Internal Logical Induction for Pixel-Symbolic Reinforcement Learning

Jiacheng Xu
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
xujc@lamda.nju.edu.cn

Chao Chen
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
chenc@lamda.nju.edu.cn

Fuxiang Zhang
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
zhangfx@lamda.nju.edu.cn

Lei Yuan
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
Polixir Technologies
Nanjing, China
yuanl@lamda.nju.edu.cn

Zongzhang Zhang*
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
zzzhang@nju.edu.cn

Yang Yu
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
Polixir Technologies
Nanjing, China
yuy@nju.edu.cn

ABSTRACT

Reinforcement Learning (RL) has experienced rapid advancements in recent years. The widely studied RL algorithms mainly focus on a single input form, such as pixel-based image input or symbolic vector input. These two forms have different characteristics and, in many scenarios, will appear together, while few RL algorithms have studied the problems with mixed input types. Specifically, in the scenario where both pixel and symbolic inputs are available, symbolic input usually offers abstract features with specific semantics, which is more conducive to the agent's focus. Conversely, pixel input provides more comprehensive information, enabling the agent to make well-informed decisions. Tailoring the processing approach based on the properties of these two input types can contribute to solving the problem more effectively. To tackle the above issue, we propose an Internal Logical Induction (ILI) framework that integrates deep RL and rule learning into one system. ILI utilizes the deep RL algorithm to process the pixel input and the rule learning algorithm to induce propositional logic knowledge from symbolic input. To efficiently combine these two mechanisms, we further adopt a reward shaping technique by treating valuable knowledge as intrinsic rewards for the RL procedure. Experimental results demonstrate that the ILI framework outperforms baseline approaches in RL problems with pixel-symbolic input, and its inductive knowledge exhibits transferability advantages when pixel input semantics change.

*Zongzhang Zhang is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599393

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Reinforcement Learning, Rule Learning

ACM Reference Format:
Jiacheng Xu, Chao Chen, Fuxiang Zhang, Lei Yuan, Zongzhang Zhang, and Yang Yu. 2023. Internal Logical Induction for Pixel-Symbolic Reinforcement Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599393

1 INTRODUCTION

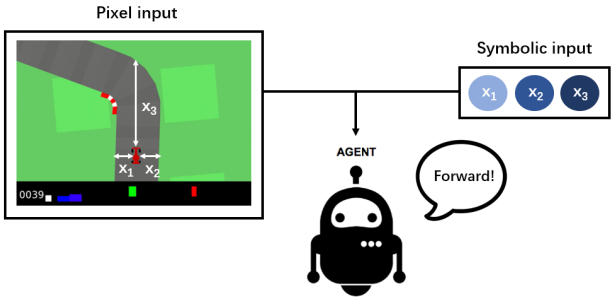


Figure 1: An illustration of the two forms of input in the self-driving car game.

Deep Reinforcement Learning (RL) is developing rapidly and has achieved remarkable success in Atari games [21], Go [44], robotics [1], etc. This reveals the tremendous potential of deep RL and it is considered to be a promising solution to real-world sequential decision-making problems. However, current RL algorithms mainly

focus on only a single type of input, such as image input expressed in pixel form or vector input expressed in symbolic form. Nevertheless, the scenario of mixed forms of input is widespread [14, 34, 52]. Take the autonomous driving scenario as an example [58]. The vehicle camera will capture the pixel input, while vehicle-borne radar, urban road information, and vehicle index information (such as speed, power, coordinates, etc.) are symbolic input.

In the mixed pixel-symbolic input scenario, the pixel input usually provides more comprehensive information. But a large number of samples are required to train the agent due to the complexity of task description in pixel form and neural networks. In contrast, the dimensionality of symbolic states is much smaller, and each state dimension has fixed and specific meanings. It may lack some information and result in the inability to get optimal performance, but training on the symbolic state will converge faster. The two forms of input are illustrated in the self-driving car game in Figure 1. The left frame is a pixel state, while the vector of the distances to each road edge is a symbolic state. We refer to this type of problem as the pixel-symbolic RL problem.

Existing RL algorithms mainly focus on single-form input, and only a few works revolve around multiple-form input, the main directions of which are: Multi-View Reinforcement Learning (MVRL) [26, 29] and Symbolic Deep Reinforcement Learning (SDRL) [4, 10, 30]. The former focuses on multi-view data, and each view shares common dynamics but adheres to different observation models. This setting concerns how to integrate information from multi-view data by representation learning [27], while the pixel input in our setting already contains the integrated information. The latter handles both high-dimensional sensory input and task-level symbol input under the hierarchical RL framework [36]. They focus on how to use task-level symbolic input for upper-level policy planning. In comparison, symbolic input in our setting can be directly employed for decision-making assistance. Thus, the granularity and usage of the symbolic input in SDRL and our setting are very different.

When encountering a mixed pixel-symbolic input scene, a common practice is to stitch the pixel input processed by convolutional neural networks with the original symbol and then use it as the input of the RL network [14, 49, 52]. This method does not make good use of the characteristics of symbolic input because the neural network still needs a large number of samples to learn to capture the semantics of symbolic input. And the high-dimensional representation of pixel input may interfere with the recognition of symbolic input. We should have special treatment for symbolic states.

One key point to solving the pixel-symbolic RL problem is to process symbolic input according to its characteristics to help the learning of deep neural network agents. Fortunately, there are many effective methods for processing symbolic data in the data mining field, and rule learning is one of the most influential and mature methods [18, 33, 50]. Usually, rule learning can discover effective logic rules even on a small amount of symbolic data, which helps quickly capture valuable experiences from interactions with the environment.

In this paper, we propose the Internal Logical Induction (ILI) framework to specifically deal with the pixel-symbolic RL problem. It integrates deep RL and rules learning into a system that separately processes pixel and symbolic input. To effectively combine these two components, we first construct a dataset from symbolic input for rule learning during the training process, based on which we can derive propositional logic knowledge. Then we adopt an adaptive mechanism to select valuable knowledge as the intrinsic reward to help the learning of deep RL agents.

Based on the above idea, we use DQN [31] as the implementation of deep RL and RIPPER [11] as the implementation of rule learning. The experimental results show that the ILI algorithm has a significant performance improvement over the baselines. In addition, the propositional logic knowledge gained from rule learning is transferable, which can help the agent learn faster on new tasks where the pixel input changes.

We summarize our contributions as follows:

(1) We formalize the RL problem with both pixel and symbolic input as the pixel-symbolic RL problem and analyze the characteristics and challenges of this problem.
(2) We propose a framework specifically designed to solve pixel-symbolic RL, called Internal Logical Induction (ILI), which brings together the advantages of deep RL and rule learning.
(3) The experimental results show that the method of constructing the rule learning dataset during training progress can derive valuable propositional logic knowledge. Then the ILI framework shows significant performance improvement in the pixel-symbolic RL problems compared to the baseline algorithm. Moreover, the propositional logic knowledge derived from the ILI framework helps migrate the agent's policy when the semantics of the pixel input change.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning (RL) [25] is a machine learning paradigm in which an agent learns an optimal policy by interacting with the environment. The common RL framework mainly consists of two parts: the environment and the agent. At each time step t, the agent chooses an action a_t and executes it according to the current state s_t of the environment, then the environment transitions to a new state s_{t+1} while giving the agent a reward r_t. The above is an interactive loop in RL that continues until the end of the task. The objective of RL is to maximize the cumulative reward $\mathbb{E}_\pi[\sum_t \gamma^t r_t]$, where γ is the discount factor.

Formally, RL can be expressed as a Markov decision process (MDP) problem. It can be defined as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where $\mathcal{S}$ is the state space, and s_t is the state at time step t; $\mathcal{A}$ is the action space, and a_t is the action at time step t; $\mathcal{R}$ is the reward function, and reward $r_t = R(s_t, a_t) \in \mathbb{R}$ is a function of state s_t and action a_t; $\mathcal{P}$ is the transition function, and $P(s, a, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ denotes the probability of state s_t transitioning to state s_{t+1} after taking action a_t.

The policy of RL agent $\pi(s)$ is a function defined over the state space. It can be a simple function or a neural network. Given an environment defined by a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, the aim of RL is to learn the policy $\pi(s)$ that can maximize accumulated reward.

Recently, much research has been done on applying deep learning to RL, which gives birth to deep RL [3]. Benefiting from the strong expression ability of the neural network, deep RL achieves remarkable performance.

The DQN [31] algorithm used as a baseline in this paper is a classic method of deep RL. It belongs to the Q-learning algorithm and uses a Q-network $Q(s, a; \theta)$ to approximate a Q-value function. We can derive the policy from the Q-value function by setting $\pi(s_t) = \arg\max_a Q(s_t, a; \theta)$. In order to stabilize the training process, DQN uses a Q-network $Q(s, a; \theta)$ and a target network $\hat{Q}(s, a; \hat{\theta})$, and parameter θ is updated as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta (r_t + \gamma \max_a \hat{Q}(s_{t+1}, a; \hat{\theta}) - Q(s_t, a_t; \theta)) , \quad (1)$$

where $\langle s_t, a_t, r_t, s_{t+1} \rangle$ is sampled from replay buffer and α is the learning rate. Parameter $\hat{\theta}$ will be regularly updated following the formula $\hat{\theta} \leftarrow (1 - \beta)\hat{\theta} + \beta\theta$, where β represents a hyperparameter.

2.2 Rule Learning

Rule learning is one of the most influential and mature fields of machine learning [18]. Many rule learning systems have been developed for propositional and relational learning [17, 19, 38]. Rule learning is particularly useful in intelligent data analysis and knowledge discovery tasks, where compact, transparent, and effective rules can be learned.

Given a set of pre-classified objects, usually described by attribute values, a rule learning system constructs one or more rules of the form:

$$\text{IF } \mathbf{f}_1 \text{ AND } \mathbf{f}_2 \text{ AND ... AND } \mathbf{f}_L \text{ THEN Class } = c_i$$

The condition part of the rule is a logical conjunction of symbolic features, where a feature $\mathbf{f}_k$ is a test that checks whether an example to be classified has a specified property. Features $\mathbf{f}_k$ typically have the form $\mathbf{A}_i = v_{i,j}$ for discrete attributes, and $\mathbf{A}_i < v$ or $\mathbf{A}_i \geq v$ for continuous attributes (here, v is a threshold value). The conclusion of the rule is a class value c_i, which means that for any example that satisfies the body of the rule, the rule predicts the class value c_i. We call $\mathbf{f}_1$ AND $\mathbf{f}_2$ AND ... AND $\mathbf{f}_L$ as rule body and Class $= c_i$ as rule head.

In this paper, we choose Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [11], which is a widely-used propositional rule learner, as the concrete rule learning implementation. It forms rules through a process of repeated growing and pruning. During the growth phase, the rules are grown by adding conditions that maximize an information gain criterion to fit the training data as closely as possible until the rule covers no negative examples. During the pruning phase, the rules are pruned by deleting conditions that maximize a pruning function until any deletion cannot improve the function's value to avoid overfitting, which can cause poor performance on unseen instances. The RIPPER algorithm has excellent performance, can handle large-scale noisy data, and supports numerical variables and multiple classes, which makes it applicable to a wider range of problems.

3 METHOD

In this paper, we focus on the RL problem with mixed input of pixel and symbolic types, which we call pixel-symbolic RL. We formulate pixel-symbolic RL problem as $M := \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, the only difference from standard MDP in Section 2.1 is that $\mathcal{S}$ here is the joint state space consists of pixel state $\mathcal{S}^p \in \mathbb{R}^{C \times H \times W}$ and symbolic state $\mathcal{S}^s \in \mathbb{R}^M$. C, H, W, and M represent the number of channels, image height, image width, and symbol vector length respectively.

The pixel state often corresponds to perceptual data, while the symbolic state can be derived from various sensor data (such as distance measurements or internal agent indicators) [55], system database queries (such as task-related numerical descriptions) [14], or some generic digital image processing techniques.

In many tasks, while the task logic embodied by symbolic input remains consistent, the semantics of pixel input may change. These changes may be due to changing lighting conditions, as visual sensors need to function both day and night [13], or scene changes, such as encountering a novel navigation scene under consistent robot navigation logic [46]. Agents originally trained to perform tasks in the original pixel state space often need to be retrained to function in the new space. Formally, both the old and new tasks share the same symbolic state space $\mathcal{S}^s$, action space $\mathcal{A}$, reward function $\mathcal{R}$, and state transition $\mathcal{P}$, while their pixel state spaces $\mathcal{S}^p$ differ. How to perform efficient policy transfer between old and new tasks is a topic worthy of further consideration.

We present an Internal Logic Induction (ILI) framework to solve the pixel-symbolic RL problem, which integrates deep RL and rule learning into one system. We train agents on pixel states using common deep RL algorithms. At the same time, the rule learning algorithm is applied to symbolic states, aiming to induce propositional logic knowledge from a small number of successful samples to guide the learning of agents. Since the samples used for logical induction are extracted from training trajectories, the training of ILI is actually a self-supervised process. A benefit of the ILI training framework is that the knowledge of propositional logic acquired during training can help us with the new task adaptation discussed above. The rest of this section will explain the ILI algorithm in detail.

First, we will describe how to derive effective propositional logic knowledge during the training progress in Section 3.1. Then, we will show how to use valuable propositional logic knowledge to help the learning of RL agents in Section 3.2. Finally, we will present an internal logical induction framework, which automatically summarizes propositional logic knowledge during the training process to assist the learning of the RL agent in Section 3.3.

3.1 Induce Propositional Logic Knowledge

Rule learning algorithms use supervised learning and require a specific dataset for training. This section details the process of preparing a dataset to induce knowledge in the form of propositional logic rules, hereafter referred to as propositional logic knowledge.

Here is a straightforward idea: we can use a pre-trained RL policy to sample in the environment, and after acquiring sufficient state-action pairs, the rule learning algorithm can be applied by setting the state as the rule body, and the action as the rule head. However, in many scenarios where rules must be generated during the training of an RL agent, it is essential to derive propositional logic knowledge directly from the training samples of the RL policy.

A vanilla solution is to use all the state-action pairs that the RL policy has collected so far in the environment as the training set for the rule learning algorithm, similar to a replay buffer in RL. But the training set has a large amount of data sampled by the near-random policy, as well as data collected by ongoing exploration,

which introduces considerable noise to the rule learning algorithm. This approach neglects the distinction between rule learning and RL, i.e., rule learning algorithms want to obtain the most accurate samples possible without the exploration mechanism in RL. Given these attributes of rule learning algorithms, enhancing the quality of samples used in rule learning is crucial.

Before introducing the method to reduce low-quality samples mentioned above, we formalize the definition as follows. Suppose that there are N trajectories in the training set, with the ith trajectory defined as $\text{Traj}_i = \{(s_0^s, a_0), (s_1^s, a_1), \cdots\}_i$ and its corresponding cumulative reward defined as R^{Traj_i}.

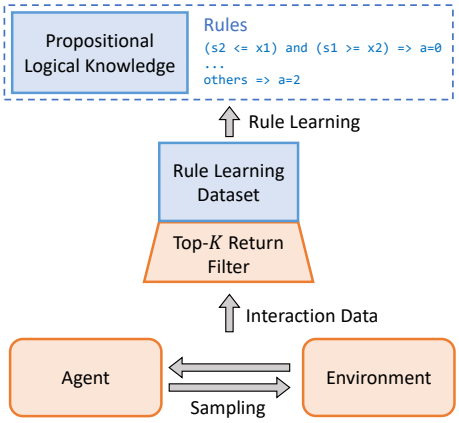


Figure 2: Propositional Logic Knowledge Induction.

We make two modifications to ensure the quality of the samples used for rule learning. First, we exclude state-action pairs where exploration actions occur, yielding filtered trajectories Traj'_i. Second, we only use the training samples with the top K returns. Specifically, the N filtered trajectories are sorted in descending order based on their cumulative rewards, and the resulting sorted trajectories are denoted as $\{\text{Traj}'_{j_1}, \text{Traj}'_{j_2}, \cdots, \text{Traj}'_{j_N}\}$. Expanding symbolic state-action pairs in each trajectory results in $\{\{(s_0^s, a_0), \cdots\}_{j_1}, \{(s_0^s, a_0), \cdots\}_{j_2}, \cdots, \{(s_0^s, a_0), \cdots\}_{j_N}\}$, and the current dataset $\mathcal{D}_{\text{rule}}$ used in rule learning will include the top K pairs. Given the processed training set $\mathcal{D}_{\text{rule}}$, the rule learning algorithm generates a propositional logic knowledge base KB in the following form:

```
(s2 <= x1) and (s1 >= x2) => a=0
(s2 <= x3) and (s1 >= x4) => a=1
...
others => a=2
```

Here s_i denotes a symbolic feature in the vector input, x_j denotes a threshold value, and a corresponds to the action suggested by the propositional logic knowledge base.

In this way, we can induce valid knowledge from a small sample of recent successes in time to help the agent focus on and learn them as rapidly as possible. The methodology diagram is shown in Figure 2 and the details of how to use the derived propositional logic knowledge will be described in the next section.

3.2 Use Propositional Logic Knowledge

In many cases, leveraging domain knowledge can significantly reduce redundant exploration. Here, we represent domain knowledge in the form of propositional rules and use propositional logic knowledge as a supervisory signal to guide the agent's learning.

We store all propositional rules in the knowledge base KB in order, and the matching process is completed once obtaining a satisfying match. At each time step, the environment's returned symbolic state is sent to the knowledge base. If the state matches the rule body, then the knowledge base will give the corresponding rule head, i.e., the action suggested by the domain knowledge. If no matching rule body is found in the knowledge base, no advice is given for this state.

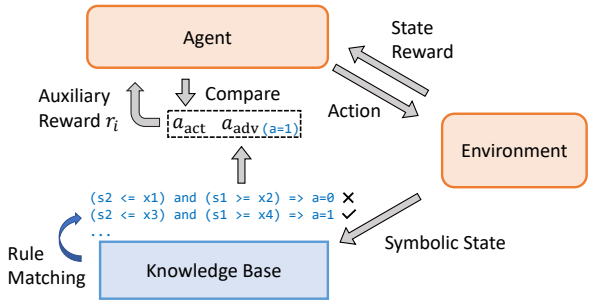


Figure 3: The Use of Propositional Logic Knowledge.

After obtaining the suggested actions, we use the knowledge base's suggestions as intrinsic rewards to facilitate RL policy training. The intrinsic reward r_i is $+x$ if the action a_{act} chosen by the agent is consistent with the action a_{adv} advised by the knowledge base (x is a positive number correlated to the environmental reward scale), and it is zero if they are inconsistent. The combined reward takes the following form:

$$r = r_e + \lambda r_i \,, \tag{2}$$

$$r_i = \begin{cases} +x, & \text{if} \quad a_{\text{act}} = a_{\text{adv}} \\ 0, & \text{otherwise} \,, \end{cases} \tag{3}$$

where r_e represents the environment reward and λ is the coefficient controlling the intrinsic rewards degree. We train the RL policy using the combined reward instead of solely the environmental reward. The specific form is shown in Figure 3.

The introduction of propositional logic knowledge can enhance the agent's performance. Furthermore, the transparency and readability of propositional rules themselves simplify the integration of human priors and constraints, offering potential advantages for the investigation of incorporating human preferences or security.

3.3 Internal Logical Induction

We have described the induction of propositional logic knowledge using rule learning algorithms in Section 3.1 and the utilization of the inducted valuable propositional logic knowledge to facilitate the agent's learning in Section 3.2. In this section, we will propose an adaptive reward reshaping technique that connects the two methods above to solve the pixel-symbolic RL problem.

Human cognition [45] is an excellent system for processing pixel-symbolic input. It consists of two remarkable capabilities: perception and induction. The former processes sensory information, while the latter primarily operates symbolically. It is frequently posited that advanced intelligent technologies will emerge when machine learning and logical induction are seamlessly integrated [56]. Inspired by this, we propose an Internal Logical Induction (ILI) framework that integrates rule learning and RL. This framework demonstrates both perception and induction capabilities, allowing the two modules to mutually enhance the agent's performance. In the ILI framework, we use a deep RL algorithm to process the pixel state, similar to standard practices. Simultaneously, we use a rule learning algorithm to process the symbolic state to obtain the propositional logic knowledge, which is used to generate intrinsic rewards that guide the deep learning algorithm. The framework's illustration is provided in Figure 4.

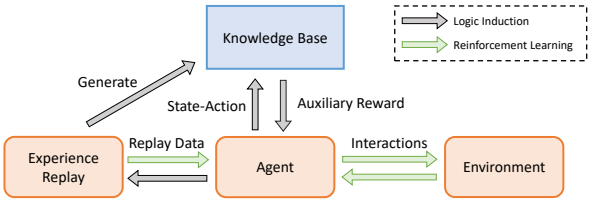


Figure 4: Internal Logical Induction.

For each time step, we can use the method outlined in Section 3.1 to induce the propositional logic knowledge base. To ensure the knowledge base's stability, we evaluate the current knowledge base in the environment at intervals and replace it when it outperforms the optimal one. Then the optimal propositional logic knowledge will be provided to the agent for guidance based on the method described in Section 3.2. It has to be mentioned that we should select the timing for generating supervision signals wisely to prevent potential adverse effects, such as training stagnation. Subsequently, we will elaborate on the design thinking about the intrinsic reward coefficient factor λ.

Propositional logic knowledge in ILI serves an exploitation role. Therefore, it is inappropriate to use a fixed pattern decay for the λ coefficient, as is customary in intrinsic motivation exploration algorithms [9], a class of algorithms that often uses intrinsic motivation to obtain exploration data. Blind exploitation can hinder agent learning. For example, in the beginning, the accuracy of the extracted propositional logic knowledge is low. Relying on this knowledge at this stage will only make it repeat the action it has tried before. During the middle phase of training, it is easy to get stuck in local optima by constantly focusing on past successes. In the later stage of training, referring to a rule of thumb [47], external reward or supervision should be removed to avoid bias. The above analysis demonstrates that finding a universal decay function is difficult.

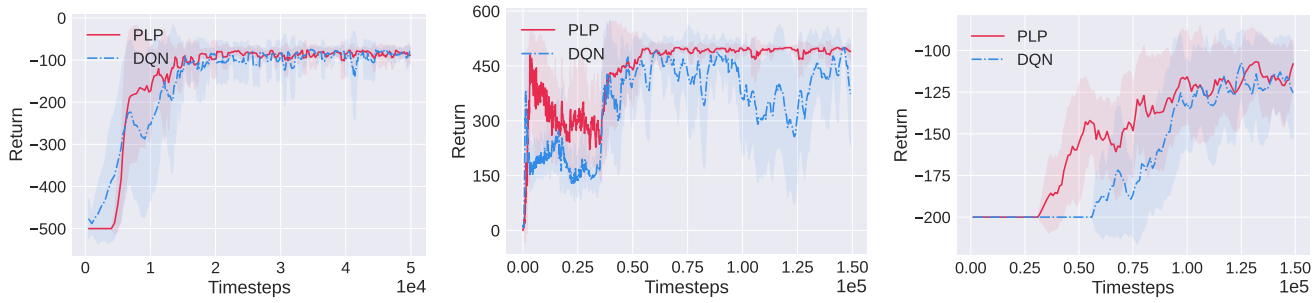
Considering the above characteristics of internal logic induction, we propose a self-adaptive λ coefficient adjustment method. We define the long-term average return as the average return of the last p episodes, denoted by $R^{\text{Traj}}_{\text{l\_avg}}$, and the short-term average return as the average return of the last q episodes satisfying $q < p$, denoted by $R^{\text{Traj}}_{\text{s\_avg}}$. When the short-term average return is greater than the long-term average return, it indicates that we have explored higher-return trajectories. We should use the propositional logic knowledge obtained by induction because rule learning can extract knowledge from successful experiences more rapidly than neural networks. Conversely, when the short-term return is less than or equal to the long-term average return, it indicates that our exploration strategy does not obtain new effective information. Emphasizing past successful experiences in this situation would cause the agent to become trapped in a local optimum, so we should not refer to the propositional logic knowledge. Finally, the λ is calculated as follows:

$$\lambda = \max\left(\frac{R^{\text{Traj}}_{\text{s\_avg}} - R^{\text{Traj}}_{\text{l\_avg}}}{R^{\text{Traj}}_{\text{s\_avg}}}, 0\right). \tag{4}$$

Algorithm 1 An Implementation of Internal Logical Induction

1: **Initialize:** Q-network $Q(s^p, a; \theta)$, target Q-network $\hat{Q}(s^p, a; \hat{\theta})$,
2: DQN replay buffer $\mathcal{D}_{\text{dqn}}$, rule learning dataset $\mathcal{D}_{\text{rule}}$,
3: propositional logic knowledge base *KB*.
4: **for** $t = 1$ to T **do**
5: With probability ϵ select a random action a_t.
6: Otherwise select $a_t = \arg\max_a Q(s^p_t, a; \theta)$.
7: Execute action a_t, get reward r_t, next pixel state s^p_{t+1} and next symbolic state s^s_{t+1}.
8: Store $< s^p_t,\ a_t,\ r_t,\ s^p_{t+1} >$ into DQN replay buffer $\mathcal{D}_{\text{dqn}}$.
9: Sample a batch of transitions $\mathcal{B}$ from $\mathcal{D}_{\text{dqn}}$.
10: Modify $\{r_t\}$ in $\mathcal{B}$ with *KB* according to Eq. 2.
11: Update Q-network with modified $\mathcal{B}$ by Eq. 1.
12: Update target Q-network every C steps.
13: **if** done **then**
14: Get current trajectory Traj_i and cumulative reward R^{Traj_i} to generate $\mathcal{D}_{\text{rule}}$ according to Section 3.1.
15: Train RIPPER on dataset $\mathcal{D}_{\text{rule}}$ to get new *KB*.
16: Adjust the intrinsic reward coefficient λ by Eq. 4.
17: **end if**
18: **end for**

In this paper, we use the deep RL algorithm DQN and the rule learning algorithm RIPPER as a concrete implementation of the internal logical induction framework. The complete pseudo-code can be found in Algorithm 1. During the vanilla DQN sampling progress, the agent selects action a_t based on Q-network. After the action is executed, the environment will return the next state s^p_{t+1}, s^s_{t+1} and reward r_t, and the state-action pair will be stored in the DQN replay buffer (lines 5~8). Then, the knowledge base will be used in the training of DQN. After sampling a batch of transitions from the replay buffer, the reward will be modified according to its consistency with the knowledge base, and the agent will update its Q-networks with the modified batch (lines 9~12). When an episode terminates, the knowledge base and intrinsic coefficient will be updated according to the methods introduced in Sections 3.1 and 3.3 (lines 13~17).



(a) Acrobot (b) CartPole (c) MountainCar

Figure 5: Performance of the compared methods on three classic control tasks.

4 EXPERIMENTS

In this section, we will assess the effectiveness of the proposed ILI framework in the following aspects. First, we will verify the validity of the propositional logic knowledge learned using rule learning. Then, we will confirm the superiority of the ILI framework on the pixel-symbolic RL problem in terms of convergence speed or performance. Finally, we will show the additional benefit of propositional logic knowledge, induced by the ILI framework, in facilitating knowledge transfer during semantic variations in pixel input.

4.1 Experiment Settings

We will perform the experiments on the first question with three classical control tasks, MountainCar, CartPole, and Acrobot. They have a continuous state space and a discrete action space. Their state spaces are of vector type, and each dimension can be viewed as a symbol with practical meaning. Take Acrobot as an example. It possesses six symbolic features: the cosine and sine of θ_1 and θ_2, and the angular velocity of θ_1 and θ_2. Here θ_1 and θ_2 are two rotational joint angles. It has three executable actions to perform -1, 0, or 1 torque to the actuated joint. Note that in the first study, no pixel input is involved, and the input is only a symbolic state. We wish to verify the validity of the method proposed in Section 3.1 for obtaining propositional logic knowledge, which is an essential guarantee of the validity of the supervised signal in the ILI framework.

The second and third experiments will be performed on the modified environments FlappyBird, CarRacing, and CollectHealth with both pixel and symbolic inputs. The decision-relevant symbolic state could either be an extraction of key information from the pixel state or a supplement to the missing information in the pixel state. For the FlappyBird environment, we provide the horizontal and vertical coordinates relative to the next pipe, but information about the flight speed, the distance to the top and bottom boundary, etc., needs to be obtained from the image input. For the CarRacing environment, we provide the distances to the left, right, and front of the track. Still, information about car speed, bends, etc. needs to be extracted from the image input. For the CollectHealth environment, we provide the relative direction to the health kits, which is not always available in the pixel state due to 3D viewpoint limitations. The rest of the task information, such as the distance to the health kits, needs to be obtained from the pixel state. Each episode in these three environments generates a completely new map, bringing a greater challenge to the tasks. In addition, since the third experiment involves the migration of propositional logic knowledge, we will generate tasks with new pixel input by modifying the pixel values.

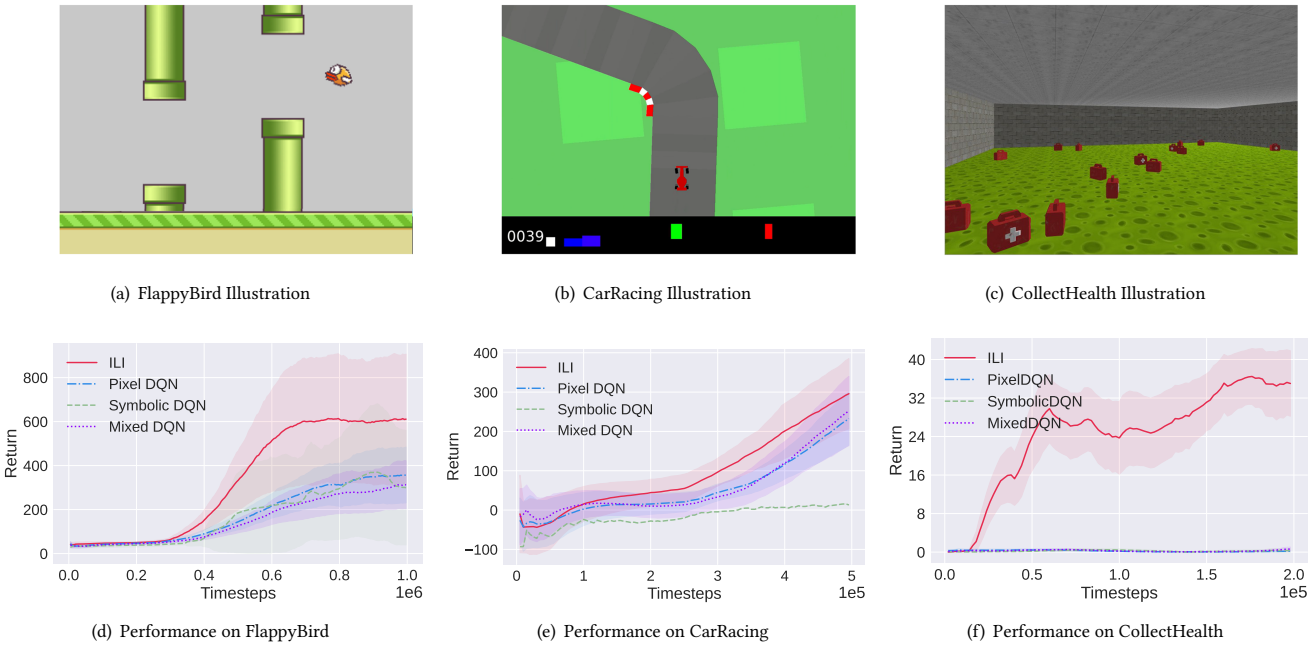
We use DQN as the RL algorithm and RIPPER as the rule learning algorithm, both are among the most influential algorithms in their respective fields. It is worth noting that the ILI framework does not restrict specific RL or rule learning algorithms, and any unilateral improvement will likely lead to further improvements in the final performance. The implementation of DQN follows Stable Baseline3 [39], and RIPPER uses Weka's default implementation [20].

4.2 Experiment Results

RQ1: How valuable is propositional logic knowledge obtained from rule learning?

In the three classical control tasks containing only the symbolic state, we use DQN to learn directly in the symbolic environment and construct a rule learning dataset using the method proposed in Section 3.1. At each evaluation moment, we test the performance of the propositional logical knowledge derived from the rule learning dataset, represented as Propositional Logic Policy (PLP). This design aims to verify the effectiveness of logic knowledge induction proposed in Section 3.1, which is the logical reasoning foundation of ILI. So the experiment does not involve using logical knowledge to assist DQN's learning.

The experimental results in Figure 5 show that the derived PLP matches the asymptotic performance of DQN across all tested environments but with superior sample efficiency. This advantage is particularly evident in the MountainCar environment, which places the highest demands on an agent's exploratory capabilities due to a sparse reward structure, only offering positive rewards upon task completion. When the agent successfully explores one trajectory, the PLP can quickly induce the knowledge from that successful trajectory to complete the task. In contrast, the DQN requires a learning period, with the PLP completing the task in 30K time steps, compared to the DQN's 50K. In the CartPole environment, the PLP additionally shows robustness when a jitter occurs during the DQN training. Minor impact is observed on it due to the fluctuations in



(a) FlappyBird Illustration (b) CarRacing Illustration (c) CollectHealth Illustration

(d) Performance on FlappyBird (e) Performance on CarRacing (f) Performance on CollectHealth

Figure 6: Three RL environments and performance comparison of different methods.

the performance of the sampling policy, as the focus remains on the most valuable samples. The observed advantage aligns with the expectation that rule learning responds more swiftly and stably to successful samples than deep RL.

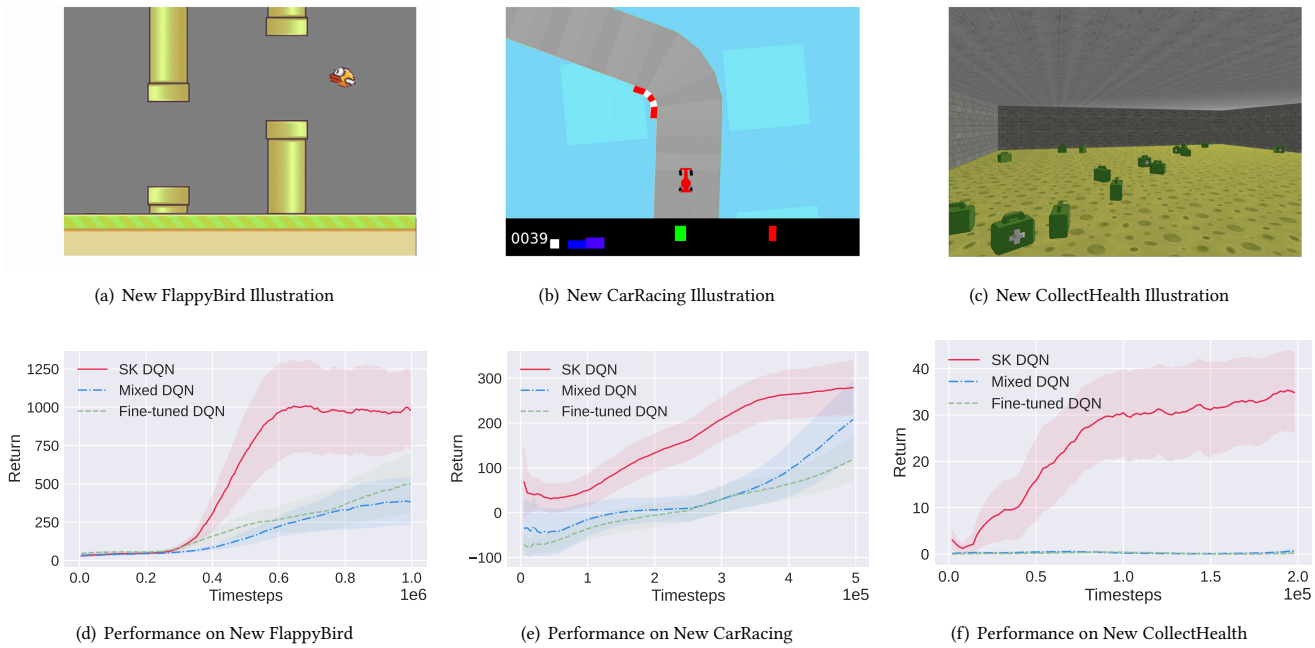
RQ2: How does ILI perform on the RL problem with pixel-symbolic inputs?

In this experiment, we modify three classic pixel-state environments FlappyBird, CarRacing, and CollectHealth to provide additional symbolic states. Therefore, the state returned by each environment consists of a high-dimensional pixel input and a low-dimensional symbolic input. We design three baseline algorithms, all are trained using DQN, but the inputs of DQN are different. The inputs of the three algorithms are pixel-only, symbol-only, and a mixture of pixel-symbol input. We call them Pixel DQN, Symbolic DQN, and Mixed DQN, respectively. Mixed DQN is designed following conventional methods [14, 49, 52]. We first process the pixel input through the convolutional neural network to yield a feature space vector and then splice it with the original symbolic input to the fully connected network for decision-making.

As seen from the curves in Figure 6, the ILI algorithm shows better performance and sample efficiency in all environments, and the performance of the Mixed DQN is comparable to that of the Pixel DQN. These results demonstrate that ILI handles the symbolic state more effectively than the spliced mixed input approach. Additionally, the Symbolic DQN struggles to learn useful strategies due to the absence of certain decision-critical information in its input. In the FlappyBird environment, ILI embodies a significant advantage because the symbolic state provided by this environment is critical for decision. Judging the horizontal and vertical distances from the gap is a prerequisite for completing the task. So the propositional logic knowledge obtained by ILI can help the agent learn to cross the tube quickly when it is very close to the gap. In the CollectHealth environment, acquiring rewards through random exploration in randomly generated 3D environments is exceedingly difficult for baseline algorithms. The sparse reward nature of the environment imposes greater demands on the agent's credit assignment, leading baseline algorithms to require numerous successful explorations before gradual performance enhancement. In contrast, logical induction enables the extraction of effective knowledge from a limited number of successful samples, while the reward shaping method efficiently guides the reinforcement learning strategy. In the CarRacing environment, the symbolic input lacks some crucial decision-making elements like car speed and road corners, so the improvement is less prominent compared to FlappyBird and CollectHealth. We infer that the quality of symbolic input has a certain influence on the performance improvement of the ILI. Handling scenarios where symbolic input cannot directly assist decision-making remains an area for further investigation.

RQ3: How transferable is the exported propositional logic knowledge on a task with a new pixel representation?

In scenarios involving pixel-symbolic input, a change in the pixel input's semantics often results in a new task, even though the symbolic input's semantics remains constant. To simulate this situation, we altered the primary entity colors in the pixel input of FlappyBird, CarRacing, and CollectHealth, as shown in the upper half of Figure 7, while the logic of the environment remains the same. The change in their image input challenges general deep learning algorithms a lot. In this situation, efficient use of consistent symbolic input can significantly reduce the difficulty of learning the new task. We will obtain a propositional logic knowledge base if we



(a) New FlappyBird Illustration (b) New CarRacing Illustration (c) New CollectHealth Illustration

(d) Performance on New FlappyBird (e) Performance on New CarRacing (f) Performance on New CollectHealth

Figure 7: Three new RL environments and performance comparison of different methods.

train deep RL algorithms within the ILI framework. Then, we can use this propositional logic knowledge to assist the deep learning algorithm on the new task with changed pixel input. The practical implementation, built upon the reward shaping method delineated in Section 3.2, employs a standard transfer-setting technique [15], which substitutes the agent's exploratory action with propositional logic knowledge's advice action, termed as Symbolic Knowledge DQN (SK DQN). For comparison, we design two baseline algorithms. One is the Mixed DQN mentioned in RQ2. The other adopts the method of model reuse for policy migration, i.e., the Mixed DQN network trained on the old task is further trained on the new task, called Fine-tuned DQN.

The experimental results in the lower half of Figure 7 depict that the propositional logic knowledge gleaned from the original task training can effectively aid policy training on the new task. One point worth noting is that in the CarRacing environment, SK DQN showed a clear advantage at the start. The reason is that there are many straightaways in a randomly generated map, and we will get a positive reward from the environment when we advance a distance. Keeping forward is easily transferable knowledge, so SK DQN knew about it at the first evaluation time. Furthermore, there is no significant difference in the performance of Mixed DQN and Fine-tuned DQN. This indicates that when the pixel state undergoes significant changes, the neural network parameters trained on the original task become nearly irrelevant, which aligns with our expectations. Preserving the information induced from symbolic input in the propositional logic knowledge base can avoid the catastrophic forgetting problem of deep learning. This has considerable potential in many migration scenarios, such as learning a generic skill, storing it in the propositional logic knowledge base, and then using it in different task scenarios.

5 RELATED WORK

5.1 RL with Multiple Input Types

Existing RL algorithms mainly focus on single-type inputs, and only a few works focus on multi-type inputs, and their representative directions are: Multi-View Reinforcement Learning (MVRL) [26, 29] and Symbolic Deep Reinforcement Learning (SDRL) [4, 30].

MVRL focuses on multi-view data, and each view shares common dynamics but adheres to different observation models. It wants to obtain as comprehensive information about the task as possible from multiple views, and its main difficulty is measuring the content similarity between the heterogeneous samples. A range of strategic approaches have been considered to address this issue, including alignment representation [35], joint representation [8], and shared and specific representation [51], among others. Despite the diversity of approaches, their central principle aligns each endeavor to establish a common representation space by exploring semantic relationships in multi-view data. This setting focuses on how to integrate information from multiple observation information sources. In contrast, our proposed pixel-symbolic RL setting assumes that the pixel input already contains the integrated information, and we are more concerned with how to utilize the symbolic information.

SDRL handles both high-dimensional sensory inputs and task-level symbol inputs. They usually use a hierarchical learning framework [36]. The upper-level strategy uses symbolic AI planning on symbolic input to select tasks, and the lower-level strategy uses RL algorithms on pixel input for task execution. Neuro-Symbolic Relational Transition Models (NSRTs) [10] is one of the representative

algorithms, which contains both symbolic and neural components, enabling a bilevel planning scheme where continuous planning with A* search in an outer loop guides the neural models in an inner loop. The symbolic input of SDRL is about the description of task status for upper-level policy planning, while symbolic input in our setting can be directly employed for decision-making assistance. Thus, the granularity and usage of the symbolic input in SDRL and our setting are very different.

5.2 Sample Efficient RL

Current RL methodologies necessitate extensive interaction data, particularly in environments with high-dimensional pixel inputs, limiting their broader implementation [54]. Consequently, enhancing sample efficiency has emerged as a central focus of RL research over recent years, with most prior work concentrating on the exploration and exploitation of agents [28]. Exploration involves gathering high-value samples from the environment, while exploitation emphasizes effectively utilizing these environmental samples.

Intrinsic motivation-based exploration algorithms are effective exploration strategies in high-dimensional environments [9]. They promote agent learning by designing intrinsic motivation functions and combining intrinsic and environmental rewards. Different algorithms use different design methods. Intrinsic Curiosity Module (ICM) [37] estimates prediction errors of the environmental dynamics, Random Network Distillation (RND) [7] counts the state novelty, and Variational Information Maximizing Exploration (VIME) [23] calculates the information gain.

Many types of algorithms focus on efficient utilization, and we choose some of these representative works to present. Prioritized Experience Replay (PER) [43] considers transitions with high TD error more valuable and replays them more frequently. Generalizable Episodic Memory (GEM) [24] can be viewed as an extreme way of exploiting past experiences because the agent repeats the same actions that gave the best outcome in the past. Hindsight Experience Replay (HER) [2] designs a mechanism to achieve implicit curriculum learning by generating goals.

We refer to the concept of intrinsic motivation in RL and use the propositional logic knowledge induced in the ILI process as an intrinsic reward, guiding agent learning together with environmental rewards. But the purpose of intrinsic reward here is mining valuable information from past sampling, similar to the ideas contained in exploitation-focused algorithms.

5.3 Transfer RL

Transfer RL aims to harness knowledge from the source domain to facilitate learning in the target domain, circumventing the need for learning from scratch [57]. Depending on the type of knowledge transferred, Transfer RL can be categorized into the following types.

The first is policy transfer, where a set of teacher policies trained on the source domain are available. The aim is to train a student policy on the target domain. This part of the work mainly focuses on two directions: policy distillation and policy reuse. Policy distillation [22] trains the student policy by assembling teacher policies, a conventional way of implementation is to minimize the divergence of action distributions between the teacher policy and student policy [40, 53]; and policy reuse directly reuses policies from source tasks to build the target policy, a straightforward way is to adjust the probability of using each teacher policy in training [15]. Additionally, a more recent work greedily chooses the action of the highest Q-value among all policies [6].

The second is representation transfer, which assumes the existence of a task-invariance subspace, allowing representations from the source domain to be transferred to the target domain, leveraging the powerful approximation capabilities of deep neural networks. By introducing innovative neural network structures such as progressive neural network [41] and PathNet [16], representations learned from previous tasks can be reused in a progressive manner. In addition, when the differences between source and target domains only lie in reward function, by learning successor representations [12] or using universal function approximation [5, 42], representations learned in the source domain can be reward-independent and thus can be transferred directly to the target domain.

We highlight that our setting in the RQ3 deviates from traditional transfer RL as the knowledge transferred is neither a policy nor a representation. Instead, it is the propositional logic knowledge that remains semantically unchanged. By leveraging the supervised signal of this logic knowledge, we can boost the learning of the agent in the target domain.

6 CONCLUSION

In this paper, we focus on RL with the mixed input of pixel and symbolic type, an area to which more research attention needs to be paid. Considering the characteristics of these two inputs, we propose an Internal Logical Induction (ILI) framework that integrates deep RL and rule learning into one system. ILI trains deep RL algorithms on the pixel input and uses rule learning algorithms to induce propositional logic knowledge from symbolic input. To connect these two modules, we further raise a reward shaping technique by treating valuable knowledge as intrinsic rewards for the agent's training. Experimental results show that the ILI framework works well and its inductive knowledge has the advantage of transferability. The ILI framework has no restrictions on the specific selection of RL and rule learning algorithms, reflecting good compatibility. Any improvement in the two components may lead to an overall improvement.

Deep learning algorithms endow RL agents with superior perception capabilities. In contrast, the reasoning ability of current agents still has a lot of room for improvement. Enabling agents to reason is an important and challenging research direction, and we hope that the pixel-symbolic RL problem can somewhat inspire researchers in this direction.

ACKNOWLEDGEMENTS

We thank the reviewers for their insightful and valuable comments. We thank Ke Xue, Pengyuan Wang, and Feng Chen for providing helpful comments. This work is supported by the National Key Research and Development Program of China (2022ZD0114804), the National Science Foundation of China (62276126), the Fundamental Research Funds for the Central Universities (14380010), and the program B for Outstanding Ph.D. candidate of Nanjing University.

REFERENCES

[1] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. 2020. Learning Dexterous In-Hand Manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5048–5058.

[3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[4] Masataro Asai and Christian Muise. 2020. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In *International Joint Conference on Artificial Intelligence (IJCAI)*. 2676–2682.

[5] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. 2018. Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. In *International Conference on Machine Learning (ICML)*. 501–510.

[6] Andre Barreto, Will Dabney, Remi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. 2017. Successor Features for Transfer in Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 4055–4065.

[7] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*.

[8] Ning Chen, Jun Zhu, Fuchun Sun, and Eric P. Xing. 2012. Large-Margin Predictive Latent Subspace Learning for Multiview Data Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 12 (2012), 2365–2378.

[9] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2004. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1281–1288.

[10] Rohan Chitnis, Tom Silver, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2022. Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 4166–4173.

[11] William W. Cohen. 1995. Fast Effective Rule Induction. In *International Conference on Machine Learning (ICML)*. 115–123.

[12] Peter Dayan. 1993. Improving Generalization for Temporal Difference Learning: The Successor Representation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 613–624.

[13] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5982–5994.

[14] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[15] Fernando Fernández and Manuela M. Veloso. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 720–727.

[16] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint arXiv:1701.08734* (2017).

[17] Eibe Frank and Ian H. Witten. 1998. Generating Accurate Rule Sets Without Global Optimization. In *International Conference on Machine Learning (ICML)*. 144–151.

[18] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. 2012. *Foundations of Rule Learning*. Springer.

[19] Brian R. Gaines and Paul Compton. 1995. Induction of Ripple-Down Rules Applied to Modeling Large Databases. *Journal of Intelligent Information Systems* 5, 3 (1995), 211–228.

[20] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

[21] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*. 3215–3222.

[22] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).

[23] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1109–1117.

[24] Hao Hu, Jianing Ye, Guangxiang Zhu, Zhizhou Ren, and Chongjie Zhang. 2021. Generalizable Episodic Memory for Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 4380–4390.

[25] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

[26] Akira Kinose, Masashi Okada, Ryo Okumura, and Tadahiro Taniguchi. 2022. Multi-View Dreaming: Multi-View World Model with Contrastive Learning. *arXiv preprint arXiv:2203.11024* (2022).

[27] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. 2020. Contrastive Representation Learning: A Framework and Review. *IEEE Aerospace Conference* 8 (2020), 193907–193934.

[28] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2021. SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 6131–6141.

[29] Minne Li, Lisheng Wu, Jun WANG, and Haitham Bou Ammar. 2019. Multi-View Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1418–1429.

[30] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. 2019. SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning. In *AAAI Conference on Artificial Intelligence (AAAI)*. 2970–2977.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level Control Through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533.

[32] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. In *International Conference on Machine Learning (ICML)*. 3878–3887.

[33] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2021. An Embedding-Based Approach to Rule Learning in Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2021), 1348–1359.

[34] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. 2019. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113* (2019).

[35] Hyunjong Park, Sanghoon Lee, Junghyup Lee, and Bumsub Ham. 2021. Learning by Aligning: Visible-Infrared Person Re-identification using Cross-Modal Correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 12026–12035.

[36] Shubham Pateria, Budhitama Subagdja, Ah-Hwee Tan, and Chai Quek. 2022. Hierarchical Reinforcement Learning: A Comprehensive Survey. *ACM Computing Surveys (CSUR)* 54, 5 (2022), 109:1–109:35.

[37] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *International Conference on Machine Learning (ICML)*. 2778–2787.

[38] J. Ross Quinlan. 1990. Learning Logical Definitions from Relations. *Machine Learning* 5 (1990), 239–266.

[39] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8.

[40] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295* (2015).

[41] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *arXiv preprint arXiv:1606.04671* (2016).

[42] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*. 1312–1320.

[43] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*.

[44] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (2016), 484–489.

[45] Robert L Solso, M Kimberly MacLin, and Otto H MacLin. 2005. *Cognitive Psychology*. Pearson Education New Zealand.

[46] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. 2021. The Distracting Control Suite - A Challenging Benchmark for Reinforcement Learning from Pixels. *arXiv preprint arXiv:2101.02722* (2021).

[47] Adrien Ali Taïga, William Fedus, Marlos C. Machado, Aaron C. Courville, and Marc G. Bellemare. 2020. On Bonus Based Exploration Methods in The Arcade Learning Environment. In *International Conference on Learning Representations*

(ICLR).

[48] Yunhao Tang. 2020. Self-Imitation Learning via Generalized Lower Bound Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS).* 13964–13975.

[49] Yang Wang. 2020. Survey on Deep Multi-modal Data Analytics: Collaboration, Rivalry and Fusion. *arXiv preprint arXiv:2006.08159* (2020).

[50] Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. 2019. Generalized Linear Rule Models. In *International Conference on Machine Learning (ICML).* 6687–6696.

[51] Jinglin Xu, Wenbin Li, Xinwang Liu, Dingwen Zhang, Ji Liu, and Junwei Han. 2020. Deep Embedded Complementary and Interactive Information for Multi-View Classification. In *AAAI Conference on Artificial Intelligence (AAAI).* 6494–6501.

[52] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, Yinyuting Yin, Bei Shi, Liang Wang, Tengfei Shi, Qiang Fu, Wei Yang, Lanxiao Huang, and Wei Liu. 2020. Towards Playing Full MOBA Games with Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS).* 621–632.

[53] Haiyan Yin and Sinno Jialin Pan. 2017. Knowledge Transfer for Deep Reinforcement Learning with Hierarchical Experience Replay. In *AAAI Conference on Artificial Intelligence (AAAI).* 1640–1646.

[54] Yang Yu. 2018. Towards Sample Efficient Reinforcement Learning. In *International Joint Conference on Artificial Intelligence (IJCAI).* 5739–5743.

[55] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* 8 (2020), 58443–58469.

[56] Zhi-Hua Zhou. 2019. Abductive Learning: Towards Bridging Machine Learning and Logical Reasoning. *Science China Information Sciences* 62, 7 (2019), 76101:1–76101:3.

[57] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv preprint arXiv:2009.07888* (2020).

[58] Zeyu Zhu and Huijing Zhao. 2022. A Survey of Deep RL and IL for Autonomous Driving Policy Learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 9 (2022), 14043–14065.

A HYPER-PARAMETERS

The implementation of DQN follows Stable Baseline3[1]. Below are three structures used in experiment.

- Symbolic DQN:
  `Symbolic state => MLP(256) => relu => MLP(256)=> relu => MLP(number of actions)`
- Pixel DQN:
  `Pixel state => Conv(32, 8, 4, 0) => relu => Conv(64, 4, 2, 0) => relu => Conv(64, 3, 1, 0) => relu => Flatten => MLP(256) => rule => MLP(128) => relu => MLP(128) => relu => MLP(number of actions)`
- Mixed DQN:
  `Pixel state => Conv(32, 8, 4, 0) => relu => Conv(64, 4, 2, 0) => relu => Conv(64, 3, 1, 0) => relu => Flatten => MLP(256) => Concat(Symbolic state) => relu => MLP(128) => relu => MLP(128) => relu => MLP(number of actions)`

Here, `MLP(n)` denotes a fully-connected layer with output size of n; `Conv(c, k, s, p)` denotes a 2D convolution layer of output channel c, kernel size k, stride s, and padding p; `Concat(x)` denotes the operation of connecting the output of last layer with x; `relu` denotes a rectified linear unit.

Other hyperparameters are listed in Table 1. The front of the slash indicates the value taken in the RQ1 experiment, and the back indicates the value taken in the RQ2 and RQ3 experiments.

Table 1: Hyperparameters in Experiments

Hyperparameter	Value
learning rate lr	0.0003 / 0.0001
discount factor γ	0.99
buffer size $\lvert\mathcal{D}_{\text{dqn}}\rvert$	30000 / 50000
batch size $\lvert\mathcal{B}\rvert$	64
target update interval C	100 / 10000
ϵ end	0.01
ϵ decay	30000 / 100000
batch size $\lvert\mathcal{B}\rvert$	64
episodes of long-term average return p	30
episodes of short-term average return q	5
size of rule learning dataset TopK	2000

B ADDITIONAL EXPERIMENT

B.1 Experiments of Combining ILI with the Self-Imitation Learning Algorithm

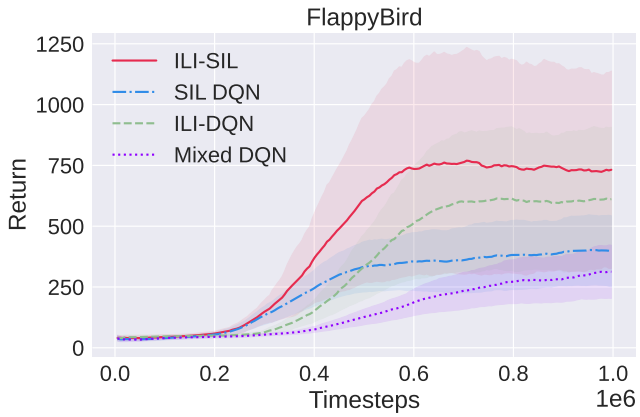


Figure 8: Empirical results of combining ILI with the self-imitation algorithm.

[1]https://github.com/DLR-RM/stable-baselines3

We claim that the Internal Logical Induction (ILI) framework can effectively integrate Reinforcement Learning (RL) algorithms and rule learning algorithms. To further illustrate its effectiveness, we chose Self-Imitation Learning DQN (SIL DQN) [32, 48] as our RL algorithm. SIL DQN is recognized for its sample efficiency and encourages the agent to imitate actions for which the Monte Carlo return, i.e., the return of the trajectory obtained by Monte Carlo sampling, is greater than the value function estimate. We named this combination ILI-SIL. Figure 8 displays the performance comparison between ILI-SIL, SIL DQN, ILI-DQN, and Mixed DQN within the FlappyBird Environment. It is clear that both DQN and SIL algorithms exhibit significant performance improvements when augmented with ILI. Furthermore, the superiority of ILI-DQN to SIL DQN indicates that guidance from logical knowledge has a greater impact than imitation based only on pixel inputs.

B.2 Experiments on the Sensitivity Analysis of TopK

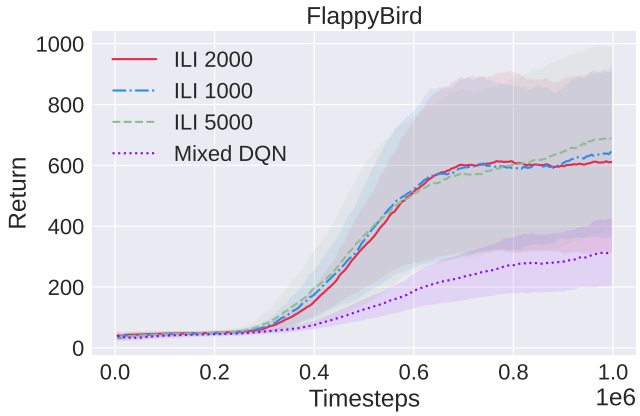


Figure 9: Empirical results on the sensitivity of the hyperparameter TopK.

We conducted sensitivity analysis on "the size of the rule learning dataset (TopK)", as constructing the rule learning dataset is crucial for the effectiveness of propositional logic knowledge. We selected a parameter range of 2 to 10 times the maximum trajectory length (namely, 1000, 2000, and 5000), and conducted experiment in the FlappyBird environment. The empirical results in Figure 9 suggest that performance remains similar within a certain range of TopK values.