End-to-End Neuro-Symbolic Reinforcement Learning with Textual Explanations

Lirui Luo [1 2] **Guoxi Zhang** [2] **Hongming Xu** [2] **Yaodong Yang** [1 2] **Cong Fang** [1] **Qing Li** [2]

Abstract

Neuro-symbolic reinforcement learning (NS-RL) has emerged as a promising paradigm for explainable decision-making, characterized by the interpretability of symbolic policies. NS-RL entails structured state representations for tasks with visual observations, but previous methods cannot refine the structured states with rewards due to a lack of efficiency. Accessibility also remains an issue, as extensive domain knowledge is required to interpret symbolic policies. In this paper, we present a neuro-symbolic framework for jointly learning structured states and symbolic policies, whose key idea is to distill the vision foundation model into an efficient perception module and refine it during policy learning. Moreover, we design a pipeline to prompt GPT-4 to generate textual explanations for the learned policies and decisions, significantly reducing users' cognitive load to understand the symbolic policies. We verify the efficacy of our approach on nine Atari tasks and present GPT-generated explanations for policies and decisions.

1. Introduction

Recent years have witnessed remarkable progress of deep reinforcement learning (RL) (Agarwal et al., 2021; Wurman et al., 2022; Degrave et al., 2022). However, deep RL still faces limitations in sensitive domains due to its opaque nature (Milani et al., 2022). For example, the opacity hinders diagnosing or rectifying policies that fails to generalize, which may be caused by exploiting peripheral information (Delfosse et al., 2024b). Neuro-symbolic reinforcement learning (NS-RL) is promising for overcoming this limitation (Verma et al., 2018; Coppens et al., 2019). It uses structured states and parameterizes policies with concise expressions, thereby guaranteeing interpretability via clear semantics of states and policies.

For tasks with visual observations, the structured states need to be learned from pixels, which involves identifying objects from images. Zheng et al. (2022) proposed to use the Spatially Parallel Attention and Component Extraction (SPACE) model (Lin et al., 2020) for this purpose, but the state representations are fixed during policy learning due to the computational overhead of SPACE. That being said, the structured states are not refined with reward signals, leading to significant performance degradation.

Meanwhile, symbolic policies can be intricate for general users, albeit being intrinsically interpretable. For instance, to interpret logical policies (Delfosse et al., 2024a) one has to be familiar with first-order logic, and in the case of programmatic policies (Qiu & Zhu, 2022) one has to learn the corresponding grammars. Such a lack of accessibility can be an obstacle for agents to gain the trust of the general public. Nevertheless, there is a lack effort in the literature of NS-RL to explain policies for non-expert users.

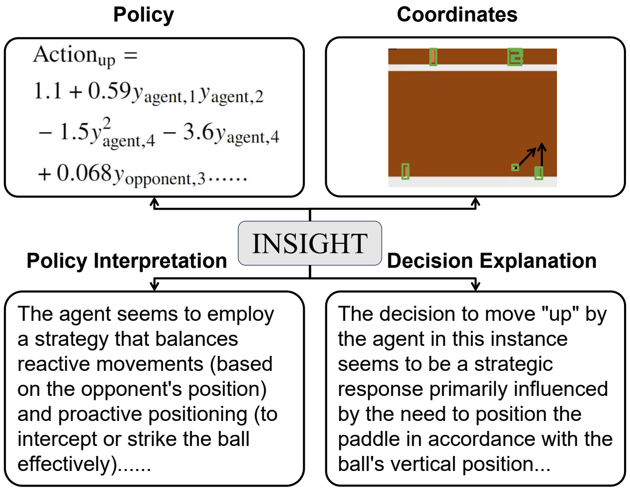
To address these issues, we present INSIGHT, an interpretable neuro-symbolic framework for visual reinforcement learning. As illustrated in Fig. 1, INSIGHT can learn the object coordinates and symbolic policies simultaneously and explain policies and specific decisions in natural language. The key idea of INSIGHT is to overcomes the efficiency drawback of previous methods by distilling vision foundation models into a scalable perception module. Accompanied by the equation learner (EQL) (Sahoo et al., 2018) for representing policies, both the perception module and symbolic policies can be learned from rewards in an end-to-end fashion. Moreover, we develop a pipeline for explaining policies and decisions in natural language via GPT-4, which reduces the cognitive load on users to understand symbolic policies learned by INSIGHT. Through a step called concept grounding, LLMs are instructed to associate quantities in symbolic policies with their semantics for the task of interest. Then, as illustrated in Fig. 1, LLMs are prompted for explaining the decision-making patterns of a symbolic policy and specific decisions made by the policy.

We verify the efficacy of INSIGHT and its model design with extensive experiments on nine Atari games. INSIGHT outperforms all existing approaches for NS-RL (CGP (Wil-

---

Project page: ins-rl.github.io. [1] School of Intelligence Science and Technology, Peking University [2] State Key Laboratory of General Artificial Intelligence, BIGAI. Correspondence to: Qing Li <dylan.liqing@gmail.com>, Cong Fang <fangcong@pku.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



$$\text{Action}_{\text{up}} = 1.1 + 0.59 y_{\text{agent},1} y_{\text{agent},2} - 1.5 y_{\text{agent},4}^2 - 3.6 y_{\text{agent},4} + 0.068 y_{\text{opponent},3} \cdots\cdots$$

The agent seems to employ a strategy that balances reactive movements (based on the opponent's position) and proactive positioning (to intercept or strike the ball effectively)......

The decision to move "up" by the agent in this instance seems to be a strategic response primarily influenced by the need to position the paddle in accordance with the ball's vertical position...

Figure 1. For tasks with visual input, INSIGHT can simultaneously learn the coordinates of objects in observations and coordinate-based symbolic policies simultaneously, and it can interpret learned policies and specific decisions in natural language. $y_{\text{agent,i}}$ represents the vertical coordinate of the agent in the ith frame. Both policy interpretation and decision explanation are produced by entering the policy and a predefined prompt template into the LLM.

son et al., 2018), Diffses (Yuan et al., 2022), DSP (Landajuela et al., 2021), and NUDGE (Delfosse et al., 2024a)). We show that its empirical performance can be attributed to the refinement of structured states with reward information and our approach for learning symbolic policies. We also present examples of policy interpretations and decision explanations.

In summary, our contributions are three-fold.

1. We propose an NS-RL framework that can refine structured states with both visual and reward information.
2. We develop a pipeline to prompt GPT-4 to explain the learned policies and decisions in natural language.
3. We demonstrate the efficacy of our framework on nine Atari games and showcase textual explanations.

2. Related work

Approaches for explainable RL can be categorized into post hoc and intrinsic methods. The former generates explanations using predefined templates (Ehsan et al., 2018; Wang et al., 2019; Hayes & Shah, 2017) or saliency maps (Greydanus et al., 2018), which can be subjective and unreliable (Dazeley et al., 2023). The latter relies on interpretable policy classes, such decision trees (Topin et al., 2021; Zhang et al., 2021), logical expressions (Dazeley et al., 2023; Delfosse et al., 2024a;b), and mathematical expressions (Zheng et al., 2022; Landajuela et al., 2021), for better transparency. Meanwhile, we argue that intrinsic transparency does not guarantee accessibility. While logic-based approaches are better for characterizing relations, they can be unfriendly to users due to their prerequisite knowledge. Recently, there is a growing interest in using LLMs to explain machine learning models (Kroeger et al., 2023; Tennenholtz et al., 2023; Singh et al., 2023; Bills et al., 2023; Zhang et al., 2023). INSIGHT is the first NS-RL method that improves accessibility by using LLMs to generate explanations automatically.

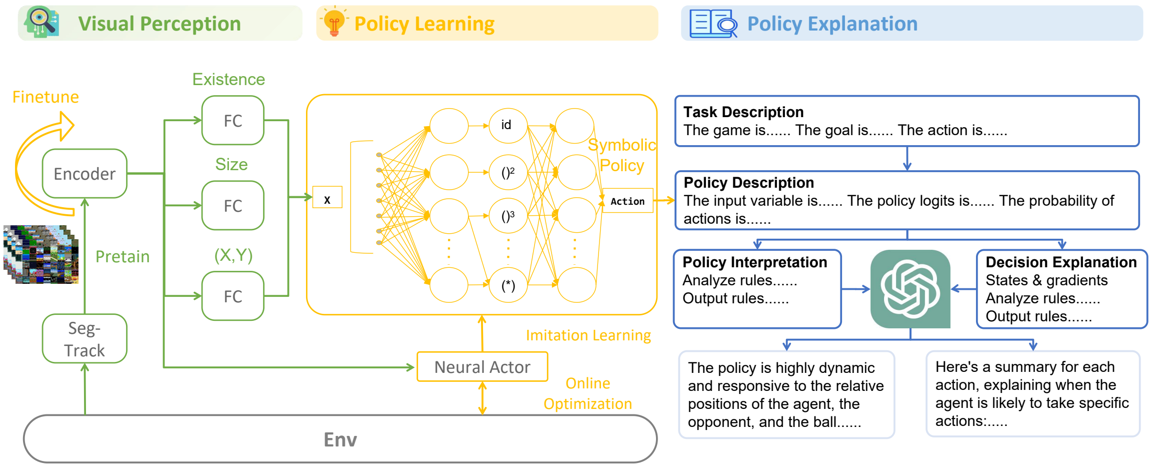
While neuro-symbolic approaches (Manhaeve et al., 2018; Li et al., 2020; 2024) have better interpretability, prior NS-RL approaches (Verma et al., 2018; Coppens et al., 2019; Verma et al., 2019; Landajuela et al., 2021) focus on tasks with low-dimensional observations that have clear semantics. For tasks with visual input, they either rely on ground truth (Delfosse et al., 2024a) or human-defined primitives (Wilson et al., 2018; Lyu et al., 2019) for state representations. As an exception, Zheng et al. (2022) proposed to extract structured states from pre-trained SPACE models, yet the representations are not refined with reward signals and lead to significant performance degradation. INSIGHT is the first NS-RL approach that learns the structured states from both visual and reward signals.

3. INSIGHT

This section presents the proposed INSIGHT framework. As illustrated in Fig. 2, it consists of three components: a visual perception module (Sec. 3.1), a policy learning module (Sec. 3.2), and a policy explanation module (Sec. 3.3).

3.1. Visual Perception

Frame-Symbol Dataset The perception module needs to extract information about objects from input images. Given the notorious sample efficiency of online RL, approaches that use image reconstruction objectives (Zheng et al., 2022; Yoon et al., 2023) are prohibitively expansive. We claim that, for NS-RL, compared to reproducing every detail in visual observations, it suffices to only recognize the positions of objects. To this end, we consider harvesting the segmentation and tracking ability of vision foundation models. For each task, we first roll out about 10,000 frames using pre-trained neural agents and then extract the object bounding boxes using the FastSAM segmentation model (Zhao et al., 2023) and the DeAot model (Yang & Yang, 2022). In specific, segmentation is repeated every ten frames to capture new objects, and the DeAot model is also capable of capturing unseen objects during tracking. Using the bounding boxes, we compute the coordinates of objects (i.e., their centers) and their width and height, normalize these quantities to $[0, 1]$, and pair them with the corresponding images to form a frame-symbol dataset $\mathcal{D}_{\text{symbol}}$. Details of the dataset gen-



Task Description
The game is...... The goal is...... The action is......

Policy Description
The input variable is...... The policy logits is...... The probability of actions is......

Policy Interpretation
Analyze rules......
Output rules......

Decision Explanation
States & gradients
Analyze rules......
Output rules......

The policy is highly dynamic and responsive to the relative positions of the agent, the opponent, and the ball......

Here's a summary for each action, explaining when the agent is likely to take specific actions:......

Figure 2. INSIGHT consists of three components: a perception module, a policy learning module, and a policy explanation module. The perception module learns to predict object coordinates using a frame-symbol dataset distilled from vision foundation models. The policy learning module is responsible for learning coordinate-based symbolic policies. In particular, to address with the limited expressiveness of object coordinates, it uses a neural actor to interact with the environment. The policy explanation module can generate policy interpretations and decision explanations using task description, policy description, and values of object coordinates.

eration are provided in Appx. A.1. By fitting to this dataset, the perception module can learn to extract the structured states before policy learning, which can improve the sample efficiency of INSIGHT.

Learning Object Coordinates We now present the details of learning from $\mathcal{D}_{\text{symbol}}$. $\mathcal{D}_{\text{symbol}}$ contains information of the existence, the shape, and the coordinates of objects. To make full use of them, INSIGHT employs a multi-task formulation for the perception module $\omega_{\text{perception}}$, which is illustrated in Fig. 2. An encoder, parameterized by convolutional neural networks (CNNs) and FCNs, is responsible for encoding information about objects and rewards into hidden representations. Three FCNs use the hidden representations to predict the existence, the shape, and the coordinates of objects, which are explained below.

Objects can appear and disappear in many tasks. Only the coordinates of present objects can be used by the symbolic policy, otherwise it will exploit the coordinates of missing objects and lose interpretability. Therefore, we need to mask out the coordinates of missing objects during policy learning, which requires predicting the existence of objects. One issue of existence prediction is that the distribution of objects can be long tailed. For example, in the BeamRider task, the agent's spaceship is present at almost every step, while torpedoes appear less frequently. This issue is handled with the distribution-balanced focal loss (Wu et al., 2020).

Specifically, for the jth object in the ith sample, let $c_{ij} = 1$ if it is present, and $c_{ij} = 0$ otherwise. Let C be the number of objects and N be the number of samples. This loss extend the focal loss (Lin et al., 2017) by weighting labels with their inverse frequency. It is given by

$$\mathcal{L}_{\text{exist}} = \sum_{j=1}^{C} \bar{\eta}_{ij} \Big[c_{ij}(1-p_{ij})^{\psi} \log(p_{ij}) + (1-c_{ij}) p_{ij}^{\psi} \log(1-p_{ij}) \Big], \tag{1}$$

where ψ is the modulating factor of the focal loss (Lin et al., 2017) and$p_{i,j}$ represents the probability generated by the existence layer. Details for computing $\bar{\eta}_{ij}$ is provided in Appx. A.2.

As for coordinate prediction, we use the L1 loss since the normalized coordinates can take small values. Let $\mathbf{x}_i \in \mathbb{R}^{2C}$ be the vector for object coordinates in the ith image and $\hat{\mathbf{x}}_i$ be its predicted value. For $j = 1, 2, \ldots, C$, $x_{i,2j}$ and $x_{i,2j+1}$ is the Y and X coordinate for the jth object. Then, the objective for coordinate prediction is given by:

$$\mathcal{L}_{\text{coor}} = \sum_{j=1}^{C} c_{ij} \left[|x_{i,2j} - \hat{x}_{i,2j}| + |x_{i,2j+1} - \hat{x}_{i,2j+1}| \right]. \tag{2}$$

These losses are then averaged over all frames. Note that the predicted coordinates are clipped to $[0, 1]$ when being used for policy learning, as they are supposed to be the coordinates of objects. For shape prediction, the network is trained to predict the width and height of object bounding boxes. The loss, $\mathcal{L}_{\text{size}}$, takes a similar form as $\mathcal{L}_{\text{coor}}$, except that the prediction targets are replaced with the width and height of objects. Overall, the objective for learning from the frame-symbol dataset is:

$$\mathcal{L}_{\text{cnn}} = \mathcal{L}_{\text{exist}} + \mathcal{L}_{\text{coor}} + \mathcal{L}_{\text{size}}. \tag{3}$$

Before policy learning, the perception module is pre-trained on frame-symbol datasets to equips agents with knowledge about objects. We will verify this design choice in Sec. 4.

3.2. Learning Symbolic Policies

EQL We represent symbolic policies with the EQL network (Martius & Lampert, 2017; Sahoo et al., 2018). Given an input vector $\mathbf{h}^{(k-1)} \in \mathbb{R}^{d_{k-1}}$, its kth layer first applies a transformation $\mathbf{g}^{(k)} = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}$, where $\mathbf{W}^{(k)} \in \mathbb{R}^{d_k \times d_{k-1}}$ and $\mathbf{b}^{(k)} \in \mathbb{R}^{d_k}$ are learnable parameters. d_{k-1} and d_k are the dimension of $\mathbf{h}^{(k-1)}$ and $\mathbf{g}^{(k)}$.

Compared to fully-connected layers (FCN), EQL has more flexible activation functions. The activation function for a vector $\mathbf{g} \in \mathbb{R}^{d_k}$, $\mathbf{f} : \mathbb{R}^{d_k} \to \mathbb{R}^{d'_k}$, is given by

$$\mathbf{f}(\mathbf{g}) = \begin{bmatrix} f_1(g_1) \\ \vdots \\ f_u(g_u) \\ f_{u+1}(g_{u+1}, g_{u+2}) \\ \vdots \\ f_{d'_k}(g_{d_k-1}, g_{d_k}) \end{bmatrix}, \tag{4}$$

where $f_1, f_2, \ldots, f_{d'_k} \in \mathcal{F}$ are scalar functions. $f_1, f_2, \ldots, f_u$ are unary functions, and the remaining are binary functions. This flexible choice of activation functions enhances expressiveness of EQL. In this work, $\mathcal{F}$ includes the square function, the cubic function, constants, the identity function, multiplication, and addition.

The final ingredient is sparsity regularization (Martius & Lampert, 2017) required for deriving concise expressions, which imposes the following regularization to the parameters of a EQL network to avoid the singularity in the gradient as the weights go to 0.

$$\mathcal{L}_{\text{reg}}(w) = \begin{cases} |w|^{1/2} & |w| \geq a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{1/2} & |w| < a \end{cases} \tag{5}$$

Here, $a = 0.05$ is a smoothing parameter.

Neural Guidance We now discuss the challenges in learning symbolic policies. The symbolic policy, referred to as the EQL actor π_{EQL}, computes action distributions using the predicted object coordinates. While being intuitive, it is worth noting that the object coordinates $\mathbf{x}$ are subject to limited expressiveness. Each element of $\mathbf{x}$ is bound to the Y or X coordinate of some objects, which means they are not distributed representations and cannot encode complex patterns as neural representations do. What exacerbates the situation is that the expressions represented by π_{EQL} are forced to be concise by $\mathcal{L}$. In consequence, π_{EQL} might not be able to represent *all possible policies* and fail to explore well during policy learning.

Algorithm 1 Procedures for learning symbolic policy

```
Input: Pretrained perception module $\omega_{\text{perception}}$ and the frame-symbol dataset $\mathcal{D}_{\text{symbol}}$
Output: Fine-tuned perception module and the EQL actor $\pi_{\text{EQL}}$
Initialize the neural actor $\pi_{\text{nerual}}$ and the EQL actor $\pi_{\text{EQL}}$ randomly.
for batch = $1, 2, \ldots$ do
    Collect online samples $\mathcal{D}_{\text{batch}}$ using $\pi_{\text{nerual}}$.
    for iteration=$1, 2, ..., n$ do
        if iteration==$n$ then
            Update $\pi_{\text{nerual}}$, $\pi_{\text{EQL}}$, and $\omega_{\text{perception}}$ by minimizing $\mathcal{L}$ using $\mathcal{D}_{\text{batch}}$ and $\mathcal{D}_{\text{symbol}}$.
        else
            Update $\pi_{\text{neural}}$ and $\omega_{\text{perception}}$ by minimizing $\mathcal{L}_{\text{ppo}}$ using $\mathcal{D}_{\text{batch}}$.
        end if
    end for
end for
```

We therefore opt for using π_{EQL} to approximate only the optimal policy, which is less demanding than approximating all possible policies. Inspired by Nguyen et al. (2021); Landajuela et al. (2021), we propose a neural guidance scheme that uses a neural actor π_{neural} to interact with the environment. π_{neural} takes as input the hidden representations produced by the encoder of $\omega_{\text{perception}}$ and is not regularized by $\mathcal{L}_{0.5}$, so it can effectively explore the state-action space. The EQL actor is trained to distill π_{neural} using symbolic expressions and coordinate. Compared to existing neural guidance schemes (Nguyen et al., 2021; Landajuela et al., 2021), in our scheme the EQL actor and the neural actor are trained simultaneously rather than separately, which results in improved sample efficiency.

The procedures for learning symbolic policies are outlined in Alg. 1. Specifically, we optimize the neural actor using the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm, whose objective is denoted by $\mathcal{L}_{\text{ppo}}$. As for the EQL actor, since we have access to π_{neural}, we minimize the cross entropy between the action distributions induced by the two actors, which is given by

$$\mathcal{L}_{\text{ng}} = -\mathbb{E}_{s,a\sim\pi_{\text{neural}}}\left[\log\left(\pi_{\text{EQL}}(a|s)\right)\right]. \tag{6}$$

The expectation is taken over online samples collected by π_{neural}. For tasks with discrete action space, the expectation over actions can be calculated analytically. To keep the accuracy of predicted coordinates, we also update $\omega_{\text{perception}}$ using $\mathcal{L}_{\text{cnn}}$. In summary, the objective for learning π_{EQL} and π_{neural} jointly is given by

$$\mathcal{L} = \mathcal{L}_{\text{ppo}} + \mathcal{L}_{\text{ng}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{cnn}}\mathcal{L}_{\text{cnn}}. \tag{7}$$

where $\lambda_{\text{reg}}, \lambda_{\text{cnn}} \in \mathbb{R}$ are hyper-parameters to balance policy learning, coordinate prediction, and sparsity regularization.

Remarks The PPO algorithm reuses online samples for multiple iterations to improve sample efficiency. In preliminary experiments, we found that optimizing $\mathcal{L}_{\text{cnn}}$ and $\mathcal{L}_{\text{ng}}$ for multiple iterations led to inferior performance, possibly due to drastic changes imposed on the perception module $\omega_{\text{perception}}$. Thus, we optimize $\mathcal{L}$ only in the last iteration and $\mathcal{L}_{\text{ppo}}$ in all other iterations in Alg. 1.

The perception module $\omega_{\text{perception}}$ sacrifices some auxiliary information for the sake of efficiency, since we only include the location, heights, and width in $\mathcal{D}_{\text{symbol}}$. This is where end-to-end policy learning comes in. By refining $\omega_{\text{perception}}$ using reward signals, the hidden layers of $\omega_{\text{perception}}$ can learn to capture features that are essential for task performance but overlooked during pre-training. As shown by our results presented in Tab. 4 and Fig. 3, end-to-end policy improves both task performance and coordinate prediction.

3.3. Explaining Policies and Decisions

The pipeline for generating explanations starts with a step called concept grounding, which is followed by separate prompts for policy interpretations and decision explanations. Full prompts and details are provided in Appx. D. Suppose a user who is familiar with a task wants to interpret a learned symbolic policy. What can be tedious for the user is to associate concepts of the task, such as the goal and the influence of actions, with the construction of the symbolic policy. For example, he or she has to find out which element in the coordinate vector corresponds to the location of a certain object. We refer the process of establishing such a correspondence as *concept grounding*. Note that concept grounding is in fact not necessary for understanding explanations for policies and decisions. For example, only knowledge of Pong is required to understand the conclusion part in the right part of Fig. 5, which are explanations for choosing action up at a particular state. This observation inspires us to use LLMs to ground concepts and generate explanations, thereby reducing the user's intellectual burden.

Specifically, the prompt for concept grounding consists of *task description* and *policy description*. The task description includes the goal of the task and the effects of actions, which is essential for steering the explanations toward task solving. For Pong, it can be "There are two paddles in the screen...The agent earns a point if its opponent fails to strike the ball back...It needs to take one of the three actions: noop (take no operation)...". As for policy description, we include the construction of the coordinate system and the expressions of the symbolic policy. The semantics of coordinates are reflected with their names. For example, the $x_{i,4}$ and $x_{i,5}$ for Pong are the X and Y coordinates of the agent's paddle. To associate decisions with the location and motion of objects, we also instruct the LLM to infer the motion of objects from the coordinates in successive frames.

As shown in the upper left of Fig. 5, learned policies are formulated as a set of equations involving input variables (i.e. object coordinates), intermediate variables (e.g., t_1 in Fig. 5), and action logits. Despite their simplicity, prompting LLMs with these equations only yields vacuous responses such as "the policy is complex, involving non-linear combinations of input variables". Therefore, we apply the chain-of-thought principle and proceed in three steps: analyze the mapping from input variables to intermediate variables, analyze the mapping from intermediate variables to action logits, and finally summarize the findings. Moreover, to mitigate issues such as hallucinations and uncontrollable outputs, we carefully design a set of rules for the LLM, as detailed in Tab. A16. For example, Rule 1 insists that explanations remain rooted in the policy's expressions, and rule 2 dictates that LLMs' policy explanations should be based on logits and action probabilities. In our preliminary experiments, these rules were able to improves the alignment between generated explanations and actual policy actions and makes outputs more precise and controllable.

To generate explanations for specific decisions, we provide the LLM with the exact values of object coordinates, the action being taken, and the gradients of action log-likelihood with respect to every object coordinate. In addition, we instruct the LLM to assess the importance of input variables from the sensitivity perspective by mentioning the semantics of gradients, i.e. the sensitivity of action log-likelihood regarding to input variables, An example of generated explanations is shown in the bottom right of Fig. 5.

4. Experiments

This section evaluates the efficacy of INSIGHT for learning symbolic policy and structured state representations. Sec. 4.2 reports the task performance of INSIGHT and validates our design choices, and Sec. 4.3 presents textual explanations for learned policies and specific decisions.

4.1. Experimental Setup

Tasks We consider the online RL setting and nine Atari tasks: Pong, BeamRider, Enduro, Qbert, SpaceInvaders, Seaquest, Breakout, Freeway, and MsPacman, which widely used in prior works (Delfosse et al., 2024a; Landajuela et al., 2021; Zheng et al., 2022). They range from simple motion control (e.g. Pong) to complex decision-making (e.g. Seaquest), and cover issues such as clean background (e.g. SpaceInvaders) vs noisy background (e.g. BeamRider) and constant object (e.g. Pong) vs varying objects (e.g. Enduro). So with them we can examine INSIGHT comprehensively. In the meantime, results on more realistic tasks are also important for validating NS-RL algorithms, so we also include results on MetaDrive (Li et al., 2022), a challenging environment for autonomous driving.

Evaluation Metrics For task performance, we report the

Table 1. **INSIGHT matches the performance of neural agents on all Atari tasks.** Performance of the proposed INSIGHT, Neural (a neural agent that has the same network architecture as INSIGHT), and existing NS-RL approaches (CGP (Wilson et al., 2018), Diffses (Zheng et al., 2022), DSP (Landajuela et al., 2021), and NUDGE (Delfosse et al., 2024a)) on nine Atari tasks. INSIGHT outperforms all NS-RL baselines and matches the performance of Neural for all nine tasks.

Task	INSIGHT	Neural	Coor-Neural	CGP[1]	DiffSES[2]	DSP[3]	NUDGE[4]	Human[5]
Pong	**20.9** ± 0.1	20.4 ± 0.6	19.8 ± 0	20 ± 0	20.2 ± 0	−1 ± 0	−7.2 ± 1.2	9.3
BeamRider	3828.1 ± 261.1	3868.1 ± 204.2	1614.4 ± 79.8	1341.6 ± 21	/	354.6 ± 20.3	/	5775
Enduro	843.7 ± 213.8	676.9 ± 730.4	933.9 ± 4.2	2 ± 0	/	41.1 ± 15.5	2.4 ± 2.1	309.6
Qbert	16978.6 ± 1936.1	17879.2 ± 1857.1	13269.2 ± 1024.4	770 ± 94	/	558.3 ± 41.5	/	13455
SpaceInvaders	**1232.6** ± 140.7	1184.6 ± 137.6	717.7 ± 15.3	1001 ± 25	792.4 ± 0	222.6 ± 9	80 ± 17	1652
Seaquest	**2665.7** ± 728.2	1804.8 ± 20.1	1410.6 ± 338.2	724 ± 26	/	193.3 ± 12	0 ± 0	20182
Breakout	**409.6** ± 11.3	259.6 ± 183.8	356.5 ± 29.2	13.2 ± 2	/	4.3 ± 0.5	3.4 ± 0.8	31.8
Freeway	**32.7** ± 0.1	28.7 ± 5.3	32.4 ± 0.1	28.2 ± 0	/	21.5 ± 0.2	21.4 ± 0.8	29.6
MsPacman	**3042.5** ± 320.1	2737.1 ± 562.3	2257.3 ± 145.2	2568 ± 724	/	2937.3 ± 892.9	/	15693

[1] Results for CGP were taken from (Wilson et al., 2018).

[2] We used the results for Pong and SpaceInvaders reported by Zheng et al. (2022), yet we were unable to obtain results for other tasks as the code is incomplete.

[3] Since DSP cannot handle visual observations, we used pre-trained peception module of INSIGHT to extract object coordinates and used its released code for policy learning.

[4] We used the results reported by Delfosse et al. (2024a) for Freeway and obtained results for Pong, Enduro, SpaceInvaders, Seaquest, and Breakout using its released code. Due to the absence of predefined templates for all tasks within the codebase, we tailored the template originally designed for Asterix to fit the unique action spaces of the additional tasks. We were unable to obtain results for BeamRider, Qbert, and MsPacman since the ground-truth object locations are not available for them.

[5] Results were taken from (Wilson et al., 2018).

Table 2. **INSIGHT outperforms neural baselines on MetaDrive.** This table shows the performance of INSIGHT, Neural and Random on MetaDrive when trained fro 1M, 2M, and 5M steps.

Success Rate	INSIGHT	Neural	Coor-Neural	Random
1M	**0.37** ± 0.21	0.17 ± 0.12	0.07 ± 0.05	0 ± 0
2M	**0.42** ± 0.13	0.19 ± 0.06	0.21 ± 0.14	0 ± 0
5M	**0.63** ± 0.11	0.49 ± 0.11	0.41 ± 0.12	0 ± 0

Table 3. **INSIGHT is as efficient as Neural.** This table shows the inference time per state on Pong. The testing conditions are provided in Appx. A.4. INSIGHT is as efficient as Neural are is an order of magnitude faster than SA-Neural and SPACE-Neural.

Method	INSIGHT	SPACE-Neural	SA-Neural	Neural	NUDGE
Time (ms)	**2**	50	40	**2**	60

means and standard deviations of test returns after training agents for 10 million environment steps.The higher, the better. For coordinate prediction, we report the mean absolute error (MAE) of predicted coordinates, which is the result of dividing the total absolute error of coordinate prediction on a sample by the number of objects, the number of stacked frames, and two as we are predicting the X and Y coordinates. It is within $[0, 1]$, and the lower the better. Since not all objects are relevant for symbolic policies, we include a variant of MAE, F-MAE, which only considers relevant objects. All experiments are repeated for three seeds.

Baselines For task performance, INSIGHT is compared with Neural, DSP, Diffses, NUDGE, and CGP. Neural is a deep RL alternative for INSIGHT that uses the same network architecture but does not learn from the frame-symbol datasets. The remaining four are approaches for NS-RL. DSP (Landajuela et al., 2021) searches for symbolic policies from low-dimensional input using a recurrent neural network, while Diffses (Zheng et al., 2022) uses frozen SPACE models to extract states and search for symbolic policies using genetic programming. NUDGE (Delfosse et al., 2024a) relies on the ground-truth structured states and represent policies with first-order differentiable logic. Lastly, CGP (Wilson et al., 2018) leverages cartesian genetic programming for learning policies from images.

INSIGHT Variants To analyze our design choices, we include results for several variants of INSIGHT. Specifically, w/o Pre-train is a variant that does not pre-train the perception model $\omega_{\text{perception}}$, and Fixed is a variant that freezes $\omega_{\text{perception}}$ in policy learning. By w/o NG, we mean a variant that removes the neural guidance scheme and optimizes the EQL actor using $\mathcal{L}_{\text{ppo}}$ directly. Meanwhile, Coor-Neural trains $\omega_{\text{perception}}$ and the neural actor π_{neural} similarly with INSIGHT, but it uses the predicted coordinates as the input for π_{neural}. SPACE-Neural extracts object coordinates from a pre-trained SPACE model, and SA-Neural utilizes the BO-QSA (Jia et al., 2023), a recent slot-attention algorithm, to extract latent representations of objects from images.

Implementation Details We use the standard preprocessing protocol for Atari tasks, which includes resizing, gray scaling, and frame stacking. As for hyper-parameters, we use 0.001 for λ_{reg}, 2 for λ_{cnn}. We use the OpenAI's GPT-4 model (Bubeck et al., 2023) when generating explanations. All the implementation details are provided in Appx. B.1.

4.2. Quantitative Results

Improved Task Performance Tab. 1 shows that except for Pong, where visual observations are less involved, NS-RL baselines are outperformed by Neural by a wide margin. In contrast, INSIGHT can match or even outperform Neural. In addition, in Tab. A8 we show that our implementation

Table 4. **INSIGHT significantly enhances coordinate prediction for policy-relevant objects through policy learning.** MAE and F-MAE measure the coordinate prediction error, with lower values indicating better performance. F-MAE measures the mean absolute error of coordinate prediction for policy-related objects. Numbers are multiplied by 100. Refer to Tab. A7 for results for the other four tasks. By comparing the results of INSIGHT with those of Fixed, we see that refining state representations with rewards improves coordinate prediction for policy-related objects. The proposed neural guidance scheme is crucial for such improvement, as implied by results of w/o NG.

	INSIGHT		w/o Pretrain		Fixed		w/o NG		Coor-Neural	
Tasks	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE
Pong	1.4 ± 0.9	$\mathbf{1.3} \pm 0.9$	2.9 ± 3.1	3.5 ± 3.8	2.8 ± 0.5	3.1 ± 0.3	1.7 ± 0.9	1.6 ± 0.9	$\mathbf{0.9} \pm 0$	/
BeamRider	$\mathbf{5.5} \pm 0$	$\mathbf{3.2} \pm 0.3$	5.8 ± 0.1	3.5 ± 0.1	$\mathbf{5.5} \pm 0.4$	4.9 ± 0.1	6.2 ± 0.1	5.9 ± 0	5.7 ± 0	/
Enduro	13.5 ± 0	$\mathbf{6.7} \pm 0.3$	13.1 ± 0.1	8.2 ± 0.2	$\mathbf{12.7} \pm 0.5$	12.2 ± 0.5	13.4 ± 0.2	13.4 ± 0.2	13.7 ± 0.1	/
SpaceInvaders	10.3 ± 0.4	8.1 ± 0.9	10.6 ± 0.2	$\mathbf{6.5} \pm 0.6$	$\mathbf{9.9} \pm 0.5$	11.1 ± 0.4	11.1 ± 0.3	11.1 ± 0.4	10.3 ± 0.3	/
Freeway	16.3 ± 0.3	$\mathbf{14.8} \pm 1.6$	23.4 ± 0	19.4 ± 5.6	$\mathbf{15.7} \pm 0.2$	18.7 ± 0.8	16.4 ± 0.2	16.4 ± 0.2	16.4 ± 0.2	/

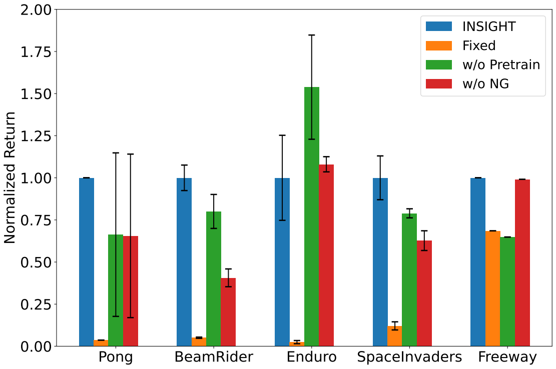


Figure 3. **Each component of INSIGHT is critical for overall performance.** Detailed performance analyses of INSIGHT and its variants across five tasks are presented. Refer to Fig. A1 for results for the remaining four tasks. Test returns are normalized so that INSIGHT corresponds to one and random policy is zero. Fixing the perception module during policy learning (i.e., Fixed) hinders performance for all tasks, indicating that it is crucial to refine states with reward signals. The results of w/o Pretrain and w/o NG show that pre-training the perception module and the proposed neural guidance scheme also improves performance.

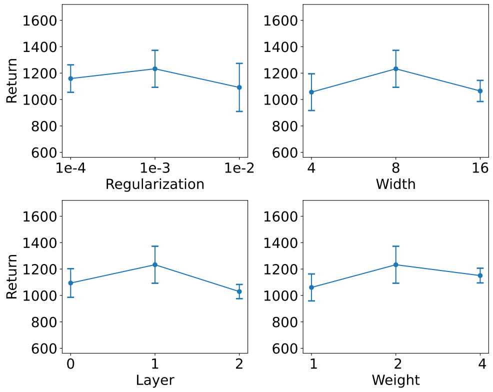


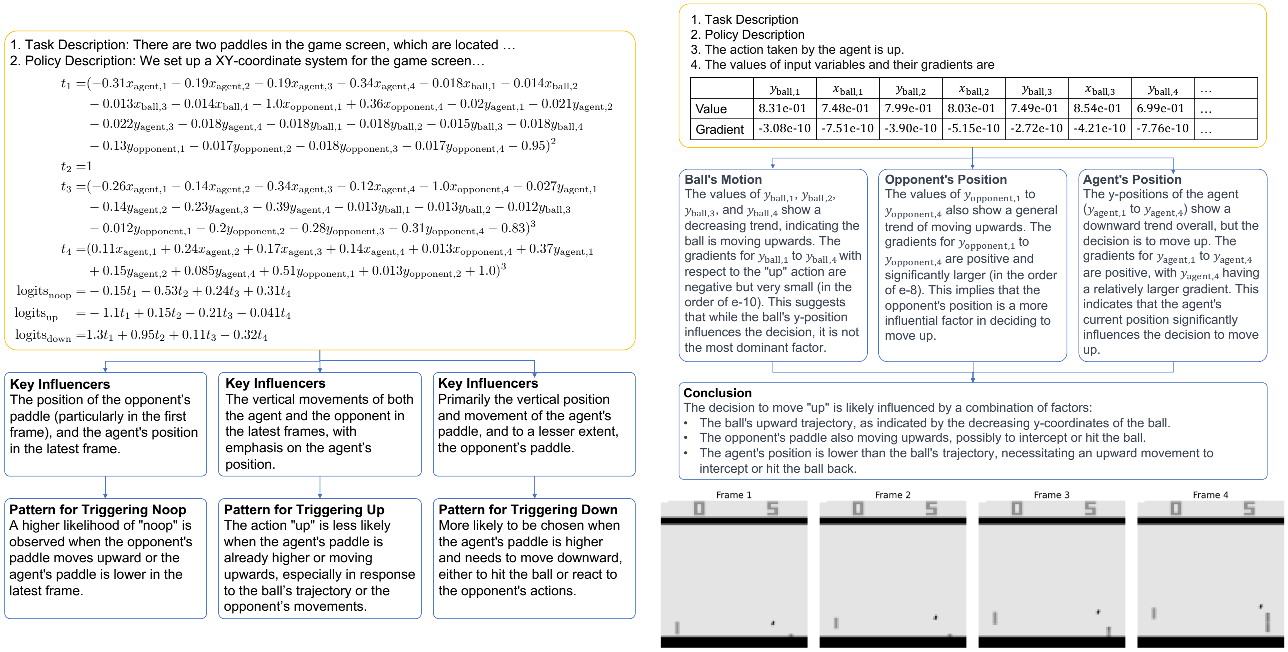
Figure 4. **INSIGHT demonstrates robustness to hyper-parameters.** Examining the influence of sparsity regularization weight λ_{reg}, the EQL actor's width/layers, and the weight of $\mathcal{L}_{\text{cnn}}$ on SpaceInvaders performance. Results for additional tasks are available in Fig. A2. Overall, INSIGHT shows substantial robustness to variations in hyper-parameters.

for Neural matches the performance of CleanRL (Huang et al., 2022), an open-source implementation for deep RL agents. Tab. 2 shows the success rates of INSIGHT and neural baselines over 1M, 2M, and 5M timesteps for Metadrive. INSIGHT outperforms the neural baselines at all timesteps. These results highlight that INSIGHT, by learning structured states and symbolic policies jointly, overcomes the performance drawback of existing NS-RL approaches. We now present an ablation analysis to explain the improvement.

Better Efficiency As mentioned in Sec. 3.1, the perception module of INSIGHT is designed to be efficient. Tab. 3 shows that INSIGHT is an order of magnitude faster than both SPACE-Neural and SA-Neural, which verifies this design. In addition, Tab. A5 shows that agents learn faster when learning from $\mathcal{D}_{\text{symbol}}$. A possible reason is that $\mathcal{D}_{\text{symbol}}$ provides direct supervision for object locations. When trained with image reconstruction objectives, agents are not forced to focused on individual objects.

Benefits of End-to-End Policy Learning We now examine our claim for refining structured states with both visual and rewards from the perspective of task performance and coordinate prediction. Fig. 3 shows that when compared to the variant Fixed, INSIGHT has significant better performance for all five task, demonstrating that INSIGHT's performance is largely determined by its ability to refine structured states with reward signals. Tab. 4 further reveals a clue for the performance improvement. Compared to Fixed, INSIGHT has higher MAE for four tasks but lower F-MAE for all five tasks. In addition, readers may refer to Fig. A3 for visualizations of predicted coordinates. These findings suggest that reward signals can guide agents to improve the coordinate prediction of policy-relevant objects.

Benefits of Pre-Training the Perception Module In Sec. 3.1, we suggest pre-training the perception module before policy learning. Fig. 3 shows that the pre-training step indeed improves performance for four tasks, and Tab. 4 indi-



1. Task Description: There are two paddles in the game screen, which are located …
2. Policy Description: We set up a XY-coordinate system for the game screen…

$$\begin{aligned} t_1 =&(-0.31x_{\text{agent},1} - 0.19x_{\text{agent},2} - 0.19x_{\text{agent},3} - 0.34x_{\text{agent},4} - 0.018x_{\text{ball},1} - 0.014x_{\text{ball},2} \\ &- 0.013x_{\text{ball},3} - 0.014x_{\text{ball},4} - 1.0x_{\text{opponent},1} + 0.36x_{\text{opponent},4} - 0.02y_{\text{agent},1} - 0.021y_{\text{agent},2} \\ &- 0.022y_{\text{agent},3} - 0.018y_{\text{agent},4} - 0.018y_{\text{ball},1} - 0.018y_{\text{ball},2} - 0.015y_{\text{ball},3} - 0.018y_{\text{ball},4} \\ &- 0.13y_{\text{opponent},1} - 0.017y_{\text{opponent},2} - 0.018y_{\text{opponent},3} - 0.017y_{\text{opponent},4} - 0.95)^2 \\ t_2 =&1 \\ t_3 =&(-0.26x_{\text{agent},1} - 0.14x_{\text{agent},2} - 0.34x_{\text{agent},3} - 0.12x_{\text{agent},4} - 1.0x_{\text{opponent},4} - 0.027y_{\text{agent},1} \\ &- 0.14y_{\text{agent},2} - 0.23y_{\text{agent},3} - 0.39y_{\text{agent},4} - 0.013y_{\text{ball},1} - 0.013y_{\text{ball},2} - 0.012y_{\text{ball},3} \\ &- 0.012y_{\text{opponent},1} - 0.2y_{\text{opponent},2} - 0.28y_{\text{opponent},3} - 0.31y_{\text{opponent},4} - 0.83)^3 \\ t_4 =&(0.11x_{\text{agent},1} + 0.24x_{\text{agent},2} + 0.17x_{\text{agent},3} + 0.14x_{\text{agent},4} + 0.013x_{\text{opponent},4} + 0.37y_{\text{agent},1} \\ &+ 0.15y_{\text{agent},2} + 0.085y_{\text{agent},4} + 0.51y_{\text{opponent},1} + 0.013y_{\text{opponent},2} + 1.0)^3 \\ \text{logits}_{\text{noop}} =& - 0.15t_1 - 0.53t_2 + 0.24t_3 + 0.31t_4 \\ \text{logits}_{\text{up}} =& - 1.1t_1 + 0.15t_2 - 0.21t_3 - 0.041t_4 \\ \text{logits}_{\text{down}} =&1.3t_1 + 0.95t_2 + 0.11t_3 - 0.32t_4 \end{aligned}$$

Key Influencers
The position of the opponent's paddle (particularly in the first frame), and the agent's position in the latest frame.

Key Influencers
The vertical movements of both the agent and the opponent in the latest frames, with emphasis on the agent's position.

Key Influencers
Primarily the vertical position and movement of the agent's paddle, and to a lesser extent, the opponent's paddle.

Pattern for Triggering Noop
A higher likelihood of "noop" is observed when the opponent's paddle moves upward or the agent's paddle is lower in the latest frame.

Pattern for Triggering Up
The action "up" is less likely when the agent's paddle is already higher or moving upwards, especially in response to the ball's trajectory or the opponent's movements.

Pattern for Triggering Down
More likely to be chosen when the agent's paddle is higher and needs to move downward, either to hit the ball or react to the opponent's actions.

1. Task Description
2. Policy Description
3. The action taken by the agent is up.
4. The values of input variables and their gradients are

	$y_{\text{ball},1}$	$x_{\text{ball},1}$	$y_{\text{ball},2}$	$x_{\text{ball},2}$	$y_{\text{ball},3}$	$x_{\text{ball},3}$	$y_{\text{ball},4}$	…
Value	8.31e-01	7.48e-01	7.99e-01	8.03e-01	7.49e-01	8.54e-01	6.99e-01	…
Gradient	-3.08e-10	-7.51e-10	-3.90e-10	-5.15e-10	-2.72e-10	-4.21e-10	-7.76e-10	…

Ball's Motion
The values of $y_{\text{ball},1}$, $y_{\text{ball},2}$, $y_{\text{ball},3}$, and $y_{\text{ball},4}$ show a decreasing trend, indicating the ball is moving upwards. The gradients for $y_{\text{ball},1}$ to $y_{\text{ball},4}$ with respect to the "up" action are negative but very small (in the order of e-10). This suggests that while the ball's y-position influences the decision, it is not the most dominant factor.

Opponent's Position
The values of $y_{\text{opponent},1}$ to $y_{\text{opponent},4}$ also show a general trend of moving upwards. The gradients for $y_{\text{opponent},1}$ to $y_{\text{opponent},4}$ are positive and significantly larger (in the order of e-8). This implies that the opponent's position is a more influential factor in deciding to move up.

Agent's Position
The y-positions of the agent ($y_{\text{agent},1}$ to $y_{\text{agent},4}$) show a downward trend overall, but the decision is to move up. The gradients for $y_{\text{agent},1}$ to $y_{\text{agent},4}$ are positive, with $y_{\text{agent},4}$ having a relatively larger gradient. This indicates that the agent's current position significantly influences the decision to move up.

Conclusion
The decision to move "up" is likely influenced by a combination of factors:

- The ball's upward trajectory, as indicated by the decreasing y-coordinates of the ball.
- The opponent's paddle also moving upwards, possibly to intercept or hit the ball.
- The agent's position is lower than the ball's trajectory, necessitating an upward movement to intercept or hit the ball back.

Figure 5. Examples for textual explanations for Pong. Readers may refer to Appx. D for full prompts. **Left**: interpretations for a learned policy. The interpretations identify influential input variables and summarize triggering patterns of actions. **Right**: explanations for an action taken at a state. The four images located at the bottom illustrate the state. The motion of the ball and the opponent's paddle are deduced from input variables, which are used for supporting explanations of actions.

cates that the improvement may be the result of better prediction for policy-relevant objects. Despite the imperfections and redundancy in the representation space obtained through pre-training, it still offers valuable information about policy-related objects, serving as a strong initialization for subsequent end-to-end training. Meanwhile, this extra step does complicate the training protocol of INSIGHT. One possible improvement is to use off-policy algorithms for policy learning, which allow the perception module to be optimized for more gradient steps.

Efficacy of the Proposed Neural Guidance Scheme We introduce the neural guidance scheme to address the limited expressiveness of object coordinates. The effectiveness of this scheme is supported by results in Fig. 3–w/o NG performs worse than INSIGHT for three tasks out of five tasks. Moreover, Coor-Neural is outperformed by Neural for six tasks in Tab. 1, suggesting that the object coordinates are not sufficiently expressive for online policy learning. Tab. 4 reveals that unlike the case of INSIGHT, the F-MAE and MAE of w/o NG are almost the same, implying that w/o NG cannot improve coordinate prediction using reward signals. Thus, the neural guidance scheme plays a key role in refining coordinate prediction with rewards.

Robustness Against Hyper-Parameters Lastly, Fig. 4 shows the influence of four hyper-parameters for SpaceInvaders, which indicates that the performance of INSIGHT are quite robust against the hyper-parameters. Results for other tasks are provided in Fig. A2. Since our goal is to illustrate the efficacy of INSIGHT rather than benchmarking, we use the same values for hyper-parameters for all tasks.

In summary, the ability to refine states with reward signals is decisive for performance improvement, possibly due to the improved coordinate prediction for policy-relevant objects. The proposed neural guidance scheme also plays a role in improving task performance and coordinate prediction.

4.3. Textual Explanations

The left part of Fig. 5 presents interpretations of a policy learned for Pong, which requires an agent to control the right paddle to hit the ball to the left-hand side. The agent earns a point if the opponent fails to hit the ball back. Both paddles move only vertically.

The bottom left part of Fig. 5 shows policy interpretations generated by INSIGHT. It shows that the influential variables are correctly identified. For example, $y_{\text{opponent},1}$ is recognized as influential for action noop due to its coefficient (-0.13), and the coordinates of the ball are omitted for their small coefficients. However, some triggering patterns are less convincing. Action up is considered to be less likely when the agent's paddle moves upward (y_{agent} decreases). While this is correct when considering t_3, it is in fact difficult to determine how t_1 changes in response to decrease of y_{agent}. These observations confirms the LLM's ability to perform step-by-step analysis for symbolic policies.

The right part of Fig. 5 showcases decision explanations. While the movements of the ball is correctly deduced, there is a mistake for the agent's paddle, possibly due to fluctuations in the coordinate predictions. The generated explanations reveals that the decision is more sensitive to the position of the opponent's paddle than the ball, suggesting that the agent is exploiting the opponent's fixed policy. Similar phenomena are also observed in recent NS-RL works (Delfosse et al., 2024b), which highlights the importance of inspecting learned policies manually. Nevertheless, INSIGHT can generate credible explanations by associating interpretations with facts.

Overall, despite the minor mistakes, both the policy interpretation and the decision explanation are friendly to non-expert users and reveal some patterns in the agent's decision-making process.

5. Conclusions

We propose INSIGHT, a framework that uses object coordinates as structured state representations and learns symbolic policies from visual input. INSIGHT is able to refine the structured states with reward signals by distilling vision foundation models into a scalable perception module, and it leverages a new neural guidance scheme to learn competitive symbolic policies from object coordinates, thereby overcoming the performance bottleneck of previous NS-RL approaches. Moreover, to improve model transparency for non-expert users, INSIGHT can generate language explanations for learned policies and specific decisions with LLMs. With experiments on nine Atari tasks and a MetaDrive task, we show that INSIGHT outperforms all NS-RL baselines and further reveal that the improvement can be explained by the improved coordinate prediction for policy-relevant objects. We also showcase language explanations for learned policies and decisions.

Limitations Currently, the EQL network in INSIGHT cannot express logical operations required by some reasoning tasks. Furthermore, quantitative evaluation of the explanations is also an interesting future topic.

Acknowledgments

C. Fang was supported by the National Science and Technology Major Project (2022ZD0114902), the NSF China (No. 62376008). The authors would like to thank four anonymous reviewers for constructive feedback.

Impact Statement

This paper aims to improve RL's transparency without sacrificing task performance, which paves the road towards trustworthy RL applications. From a societal point of view, we aim to reduce the barrier between the general public and RL models, thus advancing the broad application of RL agents.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:29304–29320, 2021. 1

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2023. 2

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 6

Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A., Miller, T., Weber, R., and Magazzeni, D. Distilling deep reinforcement learning policies in soft decision trees. In *IJCAI workshop on explainable artificial intelligence*, 2019. 1, 2

Dazeley, R., Vamplew, P., and Cruz, F. Explainable reinforcement learning for broad-xai: a conceptual framework and survey. *Neural Computing and Applications*, pp. 1–24, 2023. 2

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022. 1

Delfosse, Q., Blüml, J., Gregori, B., Sztwiertnia, S., and Kersting, K. Ocatari: object-centric atari 2600 reinforcement learning environments. *arXiv preprint arXiv:2306.08649*, 2023a. A7

Delfosse, Q., Stammer, W., Rothenbächer, T., Vittal, D., and Kersting, K. Boosting object representation learning via motion and object continuity. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, pp. 610–628. Springer Nature, 2023b. A4

Delfosse, Q., Stammer, W., Rothenbächer, T., Vittal, D., and Kersting, K. Boosting object representation learning via motion and object continuity. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 610–628. Springer, 2023c. A8

Delfosse, Q., Shindo, H., Dhami, D., and Kersting, K. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. 1, 2, 5, 6

Delfosse, Q., Sztwiertnia, S., Stammer, W., Rothermel, M., and Kersting, K. Interpretable concept bottlenecks to align reinforcement learning agents. *arXiv preprint arXiv:2401.05821*, 2024b. 1, 2, 9

Ehsan, U., Harrison, B., Chan, L., and Riedl, M. O. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 /ACM Conference on AI, Ethics, and Society*, pp. 81–87, 2018. 2

Greydanus, S., Koul, A., Dodge, J., and Fern, A. Visualizing and understanding atari agents. In *International Conference on Machine Learning (ICML)*, pp. 1792–1801. PMLR, 2018. 2

Hayes, B. and Shah, J. A. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction*, pp. 303–312, 2017. 2

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *The Journal of Machine Learning Research*, 23 (1):12585–12602, 2022. 7, A5

Jia, B., Liu, Y., and Huang, S. Improving object-centric learning with query optimization. In *International Conference on Learning Representations (ICLR)*, 2023. 6

Kroeger, N., Ley, D., Krishna, S., Agarwal, C., and Lakkaraju, H. Are large language models post hoc explainers? *arXiv preprint arXiv:2310.05797*, 2023. 2

Landajuela, M., Petersen, B. K., Kim, S., Santiago, C. P., Glatt, R., Mundhenk, N., Pettit, J. F., and Faissol, D. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 5979–5989. PMLR, 2021. 2, 4, 5, 6

Li, Q., Huang, S., Hong, Y., Chen, Y., Wu, Y. N., and Zhu, S.-C. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. *International Conference on Machine Learning (ICML)*, 2020. 2

Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., and Zhou, B. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3461–3475, 2022. 5

Li, Q., Zhu, Y., Liang, Y., Wu, Y. N., Zhu, S.-C., and Huang, S. Neural-symbolic recursive machine for systematic generalization. *International Conference on Learning Representations (ICLR)*, 2024. 2

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017. 3

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020. 1

Lyu, D., Yang, F., Liu, B., and Gustafson, S. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the Conference on Artificial Intelligence*, pp. 2970–2977, 2019. 2

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 2

Martius, G. and Lampert, C. H. Extrapolation and learning equations. *International Conference on Learning Representations (ICLR)*, 2017. 4

Milani, S., Topin, N., Veloso, M., and Fang, F. A survey of explainable reinforcement learning. *arXiv preprint arXiv:2202.08434*, 2022. 1

Nguyen, V.-Q., Suganuma, M., and Okatani, T. Look wide and interpret twice: Improving performance on interactive instruction-following tasks. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021. 4

Qiu, W. and Zhu, H. Programmatic reinforcement learning without oracles. In *International Conference on Learning Representations*, 2022. 1

Sahoo, S., Lampert, C., and Martius, G. Learning equations for extrapolation and control. In *International Conference on Machine Learning (ICML)*, pp. 4442–4450. PMLR, 2018. 1, 4

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

Singh, C., Hsu, A. R., Antonello, R., Jain, S., Huth, A. G., Yu, B., and Gao, J. Explaining black box text modules in natural language with language models. *arXiv preprint arXiv:2305.09863*, 2023. 2

Tennenholtz, G., Chow, Y., Hsu, C.-W., Jeong, J., Shani, L., Tulepbergenov, A., Ramachandran, D., Mladenov, M., and Boutilier, C. Demystifying embedding spaces using large language models. *arXiv preprint arXiv:2310.04475*, 2023. 2

Topin, N., Milani, S., Fang, F., and Veloso, M. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the Conference on Artificial Intelligence*, pp. 9923–9931, 2021. 2

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 5045–5054. PMLR, 2018. 1, 2

Verma, A., Le, H., Yue, Y., and Chaudhuri, S. Imitation-projected programmatic reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2

Wang, X., Yuan, S., Zhang, H., Lewis, M., and Sycara, K. Verbal explanations for deep reinforcement learning neural networks with attention on extracted features. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–7. IEEE, 2019. 2

Wilson, D. G., Cussat-Blanc, S., Luga, H., and Miller, J. F. Evolving simple programs for playing atari games. In *Proceedings of the genetic and evolutionary computation conference*, pp. 229–236, 2018. 1, 2, 6

Wu, T., Huang, Q., Liu, Z., Wang, Y., and Lin, D. Distribution-balanced loss for multi-label classification in long-tailed datasets. In *European Conference on Computer Vision (ECCV)*, 2020. 3

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022. 1

Yang, Z. and Yang, Y. Decoupling features in hierarchical propagation for video object segmentation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 36324–36336, 2022. 2

Yoon, J., Wu, Y.-F., Bae, H., and Ahn, S. An investigation into pre-training object-centric representations for reinforcement learning. *International Conference on Machine Learning (ICML)*, 2023. 2

Yuan, Z., Xue, Z., Yuan, B., Wang, X., Wu, Y., Gao, Y., and Xu, H. Pre-trained image encoder for generalizable visual reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:13022–13037, 2022. 2

Zhang, L., Li, X., Wang, M., and Tian, A. Off-policy differentiable logic reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pp. 617–632. Springer, 2021. 2

Zhang, X., Guo, Y., Stepputtis, S., Sycara, K., and Campbell, J. Explaining agent behavior with large language models. *arXiv preprint arXiv:2309.10346*, 2023. 2

Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., and Wang, J. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023. 2

Zheng, W., Sharan, S., Fan, Z., Wang, K., Xi, Y., and Wang, Z. Symbolic visual reinforcement learning: A scalable framework with object-level abstraction and differentiable expression search. *arXiv preprint arXiv:2212.14849*, 2022. 1, 2, 5, 6

A. Details of INSIGHT

A.1. Details of the Frame-Symbol Dataset

This section provides a detailed explanation of the frame-symbol dataset generation process, expanding on the preliminary overview presented in Sec. 3.1.

The generation of the dataset begins with a comprehensive training regimen of 10 million steps using the neural baseline. This regimen incorporates a neural policy tailored for interaction with the environment. During the final 1 million steps, specifically starting from the 9-millionth step, we captured 10,000 consecutive frames spanning multiple episodes, thereby forming an unsupervised frame dataset. This phase primarily focuses on documenting environmental interactions via image captures, with a special emphasis on acquiring objects in high-reward scenarios.

Following this, the initial frame undergoes processing through FastSAM, which identifies and segregates unique objects by extracting their masks. Without assuming prior knowledge of the number of objects in the environment. To accommodate various environments, we establish a maximum object count of 256 for all experiments, a threshold that is sufficient for all Atari games. The objects are ordered based on their confidence scores from FastSAM. These masks serve as inputs for the DeAot module, enabling object tracking across the subsequent 9,999 frames. To maintain the consistency of object IDs, these 10,000 frames were spliced into a whole video and read by DeAot. FastSAM re-evaluates the scene every tenth frame to include new objects, leading to an increase in the object count. This increment prompts DeAot to start tracking these newly identified objects. For improved segmentation and tracking, all frames are resized to a resolution of 1024×1024 pixels.

During the segmentation phase using FastSAM, objects are excluded if their confidence level does not meet or exceed a threshold of 0.9. This stringent selection criterion is pivotal for minimizing misidentifications attributable to environmental factors. Moreover, within the same frame, a distinction is made between connected and non-connected masks. Connected masks that do not overlap with previously tracked objects are deemed new. Conversely, objects that overlap with an existing mask by 50% or less are also classified as new, i.e., IOU=0.5. The rationale for the former is to avoid fragmenting a single object into multiple smaller segments as much as possible, and for the latter, it is because the tracking module tends to recognize non-connected, similar objects as a single entity, necessitating their re-identification and enumeration.

In the dataset's final development phase, the dataset was re-segmented starting at 0th frame and the FastSAM module is deactivated, allowing DeAot to autonomously track objects from the initial to the final frame. The resulting dataset encompasses detailed information like object coordinates, bounding boxes, and RGB values, linked to their corresponding frames. When an object is missing from a frame, it is noted as 'non-existent' in the dataset, and its coordinates are set to (0,0). This approach allows for a systematic way to account for absent objects without disrupting the overall tracking and identification process. Ultimately, this leads to the formation of a comprehensive frame-symbol dataset, where frames are stored at a resolution of 512×512 pixels. This process mitigates a previously identified limitation where the tracking model struggled to consistently detect objects in each frame. Conforming to the standard supervised training approach, the dataset is divided into training and test sets in an 80:20 ratio.

A.2. Label Weights of the Distribution-Balanced Focal Loss

Denote by $n_j = 1/\sum_{i=1}^{N} c_{ij}$ the inverse frequency of the jth label, where N is the number of samples in $\mathcal{D}_{\text{symbol}}$. The inverse frequency is further normalized, since the number of labels varies across samples, and transformed to the range $[\alpha, \alpha+1]$ for numerical stability. That is, the weight of label c_{ij} is given by $\bar{\eta}_{ij} = \alpha + \sigma(\beta(\eta_{ij} - \mu))$, where $\eta_{ij} = n_j/\sum_{j'}^{C} c_{ij'} n_{j'}$ and $\sigma(\cdot)$ is the sigmoid function. α, β, and μ are hyper-parameters.

A.3. Details of F-MAE

The F-MAE metric evaluates the precision of predicted object coordinates within frames of Atari tasks, with a focus on objects essential for the agent's decision-making. Unlike traditional MAE, F-MAE targets a subset of objects identified as critical through symbolic regression. For a dataset $\mathcal{D}_{\text{symbol}}$ containing N samples, the presence of the jth object in the ith sample is marked by c_{ij}, where $c_{ij} = 1$ indicates the object's presence, and $c_{ij} = 0$ its absence. Let $\mathbf{x}_i \in \mathbb{R}^{2C}$ represent the vector of object coordinates in the ith image, and $\hat{\mathbf{x}}_i$ its predicted counterpart. For objects numbered $j = 1, 2, \ldots, C$, the coordinates $x_{i,2j}$ and $x_{i,2j+1}$ correspond to the Y and X positions, respectively. The F-MAE, focusing on a critical subset of objects denoted as S, is calculated using Eq. (A1):

$$\text{F-MAE} = \frac{1}{2EN|S|} \sum_{i=1}^{N} \sum_{j=1}^{C} s_{ij} c_{ij} \left(|x_{i,2j} - \hat{x}_{i,2j}| + |x_{i,2j+1} - \hat{x}_{i,2j+1}| \right). \tag{A1}$$

Here, s_{ij} denotes the inclusion of the jth object in the ith sample within the filtered subset, taking a value of 1 when included and 0 otherwise. The term E represents the total number of frames featuring the objects after filtering. The division by 2 accounts for the mean impact of the x and y coordinates. The term $|S|$ signifies the count of objects post-filtering, with the division by $|S|$ adjusting for the impact of individual objects. This formula constrains the F-MAE's range between 0 and 1, where 0 indicates perfect accuracy and 1 denotes complete inaccuracy, thus providing a clear metric for evaluating object coordinate prediction precision within task frames.

A.4. Conditions for Testing Inference Speed

In the experiments conducted as detailed in Tab. 3, the hardware setup comprised an AMD Ryzen 9 5950X 16-Core Processor for CPU, an NVIDIA GeForce RTX 3090 Ti as the graphics card, and 24564MiB of video memory. Each experiment involved executing 1000 steps on the Pong task, from which the average single-step inference time of the model was calculated.

B. Experimental Setup

B.1. Architecture and Hyperparameters

CNN Encoder Structure In Tab. A1, the structure of the CNN encoder is elaborated. Comprising three convolutional layers, each layer is distinctively configured with varying kernel sizes, strides, padding, and channel outputs. The initial resolution of the image is defined as 84x84 pixels, encompassing four channels. These channels amalgamate the grayscale images from the previous four temporal frames, capturing motion information.

The initial convolutional layer (Conv1) utilizes a 5x5 kernel, a stride of 2, and padding of 2 to output 32 channels. This configuration begins the feature extraction process, reducing spatial dimensions while enriching the feature map's depth. The following layers (Conv2 and Conv3) maintain this configuration but with an increased channel output of 64, capturing more intricate features.

Post-convolution, the network integrates a flattening step, converting the multi-dimensional feature maps into a singular vector. This vector feeds into a fully connected linear layer with 2048 output features and ReLU activation, transforming the detailed convolutional features for subsequent analysis.

A layer normalization follows the first linear layer, enhancing the learning process's stability and efficiency. This normalization standardizes data scales and enables faster training through higher learning rates.

In the final stage, the network includes distinct output layers for existence, coordinate, and shape predictions. The Existence Layer initially compresses the feature dimensions from 2048 to 1024 using a ReLU activation function. This is succeeded by a subsequent linear layer, which maintains this reduced feature dimension. The Coordinate Layer, conversely, preserves the feature count at 2048, while the Shape Layer diminishes it to 512. Both these layers incorporate ReLU activations and are followed by linear transformations for processing.

EQL Structure The input dimension of the EQL network is configured as 2048, specifically designed to accommodate the coordinate representation generated by the encoder. These coordinates are initially processed through a hidden layer, followed by a custom activation function. The activation function employs a variety of operations to avoid excessive gradient explosion and ensure sufficient representation ability. These include squaring ($f(x) = x^2$), cubing ($f(x) = x^3$), constant ($f(x) = c$), identity ($f(x) = x$), product ($f(x, y) = xy$), and addition ($f(x, y) = x + y$). Each of these functions is iteratively applied four times within the hidden layer, as detailed in Tab. A2. Subsequently, the output layer generates a symbolic expression corresponding to each action dimension within the environment. This process includes a transition through a softmax layer, where the final action is derived by random sampling from the resulting probability distribution. Furthermore, the output of the EQL network is multiplied by a temperature coefficient $t_{eql} = 10$, enhancing sensitivity to coordinate changes and aiding in the preservation of object coordinate prediction during policy learning.

Table A1. Hyperparameters of the CNN encoder.

Hyperparameter	Value				
Resolution	84x84				
Image Channels	4				
Encoder Configuration					
Layer	Kernel Size	Stride	Padding	Channels	Activation
Conv1	5x5	2	2	32	ReLU
Conv2	5x5	2	2	64	ReLU
Conv3	5x5	1	2	64	ReLU
Post-Convolution Layers					
Layer	Out Feature	Activation			
Flatten	-	-			
Linear	2048	ReLU			
LayerNorm	2048	-			
Output Layers					
Layer	Structure	Activation			
Existence Layer	Linear(2048, 1024)	ReLU			
	Linear(1024, 1024)	-			
Coordinate Layer	Linear(2048, 2048)	ReLU			
	Linear(2048, 2048)	-			
Shape Layer	Linear(2048, 512)	ReLU			
	Linear(512, 512)	-			

Table A2. Hyperparameters of EQL network.

Hyperparameter	**Value**
Input Dimensions	2048
Activation Function 1	$f(x) = x^2$
Activation Function 2	$f(x) = x^3$
Activation Function 3	$f(x) = c$
Activation Function 4	$f(x) = x$
Activation Function 5	$f(x, y) = xy$
Activation Function 6	$f(x, y) = x + y$
Number of Repetitions	4
Number of Hidden Layers	1
t_{eql}	10

Pretraining The pretraining stage employed the loss function detailed in Eq. (3), spanning 600 epochs with a batch size of 32. To counteract overfitting, a weight decay regularization of 1×10^{-4} was applied. The learning rate was established at 3×10^{-4}, with the Adam optimizer selected for its effectiveness in gradient descent optimization. Detailed specifics of the parameters, including their command line arguments and values, are provided in Tab. A3.

Policy Learing In the policy learning phase, following the common settings of Atari tasks, INSIGHT interacts with the environment for 10M steps. PPO with a learning rate of 2.5e-4 uses the collected rewards to optimize the model. The batchsize of each update is 1024. In order to ensure that the network can learn useful policies while ensuring the coefficient of the policy, we choose to use 1e-3 regularization and increase the regularization coefficient from 0 to 1 in each update. In addition, we multiply the coefficient of Eq. (3) by 2 to enhance the accuracy of coordinate prediction. All detailed parameters are summarized in Tab. A4.

Table A3. Hyperparameter of pretraining.

Hyperparameter	**Value**
Epoch	600
Batch Size	32
Learning Rate	3×10^{-4}
Weight Decay	1×10^{-4}
α of $\mathcal{L}_{\text{exist}}$	0.1
β of $\mathcal{L}_{\text{exist}}$	10
μ of $\mathcal{L}_{\text{exist}}$	Mean Value of n_j
γ of $\mathcal{L}_{\text{exist}}$	2
Optimizer	Adam
Loss Function	Eq. (3)

Table A4. Hyperparameters of Policy Learning.

Hyperparameter	**Value**
Total Steps	10M
Learning Rate	2.5×10^{-4}
Batch Size	1024
Initial value of λ_{reg}	1×10^{-3}
Coefficient for annealing λ_{reg} linearly	$\frac{\text{Update}-1}{\text{Total Updates}}$
λ_{cnn}	2

Table A5. **Distilled structured representations significantly enhance performance.** Evaluating the performance of SA-Neural, SPACE-Neural, Coor-Neural, and Neural after training for one million steps. Coor-Neural surpasses both SA-Neural and SPACE-Neural in most tasks and stands as the only competitor to Neural. This demonstrates the accelerated learning achieved by agents using our proposed method for structured state representation.

Tasks	SA-Neural	SPACE-Neural	Coor-Neural	Neural
Pong	5.4 ± 0.5	-20.6 ± 0	10.4 ± 6.2	**16.8** ± 3.8
BeamRider	684.8 ± 587.1	579.6 ± 348.4	846.1 ± 92.2	551.6 ± 25.1
Enduro	0 ± 0	8.1 ± 8.4	45.4 ± 63.5	9.7 ± 6.9
SpaceInvaders	35.8 ± 58.2	135 ± 134.4	**492.5** ± 61.4	490.4 ± 32.3
Freeway	0 ± 0	0 ± 0	**27.8** ± 2.9	17.1 ± 12.9
Qbert	1085.2 ± 0.5	289.1 ± 0	2030.2 ± 383.9	**5659.2** ± 1271.5
Seaquest	955.2 ± 587.1	200.3 ± 348.4	718.1 ± 112.1	1297.3 ± 359.6
Breakout	**65.5** ± 0	1 ± 8.4	39.7 ± 4.2	34.7 ± 23.5
MsPacman	643.2 ± 58.2	60.4 ± 134.4	1180.2 ± 245.9	1289.2 ± 106.7

C. Additional Experimental Results

This section encompasses all supplementary experiments, providing a comprehensive overview.

C.1. Comprehensive Evaluation of Representations

We highlight an additional advantage of adopting a distilled vision-based model: enhanced efficiency in task learning through improved representations. Tab. A5 presents the task performance of different approach to extract state representations accompanied by neural actors.

Distilled Structured Representations Significantly Enhance Performance These methods were trained for only one million steps due to the poor efficiency of SPACE-Neural. Similar experiment setup has been utilized by Delfosse et al. (2023b). We also report results for three million steps in Tab. A6, by which point the performances on Pong and Freeway had stabilized. Note that compared to slot attention or the SPACE model, our perception module leads to better performance. This is because our image-symbol dataset provides direct supervision for the location of objects. On the contrary, unsupervised methods are trained to reconstruct whole images and thus less focused on individual objects.

Table A6. **Distilled structured representations significantly enhance the performance upon convergence.** After training for three million steps, the performance of SA-Neural, SPACE-Neural, Coor-Neural, and Neural is evaluated. Coor-Neural consistently outperforms SA-Neural and SPACE-Neural upon convergence.

Tasks	SA-Neural	SPACE-Neural	Coor-Neural	Neural
Pong	17.2 ± 3	-20.6 ± 0	19.7 ± 0	**19.9** ± 0.3
Freeway	19.8 ± 15	12.2 ± 3	**31.6** ± 0.1	25.3 ± 6.8

Table A7. **INSIGHT demonstrates robustness to hyper-parameters.** MAE and F-MAE are used to measure coordinate prediction accuracy for Qbert, Seaquest, Breakout, and MsPacman, with all numbers scaled up by a factor of 100.

	INSIGHT		w/o Pretrain		Fix		w/o NG		Coor-Neural	
Tasks	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE	MAE	F-MAE
Qbert	1.3 ± 0	**0.8** ± 0	1.4 ± 0.1	0.9 ± 0.1	**1** ± 0	0.9 ± 0.1	1.4 ± 0.1	1.4 ± 0.1	1.7 ± 0.1	/
Seaquest	4.2 ± 0.1	**1.9** ± 0.2	4.5 ± 0.1	2 ± 0.2	**4.1** ± 0.1	3.7 ± 0.3	4.4 ± 0.1	4.4 ± 0.1	4.5 ± 0.1	/
Breakout	7.3 ± 0.4	**5.4** ± 0.6	6.4 ± 0.1	4.2 ± 0.3	**4.5** ± 0.2	**2.5** ± 0.1	5.2 ± 0.3	5.2 ± 0.3	6.7 ± 0.7	/
MsPacman	7.2 ± 0.1	**4.3** ± 0.5	6.9 ± 0.2	5.1 ± 0.4	**6.5** ± 0.2	6.8 ± 0.7	7.2 ± 0.3	7.2 ± 0.3	6.9 ± 0.3	/

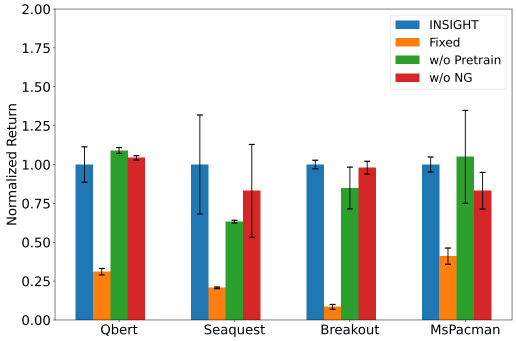


Figure A1. **Each component of INSIGHT is critical for overall performance.** Detailed performance analyses of INSIGHT and its variants across four tasks are presented. The findings in these four environments align with those in the other five environments.

C.2. Analysis of Coordinate Accuracy

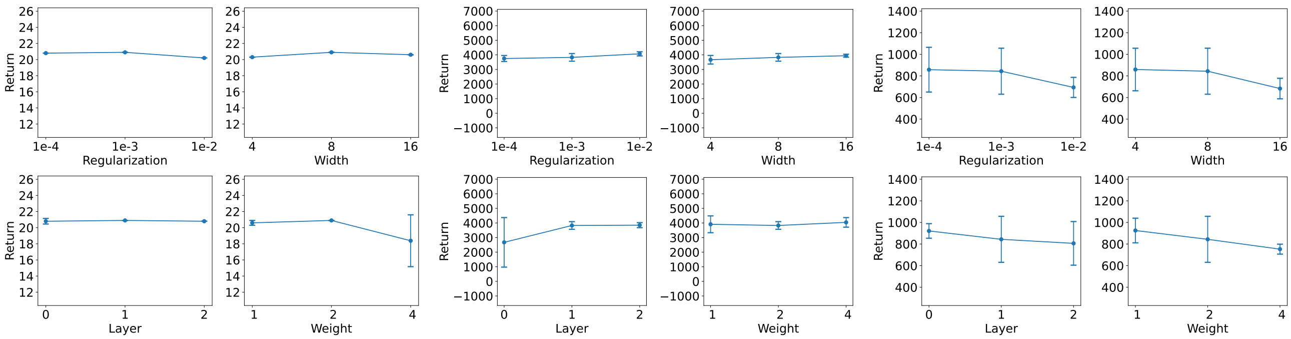
End-to-End Fine-Tuning Markedly Enhances Coordinate Prediction Accuracy in Policy-Relevant Tasks Tab. 4 presents a quantitative assessment of the coordinate prediction accuracy achieved by our method. In Tab. A7, we document the experimental results for the additional four tasks. Notably, the F-MAE on Qbert, Seaquest, and MsPacman post-training markedly surpasses the pre-training figures. This suggests that end-to-end fine-tuning significantly improves accuracy in policy-relevant object coordinates.

C.3. Extended Ablation Study Details

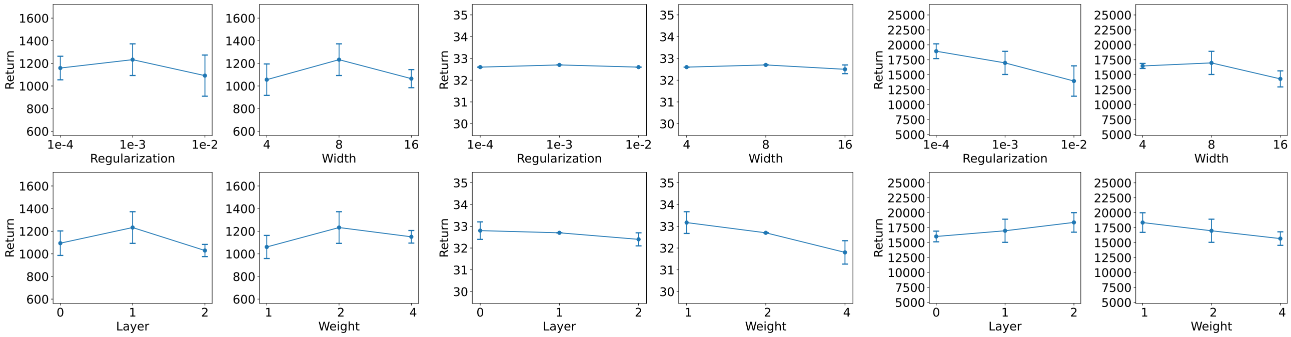
Generalizability of Ablation Experiment Conclusions Across Tasks This subsection extends the ablation studies detailed in Fig. 3 and Fig. 4 to encompass all tasks, as referenced in Fig. A1-Fig. A2i. Notably, the comprehensive ablation analysis presented in Fig. A1 demonstrates performance declines in most tasks when methods are modified. Furthermore, the consistency observed across various hyperparameters, as shown in Fig. A2a-Fig. A2i, corroborates the findings discussed in Sec. 4.2.

C.4. Additional Baseline

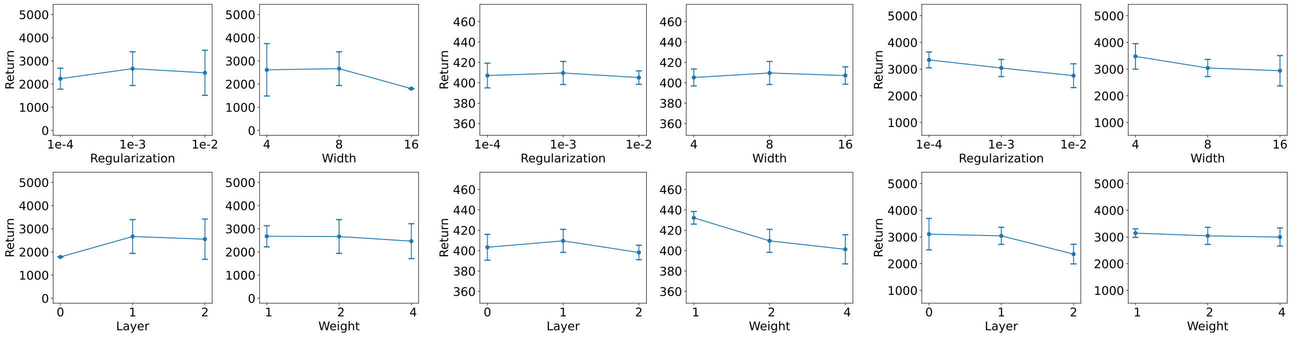
INSIGHT Outperforms CleanRL In this section, we introduce an additional baseline, CleanRL, to evaluate the performance of INSIGHT. CleanRL is an open-source implementation derived from the CleanRL library (Huang et al., 2022). The sole modification involves adjusting the CNN's output dimension from 512 to 2048 or higher to facilitate a fair comparison with our framework. It is noteworthy that our neural baseline exhibits competitive performance relative to CleanRL. As shown in Tab. A8, INSIGHT achieves superior returns in six tasks, unequivocally demonstrating its capability



(a) Pong. (b) BeamRider. (c) Enduro.



(d) SpaceInvaders. (e) FreeWay. (f) Qbert.

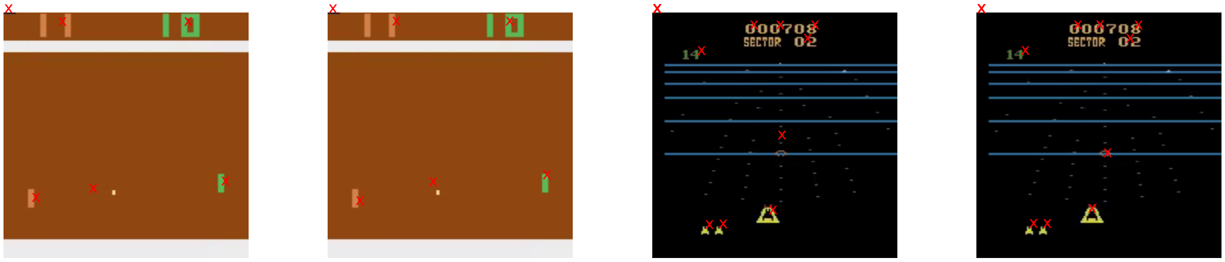


(g) Seaquest. (h) Breakout. (i) MsPacman.

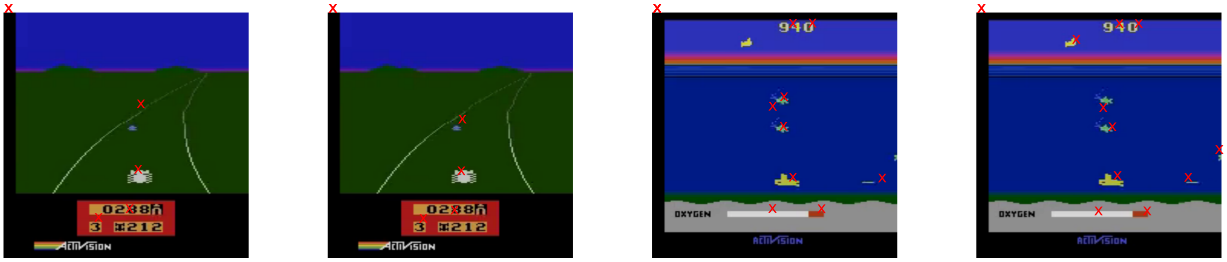
Figure A2. **INSIGHT demonstrates robustness to hyper-parameters.** Examining the effects of the regularization coefficient λ_{reg}, the width and number of hidden layers of the EQL actor, and the weight of $\mathcal{L}_{\text{CNN}}$ on performance. INSIGHT shows substantial robustness to variations in these hyper-parameters.

Table A8. **INSIGHT outperforms CleanRL.** Evaluating the performance of the proposed INSIGHT, Neural, and CleanRL (Huang et al., 2022) across nine Atari tasks. INSIGHT matches Neural's performance on all tasks and surpasses all NS-RL baselines. In most instances, our neural baseline competes closely with the CleanRL baseline.

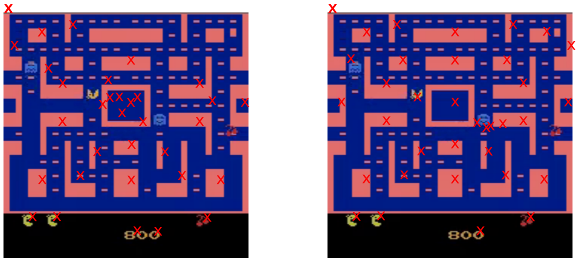
Task	INSIGHT	Neural	CleanRL
Pong	**20.9** ± 0.1	20.4 ± 0.6	20.7 ± 0.4
BeamRider	3828.1 ± 261.1	**3868.1** ± 204.2	2270.5 ± 107.3
Enduro	843.7 ± 213.8	676.9 ± 730.4	**1006.6** ± 244.1
Qbert	16978.6 ± 1936.1	**17879.2** ± 1857.1	16619.8 ± 256.3
SpaceInvaders	**1232.6** ± 140.7	1184.6 ± 137.6	1023.1 ± 160.2
Seaquest	**2665.7** ± 728.2	1804.8 ± 20.1	1841.2 ± 17.1
Breakout	**409.6** ± 11.3	259.6 ± 183.8	387.8 ± 11.8
Freeway	**32.7** ± 0.1	28.7 ± 5.3	32.2 ± 1.1
MsPacman	**3042.5** ± 320.1	2737.1 ± 562.3	2237.2 ± 163.5



(a) Pong (before). (b) Pong (after). (c) BeamRider (before). (d) BeamRider (after).



(e) Enduro (before). (f) Enduro (after). (g) Seaquest (before). (h) Seaquest (after).



(i) MsPacman (before). (j) MsPacman (after).

Figure A3. Predicted coordinates before and after policy learning.

to outperform existing neural policies in online environments.

C.5. Visualized Coordinate Tracking

In this section, we expand upon the advantages of end-to-end training for coordinate prediction, initially presented in Tab. 4, by offering a more intuitive visualization. Fig. A3 illustrates the object coordinates before and after the training process. For instance, in Figs. A3a and A3b, we note a marked reduction in the pixel coordinate shift of the ball post-training. Similarly, the laser emitted by the agent in Figs. A3c and A3d and the car on the road in Figs. A3e and A3f exhibit significantly diminished pixel coordinate shifts following training. Moreover, post-training observations in Figs. A3g and A3h reveal the successful detection of enemy coordinates at the periphery, a detail that was previously unattainable. Lastly, in Figs. A3i and A3j, the model demonstrates its ability to consistently track the blue enemy post-training, overcoming the initial limitation of losing track of the object.

C.6. Accuracy Evaluation of the FastSAM and DeAot Method

Our Approach Competes with MOC Solely Through Zero-Shot Generalization To evaluate the reliability of our dataset generation method, we further evaluate the accuracy of bounding boxes in the OCAtari dataset (Delfosse et al., 2023a). The table below shows the F-score from the Pong, SpaceInvaders and MsPacman environments within the OCAtari dataset, compared with the results of SPACE+MOC, as reported in (Delfosse et al., 2023c):

Table A9. **Our method competes with MOC solely through zero-shot generalization.** The F-score evaluation of the FastSAM and DeAot method on OCAtari.

Env	INSIGHT	SPACE+MOC(w/o OC)	SPACE+MOC
Pong	97.5	91.5	87.4
SpaceInvaders	86.5	85.1	85.2
MsPacman	46.5	88.6	90.5

Table A10. **Our method achieves high recall solely through zero-shot generalization.** The precision and recall evaluation of the FastSAM and DeAot method on OCAtari.

Env	Precision	Recall
Pong	96.3	98.7
SpaceInvaders	88.5	84.5
MsPacman	30.5	97.8

Note that INSIGHT achieves an impressive F-score of 97.5% on Pong, significantly outperforming SPACE+MOC. For SpaceInvaders, we set IOU to 1 to segment as many different objects as possible. However, MsPacman's F-score is lower than that achieved with the MOC-based method.

Our Method Achieves High Recall Solely Through Zero-Shot Generalization To further analyze this, we provide the precision and recall scores in Table A10. In the MsPacman environment, the very low precision and nearly perfect recall indicate that while all objects were segmented, many irrelevant objects not in the OCAtari dataset were also included. Given our goal during dataset generation is to segment as many complete objects as possible for downstream symbolic policy, higher precision may be more appropriate than F-score in RL scenarios.

C.7. Additional Computation Time Information

The average training time of each component on 9 Atari environments using NVidia RTX 3090 GPU is as follows:

Table A11. Training time of each component on 9 Atari environments.

	Pretraining the Visual Perception	Policy learning	Policy Learning w/o Finetune
Time (ms)	2.6 ± 0.3	7.3 ± 1.1	6.2 ± 0.8

Note that the end-to-end training time of the visual module only accounts for about 1/10 of the total pipeline, but it greatly improves the return and prediction accuracy, which reflects the high profitability.

Overall Computation Time Increases with the Number of Objects In Tab. A12, we present the computation times of FastSAM, SAM, and DeAot with varying numbers of objects: The segmentation speed of FastSAM remains relatively

Table A12. **Overall computation time increases with the number of objects.** The computation times (s) of FastSAM, SAM, and DeAot with varying numbers of objects.

Num	FastSAM	SAM	DeAot
5	0.07	4.21	0.01
10	0.07	4.57	0.02
20	0.07	5.21	0.02
50	0.08	5.83	0.03
100	0.08	5.84	0.07

constant despite varying object counts, whereas DeAot's speed significantly decreases as more objects are introduced. This suggests an increase in overall computational overhead proportional to the number of objects.

The average inference time of each component on 9 Atari environments using NVidia RTX 3090 GPU is as follows

Table A13. Inference time of each component on 9 Atari environments.

	CNN	EQL
Time (ms)	0.3	1.7

Note that we have adopted a simple CNN as the tracking module, which significantly enhances the inference speed of INSIGHT.

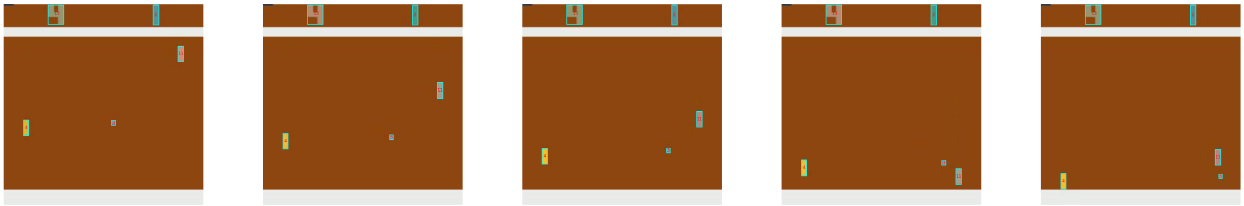
C.8. Dataset segmentation results

To visually demonstrate our segmentation accuracy, we took five frames from each Atari game, and summarized them in Appx. C.8:

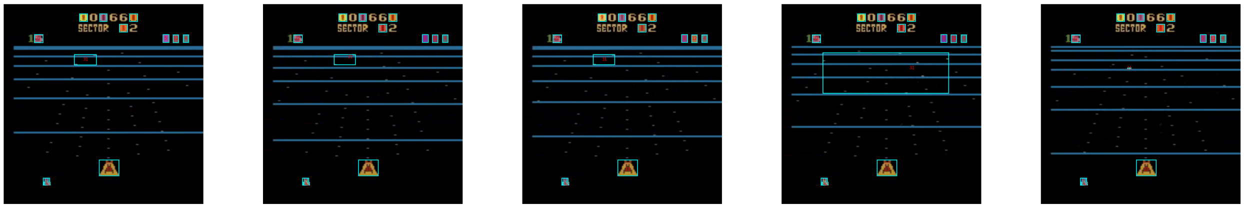
Note that our segmentation data set completes accurate segmentation of objects in most cases.

D. Comprehensive Prompt Template Overview

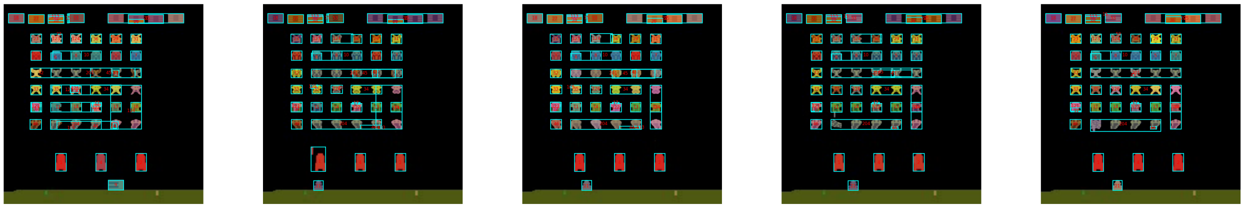
In this section, the full prompt template is presented. It comprises a public description template, along with two distinct policy illustration templates. The public description template outlines the fundamental aspects of the task and the underlying policy (refer to Tab. A14 and Tab. A15). The policy illustration templates are further bifurcated into policy explanation and decision explanation, detailed in Tab. A16 and Tab. A17, respectively.



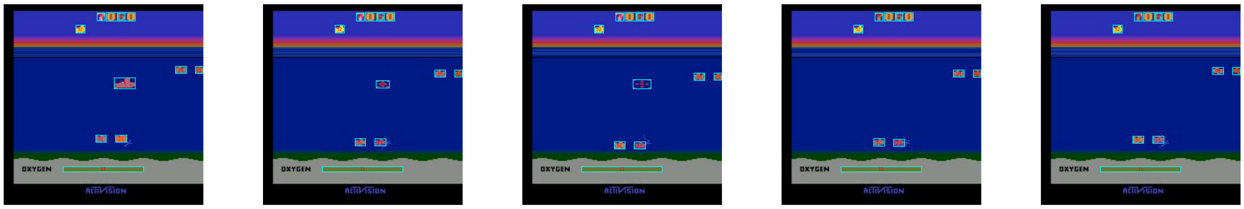
(a) Pong.



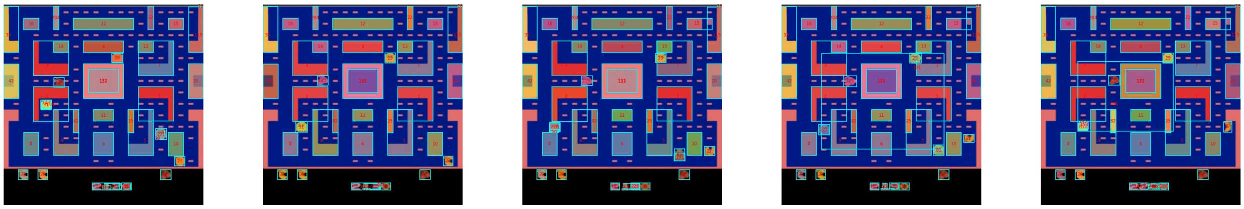
(b) BeamRider.



(c) SpaceInvaders.



(d) Seaquest.



(e) MsPacman.

Figure A4. FastSAM and DeAot Extract Segmentation Results from Five Frames per Game, Sampled Every Five Frames.

Table A14. The first part of pong's public prompt.

You need to help a user to analyze a control policy for the task Pong available in the OpenAI Gym repository. The policy is obtained with deep reinforcement learning.
You need to first understand the goal of the task and the policy.

# task Description

There are two paddles in the task screen, which are located at the left and right side of the screen. The agent controls the right paddle, and its opponent controls the left paddle. Both of them can only control the paddle to move up or down. They cannot move leftward or rightward

Like a pingpong task, the agent competes against its opponent by stricking the ball to the opponent's side (left). The agent earns a point if its opponent fails to strick the ball back.

The agent needs to solve the task in discrete steps. At each step, it takes as input the task screen, and it needs to take one of the three actions:

* noop: take no operation
* up: move its paddle upward.
* down: move its paddle downward.

# The policy

## Input Variable

We set up a xOy-coordinate system for the task screen. The origin is at the upper left corner. The positive direction of the y-axis is downwards, and the positive direction of the x-axis is to the right. We provide the agent with the latest four consecutive frames and use the coordinates of objects in these frames as input. Frame 4 is the current frame. Frame 3 is the frame obtained at one step before. Frame 2 is the frame obtained at two steps before, and frame 1 is the frame obtained at three steps before. You can use the coordinates of the same object in different time steps to infer the motion of the object.

The objects of interest are the agent, the opponent, and the ball. The input variables follows this naming convention: [x/y]_object_frame. For example, x_agent_1 is the x coordinate of the agent at frame 1. Remember, the input variables represent coordinates of some objects, and they are in the range [0,1].

Table A15. The second part of pong's public prompt.

## Logits

logits_noop1 = -0.56*y_agent_1**2 - 0.38*y_agent_1*y_agent_2 - 0.087*y_agent_1*y_opponent_1 - 0.16*y_agent_1*y_opponent_2 - 0.76*y_agent_1*y_opponent_3 - 0.51*y_agent_1*y_opponent_4 - 0.54*y_agent_1 - 0.24*y_agent_2**2 - 0.073*y_agent_2 + 0.27*y_agent_4**2 + 0.55*y_agent_4 - 0.078*y_opponent_1**2 - 0.33*y_opponent_1*y_opponent_2 - 0.2*y_opponent_1 - 0.35*y_opponent_2**2 - 0.5*y_opponent_2 - 0.34*y_opponent_3**2 - 0.45*y_opponent_3*y_opponent_4 - 0.32*y_opponent_3 - 0.15*y_opponent_4**2 - 0.19*y_opponent_4 + 1.1

logits_noop2 = -0.074*y_agent_1*y_opponent_2 + 0.059*y_agent_1*y_opponent_3 - 0.097*y_agent_4 - 0.16*y_opponent_1*y_opponent_2 - 0.18*y_opponent_2**2 - 0.27*y_opponent_2 + 0.063*y_opponent_4

logits_up1 = 0.23*y_agent_1**2 + 0.59*y_agent_1*y_agent_2 + 0.4*y_agent_2**2 + 0.11*y_agent_2 - 1.5*y_agent_4**2 - 3.6*y_agent_4 + 0.068*y_opponent_3 + 1.1

logits_down1 = 0.09*x_ball_3 + 0.12*x_ball_4 - 0.21*y_agent_1**2 + 0.12*y_agent_1*y_opponent_1 + 0.27*y_agent_1*y_opponent_2 - 0.43*y_agent_1*y_opponent_3 - 0.28*y_agent_1*y_opponent_4 + 0.13*y_agent_2 + 0.14*y_agent_4**2 + 0.43*y_agent_4 + 0.087*y_ball_3 + 0.15*y_ball_4 + 0.14*y_opponent_1**2 + 0.6*y_opponent_1*y_opponent_2 + 0.61*y_opponent_1 + 0.65*y_opponent_2**2 + 1.1*y_opponent_2 - 0.2*y_opponent_3**2 - 0.26*y_opponent_3*y_opponent_4 - 2.8*y_opponent_3 - 0.085*y_opponent_4**2 - 0.14*y_opponent_4 - 2.3

logits_up2 = 0.063*x_ball_4 - 0.078*y_agent_1 + 0.18*y_agent_2**2 + 0.52*y_agent_2*y_agent_3 + 0.35*y_agent_2*y_opponent_1 + 0.29*y_agent_2*y_opponent_2 + 0.26*y_agent_2 + 0.38*y_agent_3**2 + 0.51*y_agent_3*y_opponent_1 + 0.42*y_agent_3*y_opponent_2 + 1.6*y_agent_3 - 8.2*y_agent_4 - 0.085*y_ball_3 + 0.17*y_opponent_1**2 + 0.28*y_opponent_1*y_opponent_2 + 0.45*y_opponent_1 + 0.11*y_opponent_2**2 + 0.15*y_opponent_2 - 0.074*y_opponent_3 + 0.26

logits_down2 = -0.052*x_ball_1 - 0.068*x_ball_3 - 0.093*x_ball_4 + 0.18*y_agent_1 - 0.17*y_agent_2**2 - 0.49*y_agent_2*y_agent_3 - 0.33*y_agent_2*y_opponent_1 - 0.27*y_agent_2*y_opponent_2 - 0.39*y_agent_2 - 0.35*y_agent_3**2 - 0.48*y_agent_3*y_opponent_1 - 0.4*y_agent_3*y_opponent_2 - 0.38*y_agent_3 + 0.15*y_agent_4**2 + 0.54*y_agent_4 - 0.06*y_ball_1 - 0.064*y_ball_3 - 0.11*y_ball_4 - 0.17*y_opponent_1**2 - 0.28*y_opponent_1*y_opponent_2 - 0.58*y_opponent_1 - 0.13*y_opponent_2**2 - 0.38*y_opponent_2 + 2.2*y_opponent_3 - 0.052*y_opponent_4 - 3.6

## The Probability of Actions

action_noop = [exp(logits_noop1) + exp(logits_noop2)] / sum(exp(logits))

action_up = [exp(logits_up1) + exp(logits_up2)] / sum(exp(logits))

action_down = [exp(logits_down1) + exp(logits_down2)] / sum(exp(logits))

Table A16. Pong's policy interpretation prompt.

# Your Task
You need to analyze this policy based on its mathematical properties. You must follow the following rules.

1. You can also leverage your own knowledge about the goal of the task, but the conclusions for the policies have to be based on the mathematical properties of the policy.

2. You need to analyze the policy in these three steps: (a) analyze how changes in variables affect action logits, (b) analyze how changes in logits affect the probability of taking action, and (c) summarize the Influence of input variables on action probabilities.

3. When performing (a), remember that the input variables represent the location of an object. Take into consideration that the input variables are within [0,1]. Pay attention to the coefficients of each input variable and constants (if any).

4. When performing (b), remember that the probability of actions sum to one.

5. An increase of the logit of certain actions might results in an increase in the probability of that action. A decrease of the logit of certain actions might results in an decrease in the probability of that action.

6. When performing (c), summarize your findings from (a) and (b).

For example, for logits_up1, first think about the coefficient of x_ball_2. Since the values of x_ball_2 are within [0,1], how does it affect the logit of moving up? How does it affect the probability of moving up?

7. Be specific the effect of each term.

## Output
Organize your response as (1) equation, (2) influential variables, and (3) analysis. Render the equations into latex format. Use the object names and frame indices as subscripts. For example, y_\text{agent,1}. Use the name of actions as the subscript of logits. For example, logits_\text{noop}. Only keep two significant digits for each number.

Now, analyze action noop.

{Chatgpt response}

Analyze action up.

{Chatgpt response}

Analyze action down.

Provide a summary for your recent analysis. Follow the rules below.

1. Be specific on when will the agent chooses certain actions.
2. Your summary should be consistent with your analysis.
3. Organize your response in markdown format.

Here is a recap for our set up.

{Recap in public prompt}

Table A17. Pong's decision explanation prompt.

# Your Task
We used this policy to play the task and collected some data. You need to explain why the agent took a specific action when the input variables took specific values.

The action taken by the agent is up.
The value of y_ball_1 is 0.9018810391426086
The gradient of the log-likelihood for action up with respect to y_ball_1 is 2.75e-04.
The value of x_ball_1 is 0.5703107714653015
The gradient of the log-likelihood for action up with respect to x_ball_1 is -4.13e-04.
The value of y_ball_2 is 0.5564423203468323
The gradient of the log-likelihood for action up with respect to y_ball_2 is -4.36e-04.
The value of x_ball_2 is 0.6364298462867737
The gradient of the log-likelihood for action up with respect to x_ball_2 is -7.43e-06.
The value of y_ball_3 is 0.4664875864982605
The gradient of the log-likelihood for action up with respect to y_ball_3 is -8.24e-04.
The value of x_ball_3 is 0.7508251070976257
The gradient of the log-likelihood for action up with respect to x_ball_3 is 3.87e-04.
The value of y_ball_4 is 0.7325012683868408
The gradient of the log-likelihood for action up with respect to y_ball_4 is 7.06e-04.
The value of x_ball_4 is 1.0
The gradient of the log-likelihood for action up with respect to x_ball_4 is 1.34e-03.
The value of y_opponent_1 is 1.0
The gradient of the log-likelihood for action up with respect to y_opponent_1 is 2.09e-02.
The value of y_opponent_2 is 1.0
The gradient of the log-likelihood for action up with respect to y_opponent_2 is 1.67e-02.
The value of y_opponent_3 is 1.0
The gradient of the log-likelihood for action up with respect to y_opponent_3 is -2.63e-03.
The value of y_opponent_4 is 1.0
The gradient of the log-likelihood for action up with respect to y_opponent_4 is -8.92e-04.
The value of y_agent_1 is 1.0
The gradient of the log-likelihood for action up with respect to y_agent_1 is -1.39e-02.
The value of y_agent_2 is 1.0
The gradient of the log-likelihood for action up with respect to y_agent_2 is 4.85e-03.
The value of y_agent_3 is 1.0
The gradient of the log-likelihood for action up with respect to y_agent_3 is 4.58e-02.
The value of y_agent_4 is 0.0
The gradient of the log-likelihood for action up with respect to y_agent_4 is -4.94e-02.

## Output

You need to provide a concise explanation for why the agent took this action when the input variables took these values. For example, would the agent earn a point by choosing such an action?

There are a few rules that need to be followed.
1. Your explanations should be specific. You should explain why the action up is preferred over other actions.
2. Your explanations should be easy to read.
3. Your explanations should be entirely based on the equations for the policy, the values of input variables, and the gradients of action log-likelihood with respect to input variables.
4. Your explanations should be consistent with the definition of the input variables and the coordinate system.

Render the equations into latex format. Use the object names and frame indices as subscripts. For example, y_\text{agent,1}. Use the name of actions as the subscript of logits. For example, logits_\text{noop}. Only keep two significant digits for each number.