state and action space, respectively. Upon contrasting the hierarchical policy with the monolithic policy, it becomes clear that the state and action space (i.e., \( S_t \) and \( A_t \)) must be replaced with their extended versions (i.e., \( \tilde{S}_t \) and \( \tilde{A}_t \)) within the original AIRL algorithm to facilitate the hierarchical extension.

By substituting \( (S_t, A_t) \) with \( (\tilde{S}_t, \tilde{A}_t) \) and \( \tau_E \) with the hierarchical trajectory \( (X^E_{0:T}, Z^E_{0:T}) \) in (1), we can get an MLE problem shown as (8), from which we can recover the hierarchical reward function and policy. The derivation of (8) is available in Supplement Appendix D. Note that we assume the approximation in (1) to be exact as in AIRL [14]

\[
\max_\theta \mathbb{E}_{(X^E_{0:T}, Z^E_{0:T}) \sim p_{\text{E}}} \left[ \log P_\theta(X^E_{0:T}, Z^E_{0:T}) \right] \\
= \mu(\tilde{S}_0) \prod_{t=0}^{T-1} \mathbb{P}(\tilde{S}_{t+1}|\tilde{S}_t, \tilde{A}_t) \exp(R_\phi(\tilde{S}_t, \tilde{A}_t)) \\
= \mu(\tilde{S}_0) \prod_{t=0}^{T-1} \mathbb{P}(\tilde{S}_{t+1}|\tilde{S}_t, \tilde{A}_t) e^{R_\phi(S_t, Z_t, Z_{t+1}, A_t)}. \tag{8}
\]

As mentioned in Section III-A, the MLE problem can be efficiently solved with an adversarial learning framework, which is summarized as (9). At optimality, we can recover the hierarchical reward function (i.e., \( f_\phi \)) and policy (i.e., \( \pi_{0,\phi} \)) of the expert with these objectives, of which the justification is provided in Supplement Appendix E

\[
\min_\theta -\mathbb{E}_{(X^E_{0:T}, Z^E_{0:T}) \sim p_{\text{E}}} \left[ \sum_{t=0}^{T-1} \log D_\phi(S^E_t, Z^E_t, Z^E_{t+1}, A^E_t) \right] \\
- \mathbb{E}_{(X^E_{0:T}, Z^E_{0:T}) \sim p_{\text{E}}} \left[ \sum_{t=0}^{T-1} \log(1 - D_\phi(S_t, Z_t, Z_{t+1}, A_t)) \right] \\
\max_{\theta,\phi} L^E = \mathbb{E}_{(X^E_{0:T}, Z^E_{0:T}) \sim p_{\text{E}}} \left[ \sum_{t=0}^{T-1} R^E_{\theta,\phi} \right]. \tag{9}
\]

In the above equation, the reward is defined as \( R^E_{\theta,\phi} = \log D^\theta - \log(1 - D^\phi) \) where \( D^\theta = D_\theta(S_t, Z_t, Z_{t+1}, A_t) = \exp(f_\phi(S_t, Z_t, Z_{t+1}, A_t)) / \exp(f_\phi(S_t, Z_t, Z_{t+1}, A_t)) + \pi_{0,\phi}(Z_{t+1}, A_t|S_t, Z_t) \).

Usually, we can only acquire the trajectories of state-action pairs, i.e., \( \{X^E_{0:T}\} \), from the expert. In this case, we view the option choices \( Z^E_{0:T} \) as hidden variables in this MLE problem and adopt an EM-style adaption of our algorithm. In the expectation (E) step, we sample possible option choices with \( Z^E_{0:T} \sim P_{\text{E}}(\cdot|X^E_{0:T}) \). As introduced in Section IV-A, \( P_{\text{E}} \) represents the trained posterior distribution network for the option choices, with the parameter \( \overline{\omega} \), i.e., the old parameters before being updated in the M step. Then, in the maximization (M) step, we optimize the objectives shown in (9) for iterations, by replacing the samples in the first term of (9) with \( (X^E_{0:T}, Z^E_{0:T}) \) collected in the E step. The theoretical justification of the effectiveness of this EM-like algorithm is provided in Supplement Appendix F.

To sum up, there are in total four networks to learn in our system: the high-level policy \( \pi_0 \), low-level policy \( \pi_\phi \), discriminator \( D_\theta \), and variational posterior \( P_\omega \). \( D_\theta \) can be trained by minimizing the cross-entropy loss shown in (9).
Algorithm 1 Hierarchical Adversarial Inverse Reinforcement Learning (H-AIRL)

1: **Input:** Expert demonstrations \( \{X^E_{0:T}\} \) (If the option annotations, i.e., \( \{Z^E_{0:T}\} \), are provided, step 5 is not required.)
2: Initialize the posterior network \( P_\phi \), high-level policy \( \pi_\theta \), low-level policy \( \pi_\varphi \), and discriminator \( D_\beta \)
3: **for each training episode**
   4: Generate \( M \) trajectories \( \{(X_{0:T}, Z^E_{0:T})\} \) with \( \pi_\theta \) and \( \pi_\varphi \) by interacting with the simulator
5: Sample option choices corresponding to the expert trajectories using the posterior, i.e., \( Z^E_{0:T} \sim P_\phi(\cdot|X^E_{0:T}) \)
6: Update \( P_\phi \) by minimizing \( L^\mathcal{D}\) (Equation (5)) using Stochastic Gradient Descent [43] with \( \{(X_{0:T}, Z^E_{0:T})\} \)
7: Update \( D_\beta \) by minimizing the cross-entropy loss in Equation (9) based on \( \{(X_{0:T}, Z^E_{0:T})\} \) and \( \{(X^E_{0:T}, Z^E_{0:T})\} \)
8: Train \( \pi_\theta \) and \( \pi_\varphi \) by maximizing the return function \( L \) defined in Equation (10) using the HRL algorithm SA [37]
9: **end for**

While the update of the other three networks should follow:

\[
\max_{\theta, \varphi, \omega} L(\theta, \varphi, \omega) = \alpha_1 L^\mathcal{D}(\theta, \varphi, \omega) + \alpha_2 L^\mathcal{L}(\theta, \varphi)
\]

where \( \alpha_1, \alpha_2 > 0 \) are the weights for each objective term and fine-tuned as hyperparameters, \( L^\mathcal{D} \) and \( L^\mathcal{L} \) are defined in (5) and (9), respectively. While introducing \( L^\mathcal{D} \) may potentially impact the optimality of the MLE problem as stated in (8), our empirical evidence suggests that a hierarchical policy trained using \( L \) can outperform one trained solely with \( L^\mathcal{L} \). This is achieved by carefully balancing the tradeoff between causality enhancement and likelihood maximization through the adjustment of parameters \( \alpha_1 \) and \( \alpha_2 \). The interaction among the four networks in our algorithm framework is illustrated in Fig. 2, and a detailed pseudo code of our algorithm is available in Algorithm 1.

V. EVALUATION AND MAIN RESULTS

In this section, we compare H-AIRL with SOTA HIL algorithms: Option-GAIL [10] and DI-GAIL [9], to justify the superiority of our algorithm, and we provide comparisons with SOTA IL algorithms: GAIL [11] to show the importance of hierarchical policy learning for challenging long-horizon tasks. To keep it fair, we use the original implementations of these baseline algorithms provided by the authors. Moreover, we provide ablation study of our algorithm to evaluate the key components of our algorithm design. Specifically, we compare with: 1) Option-AIRL: a version of our algorithm by only keeping the AIRL-related term in the objective to update the hierarchical policy, i.e., \( L^\mathcal{L} \) in (10); and 2) H-GAIL: a variant by replacing the AIRL component of our algorithm with GAIL, of which the details are provided in Supplement Appendix G.

To validate our algorithm’s ability to outperform SOTA HIL and IL baselines, and to determine the necessity of each component in our algorithm’s design, we contrast our algorithm against the aforementioned baselines. We conduct this comparison across three demanding robotic control tasks constructed with Mujoco [18], namely, Hopper, Walker, and AntPusher, as depicted in Fig. 3. Hopper and Walker are locomotion tasks where the robot agents are required to move toward a certain direction by learning to coordinate their legs. Both of them have continuous state and action spaces. Specifically, Hopper has a 11-dim state space and 3-dim action space, and Walker has a 17-dim state space and 6-dim action space. While, the AntPusher needs not only to learn locomotion skills for the Ant agent [Fig. 3(c)] but also to learn to navigate into a room that is blocked by a movable box [i.e., the red one in Fig. 3(d)]. In particular, the Ant agent needs to first navigate to the left side of the box and push it away, and then enter the blocked room to complete the task, which is much more challenging with a 107-dim continuous state space and 8-dim continuous action space.

Further, in order to illustrate the learned hierarchical policy from the demonstrations and analyze the transferability of the learned skills (i.e., options), we build two Mujoco Maze tasks with the point agent, shown as Fig. 5(a) and (d). The point
Fig. 4. Comparisons with SOTA HIL and IL algorithms on Mujoco. Each training is repeated three times with different random seeds, and the mean and standard deviation are plotted as the solid lines and shaded areas, respectively. It can be observed that our algorithm, i.e., H-AIRL, performs the best, the GAIL-based algorithms suffer from unstableness, and the two-stage learning of the hierarchical policy in Directed-Info GAIL leads to poor performance. (a) Hopper. (b) Walker. (d) AntPusher.

Table II

|          | Expert     | H-AIRL (Ours) | Option-AIRL | H-GAIL     |
|----------|------------|---------------|-------------|------------|
| Hopper   | 3139.85 ± 712.02 | 3501.81 ± 110.79 | 1841.19 ± 401.22 | 2574.06 ± 920.34 |
| Walker   | 5317.56 ± 99.90  | 4354.14 ± 193.28  | 3951.41 ± 631.48  | 3812.99 ± 712.83  |
| AntPusher| 116.00 ± 3.99    | 113.94 ± 2.54     | 81.58 ± 39.75     | 80.23 ± 30.29     |

A. Comparisons With SOTA HIL and IL Baselines

In this section, we compare our algorithm, i.e., H-AIRL, with the SOTA HIL and IL baselines mentioned above on the three Mujoco locomotion tasks. As shown in Fig. 4, we plot the change of the episodic rewards (i.e., the sum of the rewards at each time step within an episode) in the training process. Note that these episodic rewards are specifically designed by OpenAI Gym [47] to encourage the Mujoco agents to complete the corresponding tasks as fast as possible at the least control cost, which can be used as evaluation metrics for the agents’ learning performance.

We repeat each training for three times with different random seeds (simply chosen as 0, 1, 2), plot the average value as the solid lines and the standard deviation as the shaded areas. It can be observed from Fig. 4 that H-AIRL outperforms the baselines significantly in terms of both the convergence speed and final performance. Also, Option-GAIL has a better performance than GAIL, which shows the advantages of hierarchical policy learning for challenging long-horizon tasks. While the fluctuations during the training process of GAIL and Option-GAIL show the unstableness of the GAIL-based algorithms. Moreover, it can be observed that DI-GAIL performs even worse than GAIL which does not take advantage of options in the learning process, showing that the separate learning of the high-level and low-level policies (i.e., learning the high-level policy first at the pretraining stage and then fixing it during the low-level policy training) will harm the agents’ performance and lead to convergence to a poor local optimum.

B. Ablation Study

Here, we provide comparisons of H-AIRL with Option-AIRL and H-GAIL as the ablation study. As shown in (10), the objective function for updating the hierarchical policy includes two parts, i.e., the directed information term $L^\text{DI}$ and AIRL term $L^\text{IL}$. In order to evaluate the importance of $L^\text{DI}$, we implement the baseline Option-AIRL, for which we only keep $L^\text{IL}$, i.e., the AIRL objectives on the extended state and action spaces, for updating the hierarchical policy. On the other hand, we replace the AIRL objective with the one defined with GAIL (Equation (24) in Supplement Appendix), denoted as H-GAIL, to show the necessity to adopt AIRL as our base IL algorithm.

In Table II, we provide numeric results of the performance of the expert demonstrations, our algorithm, and the ablation baselines on the three Mujoco benchmarks. To be specific, we repeat the training with each algorithm on each task for three times with different random seeds (chosen as 0, 1, 2), and calculate the mean and standard deviation of the episodic rewards after they converge across different random seeds as the metric of their final performance. It can be observed that our algorithm outperforms the baselines in both the average performance and the stableness, showing the effectiveness of the key components of our algorithm design. On the other
Fig. 5. (a)–(d) Mujoco Maze tasks for evaluation, where the orange arrows and green arrows represent goal-achieving task 1 and 2, respectively. The skills learned in task 1 are adopted to task 2 as initialization to testify their effect. (b) and (c) Trajectories of the expert agent and the agent trained with H-AIRL for goal-achieving task 1 in Point Room. (e) and (f) Trajectories of the expert agent and the agent trained with H-AIRL for goal-achieving task 1 in Point Corridor. The skills (i.e., options) are divided according to the forward direction of the point agent. It can be observed that, in both tasks, the agent learns the intraoption policy for each skill and knows when and how to switch among these skills using its high-level policy.

Fig. 6. Learning curves of different algorithms on goal-achieving task 2 of (a) point room and (b) point corridor, where the yellow lines denote the level of the expert demonstrations. Each training is repeated three times with different random seeds, and the mean and standard deviation are plotted as the solid lines and shaded areas, respectively. We can observe that H-AIRL outperforms the baselines significantly. Moreover, the convergence speed can be further improved if we transfer the skills learned in task 1 to task 2 as initialization (i.e., H-AIRL-init).

C. Analysis on Learned Skills

HIL can be especially effective for long-horizon structured tasks by forming low-level policies (i.e., skills) for each subtask and a high-level policy for scheduling the switch of the skills in the meantime. Moreover, the learned skills have the potential to be transferred to other tasks in this scenario to reduce the learning cost. To justify these aspects, we provide evaluations on Mujoco Maze tasks shown as Fig. 5(a) and (d), where the learned hierarchical policies can be explicitly visualized as the trajectories. In these two scenarios, the learning is based on structural demonstrations shown in Fig. 5(b) and (e). Different from the ones used in the three Mujoco locomotion tasks mentioned above, these demonstrations are with option annotations which are divided according to the forward direction of the agent. For example, all the trajectories where the agent goes up correspond to the same option and are marked as green in Fig. 5(e). Through training with our algorithm (i.e., H-AIRL), we can get the hierarchical policy shown in Fig. 5(c) and (f). It can be observed that our algorithm can recover the skills for the specific control of each subtask and the high-level policy for choosing the required skill at each time step.

Further, to evaluate the transferability of the learned skills to other tasks in similar scenarios, we adopt these skills to initialize the training for the goal-achieving task 2 in the Mujoco Mazes, which are shown as the green arrows in Fig. 5(a) and (d). The training of each algorithm is repeated three times with different random seeds (chosen as 0, 1, 2), and the mean and standard deviation are plotted as the solid lines and shaded areas, respectively. We can observe from Fig. 6 that H-AIRL with initialization (i.e., H-AIRL-init) can converge and achieve the expert level (i.e., the yellow line) faster than H-AIRL. Thus, the learned skills can be used to reduce the training cost in different but related tasks, which is another advantage of HIL compared with canonical IL. To justify the learning performance of our algorithm, we also provide comparisons with the baselines: Option-GAIL and GAIL, where both H-AIRL and H-AIRL-init significantly outperform. We do not compare with DI-GAIL because this algorithm...
cannot be applied to learning from expert demonstrations with option annotations.

VI. CONCLUSION AND FUTURE WORK
HIL has demonstrated superior performance over conventional IL in the context of complex tasks with a subtask structure. This article presents a novel HIL algorithm that integrates extended AIRL objectives with a directed information term. Notably, our learning framework allows for concurrent updates to high-level and low-level policies, mitigating suboptimality associated with two-stage training found in prior works. Additionally, we propose an EM adaptation of our algorithm suitable for unsegmented expert demonstrations, enhancing its practical applicability. Theoretical analysis and an ablation study are provided for each crucial component of our algorithm design to justify their validity. Furthermore, our algorithm's superior performance is validated through comparisons with SOTA HIL and IL baselines on Mujoco control tasks. As for future works, integrating H-AIRL with Meta/Multi-task Learning techniques [48], [49] for novel multitask HIL algorithms is also an interesting direction.

REFERENCES
[1] J. Chen, A. K. Umrawal, T. Lan, and V. Aggarwal, “DeepFreight: A model-free deep-reinforcement-learning-based algorithm for multi-transfer freight delivery,” Proc. Int. Conf. Automated Planning Scheduling, vol. 31, no. 1, 2021, pp. 510–518.
[2] A. O. Al-Ambabi, A. Ghosh, and V. Aggarwal, “DeepPool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 12, pp. 4714–4727, Dec. 2019.
[3] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, “Visual foresight: Model-based deep reinforcement learning for vision-based robotic control,” 2018, arXiv:1812.00568.
[4] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” 2015, arXiv:1509.02971.
[5] H. Li, Q. Zhang, and D. Zhao, “Deep reinforcement learning-based automation exploration for navigation in unknown environment,” IEEE Trans. Neurol. Netw. Learn. Syst., vol. 31, no. 6, pp. 2064–2076, Jun. 2020.
[6] T. T. Nguyen and V. J. Reddi, “Depreciation learning for cyber security,” IEEE Trans. Neurol. Netw. Learn. Syst., vol. 34, no. 9, pp. 3779–3795, Aug. 2023.
[7] A. M. K. Adawadkar and N. Kulkarni, “Cyber-security and reinforcement learning—A brief survey,” Eng. Appl. Artif. Intell., vol. 114, Sep. 2022, Art. no. 105116.
[8] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” Artif. Intell., vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.
[9] M. Sharma, A. Sharma, N. Rhinehart, and K. M. Kilany, “Directed-info GAIL: Learning hierarchical policies from unsegmented demonstrations using directed information,” in Proc. 7th Int. Conf. Learn. Represent., 2019, pp. 1–15.
[10] M. Jing et al., “Adversarial option-aware hierarchical imitation learning,” in Proc. 38th Int. Conf. Mach. Learn., 2021, pp. 5097–5106.
[11] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1–6.
[12] J. Massey et al., “Causality, feedback and directed information,” in Proc. Int. Symp. Inf. Theory Appl., 1990, pp. 303–305.
[13] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in Proc. 14th Int. Conf. Artif. Intell. Statist., 2011, pp. 627–635.
[14] J. Fu, K. Luo, and S. Levine, “Learning robust rewards with adversarial inverse reinforcement learning,” 2017, arXiv:1710.11249.
[15] R. Jena, C. Liu, and K. Sycara, “Augmenting GAIL with bc for sample efficient imitation learning,” in Proc. Conf. Robot. Learn., 2021, pp. 80–90.
[16] P. Wang, D. Liu, J. Chen, H. Li, and C.-Y. Chan, “Decision making for autonomous driving via augmented adversarial inverse reinforcement learning,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2021, pp. 1036–1042.
[17] T. K. Moon, “The expectation-maximization algorithm,” IEEE Signal Process. Mag., vol. 13, no. 6, pp. 47–60, Nov. 1996.
[18] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2012, pp. 5026–5033.
[19] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” Robot. Auto. Syst., vol. 57, no. 5, pp. 469–483, May 2009.
[20] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, “Imitation learning: Progress, taxonomies and challenges,” IEEE Trans. Neural Netw. Learn. Syst., early access, Oct. 25, 2022, doi: 10.1109/TNNLS.2022.3213246.
[21] A. Y. Ng and S. Russell, “Algorithms for inverse reinforcement learning,” in Proc. 7th Int. Conf. Mach. Learn. San Mateo, CA, USA: Morgan Kaufmann, 2000, pp. 663–670.
[22] B. Piot, M. Geist, and O. Pietquin, “ Bridging the gap between imitation learning and inverse reinforcement learning,” IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 8, pp. 1814–1826, Aug. 2017.
[23] D. A. Pomerleau, “Efficient training of artificial neural networks for autonomous navigation,” Neural Comput., vol. 3, no. 1, pp. 88–97, Mar. 1991.
[24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in Proc. 33rd AAAI Conf. Artif. Intell. Palo Alto, CA, USA: AAAI Press, 2008, pp. 1433–1438.
[25] S. Levine, Z. Popovic, and V. Koltun, “Nonlinear inverse reinforcement learning with Gaussian processes,” in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 19–27.
[26] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” 2015, arXiv:1507.04888.
[27] B. Lian, W. Xue, F. L. Lewis, and T. Chai, “Inverse reinforcement learning for adversarial apprentice games,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 8, pp. 4596–4609, Aug. 2023.
[28] W. Xue, B. Lian, J. Fan, P. Kolaric, T. Chai, and F. L. Lewis, “Inverse reinforcement Q-learning through expert imitation for discrete-time systems,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 5, pp. 2386–2399, May 2023.
[29] H. M. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and H. D. III, “Hierarchical imitation and reinforcement learning,” in Proc. 35th Int. Conf. Mach. Learn., vol. 80, 2018, pp. 2923–2932.
[30] T. Kipf et al., “Compile: Compositional imitation learning and execution,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 3418–3428.
[31] C. Daniel, H. van Hoof, J. Peters, and G. Neumann, “Probabilistic inference for determining options in reinforcement learning,” Mach. Learn., vol. 104, nos. 2–3, pp. 337–357, Sep. 2016.
[32] Z. Zhang and I. C. Paschalidou, “Provably hierarchical imitation learning via EM,” in Proc. 24th Int. Conf. Artif. Intell. Statist., 2021, pp. 883–891.
[33] S. Fine, Y. Singer, and N. Tishby, “The hierarchical hidden Markov model: Analysis and applications,” Mach. Learn., vol. 32, no. 1, pp. 41–62, Jul. 1998.
[34] A. Gleave and S. Toyer, “A primer on maximum causal entropy inverse reinforcement learning,” 2022, arXiv:2203.11409.
[35] P. Henderson, J. Harb, and D. Precup, “The option-critic architecture,” in Proc. 31st AAAI Conf. Artif. Intell. Palo Alto, CA, USA: AAAI Press, 2017, pp. 1726–1734.
[36] S. Zhang and S. Whiteson, “DAC: The double-actor-critic architecture for learning options,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 2010–2020.
[37] C. Li, D. Song, and D. Tao, “The skill-action architecture: Learning abstract action embeddings for reinforcement learning,” in Proc. 9th Int. Conf. Learn. Represent., 2017, pp. 1–30.
[38] P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, and D. Precup, “OptionGAN: Learning joint reward-policy options using generative adversarial inverse reinforcement learning,” 2017, arXiv:1709.06683.
[39] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, “Hierarchical deep reinforcement learning for continuous action control,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 11, pp. 5174–5184, Nov. 2018.
[40] S. Pateria, B. Subagdja, A.-H. Tan, and C. Quek, “End-to-end hierarchical reinforcement learning with integrated subgoal discovery,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 12, pp. 7778–7790, Dec. 2022.

[41] J. Chen, J. Chen, T. Lan, and V. Aggarwal, “Scalable multi-agent covering option discovery based on Kronecker graphs,” in Proc. Adv. Neural Inf. Process. Syst., vol. 35, 2022, pp. 30406–30418.

[42] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

[43] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in Proc. 19th Int. Conf. Comput. Statist. Cham, Switzerland: Springer, 2010, pp. 177–186.

[44] T. M. Cover, Elements of Information Theory, Hoboken, NJ, USA: Wiley, 1999.

[45] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA, USA: MIT Press, 2018.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[47] G. Brockman et al., “OpenAI gym,” CoRR, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[48] J. Chen, D. Tamboli, T. Lan, and V. Aggarwal, “Multi-task hierarchical adversarial inverse reinforcement learning,” 2023, arXiv:2305.12633.

[49] A. Singh et al., “Scalable multi-task imitation learning with autonomous improvement,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2020, pp. 2167–2173.

[50] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton, “Stacked capsule autoencoders,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 15486–15496.

[51] D. Galvin, “Three tutorial lectures on entropy and counting,” 2014, arXiv:1406.7872.

[52] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, “A recurrent latent variable model for sequential data,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 2980–2988.

[53] J. L. W. V. Jensen, “Sur les fonctions convexes et les inégalités entre les valeurs moyennes,” Acta Mathematica, vol. 30, no. 1, pp. 175–193, 1906.

Jiayu Chen (Graduate Student Member, IEEE) received the B.E. degree from the College of Engineering, Peking University, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree with the School of Industrial Engineering (Operation Research), Purdue University, West Lafayette, IN, USA. He is a Research Assistant with the School of Industrial Engineering (Operation Research), Purdue University. His research interests include deep reinforcement learning and robotic learning, specifically focusing on unsupervised reinforcement learning and hierarchical imitation learning.

Tian Lan (Senior Member, IEEE) received the B.A.Sc. degree from Tsinghua University, Beijing, China, in 2003, the M.A.Sc. degree from the University of Toronto, Toronto, ON, Canada, in 2005, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 2010. He is currently a Full Professor of electrical and computer engineering with George Washington University, Washington, DC, USA. His research interests include network optimization, algorithms, and machine learning.

Vanneet Aggarwal (Senior Member, IEEE) received the B.Tech. degree from the Indian Institute of Technology Kanpur, Kanpur, India, in 2005, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 2007 and 2010, respectively, all in electrical engineering. He is currently a Full Professor at Purdue University, West Lafayette, IN, USA, where he has been since 2015. Prior to that, he was with AT&T Labs Research from 2010 to 2014. His current research interests include supervised learning, reinforcement learning, and quantum machine learning.

Dr. Aggarwal received the Princeton University’s Porter Ogden Jacobus Honors Fellowship in 2009 and the Purdue University’s Most Impactful Faculty Innovator Award in 2020. In addition, he received the IEEE Jack Neubauer Memorial Award in 2017, the IEEE INFOCOM Workshop Best Paper Award in 2018, and the NeurIPS Workshop Best Paper Award in 2021. He was an Associate Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING (2017–2020) and IEEE TRANSACTIONS ON COMMUNICATIONS (2017–2022). He is currently serving on the Editorial Board for IEEE/ACM TRANSACTIONS ON NETWORKING (since 2019) and the Coeditor-in-Chief for the ACM Journal on Autonomous Transportation Systems (since 2022).