Interpretable Imitation Learning with Symbolic Rewards

NICOLAS BOUGIE, NEC-AIST AI Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology, Japan
TAKASHI ONISHI, NEC-AIST AI Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology, Japan and NEC Corporation Data Science Research Laboratories, Japan
YOSHIMASA TSURUOKA, NEC-AIST AI Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology, Japan and Department of Information and Communication Engineering, The University of Tokyo, Japan

Sample inefficiency of deep reinforcement learning methods is a major obstacle for their use in real-world tasks as they naturally feature sparse rewards. In fact, this from-scratch approach is often impractical in environments where extreme negative outcomes are possible. Recent advances in imitation learning have improved sample efficiency by leveraging expert demonstrations. Most work along this line of research employs neural network-based approaches to recover an expert cost function. However, the complexity and lack of transparency make neural networks difficult to trust and deploy in the real world. In contrast, we present a method for extracting *interpretable* symbolic reward functions from expert data, which offers several advantages. First, the learned reward function can be parsed by a human to understand, verify and predict the behavior of the agent. Second, the reward function can be improved and modified by an expert. Finally, the structure of the reward function can be leveraged to extract *explanations* that encode richer domain knowledge than standard scalar rewards. To this end, we use an autoregressive recurrent neural network that generates hierarchical symbolic rewards represented by simple symbolic trees. The recurrent neural network is trained via risk-seeking policy gradients. We test our method in MuJoCo environments as well as a chemical plant simulator. We show that the discovered rewards can significantly accelerate the training process and achieve similar or better performance than neural network-based algorithms.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; *Inverse reinforcement learning*; *Apprenticeship learning;*

Additional Key Words and Phrases: Interpretable reinforcement learning, imitation learning, interpretable imitation learning, reinforcement learning

ACM Reference format:
Nicolas Bougie, Takashi Onishi, and Yoshimasa Tsuruoka. 2023. Interpretable Imitation Learning with Symbolic Rewards. *ACM Trans. Intell. Syst. Technol.* 15, 1, Article 4 (December 2023), 34 pages.
https://doi.org/10.1145/3627822

Authors' addresses: N. Bougie, NEC-AIST AI Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology, 2-4-7 Aomi, Koto-ku, Tokyo 135-0064, Japan; e-mail: nicolas-bougie@aist.go.jp; T. Onishi, NEC Data Science Research Laboratories, 1753 Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8666, Japan; e-mail: takashi.onishi@aist.go.jp; Y. Tsuruoka, Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan; e-mail: yoshimasatsuruoka@g.ecc.u-tokyo.ac.jp.

2157-6904/2023/12-ART4 $15.00
https://doi.org/10.1145/3627822

1 INTRODUCTION

In recent years, **deep reinforcement learning (DRL)** has achieved great successes in sequential decision-making problems, including game playing and robot control. However, it remains hard to apply DRL to practical problems due notably to its sample inefficiency. Namely, these methods require many training trials for converging to an optimal action policy. In addition, a difficulty in applying DRL to real-world problems is that for many tasks of interest, there is no obvious dense reward function to maximize—dense rewards lead to less frequent updates and reduction in the training time. Therefore, an important challenge of DRL methods is sample efficiency.

Among the current state-of-the-art approaches to improve learning efficiency, a promising direction is imitation learning, a framework of learning an agent's objectives, values, or rewards by observing expert demonstrations. The simplest form of imitation learning is behavioral cloning [34, 63], which aims to clone the provided demonstrations. Another versatile form of supervision is **inverse reinforcement learning (IRL)** [20, 102], which infers the expert cost function that prioritizes entire trajectories over others. More recently, **Generative Adversarial Imitation Learning (GAIL)** [35] has been introduced to infer the expert cost function from expert data by training a discriminator neural network to distinguish the agent and expert behaviors through its observations and actions.

However, most of the previous work on imitation learning is centered around deep neural networks. Although the representational power of **deep neural networks (DNNs)** enables imitation learning in complex and high-dimensional environments, such DNN-based models are "black box" models in nature. In other words, it is hard to explain what knowledge the model has learned and why the expert reward has been assigned. In addition, the discovered cost function cannot be easily tuned or modified by a human. However, if the expert reward is understandable and modifiable, then a human could adjust it and design restrictions to achieve the intended agent behavior. Thus, DNN-based methods are difficult to use in cooperation with a human as it is not possible to understand the reasons behind a reward. The ability to explain what has been learned is important in earning people's trust and developing a robust and responsible system, especially in industrial domains.

In recent years, it has been shown that the lack of interpretability poses a serious problem when humans need to be involved in the decision-making loop [68]. Being unable to explain the reasoning behind black-box decisions is unacceptable for safety-critical systems [48]. DNNs exhibit complex functional forms, involving thousands of non-linear operators and transformations. This complexity poses a significant barrier to the deployment of DRL policies in real-world settings due to the difficulty to understand, verify, trust, and predict the behavior of the **reinforcement learning (RL)** agent. These challenges are especially relevant in industrial applications such as process control (e.g., chemical plants). Many previous studies have been dedicated to making DNNs more interpretable [17, 25, 60]. However, none of them focuses on the explanation of knowledge learned by imitation learning models (e.g., GAIL) in a way that the model can be adjusted, modified, or improved by a human. Perhaps the closest work to ours is that by Pan et al. [64], in which visual explanations are provided via post hoc explanations of a trained GAIL model. Our method, however, directly learns interpretable symbolic rewards that can be understood, verified, and improved by humans. In this work, "verifiable" specifically means that a human can manually validate or confirm the results rather than relying on any form of formal methods or automated verification processes.

In light of this, we propose an interpretable **imitation learning approach (iIL)** that learns symbolic reward functions. Learning interpretable symbolic reward functions has several desirable features. (1) Interpretability: Insights from simple mathematical expressions can often be gleaned by inspecting them. In addition, the learned reward function can be modified or improved by a

human to adjust the agent's behavior. (2) Verifiability: Unlike neural network-based approaches, which can be difficult to reproduce, a symbolic reward is easily verifiable, and the reward function exhibits predictable behaviors. (3) Explainability: The structure of the reward function can be used to extract *explanations*. In contrast with standard IRL approaches that solely provide to the agent a scalar reward, our method provides rich and complex information that explains the decisions of the expert. As a result, the agent can quickly capture important features and dynamics of the system.

We present a two-staged algorithm (iIL) for learning symbolic reward functions. First, we extract an expert cost function from expert demonstrations by employing a (deep) IRL model, which can include any state-of-the-art algorithms. Second, we construct a surrogate symbolic reward. The symbolic reward function is represented by a symbolic tree, where the leaves are input variables and the nodes are simple mathematical operators. To deal with large state–action spaces, the symbolic reward tree is decomposed in a hierarchical fashion, where a high-level tree coordinates a set of low-level trees. At execution, the high-level and low-level trees operate as a single tree—the reward is calculated by starting from the root node of the high-level tree. The trees are learned by training an autoregressive **recurrent neural network (RNN)** via risk-seeking policy gradients. We then learn a policy from that symbolic reward function with reinforcement learning. We further show that the structure of the discovered reward function can be used to extract explanations that encode more complex and richer information than standard scalar rewards, significantly improving the final performance. The experiments reveal that our method can accurately capture the expert cost function in robotic tasks and a chemical plant simulator. Furthermore, we compare the performance of agents trained with our symbolic reward function and deep IRL models and demonstrate that our agent can achieve comparable performance to "black-box" methods. We further show that the explanations extracted from the trees are critical to enhance sample efficiency and final performance. Finally, we discuss how the learned symbolic reward function can be used by a human to understand task-relevant features of the environment, as well as two strategies for *manually* modifying symbolic rewards.

We summarize our contributions as follows:

- A novel interpretable imitation learning method that leverages black-box IRL methods to produce a fully interpretable symbolic reward.
- A RNN-based algorithm that searches the space of symbolic reward functions directly within the RL loop.
- A hierarchical representation of the reward function to enable learning in tasks featuring large state–action spaces.
- A novel technique to provide more complex and richer information than standard scalar rewards—explanations, being extracted by parsing the symbolic tree.

2 RELATED WORK

Our work lies at the intersection of imitation learning and interpretable/explainable AI. This section provides a comprehensive comparison with those studies. Note that for the brevity of the description, we outline together interpretable and explainable methods. We further provide a brief comparison with reward shaping methods. In this article, explainability refers to a technical understanding of the connection between the inputs and outputs of a particular AI model. Therefore, given a generated output, a human is able to understand how it originated from the input features. For instance, in a previous study [61], an explanation is defined as a collection of features of the interpretable domain that have contributed for a given example to produce a decision. Another possible way to provide such an explanation may be to generate a heatmap that highlights the pixels of an input image that had the most influence on the decision [32].

However, interpretability relates to literally explaining what is happening behind the curtain. Thus, interpretability refers to the ability to explain or present results in understandable terms to humans [16]. A similar definition is provided by Rudin et al. [71], which consider interpretability as the ability of intrinsically interpretable models and explainability as the ability to explain models by using post hoc interpretability techniques. Intrinsic interpretability relates to a machine learning model that is constructed to be inherently interpretable or self-explanatory at the time of training by restricting the complexity of the model, including building decision tree policies [3, 56]. These definitions were also employed in recent surveys [26, 97].

2.1 Imitation Learning

Imitation learning has been a successful paradigm for improving the sample efficiency of reinforcement learning agents by leveraging prior knowledge about the task. A major line of work is behavioral cloning [34, 63], in which the agent aims to clone the provided expert demonstrations. Namely, it directly maximizes the likelihood of the expert actions under the training agent policy. In a different spirit, IRL approaches [20, 102] such as GAIL [35] aim to discover a reward or an expert cost function based on the provided demonstration data. The central idea in GAIL is to recover the expert cost function by training a (neural network) discriminator that distinguishes expert trajectories from trajectories of the learned policy. Most of the previous studies in imitation learning employ neural networks to learn from expert data. However, these approaches are not directly applicable to safety-critical applications where interpretability is of utmost importance. Besides, such methods can be challenging to use in cooperation with humans as the discovered expert cost function or reward cannot be straightforwardly tuned, modified, or improved by an expert. Finally, it can be difficult to encode complex and/or multiple pieces of information with a single scalar reward, which differs from human learning that can access complex information signals such as language [22]. In contrast, we present a method that extracts symbolic reward functions from expert data, allowing the use of iIL in safety-critical applications. Moreover, we demonstrate that the structure of the discovered reward function can be used to generate explanations that have richer expressivity than scalar rewards.

2.2 Interpretable Machine Learning

In recent years, a number of approaches have attempted to make deep networks more interpretable. Most methods can be grouped into three broad classes [61, 101]: (i) explaining the learned concepts in the abstract domain of DNN, (ii) explaining the decisions by relevance propagation and estimating corresponding concepts in the input domain, and (iii) leveraging symbolic techniques to explain and interpret a DNN. For instance, some methods [14, 27] in the first category produce logic rules to explain the learned concepts of a DNN. The second type of explanation is the meaning of hidden neurons. The idea is to associate abstract concepts with the activation of some hidden neurons [1, 19, 40, 43, 67, 70]. Finally, the third category generally trains a surrogate model expressed through symbolic expressions that comprise mathematical expressions [2, 38, 80, 90]. The proposed method belongs to the third group but specifically targets some challenges inherent in DRL such as discovering interpretable reward functions and using them to guide the agent's training. To this end, we propose an algorithm to discover mathematical expressions that mimic the expert cost function found by a deep IRL model. A more detailed comparison with interpretable DRL is provided in the next section.

2.3 Interpretable Reinforcement Learning

Several approaches have been proposed to address the interpretability issues inherent in DRL. One line of work is post hoc methods, where these methods are used in addition to the original model

to help users understand the reasons for the decisions. For instance, the user can receive visual explanations that highlight the most relevant regions of the state space [39, 76, 87, 98]. Another solution is to employ an attention mechanism to identify task-relevant information [52]. In this framework, the output of the attention layer is leveraged to identify the most important features of the state space. However, the interpretation of strategic states for real-world applications may not be as simple as the objects in a grid game. However, some methods generate textual explanations for choosing an action [31, 88]. Another study proposed to learn a vector Q-function, where each component explains preferences between actions based on reward decomposition [37]. Another popular post hoc explainer is Shapley Additive Explanations [96], which attributes feature importance to the inputs of a (deep) predictor for a single data sample by "removing" input features and measuring the changes on the output [51]. Hence, in post hoc methods, an explanation can be used to clarify, justify, or explain an action choice. However, such approaches do not offer full interpretability, as they focus on explaining the local reasons for a decision. In a different spirit, the idea of explaining the knowledge learned by imitation learning models (e.g., GAIL) has been employed by Pan et al. [64]. The authors proposed to discover visual explanations via post hoc interpretation of a trained GAIL model. Our method, however, directly learns interpretable symbolic rewards that can be understood and improved by humans. To the best of our knowledge, we propose the first approach that can discover knowledge learned by IRL approaches in a way that the model can be adjusted, modified, or improved by a human

Since achieving full interpretability is very challenging, another line of work focuses on high-level interpretability. In particular, these methods focus on their high-level interpretability, as their lower-level components rarely claim to be interpretable. For instance, in Reference [94], the high-level agent forms a representation of the world and task at hand that is interpretable for a human operator while the low-level employs a neural network [6]. Although such hierarchical approaches can rarely claim full interpretability, they benefit from the flexibility of neural approaches while providing explanations regarding the strategy of the agent [45, 78, 81, 94, 100].

Model approximation is an approach that employs a self-interpretable model to mimic the target agent's policy and then derives explanations from the self-interpretable model for the target DRL agent. For instance, VIPER [3] leverages ideas from model compression and imitation learning to learn decision tree policies guided by a DNN policy. Specifically, they learn a decision tree policy that plays Atari Pong on a symbolic abstraction of the state space rather than from pixels. In a similar spirit, another study [50] introduced Linear model U-trees to approximate a neural network's predictions, using a tree structure that is transparent, allowing rule extraction and measuring feature influence. A similar approach [89] presented **Neurally Directed Program Search (NDPS)**, for solving the challenging non-smooth optimization problem of finding a programmatic policy with maximal reward. NDPS works by first discovering a neural policy using DRL and then performing a local search over programmatic policies that aims to minimize the distance from this neural target policy.

Since most previous work in explainable deep learning focuses on explaining only a single decision in terms of input features, a recent study [86] extends explainability to the trajectory level. They introduced Abstracted Policy Graphs that are Markov chains of abstract states. This representation summarizes a policy so that individual decisions can be explained in the context of expected future transitions. A recent follow-up [5] presented a policy representation via a novel variant of the CART decision tree algorithm. Instead of mimicking the target agent's policy, self-interpretable modeling builds a self-explainable model to replace the policy network. Since the new model is interpretable, one can easily derive an explanation for the target agent [99]. A top-down attention may be employed to direct observation of the information used by the agent to select its actions, providing easier interpretation than of traditional models [62]. Neuroevolution

can also be used for training self-attention architectures for vision-based reinforcement learning tasks [84]. To avoid making any assumptions as to either unbiasedness of beliefs or optimality of policies, INTERPOLE [36] aims to discover the most plausible explanation in terms of decision dynamics and boundaries. In EDGE [30], the authors proposed a novel self-explainable model that augments a Gaussian process with a customized kernel function and an interpretable predictor, capturing both correlations between timesteps and the joint effect across episodes. Our work differs fundamentally from the DRL explanation research mentioned above in terms of the objective pursued. Although the focus of DRL explanation research is typically on developing methods to explain the behavior of a trained agent, our work is focused on discovering interpretable reward functions through expert demonstrations, which can align the agent's behavior with the desired behavior. Consequently, our main goals are to both understand/verify the learned behavior and accelerate the agent's exploration.

Another relevant idea is reward decomposition [37], which decomposes rewards into sums of semantically meaningful reward types, so that actions can be compared in terms of tradeoffs among the types. Decomposing the reward function and seeing the influence of aspects in the reward toward the decision-making process as well as the correspondence between each other is a reasonable way to explainability. To explain skill learning, Shu et al. [78] utilized hierarchical policies that decide when to use a previously learned policy and when to learn a new skill. However, action preferences can then be explained by contrasting the future properties predicted for each action [47]. For multi-agent tasks, a method called counterfactual multi-agent policy gradient utilizes a counterfactual advantage function to perform local agent training [21]. Nevertheless, this method ignores the correlation and interaction between local agents, leading to poor performance on more complex tasks. Recently, Wang et al. [92] combine the Shapley value with the Q-value and perform reward decomposition at a higher level in multi-agent tasks to guide the policy gradient process, allowing us to explain how the global reward is divided during training and how much each agent contributes. However, the present study seeks to guide learning while providing reward interpretability in sparse reward tasks. In such tasks, reward decomposition may be difficult to apply as the extrinsic reward is zero for most of the timesteps. To alleviate this challenge, we seek to discover dense symbolic rewards from expert trajectories rather than decomposing extrinsic rewards.

In a different spirit, when inputs are high-dimensional raw data, one solution is to extract symbolic representations on which a human can reason and make assumptions. Since such methods tend to abstract away irrelevant details, the reasons for a decision can be quickly and effectively understood by humans. For instance, a few methods have proposed to distill DRL into decision trees [18, 46]. An expert may also bootstrap the learning process, as shown by Silva et al. [79], where the policy tree is initialized from human knowledge. Some previous studies [23, 24] proposed to learn a symbolic policy by learning a relevant symbolic representation prior to using Q-learning. The symbolic representation includes interactions between objects in the environment. However, it is not clear how to apply such methods to tasks without well-defined *objects* such as in process industries or robot control. In Soft Decision Tree [12], the authors proposed to train a soft decision tree to mimic the action classification of a DRL policy. The resulting soft decision trees provide a form of interpretability of how the policy operates. A well-known framework for learning symbolic policies is genetic programming [58]. In GPRL [33], a genetic programming model is trained to discover a symbolic policy on an environment approximation referred to as a world model. In this work, we rely on an autoregressive RNN to learn a symbolic reward function from expert data and show that the learned reward can be used to guide the agent's training. One common problem when learning symbolic policies is that the performance is capped by the policy being imitated. This is because most of the approaches learn symbolic policies as a classification problem—without

interacting with the environment. However, we focus on learning symbolic reward functions from expert data. Namely, the proposed approach differs from most of the previous work in that we learn interpretable symbolic reward functions from expert data rather than modeling the agent's policy. Since our agent still has access to the environmental reward, the performance is not capped by the quality of the demonstrations. Besides, using expert data provides a significant speedup in the training process, while highlighting the most important features in the environment.

A study closely related to ours employs genetic programming to mimic the rewards provided by the environment [77]. Namely, they use a genetic programming model to clone the rewards received by the agent. However, we argue that expert data can provide more meaningful explanations as they are likely to cover task-relevant regions of the environment. In addition, the discovered reward function can be used to accelerate the agent's training, and the expert can directly modify and improve the learned reward function. Furthermore, rather than learning reward functions with genetic programming, we propose to employ an auto-regressive RNN that tends to perform better on high-dimensional and complex data.

2.4 Symbolic Regression in Machine Learning

Symbolic regression is a search technique that seeks to discover symbolic expressions. As mentioned above, the standard paradigm for symbolic regression is genetic programming, where a population of trees is evolved based on a fitness function. However, to deal with high-dimensional problems, it may be possible to employ DNNs for symbolic regression [104].

This idea of employing a RNN to generate symbolic expressions has been previously employed in the field of neural architecture search [66, 104]. In particular, a RNN produces a sequence of tokens that represents the architecture and activation functions of a RNN. However, to the best of our knowledge iIL is the first attempt to utilize an autoregressive RNN to learn symbolic reward functions. To overcome challenges inherent in large and complex inputs, we further propose a hierarchical decomposition of the reward function (see Section 3.2.1). A few approaches have applied the idea of symbolic regression to recover mathematical expressions from a dataset using a neural network [72] that emits batches of expressions as a sequence of mathematical operators. In a prior study [42], the authors directly learn a symbolic policy and propose an *anchoring* algorithm to deal with multi-dimensional action spaces. Although these methods have achieved significant results in some tasks, neural network–based methods still remain superior in terms of final performance. Besides, it is not clear how to scale such symbolic policies to large and complex tasks. However, we propose to learn a surrogate symbolic reward function that offers several advantages: (1) our method does not decrease the final performance, (2) the symbolic reward can be used to guide the agent's training, and (3) the discovered reward can be used to identify meaningful features of the environment and explain the expert's intention.

2.5 Reward Shaping

Reward shaping methods involve designing a reward function that encourages the agent to take actions that lead to a desired behavior. This involves modifying the original reward function with a shaping reward that incorporates domain knowledge. The additional reward terms can be either hand-crafted or learned from data and are designed based on the domain knowledge of the problem. Early work of reward shaping [15, 69] focused on crafting the shaping reward function, but did not consider that the shaping rewards may change the optimal policy. Besides the shaping approaches mentioned above, other important approaches of reward shaping include the automatic shaping methods [29, 55], multi-agent reward shaping [13], and some recent ideas such as ethics shaping [95], belief reward shaping [54], and reward shaping via meta-learning [105]. Mirchandani et al. [57] proposed a reward-shaping method that leverages interactions between an agent

and a human to shape sparse rewards associated with human instruction goals and the current state of the environment. Specifically, their method utilizes termination and relevance classifiers to shape the reward signal. Similarly, Tabrez and Hayes [83] presented a framework called Reward Augmentation and Repair through Explanation, which employs partially observable Markov decision processes to approximate the understanding of collaborators in joint tasks. However, the framework requires careful consideration of the tradeoff between reward modification and abandonment, and it may be limited by the complexity of the POMDP approximation. In addition, reward shaping can also suffer from several limitations. One of the main challenges is that the additional reward terms must be carefully designed to avoid overfitting to the specific problem domain. This can be a difficult and time-consuming task, and it can limit the generalization of the approach to other problem domains. Besides, the use of reward shaping can introduce human bias in the learning process, which can lead to suboptimal solutions [44]. Instead of investigating how to learn helpful shaping rewards, our work studies a different problem where an exploration reward is discovered from *expert trajectories*. The proposed exploration reward can be viewed as a form of reward shaping that distills valuable priors about the domain from expert trajectories. One key advantage is that this framework does not require any handcrafted rewards or additional labeled data for discovering the exploration bonus. In addition, since the exploration reward is represented as a symbolic tree, it can be understood and verified by humans, providing a form of explainability.

In recent years, curiosity [65] has also been proposed as a form of reward shaping. The intrinsic motivation measure may include mutual information between actions and end states [28], surprise [65], state prediction error [9], learned skills [8], state visit counts [82], empowerment [73], or progress [7]. Curiosity seeks to accelerate exploration by providing an additional intrinsic reward. The present work differs from curiosity-driven learning in that it attempts to learn complex reward functions through the use of expert priors, rather than replacing reward with a fixed intrinsic objective that aims to encourage exploration of the entire state–action space. Additionally, unlike curiosity-driven approaches that encode rewards primarily through deep neural networks, our goal is to discover interpretable reward functions. By doing so, we aim to achieve a higher level of transparency and understanding in the learning process, allowing for better analysis and interpretation of the training objective.

3 METHOD

The central idea behind iIL is to discover symbolic rewards from expert data and then extract a policy from that symbolic reward function with reinforcement learning. As illustrated in Figure 1, our approach involves two main stages: (1) recovering the expert's cost function with (deep) inverse reinforcement learning and (2) discovering a surrogate symbolic reward function by cloning the expert's cost function. Then, a policy is trained with reinforcement learning guided by the reward and explanations extracted from the symbolic reward function.

3.1 Recovering the Expert's Cost Function

The first stage involves a deep IRL model that is trained to recover the expert cost function. To do so, we assume access to M trajectories $\{(s_0, a_0), (s_1, a_1), \ldots, (s_M, a_M)\}$ of state–action pairs (s, a) that were collected by observing an *expert* attempting to achieve the goal being pursued in the task.

In this work, we employ the well-known GAIL model [35]. At its core, GAIL aims to recover an estimate of the reward signals underlying the Markov Decision Process from the expert's demonstration. Since directly recovering reward functions from expert data is often intractable in high-dimensional state–action spaces, it turns the inverse reinforcement learning problem into

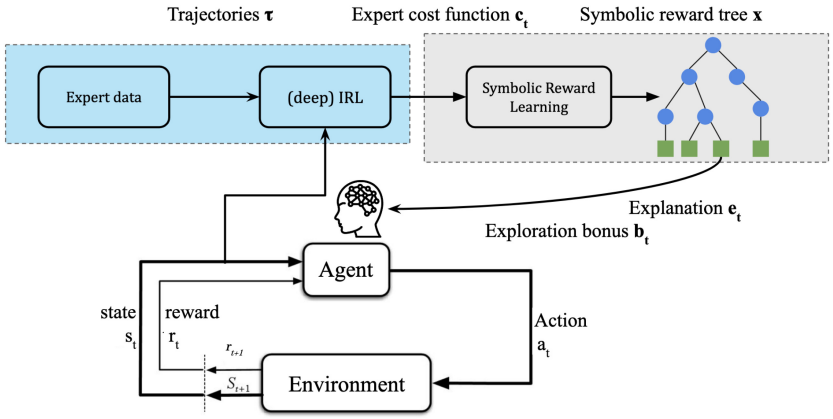


Fig. 1. Overview of the iIL method. In the first stage, an inverse-based model recovers the expert cost function based on an expert dataset and transitions experienced by the agent. In the second stage, a symbolic reward function represented as a symbolic tree is learned. Finally, the symbolic tree is used to guide the agent with an exploration reward b and an explanation e. Consequently, along with the extrinsic reward the agent tries to maximize the weighted sum of the exploration and extrinsic reward, distilling the expert behavior via the exploration bonus.

its equivalent dual problem of occupancy measure matching. Thus, the agent seeks to match the distribution of state–action transitions generated by its own policy to the distribution generated by the expert's *policy*. After recovering the expert cost function with GAIL, we utilize it as a surrogate objective for discovering symbolic reward functions, as described in the following sections. In contrast with the original GAIL method that directly trains the agent to maximize the expert cost function, in iIL the agent learns to maximize the sum of the extrinsic and symbolic reward.

To recover an estimate of the reward signals, GAIL trains a discriminative classifier to distinguish between state–action pairs generated from the agent's policy and state–action pairs generated from the expert's policy. In practice, a discriminator D_ϕ is trained to distinguish agent transitions $(s, a) \sim \pi_{agent}$ from expert transitions $(s, a) \sim \pi_{expert}$. The discriminator D_ϕ is trained via gradient descent to minimize the following loss with respect to its parameters ϕ:

$$\mathcal{L}_{GAIL} = \mathbb{E}_{(s,a)\sim\pi_{agent}}[\log(D_\phi(s,a))] + \mathbb{E}_{(s,a)\sim\pi_{expert}}[\log(1 - D_\phi(s,a))]. \tag{1}$$

In iIL, we use as the expert cost c the output of the fitted discriminator: $c = D_\phi(s, a)$, where (s, a) is a state–action transition. The agent transitions used to train the discriminator are progressively collected throughout the training process (see Algorithm 2) in Section 4). Then we construct a dataset $\mathcal{D}$ of all state-action-cost tuples (s_i, a_i, c_i), which contains both policy and expert transitions augmented with the expert cost c_i discovered by GAIL,

$$\mathcal{D} = \{(s_0, a_0, D_\phi(s_0, a_0)), (s_1, a_1, D_\phi(s_1, a_1)), \dots\} = \{(s_0, a_0, c_0), (s_1, a_1, c_1), \dots\}. \tag{2}$$

3.2 Discovering a Symbolic Reward Function

Now that we have described how the expert cost function is recovered, how does it translate to our symbolic reward function? In the second stage, to discover a symbolic reward, we train a surrogate model to emulate the expert cost function. Namely, we propose to use an auto-regressive recurrent generator network. The generator network models a distribution over symbolic operators, inputs, and constants (Figure 2). Each operator is a simple mathematical function such as *max* or *sum*. The operators are selected from a library O, where the operators can be unary, binary, or ternary.

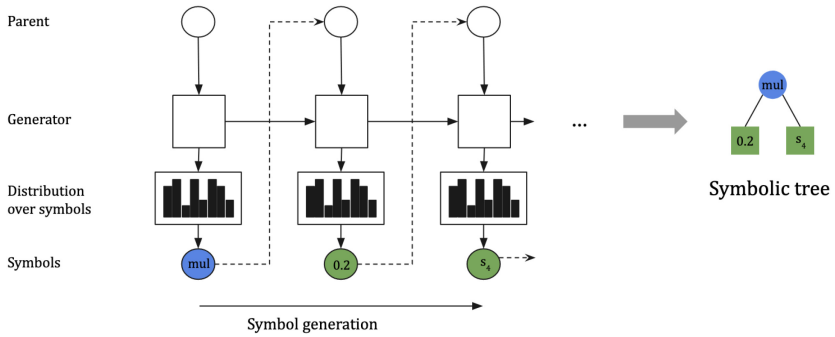


Fig. 2. Overview of symbolic tree generation. We present a simple tree generation, comprising two leaves and one node. An auto-regressive recurrent generator receives as input a representation of the parent token, and outputs a distribution over the possible symbols. The produced symbol is then added to the symbolic tree. This process generates a simple symbolic reward function: $0.2 \times s_4$.

$\mathcal{O}$ includes simple arithmetic functions as well as logic operators and constants. A description is provided in the supplementary information. For the brevity of description it does not include constants.

The set of tokens produced by the generator network can be viewed as a symbolic reward tree, a type of tree in which internal nodes are mathematical or logical operators, and the leaves are input variables or constants. The input variables are denoted as s_j, where $1 \leq j \leq q$, q are the number of input variables. A tree $x : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$ maps an observation of the observation space $\mathbb{S}$ and an action of the action space $\mathbb{A}$ to a scalar reward $b \in \mathbb{R}$.

In detail, the generator network parameterized by θ sequentially emits a categorical distribution over the operators in $\mathcal{O}$. For the ease of the notation, a tree x is represented by a sequence of tokens $x = \{x_1, \ldots, x_T\}$, where x_i is the token at the ith position and T is the maximum tree size. The tokens are emitted sequentially while building the corresponding tree. Namely, each time a token is provided by the generator, it is added to the symbolic tree. The tree is obtained by using a pre-order traversal visit of the nodes and leaves. The process repeats until the symbolic tree is complete (i.e., all branches are terminal nodes) or the tree reaches the maximum length allowed. To capture the structure of the tree, we feed a representation of the parent token as the input to the RNN. By doing so, each token sampled by the RNN takes into account the previous tokens. The likelihood for the ith token of the traversal is given by

$$p(x, \theta) = \prod_{i=1}^{|x|} p(x_i \mid x_{1:(i-1)}; \theta), \tag{3}$$

where $p(x_i \mid x_{1:(i-1)}; \theta)$ is the likelihood for the ith token to be emitted by the generator given the previously emitted tokens $x_{1:(i-1)}$.

The trees generated by the RNN are evaluated on the dataset $\mathcal{D}$ that has been collected in the first stage. Given a candidate tree x, we compute the mean-square error for each example $(s, a, c) \sim \mathcal{D}$ in the dataset as

$$MSE = \frac{1}{|\mathcal{D}|} \sqrt{\sum_{i=1}^{|\mathcal{D}|} (c_i - x(s_i, a_i))^2}, \tag{4}$$

where $x(s_i, a_i)$ is the reward produced by the tree x given the transition (s_i, a_i). We can use the prediction error R as the training signal for updating the RNN: $R(x) = \frac{1}{1+MSE}$.

Since R is non-differentiable with respect to θ, we rely on REINFORCE [93] to optimize the generator. The list of tokens that the generator produces can be viewed as a list of actions $a_{1:T}$ to design a symbolic reward function. The generator can therefore be trained with reinforcement learning with R as the reward signal. Note that in the remainder of this article, to avoid confusion with the symbolic reward we refer to R as the *training signal*. To prevent premature convergence, we add the weighted generator's sample entropy to the training signal R. The policy gradient objective $J(\theta)$ maximizes the expectation of a training signal $R(x)$ under symbolic rewards from the policy

$$J(\theta) = \mathbb{E}_{x\sim p(x|\theta)}[R(x)]. \tag{5}$$

The REINFORCE policy gradient can be used to maximize this expectation via gradient ascent,

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim p(x|\theta)}[\nabla_\theta \log p(x \mid \theta) R(x)], \tag{6}$$

where an empirical estimation of this quantity can be computed over a batch of m sample expression,

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{k=1}^{m}[\nabla_\theta \log p(x^{(k)} \mid \theta) R(x^{(k)})], \tag{7}$$

where m is the number of symbolic trees that the generator samples in one batch and $x^{(k)}$ is the kth tree.

The above update is an unbiased estimate for our gradient but tends to have a high variance. To reduce the variance, it is a common practice to employ a baseline function $\bar{b}$ that is subtracted from the training signal $R(x^{(k)})$,

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{k=1}^{m}[\nabla_\theta \log p(x^{(k)} \mid \theta)(R(x^{(k)}) - \bar{b})]. \tag{8}$$

As long as the baseline $\bar{b}$ is not a function of the current batch of trees, the gradient estimate remains unbiased. In this article, we use a moving average baseline of the previous training signals.

3.2.1 Hierarchical Symbolic Trees. One possible issue with the above algorithm is learning large trees. The complexity of real-world tasks often requires building large trees to capture the underlying complexity of the expert cost function, which may cause the generator to be slow to train. To overcome this problem, we introduce a hierarchical decomposition of the tree learning problem. The central concept is to learn a set of K *shallow* trees that focus on different aspects of the expert cost function and coordinate their outputs with a high-level tree (see Figure 3). The idea of hierarchical symbolic trees is primarily designed to enable the swift discovery of large trees, as at execution we construct a single tree x that encapsulates both the high-level and low-level trees. To do so, we replace the high-level tree's leaves that represent the output of a low-level tree with the associated symbolic trees.

We consider a high-level generator that organizes a set of K low-level trees, $\{x_1, \ldots, x_K\}$. Each low-level tree is learned by a low-level generator. The high-level tree's leaves are input variables or the output of a low-level tree. However, the low-level trees' leaves are input variables or constants. The hierarchical learning problem is to simultaneously learn the *HI*-level generator g_{HI}, called a *HI*-generator, as well as the *LO*-level generators $\{g^1_{LO}, \ldots, g^K_{LO}\}$. The aim of the learner is to achieve a high training signal (in the sense of REINFORCE) when the *HI*-generator and the low-level generators are run together.

In practice, we use a two-staged optimization—we alternate between optimizing the g_{HI} network and g_{LO} networks (Algorithm 1). During the first stage, the *HI*-level generator is trained by using the best low-level trees that have been previously discovered by the *LO*-level generators. During

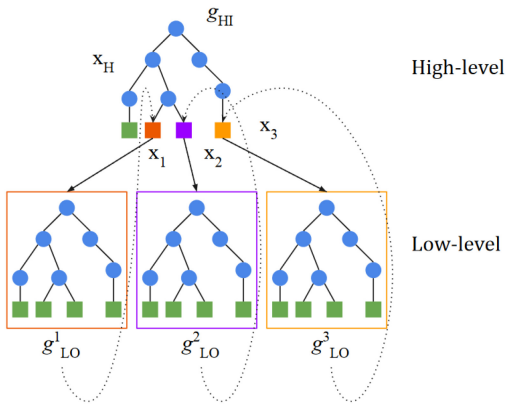


Fig. 3. An illustration of the hierarchical symbolic reward generation scheme. The *HI*-level generator g_{HI} is trained to *organize* the low-level trees $\{x_1, x_2, x_3\}$ learned by the low-level generators $\{g^1_{LO}, g^2_{LO}, g^3_{LO}\}$. The final symbolic reward is based on the joint evaluation of the high-level x_H and low-level $\{x_1, x_2, x_3\}$ trees.

the second stage, the *LO*-level generators aim to discover new solutions that will improve the global prediction based on the best *HI*-level tree discovered so far. The process continues until the end of the learning or an optimal solution is found. Note that at the beginning of the training, the initial *best* trees are randomly sampled by the corresponding generators. By doing so, the algorithm can incrementally improve the overall symbolic reward function—this can be viewed as progressively building the branches of the global tree. It should be emphasized that since the training of *LO*-level generators can be done concurrently, the method can result in a running time similar to that of a single *LO*-level generator.

3.3 Training RL Guided by the Symbolic Reward Function

Now that we have described how the symbolic reward function is learned, we can train any on-policy or off-policy method such as **Proximal Policy Optimization (PPO)** [75] to select action sequences that explore states for which the symbolic reward function is large. In other words, we would like the agent to explore states surrounding expert trajectories.

To do so, we evaluate the symbolic tree x given the current state and action (s_t, a_t) to generate an exploration reward b_t, which is further summed up with the task-reward r_t to give an augmented reward $\hat{r_t} = r_t + \beta b_t$, where β is a hyperparameter. The augmented reward has a nice property from the reinforcement learning point of view—it is a dense reward that smoothly encourages the agent to take actions toward the final goal. In addition, as the exploration bonus distill the expert's intentions, it can be used to effectively guide the agent's exploration and provide meaningful priors about the task. As a result, learning with such a reward is significantly faster and often leads to better final performance in terms of the cumulative task reward.

3.4 Explaining the Reward to the Agent

In addition to the exploration reward b_t, we propose to guide the agent's training with explanations. One issue in RL, and especially IRL, is the nature of the rewards received by the learner. The learning agent is given a single scalar value of reward at each timestep that encodes a single piece of information such as *the agent has reached the goal*, *the agent hit a wall*, and so on. Although multiple bits of information can be encoded by summing up the associated rewards or providing multiple rewards, it can throw away vital information, lead to incorrect solutions, and prevent

ALGORITHM 1: Hierarchical training of the *HI* and *LO*-level generator networks.

Initialize g_{HI} and $\{g^1_{LO}, \ldots, g^K_{LO}\}$
Sample initial best high-level tree $x^*_H \sim g_{HI}$
Sample initial best low-level trees $\{(x^*_1 \sim g^1_{LO}), \ldots, (x^*_K \sim g^K_{LO})\}$
for t=0, …,T steps **do**
  Generate a batch $\mathcal{B}$ of *HI*-level trees $\{x_H, ...\} \sim g_{HI}$
  Evaluate the expected training signal for each tree $R(x_H) \leftarrow evaluate(x_H, x^*_1, \ldots, x^*_K)$
  Select the best tree x^{**}_H in $\mathcal{B}$
  if $R(x^{**}_H) > R(x^*_H)$ **then**
    Store the new best tree $x^*_H \leftarrow x^{**}_H$
  end if
  Train the generator g_{HI} on batch $\mathcal{B}$
  for k=0, …,K **do** ▷ Training of low-level generators is done in parallel
    Generate a batch $\mathcal{B}$ of *LO*-level trees $\{x_{LO}, ...\} \sim g^k_{LO}$
    Evaluate the expected training signal for each tree $R(x_{LO}) \leftarrow evaluate(x^*_H, x^*_0, \ldots, x_{LO}, \ldots, x^*_K)$
    Select the best tree x^{**}_{LO} in $\mathcal{B}$
    if $R(x^{**}_{LO}) > R(x^*_k)$ **then**
      Store the new best tree $x^*_k \leftarrow x^{**}_{LO}$
    end if
    Train the generator g^k_k on batch $\mathcal{B}$
  end for
end for

the agent from directly understanding how and why the environment has been affected by its actions. However, one might notice that the structure of the symbolic reward—a symbolic tree, can be naturally parsed to explain the underlying reasons for the reward. We call this parsing an *explanation* and we augment the agent's input with such an explanation. As a result, the agent can quickly capture the important features and underlying factors that affect the reward function, significantly reducing the training time.

3.4.1 Generating Explanations. An explanation is a vector e that contains the evaluation of certain sub-trees (e.g., the branches) of the reward tree x, $e = [v_1, \ldots, v_P]$, where v_1 corresponds to the evaluation of a sub-tree of the reward tree. For instance, v_1 may represent whether the input s_9 is larger than $\frac{s_1}{s_8}$ or whether *the temperature in the distillation column of a chemical plant is larger than 20 degrees*. Thus, each sub-tree partially explains a reason underlying the symbolic reward at different levels of abstraction, depending on the sub-tree being evaluated.

The vector e is filled by using a pre-order traversal (i.e., by visiting each node depth-first, then left-to-right) and evaluating the value of the nodes. To ensure that only relevant attributes are kept, we store the values of the nodes with a depth d at most. As a result, the agent becomes aware of the reasons for the reward. In future work, we anticipate using more sophisticated algorithms to identify the most task-relevant sub-trees for generating explanations. An example of the tree parsing is shown in Figure 4.

3.4.2 Learning from Explanations. Let us define e_{t-1} to be the explanation generated at time $t-1$. To capture temporal information the present work relies on an aggregator function that is implemented as a RNN, a **Gated Recurrent Unit (GRU)** [10], which is a variant of a RNN. Since temporality is critical in the context of explanations, introducing a RNN into DRL networks

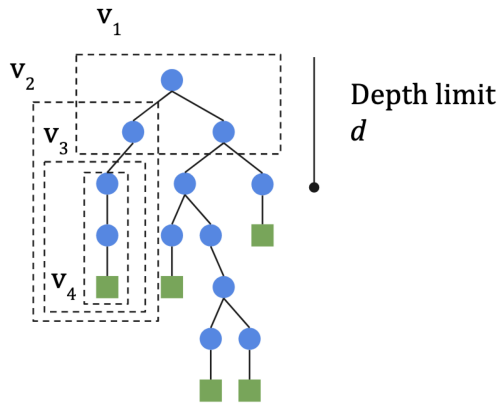


Fig. 4. An illustration of the explanation extraction. In this example, we only display four sub-trees v_1, v_2, v_3, and v_4 being contained in the first branch.

to capture temporal dependencies is an efficient solution to improve practicability of explanations. We refer to the explanation that is perceived by the policy π_{agent} at time t as $\bar{e}_t$. First, $\bar{e}_t$ is obtained by feeding the previous explanation e_{t-1} and the previous hidden states h_{t-2} to a RNN. We define $\bar{e}_t$ as the previous hidden state h_{t-1} outputted by the RNN, as follows:

$$\bar{e}_t = h_{t-1} = GRU(h_{t-2}, [\phi(e_{t-1})]), \tag{9}$$

where h_{t-1} is the hidden states at time $t-1$ and ϕ is an embedding function that embeds the explanation e_{t-1} to a latent space.

Then, $\bar{e}_t$ is used to condition the input of the policy π_{agent}. Namely, the policy parameters θ_π are obtained by maximizing (augmented) rewards with the following constraints for action generation:

$$a_t = \pi_{agent}(o_t, \bar{e}_t; \theta_\pi), \tag{10}$$

where a_t is the action selected by the agent at time t and o_t is the current observation.

4 EXPERIMENTS

Environments. Experiments are conducted on two sets of environments. First, we evaluate the proposed method on robotic tasks implemented in MuJoCo [85]. We consider the following four tasks: (1) Half Cheetah, (2) Hopper, (3) Walker 2d, and (4) Ant. The relative simplicity of this environment allowed us to measure the performance of the present method as well as the quality of the symbolic rewards and explanations. To further verify that the proposed method can scale to complex control tasks, we also conducted an evaluation on four hand manipulation tasks: (1) Hand Manipulate Block, (2) Hand Manipulate Egg, (3) Hand Manipulate Pen, and (4) Hand Reach. In those tasks, the agent is trained to manipulate physical objects via a humanlike robot hand. In the second set of experiments, we evaluate our framework on significantly more challenging tasks that feature sparse rewards, large state–action spaces, and complex dynamics. Namely, the experiments are conducted on a simulator of a **vinyl monomer acetate (VAM)** plant [53], a type of chemical plant widely used in process industries. This set of experiments aims to demonstrate that iIL can be used on real-world industrial problems. The agent seeks to control the chemical plant under disturbances, which entails returning to steady conditions to maximize the production load.

VAM plant The VAM plant reflects the characteristics and practical problems of real process plants. The simulator is composed of eight components for materials feeding, reacting, and

recycling. The process is observed via 109 sensors that measure the volume, flux, temperature, concentration, and pressure of the chemical substances. To complete the task, the agent has to (1) avoid failures in equipment that can be triggered by disturbances; (2) stabilise and correct internal or external disturbances—recover from disturbances; and (3) maintain the process in a steady state. The environment includes 19 disturbance scenarios that can be used to evaluate these properties.

We selected the following disturbance scenarios:

- Change Feed Pressure AcOH ("Pressure AcOH"): The raw material acetic acid feed composition is changed due to condition changes of the acetic acid plant. This disturbance can be observed by detecting changes in the raw material feed and water composition. The intensity level varies randomly between $[1, 50]$. The agent controls the PIDs: PC130, LC130, FC130, FC170, and PC210.
- Change Feed Pressure C2H4 ("Pressure C2H4"): The raw ethylene feed pressure is changed due to condition changes of the ethylene plant. This disturbance can be observed by detecting changes in the raw material flow rate. The intensity level varies randomly between $[70, 140]$. The agent controls the PIDs: PC130, LC130, FC130, and TC150 ("Pressure C2H4-4").
- Day and Night ("Day/Night"): A day and night cycle leads to atmosphere changes, resulting in non-steady conditions and fluctuations in internal temperatures. This disturbance can be observed by detecting changes in cooling water consumption and temperature sensor values. The intensity level varies randomly between $[1, 50]$. The agent controls the PIDs: PC130, LC130, FC310, TC150, and FC170.
- Heavy rain ("Rain"): The heat dissipation for the atmosphere is increased at all heaters due to cooling by heavy rain. The intensity level varies randomly between $[1, 50]$. The agent controls the PIDs: PC210, FC501, TC501, FC130, and TC201.

Concretely, the *state* space consists of the sensor readings. The *action* space consists of a set of PIDs to control—these PIDs are selected based on their relevance with the scenario. The ranges of actions are defined as $[-x_1\%,+x_2\%]$ from the initial values. In our experiments, we set $x_1 = 0.60$ and $x_2 = 1.35$. The agent interacts with the environment once every minute for 60 virtual minutes, which corresponds to one episode. In the absence of domain knowledge and to replicate real-world problems where rewards are naturally sparse, a general-purpose choice is to set the *reward* function as

$$r(x, x_t) = \begin{cases} 1.0 & if(\mid x - x_t \mid)/(x_s) < \epsilon) \\ 0.0 & otherwise \end{cases}, \tag{11}$$

where x is the current state, x_t is the target state, x_s is a steady state value, and ϵ is a threshold value. In practice, we set $x_t = x_s$ and $\epsilon = 0.01$.

Experimental settings. We choose the commonly used PPO algorithm as our basic RL algorithm. The actor and critic networks consist of three fully connected layers with 128 hidden units. Tanh is used as the activation function. Training is carried out with a fixed learning rate of 7.0×10^{-4} using the Adam optimizer [41], with a batch size of 128. The policy is trained for four epochs after each episode. To facilitate the learning of agents, we use an observation normalization scheme. That is, we whiten each dimension by subtracting the running mean and then dividing by the running standard deviation.

The dataset of expert trajectories consists of $M = 10^4$ transitions recorded by observing an expert controlling the agent. To ensure reproducibility, the expert is a teacher policy (PPO) that has been trained to achieve a near-optimal solution. GAIL uses the same hyperparameters as in the original implementation. The agent transitions are sampled uniformly from a policy buffer consisting of 10^6 transitions. The dataset D is balanced as we stored the same number of expert and

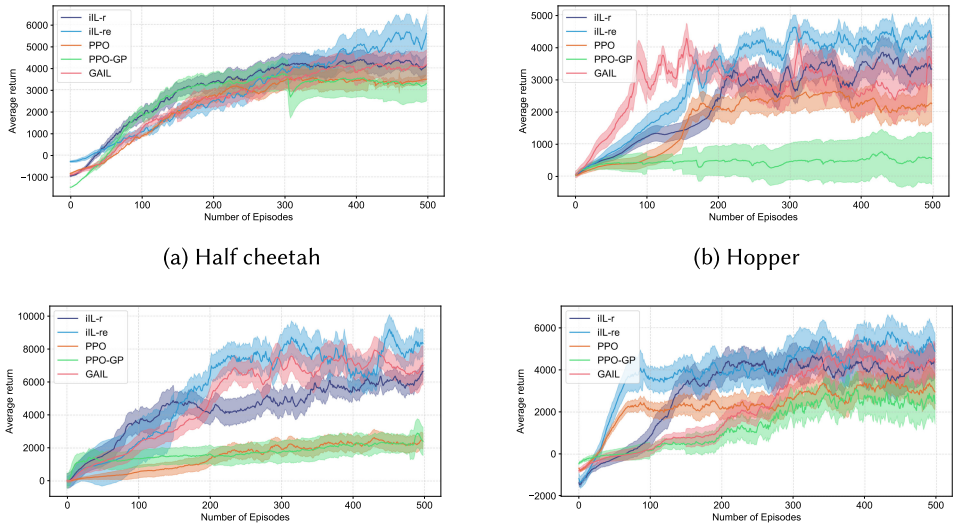
ALGORITHM 2: Pseudo-code of the update-rule of the models.

```
Fill expert buffer D^expert with expert transitions
Initialize policy buffer D^policy by running the policy
Initialize D ← ∅
while true do
    Recover the expert cost function based on D^expert and D^policy        ▷ Stage 1
    Construct the dataset D → {(s_0, a_0, c_0), (s_1, a_1, c_1), ...}
    Discover a surrogate symbolic reward function from D                   ▷ Stage 2
    Train the agent policy π_agent for H epochs
    Store agent transitions in D^policy
end while
```

agent transitions augmented with the expert cost. The generator networks are RNNs comprising an LSTM layer with 64 hidden units. Training is carried out with a fixed learning rate of 10^{-3} using the Adam optimizer [41], with a batch size of $m = 256$. The list of operators can be found in the supplementary information. In addition to these operators, the tree could contain the following constants: $\{0.1, 0, 1, 0.5, 10\}$. Only the high-level tree uses logic operators. Since a tree takes as input a state and action, the input variable s_j denotes the element at index j of the tuple (s, a). We set the maximum length of high-level trees to 40 and 15 for low-level trees, and $K = 5$. We limit the depth of the node evaluated during the explanation extraction to $d = 3$. The GRU network that encodes the explanations features 32 hidden units and takes as input the embedded representation of an explanation passed through a fully connected layer of size 32. We set $\beta = 0.3$ and the exploration reward is normalized by dividing it by a running estimate of the standard deviations of the exploration returns. To prevent premature convergence, we also use a tanh constant of 2.5 and a temperature of 5.0 for the sampling logits [4] and add the generator's sample entropy to the training signal weighted by 0.0003. Participants involved in the ablation studies had backgrounds unrelated to robot control and process control, but were provided a brief description of the task prior to conducting the experiments.

To adapt the symbolic reward to novel situations encountered by the agent, we recompute the expert cost function and fine-tune the generators every $H = 50$ epochs. This is because in GAIL the training of the deep inverse model (stage 1) and that of the agent are interleaved. Therefore, we periodically update the expert cost function (stage 1) and then fine-tune the symbolic reward function (stage 2). The pseudo-code of this update-rule is shown as Algorithm 2. Note that repeating stages 1 and 2 may not be necessary depending on the choice of the deep IRL model that recovers the expert cost function from demonstration data.

Baseline methods. The simplest baseline for our approach is just the basic RL algorithm (i.e., PPO) applied to the task reward. As suggested by some prior work and our experiments, this is a relatively weak baseline in the tasks where the reward is sparse. As the second baseline, we take the state-of-the-art IRL method GAIL combined with PPO. The agent trained with GAIL learns to maximize the sum of the extrinsic reward and expert cost function c. The cost function is defined as the output of the fitted discriminator, $c = D_\phi(s, a)$, where (s, a) is a state–action transition, as described in Section 3.1. This comparison is of interest, since GAIL leverages neural network-based rewards (expert cost function) to guide its agent's exploration, while in iIL the agent is guided with symbolic rewards. We also introduce another baseline based on Reference [77] (PPO-GP), where the symbolic reward function is learned via the genetic programming method described in the article. Finally, we compare iIL that uses rewards to guide the agent (iIL-r), and the same setting augmented with the explanations provided by our method (iIL-re).



(a) Half cheetah (b) Hopper

(c) Walker 2d (d) Ant

Fig. 5. Performance of different imitation learning algorithms and DRL baselines on MuJoCo tasks. All methods are tested with 10 random seeds.

4.1 Robotic Tasks

We first perform experiments on four different robotic tasks built on top of MuJoCo: (1) Half Cheetah, (2) Hopper, (3) Walker 2d, and (4) Ant. As shown in Figure 5, our method can learn comparable or superior policies, which entails that the discovered symbolic rewards are relevant to the agent's learning. As expected, iIL-r ends up reaching similar final performance with GAIL. However, our method has a slightly faster convergence rate. One possible explanation is that iIL tends to abstract away irrelevant details and remove noise, compared to GAIL, which may overfit the demonstration data. In addition, we can observe that the extracted explanations provide a significant speedup (iIL-re) in the training process. This is because the agent has access to the underlying factors affecting the reward function, conveying richer insights than scalar rewards. Besides, explanations can be viewed as a form of feature engineering where the novel features are automatically extracted from demonstration data, providing additional prior assumptions about the domain to the agent. Overall, this experiment highlights that iIL drastically reduces the training time in environments with sparse rewards and demonstrates that the proposed approach can achieve similar performance to (non-interpretable) NN-based IRL models.

4.2 ShadowHand Robotic Tasks

Next, we tried to verify that our method can scale to more complex robot control tasks, trained as before with the same set of symbols. Experiments are conducted on four robotic tasks implemented in MuJoCo, where the agent learns to manipulate physical objects via a humanlike robot hand: (1) Hand Manipulate Block, (2) Hand Manipulate Egg, (3) Hand Manipulate Pen, and (4) Hand Reach. The performance metric we use is the percentage of goals that the agent is able to reach. An episode is considered successful if the distance between the agent and the goal at the end of the episode is less than a threshold defined by the task. In Table 1, we can observe that our strategy helps to greatly improve final performance, indicating that meaningful symbolic rewards can be captured. The results further demonstrate that using explanations is beneficial for accelerating

Table 1. Learning Hand Control in MuJoCo

	Percentage of goals achieved			
Method	Hand Manipulate Block	Hand Manipulate Egg	Hand Manipulate Pen	Hand Reach
iIl-r	0.75±0.04	0.74±0.07	0.58±0.06	**0.84±0.06**
iIl-re	**0.78±0.02**	**0.79±0.09**	**0.62±0.04**	**0.84±0.07**
PPO	0.32±0.08	0.49±0.05	0.02± 0.03	0.53±0.07
PPO-GP	0.28±0.09	0.47±0.09	0.03± 0.04	0.70±0.07
GAIL	0.75±0.11	0.72±0.07	0.59± 0.08	0.82±0.05

Results are averaged over 10 random seeds (±std). No seed tuning is performed. Bold values indicate the best performing method.

the agent's training. The superior performance of iIL can be attributed to its ability to leverage expert knowledge that captures the underlying structure of the task, as well as dense rewards that accelerate the learning process. In addition, iIL manages to solve some highly challenging tasks on which the other methods fail to get any reward.

4.3 Chemical Plant Control under Disturbances

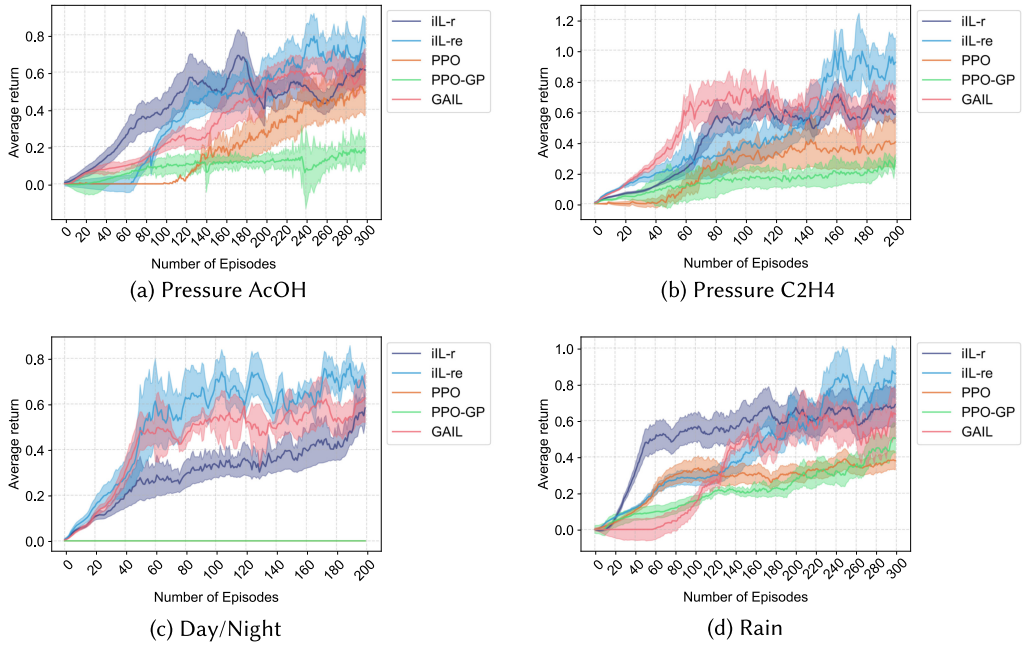
Second, we evaluate the proposed method on a set of tasks from the VAM environment, which replicates the features and problems of real-world process control plants. In process industries, explainability is critical to enable cooperation with human operators as well as explaining and verifying what has been learned. Note that this environment is significantly more complex than MuJoCo due to its dynamics, large state–action spaces, and low sample efficiency.

Figure 6 plots the learning curves of all the models. In the four tasks, our strategy greatly improves convergence speed and performance compared to plain PPO. Unlike our algorithm, PPO that learns without reusing prior knowledge about the task fails to capture the dynamics of the task and/or learn from sparse rewards, resulting in poor performance. By examining its learned policies, we notice that soon after the beginning of a disturbance, PPO cannot avoid failures in equipment. However, our method can quickly discover how to recover from a disturbance by using both the symbolic rewards as well the explanations extracted from the symbolic trees. As in the previous experiment, iIL and GAIL achieve similar final performance. In addition, by gleaning insights from the symbolic tree, a human operator can swiftly discover task-relevant features and partially explain the reasons for the agent's behavior. For instance, by examining the learned symbolic rewards we could notice that during heavy rain, it is important to maintain the temperature of the reactor's catalyst bed by increasing the TC201 unit. In another example (Pressure C2H4), if ethylene feed pressure is decreased, then it is necessary to decrease the gas pipeline head pressure (PC130) to introduce more ethylene in the feed. We further discuss this question in Section 5. Overall, this experiment demonstrates that our method can scale to real-world industrial problems and provides enough prior knowledge to the agent to improve its sample efficiency.

4.4 Ablation Analysis

We also present ablation studies to investigate (1) the quality of the discovered reward functions, (2) the quality of the extracted explanations, (3) the disagreement with the IRL model, (4) the robustness to noisy data, (5) the impact of the amount of data, (6) the impact of the symbolic tree size, (7) the use of GRU to capture temporal information, (8) the impact of the RL algorithm, (9) a strategy for manual reward fine-tuning, (10) reward fine-tuning via human preferences, and (11) predicting the agent's behavior via reward parsing.

4.4.1 Quality of the Symbolic Rewards. To interpret how relevant/good is the learned symbolic reward, we present in Figure 7 an example of a symbolic reward in Hopper and Pressure C2H4.



(a) Pressure AcOH (b) Pressure C2H4

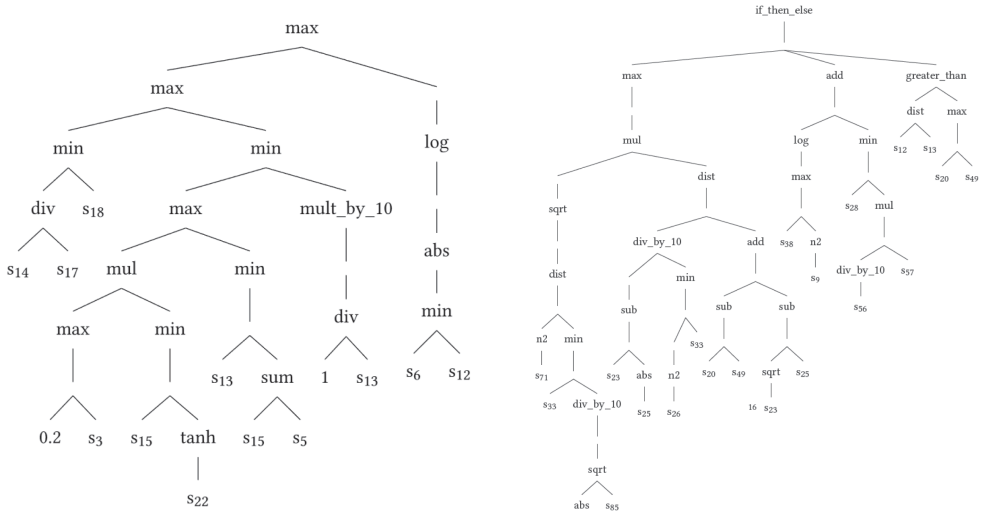
(c) Day/Night (d) Rain

Fig. 6. Performance for different disturbance scenarios on the VAM plant. Results are averaged over 10 runs (±std).

While qualitatively evaluating the reward functions is challenging for a non-expert, we can draw several observations. First, the learned rewards feature a small number of operators compared to standard DNNs, which entails that it is significantly easier to understand the reward function. Second, the second reward function primarily relies on inputs that are notoriously task-relevant in the VAM plant. Finally, the previous experiments demonstrated that an agent trained from them can achieve a large return in all tasks, which suggests that they accurately capture the expert's intention.

Moreover, we can observe that it would be relatively easy for a knowledgeable human to modify the symbolic tree to incorporate prior knowledge, such as by adding a new branch or changing an operator. A new branch may cover a novel situation that is not presented in the training data. It would also be possible to fine-tune the reward function by modifying the constants. However, the GAIL model is largely opaque and cannot be manually tuned.

4.4.2 Quality of the Explanations. We present a quantitative analysis of the quality of the explanations. The question we aim to answer in this evaluation is "Are the extracted explanations found by iIL really relevant to the agent policy?" We propose a measurement that the impact of the explanations on the output policy can be quantified by evaluating the policy solely augmented with the explanations (iIL-e). Table 2 shows the results of PPO that solely receives explanations. The results demonstrate that leveraging explanations leads to a higher average return compared to plain PPO. In other words, the explanations carry task-relevant information that can be used to guide the agent's training. Note that in the absence of well-shaped rewards provided by the symbolic tree, PPO is trained from sparse reward signals. Therefore, the final return achieved by iIL-e is lower than the one of iIL-re. In addition, by inspecting the explanations, we noticed that they primarily carry information with a high level of abstraction. For instance, in the *rain* task, on



(a) Hopper (b) Pressure C2H4

Fig. 7. Two examples of discovered symbolic rewards on Hopper and Pressure C2H4. We report the symbolic rewards after 100 training epochs.

Table 2. Final Mean Performance (± std) of Our Method Augmented with Explanations

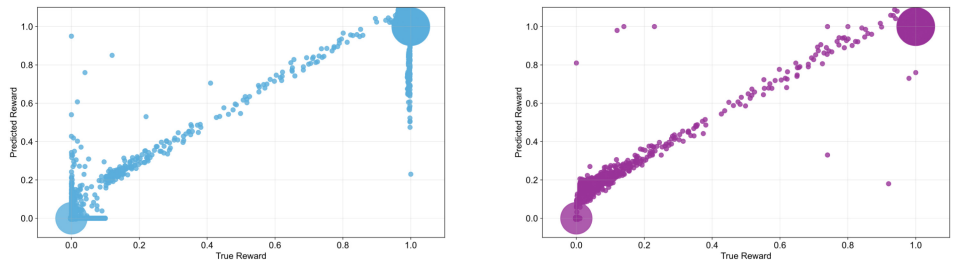
	Robot Control				VAM Plant			
Method	Half Cheetah	Hopper	Walker 2d	Ant	Pressure AcOH	Pressure C2H4	Day/Night	Rain
iIL-e	3,922±655	3,099±541	4,622±608	3,563±404	0.54±0.14	0.49±0.11	0.41±0.09	0.59±0.12
iIL-re	5,582±877	4,925±489	8,348±818	4,884±862	0.75±0.13	0.92±0.15	0.66±0.07	0.85±0.13
PPO	3,470±185	2,522±596	2,420±483	2.954±850	0.49±0.12	0.40±0.16	0.0±0.01	0.37±0.05
GAIL	4,144±682	3,490±829	6,998±1,176	4,371±809	0.63±0.09	0.68±0.08	0.62±0.10	0.63±0.13

Averages over 10 runs.

average 12 attributes of the explanations are related to the temperature in the components of the plants. By augmenting the agent's input with such high-level information, the agent can quickly learn how to act—to restore the stability of the plant, without extensive exploration.

4.4.3 Disagreement with the IRL Model. In this section, we analyze the disagreement between the learned symbolic reward function and the IRL model that initially recovered the expert cost function (i.e., GAIL). We report in Figure 8 the $error = \mid x(s_t, a_t) - c_t \mid$ of our model that does not employ a hierarchical decomposition of the reward tree (left) and with hierarchical tree learning (right). To measure the ability of iIL to scale to complex environments, we report the results obtained in the Pressure C2H4 task. As can be observed, both approaches accurately mimic the expert cost function. However, the hierarchical approach further reduces the average prediction error as it learns larger trees that cover a broader range of situations.

We also computed the percentage of timesteps where our reward function is significantly different from the expert cost function (error > 0.1). We found that the hierarchical approach can accurately imitate the expert cost function 98.3% of the time and 72.4% of the time for the non-hierarchical approach. In addition, in both environments, the present method exhibits a small error. This suggests that the symbolic reward function can generalize well and capture important features of the system. As a result, one can expect our agent trained with a symbolic reward to reach



(a) Flat tree learning (b) Hierarchical tree learning

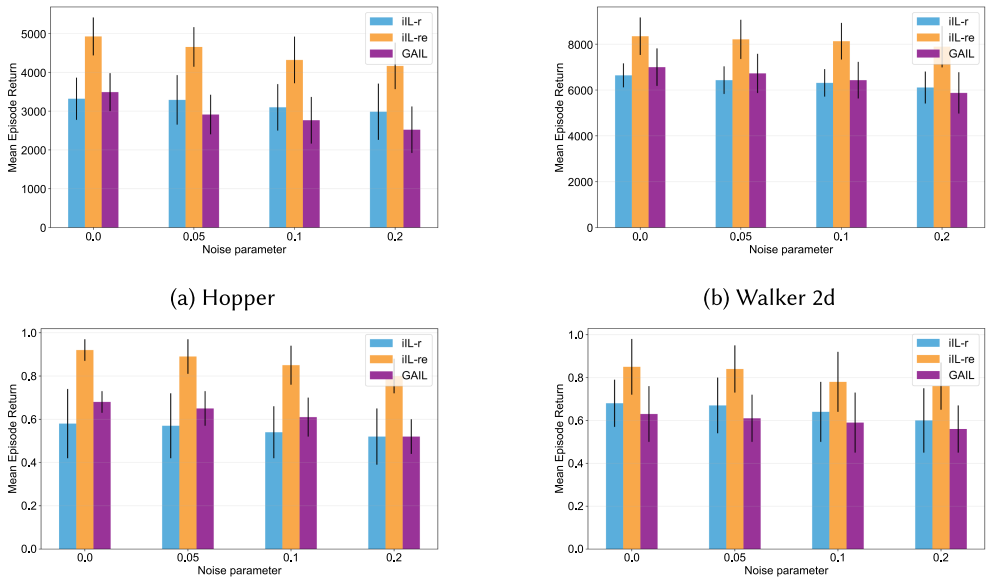
Fig. 8. True vs. learned reward in the Pressure C2H4 task. A fully aligned reward model would have all points on a straight line. The size of the dots corresponds to the number of points with the same true and learned reward.

performance comparable to that of "black-box" methods. Overall, the results highlight that the hierarchical reward approach enables the discovery of symbolic rewards that mimic more accurately the expert cost function, leading to a small disagreement with GAIL.

4.4.4 Noisy Data. We evaluated the robustness of the proposed method to noisy data by adding different levels of noise to the expert cost function. Namely, we add with probability $\{0.0, 0.05, 0.10, 0.20\}$ independent Gaussian noise to the expert cost, with mean zero and standard deviation proportional to the root mean square of the dependent variable in the cost of the expert data (Figure 9). Because symbolic rewards focus on the most important features of the dataset, the learned reward function discards noisy and irrelevant details. Namely, although the expert cost function is noisy, the model can leverage other (accurate) transitions without being oversensitive to noise in the data. Thus, the final performance of the agent trained with a symbolic reward function achieves similar performance to the original one. However, the agent trained with GAIL directly perceives the noisy expert cost function, resulting in lower performance. As a result, iIL can be used even when the expert demonstrations are noisy and inaccurate. Note that if the cost function is very noisy, then iIL is likely to learn noise in the data, resulting in low final performance comparable to GAIL.

4.4.5 Amount of Data. One legitimate question is to study the impact of the amount of data on the agent's performance. Ideally, (1) the policy performance should not be too sensitive to this hyperparameter as the generator should capture a relevant symbolic reward from a small amount of data. (2) The generator should be able to improve the quality of the learned reward function as the amount of data increases, since novel data can cover different regions of the environment. We perform a study for a various number of examples (s, a, c) on Hopper, Walker 2d, Pressure C2H4, and Rain. Figure 10 shows that the iIL performance is robust to the choice of this hyperparameter. In addition, even a small amount of data is sufficient to learn an effective symbolic reward function. Finally, as expected a larger dataset increases the quality of the learned reward function. However, there is no significant performance increase when using 100K training examples, as 30K expert transitions are sufficient to cover most of the situations.

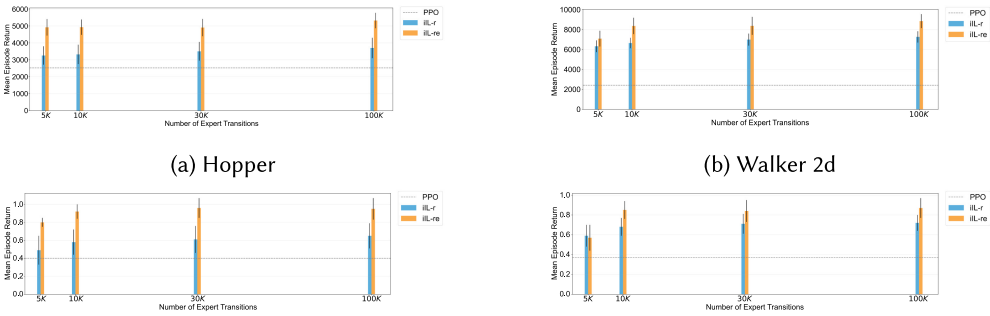
4.4.6 Impact of the Tree Size. We now report in Figure 11 evaluations showing the effect of different maximum tree size. The maximum tree size indicated on the x-axis depicts the maximum size for the high-level tree (left) and low-level trees (right). Figure 11 demonstrates that agents trained with a larger maximum tree size budget obtain higher mean returns after similar numbers of updates. However, despite a small maximum tree size (20/10), our method can still learn accurate



(a) Hopper (b) Walker 2d

(c) Pressure C2H4 (d) Rain

Fig. 9. Effect of sub-optimal demonstrations on iIL and GAIL. We produce sub-optimal demonstrations by adding Gaussian noise to the expert cost function with the probability shown on the x-axis. The vertical lines depict the standard deviation across 10 runs of each experiment.



(a) Hopper (b) Walker 2d

(c) Pressure C2H4 (d) Rain

Fig. 10. Average return obtained after training iIL with different amounts of expert data. The vertical lines depict the standard deviation across 10 runs of each experiment.

symbolic reward functions. We can draw the observation that as the maximum tree size increases, the learning effect on the agent gradually improves. Nonetheless, for the results with (40/15) and (60/40) maximum size, we can see that even though the maximum tree size significantly differs, the difference in learning effect can be negligible. This can happen because the smallest setting is sufficient to clone the expert cost function. As a result, our method can be trained with a relatively small maximum tree size, facilitating the interpretation of the symbolic rewards by humans.

4.4.7 Temporal Information. In this study, we conducted an ablation analysis to evaluate the impact of a GRU on the performance of the proposed method. Specifically, we compared the

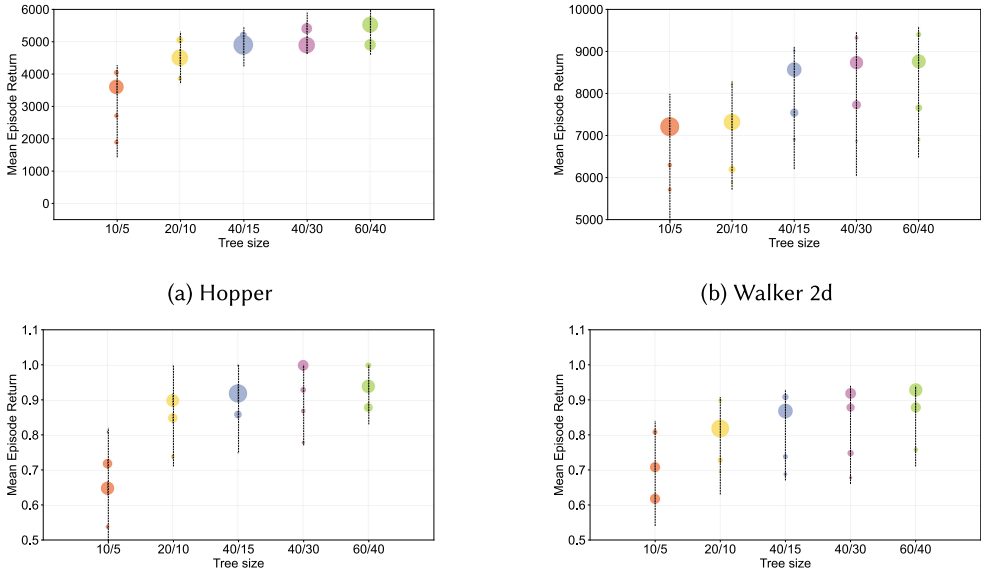


Fig. 11. Distributions of the accumulated reward over the 50 trials for different maximum tree sizes, each corresponding to a vertical line. On each vertical, the 50 trial results are binned in five intervals. Each bin is then displayed as a circle at height equal to the average value of the bin and of size proportionate to the number of results in the bin. The dashed line represents the lowest and highest return obtained over the 50 trials. The maximum tree size indicated on the x-axis (•/•) depicts the maximum size for the high-level tree (left) and low-level trees (right).

Table 3. Final Mean Performance (± std) of Our Method with and without GRU to Capture Temporal Information

	Robot Control				VAM Plant			
Method	Half Cheetah	Hopper	Walker 2d	Ant	Pressure AcOH	Pressure C2H4	Day/Night	Rain
iIL-e	3,922±655	3,099±541	4,622±608	3,563±404	0.54±0.14	0.49±0.11	0.41±0.09	0.59±0.12
iIL-re	5,582±877	4,925±489	8,348±818	4,884±862	0.75±0.13	0.92±0.15	0.66±0.07	0.85±0.13
iIL-e (no GRU)	3,151±312	2,712±418	4,211±284	3,042±511	0.51±0.13	0.39±0.12	0.38±0.09	0.56±0.11
iIL-re (no GRU)	5,120±918	4,486±453	7,543±704	4,684±752	0.71±0.13	0.79±0.17	0.60±0.11	0.81±0.12

Averages over 10 runs.

performance of the method with and without the GRU to determine the extent to which the GRU is necessary for capturing temporal information of explanations. The results of the ablation analysis (Table 3) show that the version of the method with a GRU significantly outperforms the version without a GRU on all tasks. Specifically, the version with the GRU achieves higher final mean performance. These results suggest that a GRU is an important component of the method and is useful for capturing the temporal information. One compelling reason for the effectiveness of the GRU is the model's ability to recall previous explanations. By recalling previous explanations, our method can aggregate several explanations that can guide the agent's exploration. This is because when acting, the agent should account for previous actions and states visited, which can be captured by the GRU. Overall, the results of the ablation analysis demonstrate the importance of using the GRU for aggregating explanations.

Table 4. Final Mean Performance (± std) of Our Method Trained with Different Learning Algorithms (TRPO and A2C)

	Robot Control				VAM Plant			
Method	Half Cheetah	Hopper	Walker 2d	Ant	Pressure AcOH	Pressure C2H4	Day/Night	Rain
iIL(TRPO)	4,123±1,012	4,623±711	8,223±1,003	4,398±913	0.68±0.16	0.91±0.14	0.60±0.12	0.79±0.15
iIL(A2C)	5,730±788	5,540±505	8,525±818	5,019±699	0.77±0.12	0.91±0.15	0.64±0.08	0.83±0.15
iIL(PPO)	5,582±877	4,925±489	8,348±818	4,884±862	0.75±0.13	0.92±0.15	0.66±0.07	0.85±0.13
PPO	3,470±185	2,522±596	2,420±483	2.954±850	0.49±0.12	0.40±0.16	0.0±0.01	0.37±0.05
GAIL	4,144±682	3,490 ±829	6,998±1,176	4,371±809	0.63±0.09	0.68±0.08	0.62±0.10	0.63±0.13

Averages over 10 runs.

4.4.8 Learning Algorithm. To further evaluate the effectiveness and generalizability of the proposed method, we performed experiments with two additional RL algorithms: **Trust Region Policy Optimization (TRPO)** [74] and **Advantage Actor-Critic (A2C)** [59]. For both algorithms, we used the same architecture and hyperparameters as in the PPO experiments, except for the specific changes required to accommodate each algorithm. The results of the experiments (Table 4) highlight that our method is effective with different RL algorithms, achieving similar or better performance compared to the baseline methods. We can further observe that in some tasks, iIL(A2C) could achieve higher final mean performance than iIL(PPO). Notably, the proposed method trained with TRPO and PPO achieved similar performance; however, PPO is more stable than TRPO. When trained with A2C, we noticed a significant improvement in the learning efficiency compared to PPO trained with the original reward signal. These findings indicate that the proposed method is not constrained to a particular reinforcement learning algorithm and can be applied to different algorithms with similar effectiveness. This highlights the versatility of iIL and its potential to be utilized in a wide variety of RL applications.

4.4.9 Manual Reward Fine-tuning. In some reinforcement learning applications, the reward function provided by the environment may not be sufficient to achieve the desired behavior or performance. In these cases, a human can manually modify the symbolic reward function to guide the agent toward the desired behavior. Here we discuss the process of manually fine-tuning the reward function and how it can be used to improve the performance of the agent. We should emphasize that the interpretability of the learned reward is a crucial aspect that empowers humans to take an active role in guiding the agent's behavior. The process of manually modifying the reward function involves adjusting the weights and symbols of the reward function to prioritize certain objectives or penalize certain behaviors. This can be done by analyzing the agent's behavior and identifying areas where it can be improved. Specifically, synthesized symbolic rewards are not only readable to human users but also interactive, allowing non-expert users with a basic understanding of the task to diagnose and make edits to improve their performance.

To demonstrate this, we asked three humans to read, interpret, and edit symbolic rewards to improve their performance. For example, if the agent is exhibiting risky behavior, then the penalty for taking risky actions can be increased to discourage such behavior. However, if the agent is not exploring the environment enough, then the weights of branches corresponding to exploration can be increased to encourage exploration. Symbols can further be modified or deleted according to the human's intention. To enable interactive fine-tuning, there were seven rounds of fine-tuning for each reward. Specifically, we performed a case study on Hopper and Walker 2d. In Hopper, we noticed that humans manually fine-tuned the reward function by increasing the weight of the branch of the symbolic tree that encourages the agent to go further, while decreasing the weight of a branch of the tree that dissuades large action values. In Walker 2d, performance could be enhanced by pruning a branch of the tree that encouraged too-high velocity, making early

Table 5. Final Mean Performance (± std) of Our Method Trained Using the Original Symbolic Reward and the Fine-tuned Version

	Robot Control	
Method	Hopper	Walker 2d
iIL-e	3,099±541	4,622±608
iIL-re	4,925±489	8,348±818
iIL-e (fine-tuned)	3,242±518	5,110±587
iIL-re (fine-tuned)	5,349±466	8,531±823

We asked each of the three humans to modify two symbolic trees that were discovered via iIL. There were five rounds to enable humans to visualize their changes. Averages over 10 runs.

exploration too challenging. A few other constants were adjusted to improve the symbolic reward. The results can be found in Table 5. Overall, after a few iterations of modifying the reward function and analyzing the agent's behavior, we were able to achieve a significant improvement in the agent's performance. We discuss limitations and possible areas of improvement in Section 5.

4.4.10 Reward Fine-tuning via Human Preferences. As an alternative to directly modifying the symbolic reward, it is possible to act on the generator network to generate symbolic rewards more aligned with the demonstrator's intention. As a proof of concept, we propose to fine-tune the generator model with human preferences. That is, following Christiano et al. [11], we asked human labellers to pick the most suitable symbolic reward for given short trajectories. Namely, the human overseer was given two short trajectory segments, in the form of short movie clips (60 frames) with their associated rewards. The human overseer could see five short clips and then was requested to rank the two symbolic rewards. We then derived a new expert cost function in the form of a preference predictor, similarly to a previous work [11]. The generator network was then fine-tuned using those preference-based rewards so that the generated trees will maximize the preference-based rewards. In this set of experiments, we collected a total of 300 preferences. As shown in Table 6, there is a large gap between original and fine-tuned models. Besides, experimental results verify that the fine-tuned model aligns better with human judgment than the original model, effectively distilling human's intention. We can further observe that preference-based fine-tuning could achieve slightly higher performance than manual reward fine-tuning (Section 4.4.9), which suggests that human preferences are an effective way of fine-tuning iIL.

4.4.11 Predicting Agent Behavior via Reward Parsing. As mentioned above, parsing symbolic rewards offers the potential to understand and predict the agent's behavior. To demonstrate this potential, we designed an ablation analysis based on Pressure AcOH, Pressure C2H4, Day/Night, and Rain tasks. We collected trajectories of agents trained with symbolic and/or extrinsic rewards. Then, we asked three humans to compare pairs of final states—a true final state and a *negative* final state, and judge which states was the most likely to be reached given the reward used to train the corresponding agent. In summary, humans were shown pairs of final states (50 pairs), and were requested to pick the most likely state to be reached given the reward used to train the agent. We report in Figure 12 the accuracy for (1) symbolic and (2) extrinsic rewards. Experimental results demonstrate that the symbolic reward-trained agents had a significantly higher agreement rate with the human evaluators compared to the extrinsic reward-trained agents. The results suggest that parsing symbolic rewards can improve the interpretability and predictability of agent

Table 6. Final Mean Performance (± std) of Our Method Trained Using the Original Symbolic Reward and the Fine-tuned Version via Human Preferences

	Robot Control	VAM Plant
Method	Hopper	Walker 2d
iIL-e	3,099±541	4,622±608
iIL-re	4,925±489	8,348±818
iIL-e (manually fine-tuned)	3,242±518	5,110±587
iIL-re (manually fine-tuned)	5,349±466	8,531±823
iIL-e (preference fine-tuned)	3,429±542	5,594±602
iIL-re (preference fine-tuned)	5,621±499	8,778±788

We also report results of manual fine-tuning obtained in Section 4.4.9.
Averages over 10 runs.

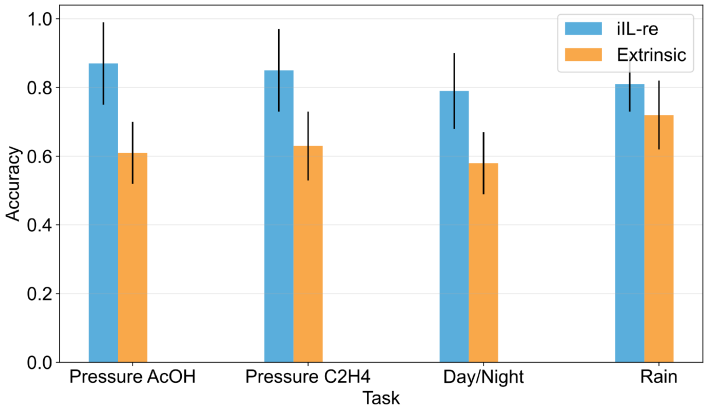


Fig. 12. Accuracy for behavior prediction for our method and using extrinsic rewards. We report the results averaged for three humans and 50 comparisons for each.

behavior, enabling humans to better understand and control the behavior of AI systems. One possible reason for this difference in performance is that symbolic rewards provide a more granular and transparent framework for designing and evaluating reward functions. In contrast, extrinsic rewards may be more complex and difficult for humans to understand, as they often involve indirect and implicit incentives that may not be directly aligned with the task objectives. Overall, our study highlights the potential of symbolic reward parsing as a valuable tool for understanding and predicting agent behaviors. We further discuss this question and future work in Section 5.

5 DISCUSSION

Our work takes a step toward achieving interpretable imitation learning. We have constructed a mechanism based on symbolic rewards and showed that the method can help exploration in challenging sparse-reward environments. The symbolic reward functions learned by using our method exhibit significantly fewer operators compared to standard DNN methods. The experiments demonstrate the effectiveness of this approach by achieving improvements on notoriously difficult tasks such as controlling a chemical plant under disturbances. Notably, iIL could scale to real-world applications including tasks featuring complex dynamics and large state–action spaces.

They also clearly demonstrate the benefits of iIL to improve sample efficiency in RL, including in safety-critical applications. Other advantages of the proposed symbolic method lie in the compactness of the discovered rewards as well as their verifiably and interpretability.

It can be observed that manually verifying the reward model is a tedious task in tasks featuring a large number of parameters. Despite our efforts to reduce human involvement, the most complex tasks still require more feedback than we would like. Therefore, minimizing the amount of data required for verifying the reward function or developing new types of automatic verification remains an important direction to explore. Future work should explicitly consider how easy it is for a human to verify a reward function. In the remainder of the discussion section, we further discuss possible methods to reduce human effort, such as visualization tools of the rewards or counterfactual explanations.

That being said, we acknowledge that our approach has certain limitations and potential avenues for future research. As shown in Section 4.4.1, the learned symbolic reward functions are relatively easy to parse and understand for a human. However, if the tree size becomes very large, then it may become difficult for a human to quickly inspect the reward function. The complexity of the symbolic reward trees is defined in terms of the maximum number of features and complexity of the operators. As shown in the Appendix, we selected simple operators to ensure that insights can be gleaned at inspection—without relying on complex analysis. Besides, we set the maximum length to 40 and 15 for high-level trees and low-level trees, respectively. Setting this parameter is a tradeoff between accuracy and interpretability. As long as the length of the trees remains within an acceptable range, we argue that iIL is drastically easier to understand for a human compared to DNN-based methods that involve thousands of nested operators and non-linear transformations. To facilitate the parsing of the symbolic rewards, we are interested in building a visualization framework, but we leave it to future work. For instance, the framework may highlight which branches in the tree are executed based on the input and logic operators being used in the tree. We also anticipate generating short textual descriptions of the reward function by parsing the symbolic tree, which can be provided to a human. Such a description may include the weight of each input variable on the final prediction, providing to the operator a method to grasp the importance of the features at each timestep.

Since iIL involves several learnable components, hyperparameter tuning can be a non-trivial task given the number of hyperparameters that need to be tuned. We agree that this is a potential limitation that may affect the performance of the method. In our experiments, we conducted a simple hyperparameter tuning process to obtain a set of hyperparameters that achieve good performance across multiple tasks. Nonetheless, we acknowledge that the optimal set of hyperparameters may vary depending on the specific task and the characteristics of the environment. Therefore, automatically adjusting the maximum length of high-level and low-level trees is an important direction that we are willing to explore to reduce the burden of parameter tuning. One way would be to dynamically adjust maximum lengths based on the training signal. Ideally, we would want maximum tree lengths to automatically increase in complex environments to enable the discovery of complex symbolic rewards, but we leave it to future work to explore this direction further.

As mentioned above, an important interpretability feature of symbolic rewards is the possibility to be parsed by a human to understand and predict the agent's behavior. In Section 4.4.10, we suggest that iIL is a versatile first step toward understanding agent behaviors. The reward for a state can be explained factually by visualizing the symbolic tree. However, it is argued that a more natural [49] and persuasive [91] form of explanation is the counterfactual, which provides reasons why an alternative outcome does not occur instead. In ongoing work, we want to continue to explore the potential of predicting agent behaviors, refining and expanding our methods of

analysis, and evaluating the model in more challenging environments. This involves evaluating counterfactual (symbolic) explanations for the generated rewards and building visualization tools. To further extend this research, future work should focus on evaluating the ability of humans to predict agent behaviors via reward parsing on a more extensive range of tasks and a larger number of human participants.

One may notice that the choice of suitable operators is an important aspect of symbolic regression. The choice of the elementary functions used by iIL can significantly affect the performance of the method. Optimizing the operator set is not the main objective of this work. Therefore, we used a rather small, yet sufficient set of elementary operators consisting of arithmetic and logic operators. One avenue for research is to automatically ground the operators to adapt iIL to the target domain.

Another question is the computational complexity of the method. The time needed for training the generators ranges from several seconds to a few minutes for a very large number of training epochs. However, the running time of the algorithm increases linearly with the input size. To accelerate the training under this setting, we introduced a hierarchical decomposition of the reward function. By analyzing the flat and hierarchical approaches, we observed that the latter improves both the prediction accuracy of the reward as well as the training speed. In this article, we trained the model with relatively large inputs (>100) and found that iIL remains significantly faster than training PPO to achieve similar final performance. Therefore, we argue that the computational overload is acceptable in comparison with the benefits offered by the proposed approach.

An avenue for research is a qualitative evaluation of symbolic rewards that should be investigated more deeply. While our experiments allowed us to quantitatively demonstrate that the symbolic rewards can help the agent to learn, it remains challenging to qualitatively evaluate the discovered rewards. In Section 4.4.1, we show two examples of symbolic rewards; however, we acknowledge that it would be necessary to qualitatively evaluate a larger number of discovered symbolic rewards. Asking human experts to rate the rewards may be a solution. Another solution for qualitative evaluation is to ask experts to design rewards and then compare them with the discovered symbolic rewards. This research direction is left for future work.

Another exciting future direction is to evaluate the proposed method on physical robots and plants. In this article, we focused on simulations that reflect real-world characteristics to verify our approach. Namely, we carefully considered scenarios as close as possible to reality. Based on the presented results, we can expect our method to benefit in sample efficiency and to significantly reduce the number of interactions. While testing on real robots would be ideal for pragmatic applications, due to practical limitations we leave this direction for future work. In addition, we aspire to expand the evaluation of our method to more diverse environments such as game playing, three-dimensional navigation, or traffic simulation.

As discussed in the Introduction, a key motivation in discovering interpretable rewards from expert data is to enable humans to modify or improve the reward. By doing so, prior knowledge about the task can be distilled, and the behavior of the agent can be adjusted to quickly reach the goal being pursued. In contrast, deep neural network approaches are generally impossible to use in cooperation with humans. As a first step toward achieving this goal, we presented an algorithm to discover symbolic rewards from expert data. In this work, we focus on the method to discover these interpretable rewards, and we have demonstrated that the symbolic rewards can be parsed by a human. In Section 4.4.9, we presented an ablation study to demonstrate how a human could improve and fine-tune the learned symbolic reward function. As shown, manually fine-tuning the reward function can be effective; however, we also acknowledge that interactive fine-tuning can be a time-consuming task. In the future, we anticipate fine-tuning the generator via human preferences [11]. Rather than directly modifying the reward function, it may be possible to achieve

the desired agent behavior by leveraging human preferences and reinforcement learning. Such an idea has been recently introduced for fine-tuning large language models [103]. In Section 4.4.10, we presented a proof of concept where the generator is fine-tuned with a small amount of human preferences. We believe that similar approaches hold great potential in fine-tuning iIL, where the learned reward function can be better aligned with the human's preferences without the need to directly handcraft the symbolic reward. This article merely hints at the potential applications for a symbolic reward, applications that we aim to exploit more fully in future work.

6 CONCLUSION

Standard imitation learning approaches have provided a significant speedup in the training process. However, they generally rely on DNN models, which entails that it is hard to explain what knowledge the models has learned from expert data, and they cannot be verified or improved by a human. To bridge this gap, we propose to learn a surrogate symbolic reward function from an expert cost function. Namely, we propose to train an autoregressive RNN to generate symbols that together represent a tree. To deal with large state–action spaces, we present a hierarchical decomposition of the reward function. We further introduce a mechanism that extracts explanations by leveraging the structure of the tree to enrich the reward signal. We demonstrate the approach on two sets of control tasks, achieving similar or better performance than DNN-based IRL models in terms of average return and sample efficiency. In addition, the experiments showed that the learned symbolic rewards can be used in cooperation with a human. Namely, a human expert can (1) modify or improve the reward and (2) glean important features of the environment by parsing the reward. The results suggest that symbolic rewards can be a strong alternative to NN-based IRL models, especially when human interpretability is of utmost importance.

APPENDIX

A SYMBOLIC REWARD TREES

The list of operators being used in our experiments is described in the table below. For describing the operators, we use a, b, and c as the name of the input(s), which are respectively the first, second, and third inputs of the operator (depending on the arity).

Operator (name)	Arity	Type	Description
add	2	arithmetic	a+b (sum)
sub	2	arithmetic	a-b (subtraction)
mul	2	arithmetic	a×s b (multiplication)
div	2	arithmetic	a/b (division)
cos	1	arithmetic	Trigonometric cosine
sin	1	arithmetic	Trigonometric sine
tan	1	arithmetic	Trigonometric tangent
exp	1	arithmetic	Exponential
log	1	arithmetic	Natural logarithm
sqrt	1	arithmetic	Square-root
n2	1	arithmetic	a raised to power 2
neg	1	arithmetic	Numerical Negative
abs	1	arithmetic	Absolute value
tanh	1	arithmetic	Hyperbolic tangent
inv	1	arithmetic	Reciprocal
min	2	arithmetic	Minimum of a and b
max	2	arithmetic	Maximum of a and b
dist	2	arithmetic	Absolute *distance* between a and b
div_by_10	1	arithmetic	Divide a by 10.0
mult_by_10	1	arithmetic	Multiply a by 10.0
binom	1	arithmetic	Binomial
is_negative	1	logic	Return 1 if a is negative or 0
is_positive	1	logic	Return 1 if a is positive or 0
greater_than	2	logic	Return 1 if a is greater than b
smaller_than	2	logic	Return 1 if a is smaller than b
equal_to	2	logic	Return 1 if a is equal to b
or	2	logic	Logical AND between a and b
and	2	logic	Logical OR between a and b
if_then_else	3	logic	Return a if the condition c is True and b otherwise

REFERENCES

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. *Adv. Neural Inf. Process. Syst.* 31 (2018), 9525–9536.

[2] Ahmed M. Alaa and Mihaela van der Schaar. 2019. Demystifying black-box models with symbolic metamodels. *Adv. Neural Inf. Process. Syst.* 32 (2019), 11304–11314.

[3] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable reinforcement learning via policy extraction. *Adv. Neural Inf. Process. Syst.* 31 (2018), 2499–2509.

[4] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. 2017. Neural optimizer search with reinforcement learning. In *International Conference on Machine Learning*. PMLR, 459–468.

[5] Tom Bewley and Jonathan Lawry. 2021. Tripletree: A versatile interpretable representation of black box agents and their environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11415–11422.

[6] Benjamin Beyret, Ali Shafti, and A. Aldo Faisal. 2019. Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '19)*. IEEE, 5014–5019.

[7] Nicolas Bougie and Ryutaro Ichise. 2020. Exploration via progress-driven intrinsic rewards. In *Artificial Neural Networks and Machine Learning—Proceedings of the 29th International Conference on Artificial Neural Networks (ICANN'20)*. Springer, 269–281.

[8] Nicolas Bougie and Ryutaro Ichise. 2020. Skill-based curiosity for intrinsically motivated reinforcement learning. *Mach. Learn.* 109 (2020), 493–512.

[9] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=H1lJJnR5Ym
[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, 1724–1734.
[11] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems* 30, (2017).
[12] Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. 2019. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence*. 1–6.
[13] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. ACM, 225–232.
[14] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Adv. Neural Inf. Process. Syst.* 31 (2018).
[15] Marco Dorigo and Marco Colombetti. 1994. Robot shaping: Developing autonomous agents through learning. *Artif. Intell.* 71, 2 (1994), 321–370.
[16] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. 2018. Explainable artificial intelligence: A survey. In *Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO '18)*. IEEE, 0210–0215.
[17] Mengnan Du, Ninghao Liu, and Xia Hu. 2019. Techniques for interpretable machine learning. *Commun. ACM* 63, 1 (2019), 68–77.
[18] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. 2001. Relational reinforcement learning. *Mach. Learn.* 43, 1 (2001), 7–52.
[19] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal* 1341, 3 (2009), 1.
[20] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*. PMLR, 49–58.
[21] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[22] Eli Friedman and Fred Fontaine. 2018. Generalizing across multi-objective reward functions in deep reinforcement learning. arXiv:1809.06364. Retrieved from https://arxiv.org/abs/1809.06364
[23] Artur d'Avila Garcez, Aimore Resende Riquetti Dutra, and Eduardo Alonso. 2018. Towards symbolic reinforcement learning with common sense. arXiv:1804.08597. Retrieved from https://arxiv.org/abs/1804.08597
[24] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. 2016. Towards deep symbolic reinforcement learning. arXiv:1609.05518. Retrieved from https://arxiv.org/abs/1906.05518
[25] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of the IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA '18)*. IEEE, 80–89.
[26] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. 2021. A survey on interpretable reinforcement learning. arXiv preprint arXiv:2112.13112 (2021).
[27] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *International Conference on Machine Learning*. PMLR, 2376–2384.
[28] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. 2016. Variational intrinsic control. arXiv:1611.07507. Retrieved from https://arxiv.org/abs/1611.07507
[29] Marek Grzes and Daniel Kudenko. 2008. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings AAMAS Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg '08)*. 17–23.
[30] Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. 2021. Edge: Explaining deep reinforcement learning policies. *Adv. Neural Inf. Process. Syst.* 34 (2021), 12222–12236.
[31] Bradley Hayes and Julie A. Shah. 2017. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, 303–312.
[32] Lei He, Nabil Aouf, and Bifeng Song. 2021. Explainable deep reinforcement learning for UAV autonomous path planning. *Aerosp. Science Technol.* 118 (2021), 107052.

[33] Daniel Hein, Steffen Udluft, and Thomas A. Runkler. 2018. Interpretable policies for reinforcement learning by genetic programming. *Eng. Appl. Artif. Intell.* 76 (2018), 158–169.
[34] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2018. Deep Q-learning from demonstrations. In *Proceedings of the Annual Meeting of the Association for the Advancement of Artificial Intelligence*.
[35] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Proceedings of Advances in Neural Information Processing Systems*. 4565–4573.
[36] Alihan Hüyük, Daniel Jarrett, Cem Tekin, and Mihaela Van Der Schaar. 2021. Explaining by imitating: Understanding decisions by interpretable policy learning. In *International Conference on Learning Representations*.
[37] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. 2019. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.
[38] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. 2022. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems* 35, (2022), 10269–10281.
[39] Jinkyu Kim and Mayank Bansal. 2020. Attentional bottleneck: Towards an interpretable deep driving network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 322–323.
[40] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2019. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 267–280.
[41] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980
[42] Mikel Landajuela, Brenden K. Petersen, Sookyung Kim, Claudio P. Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F. Pettit, and Daniel Faissol. 2021. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5979–5989.
[43] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. *Nat. Commun.* 10, 1 (2019), 1–8.
[44] Adam Daniel Laud. 2004. *Theory and Application of Reward Shaping in Reinforcement Learning*. Ph. D. Dissertation. Advisor(s) Dejong, Gerald. AAI3130966.
[45] Matteo Leonetti, Luca Iocchi, and Peter Stone. 2016. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artif. Intell.* 241 (2016), 103–130.
[46] Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. 2020. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robot. Auton. Syst.* 131 (2020), 103568.
[47] Zhengxian Lin, Kin-Ho Lam, and Alan Fern. 2021. Contrastive explanations for reinforcement learning via embedded self predictions. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=Ud3DSz72nYR
[48] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable ai: A review of machine learning interpretability methods. *Entropy* 23, 1 (2020), 18.
[49] Peter Lipton. 1990. Contrastive explanation. *Roy. Inst. Philos. Suppl.* 27 (1990), 247–266.
[50] Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. 2019. Toward interpretable deep reinforcement learning with linear model u-trees. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '18)*. Springer, 414–429.
[51] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* 30 (2017).
[52] Ronny Luss, Amit Dhurandhar, and Miao Liu. 2022. Interpreting reinforcement policies through local behaviors. Retrieved from https://openreview.net/forum?id=7qaCQiuOVf
[53] Yuta Machida, Shigeki Ootakara, Hiroya Seki, Yoshihiro Hashimoto, Manabu Kano, Yasuhiro Miyake, Naoto Anzai, Masayoshi Sawai, Takashi Katsuno, and Toshiaki Omata. 2016. Vinyl Acetate Monomer (VAM) plant model: A new benchmark problem for control and operation study. *Proc. IFAC Symp. Dynam. Contr. Process Syst. Incl. Biosyst.* 49, 7 (2016), 533–538.
[54] Ofir Marom and Benjamin Rosman. 2018. Belief reward shaping in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[55] Bhaskara Marthi. 2007. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine Learning*. 601–608.
[56] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles Kamhoua, Evangelos E Papalexakis, and Fei Fang. 2023. MAVIPER: Learning decision tree policies for interpretable multi-agent reinforcement learning.

In *Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD'22)*. Springer, 251–266.
[57] Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. 2021. Ella: Exploration through learned language abstraction. *Adv. Neural Inf. Process. Syst.* 34 (2021), 29529–29540.
[58] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT Press.
[59] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. 1928–1937.
[60] Christoph Molnar. 2020. *Interpretable Machine Learning*. Lulu.com.
[61] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digit. Sign. Process.* 73 (2018), 1–15.
[62] Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. 2019. Towards interpretable reinforcement learning using attention augmented agents. *Adv. Neural Inf. Process. Syst.* 32 (2019).
[63] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 6292–6299.
[64] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. 2020. XGAIL: Explainable generative adversarial imitation learning for explainable human decision analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1334âĂŞ1343.
[65] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*. PMLR, 2778–2787.
[66] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.
[67] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. 2018. Model agnostic supervised local explanations. *Advances in Neural Information Processing Systems* 31 (2018).
[68] Erika Puiutta and Eric Veith. 2020. Explainable reinforcement learning: A survey. In *International Cross-domain Conference for Machine Learning and Knowledge Extraction*. Springer, 77–95.
[69] Jette Randløv and Preben Alstrøm. 1998. Learning to drive a bicycle using reinforcement learning and shaping. In *International Conference on Machine Learning (ICML '98)*, Vol. 98. 463–471.
[70] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. Association for Computing Machinery, 1135–1144.
[71] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* 1, 5 (2019), 206–215.
[72] Subham Sahoo, Christoph Lampert, and Georg Martius. 2018. Learning equations for extrapolation and control. In *International Conference on Machine Learning*. PMLR, 4442–4450.
[73] Christoph Salge, Cornelius Glackin, and Daniel Polani. 2014. Empowerment—An introduction. In *Guided Self-Organization: Inception*, 67–114.
[74] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. PMLR, 1889–1897.
[75] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv:1707.06347. Retrieved from https://arxiv.org/abs/1707.06347
[76] Pedro Sequeira and Melinda Gervasio. 2020. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artif. Intell.* 288 (2020), 103367.
[77] Hassam Ullah Sheikh, Shauharda Khadka, Santiago Miret, Somdeb Majumdar, and Mariano Phielipp. 2022. Learning intrinsic symbolic rewards in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN'22)*. 1–8. DOI: https://doi.org/10.1109/IJCNN55064.2022.9892256
[78] Tianmin Shu, Caiming Xiong, and Richard Socher. 2018. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=SJJQVZW0b
[79] Andrew Silva and Matthew Gombolay. 2019. Neural-encoding human experts' domain knowledge to warm start reinforcement learning. arXiv:1902.06007. Retrieved from https://arxiv.org/abs/1902.06007
[80] Sarath Sreedharan, Utkarsh Soni, Mudit Verma, Siddharth Srivastava, and Subbarao Kambhampati. 2022. Bridging the Gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. In *The 10th International Conference on Learning Representations (ICLR'22)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=o-1v9hdSult

[81] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. 2015. A refinement-based architecture for knowledge representation and reasoning in robotics. arXiv:1508.03891. Retrieved from https://arxiv.org/abs/1508.03891

[82] Alexander L. Strehl and Michael L. Littman. 2008. An analysis of model-based interval estimation for Markov decision processes. *J. Comput. Syst. Sci.* 74, 8 (2008), 1309–1331.

[83] Aaquib Tabrez and Bradley Hayes. 2019. Improving human-robot interaction through explainable reinforcement learning. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI '19)*. IEEE, 751–753.

[84] Yujin Tang, Duong Nguyen, and David Ha. 2020. Neuroevolution of self-interpretable agents. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 414–424.

[85] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

[86] Nicholay Topin and Manuela Veloso. 2019. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2514–2521.

[87] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, 11 (2008).

[88] Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark Neerincx. 2018. Contrastive explanations for reinforcement learning in terms of expected consequences. arXiv:1807.08706. Retrieved from https://arxiv.org/abs/1807.08706

[89] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5045–5054.

[90] Marco Virgolin, Andrea De Lorenzo, Francesca Randone, Eric Medvet, and Mattias Wahde. 2021. Model learning with personalized interpretability estimation (ml-pie). In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1355–1364.

[91] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL Tech.* 31 (2017), 841.

[92] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020. Shapley q-value: A local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7285–7292.

[93] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 3 (1992), 229–256.

[94] Bohan Wu, Jayesh K. Gupta, and Mykel Kochenderfer. 2020. Model primitives for hierarchical lifelong reinforcement learning. *Auton. Agents Multi-Agent Syst.* 34, 1 (2020), 1–38.

[95] Yueh-Hua Wu and Shou-De Lin. 2018. A low-cost ethics shaping approach for designing reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[96] H. Peyton Young. 1985. Monotonic solutions of cooperative games. *Int. J. Game Theory* 14, 2 (1985), 65–72.

[97] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE Trans. Pattern Anal. Mach. Intell.* (2022).

[98] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. 2016. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*. PMLR, 1899–1908.

[99] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. 2019. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*.

[100] Haodi Zhang, Zihang Gao, Yi Zhou, Hao Zhang, Kaishun Wu, and Fangzhen Lin. 2019. Faster and safer training by embedding high-level knowledge into deep reinforcement learning. arXiv:1910.09986. Retrieved from https://arxiv.org/abs/1910.09986

[101] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. 2021. A survey on neural network interpretability. *IEEE Trans. Emerg. Top. Comput. Intell.* (2021).

[102] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, Anind K. Dey, et al. 2008. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI International Conference on Artificial Intelligence (AAAI '08)*, Vol. 8. 1433–1438.

[103] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. arXiv:1909.08593. Retrieved from https://arxiv.org/abs/1909.08593

[104] Barret Zoph and Quoc Le. 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=r1Ue8Hcxg

[105] Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. 2019. Reward shaping via meta-learning. arXiv:1901.09330. Retrieved from https://arxiv.org/abs/1901.09330

Received 5 September 2022; revised 20 September 2023; accepted 25 September 2023