Mental Model Alignment: Building Cognitive Interfaces for Explainable Reinforcement Learning

Kejia Wan (wankejiaai@163.com), Hengzhu Liu (liuhengzhu@nudt.edu.cn), Jinlong Tian (tianjinlong@nudt.edu.cn) and Hao Tang (tanghao@nudt.edu.cn)
College of Computer Science and Technology, National University of Defense Technology
No. 137 Yanwachi Street, Changsha, Hunan, 410073, China

Xinhai Xu (xuxinhai_nudt@163.com), Yuntao Liu (liuyuntao@nudt.edu.cn) and Xianglong Li (lixianglong@163.com)
Academy of Military Sciences
No. 128 Xiangshan Road, Qinglongqiao Street, Haidian District, Beijing, 100089, China

Abstract

Deep reinforcement learning has achieved remarkable success in complex decision-making tasks, yet its black-box nature limits practical deployment in safety-critical domains. Current explainable reinforcement learning methods often fail to align with the hierarchical and temporal structure of human mental models, which are central to cognitive science theories of decision making. To bridge this gap, we propose Mental Model Alignment (MMA), a novel framework that constructs cognitive interfaces using behavior trees to harmonize AI decision-making with human-understandable reasoning. MMA introduces three innovations: (1) a mental model encoder that captures the hierarchical decomposition of tasks into subgoals, mirroring human cognitive processes; (2) a cognitive pruning algorithm that simplifies BTs while preserving decision-critical nodes aligned with human mental schemas; and (3) a mental effort metric to quantify the cognitive load required for users to interpret policies. Evaluated across six benchmark environments, MMA outperforms state-of-the-art methods in interpretability, policy fidelity, and computational efficiency. Our results demonstrate that aligning AI policies with human mental models significantly enhances trust and usability in real-world applications.

Keywords: Artificial Intelligence; Decision making; Intelligent agents; Machine learning; Theory of Mind

Introduction

The transformative potential of deep reinforcement learning (DRL) in autonomous decision making has been demonstrated in domains as diverse as robotics, game theory, and resource management (Seo et al., 2024; Zhou, Tian, Buyya, Xue, & Song, 2024; Dai, 2024). Yet, the very complexity that enables DRL to excel in these tasks also renders its decision-making processes opaque, hindering its adoption in scenarios where human oversight and trust are non-negotiable—such as medical diagnostics, autonomous vehicle navigation, or educational tutoring systems (Bodria et al., 2023; Han, Choi, Lee, & Kim, 2023). Although explainable reinforcement learning (XRL) seeks to address this opacity, the prevailing methods often do not resonate with the cognitive frameworks that humans naturally employ to understand and reason about complex systems (Sreedharan, Soni, Verma, Srivastava, & Kambhampati, 2021; Iyer et al., 2018). Human decision making is inherently grounded in mental models, dynamic hierarchical constructs that decompose tasks into sub-goals, anticipate outcomes through causal reasoning, and adapt strategies dynamically when faced with failures. These models, as described by Johnson-Laird, are not mere collections of isolated rules, but interconnected schemas shaped by experience and context (Johnson-Laird, 1983). Traditional XRL approaches, such as saliency maps or simplified decision trees, provide fragmented insights that neglect this hierarchical and temporal structure, leaving users unable to trace the rationale behind AI actions or predict their long-term consequences (Stephanie et al., 2024). This misalignment between machine logic and human cognition undermines trust and limits effective collaboration between humans and AI agents (Prystawski, Li, & Goodman, 2024).

The limitations of current XRL methods stem from a fundamental disconnect between technical explanations and cognitive principles. For instance, while feature importance analyses highlight which inputs influenced a single decision, they ignore the sequential logic that binds actions into coherent plans—a critical aspect of how humans reason about tasks like navigating a city or diagnosing a patient. Similarly, rule-based models like finite-state machines impose rigid hierarchies that clash with the fluid, context-dependent nature of mental models (Tasse, Jarvis, James, & Rosman, 2023; Iovino et al., 2023). Compounding this issue is the problem of cognitive overload: overly intricate explanations overwhelm users, violating the "chunking" principle of human working memory, which posits that individuals can retain only a limited number of information units simultaneously (Miller, 1956). To bridge this gap, we propose Mental Model Alignment, a framework that reimagines XRL through the lens of cognitive science. By structuring reinforcement learning policies as behavior trees (BTs), a modular, hierarchical formalism, MMA mirrors the way humans decompose tasks into subgoals (e.g. "reach a checkpoint" followed by "avoid obstacles") and contingency plans (e.g., "if traffic congestion is detected, reroute") (Wan, Liu, Liu, & Xu, 2024; Yang, Mao, Lu, & Xu, 2022; Khatri, 2023). This alignment ensures that AI policies not only perform effectively but also communicate their reasoning in a manner that aligns with human intuition.

Central to MMA is the integration of cognitive theories into technical design. The framework introduces a mental model encoder that distills raw trajectory data into subgoal-oriented representations, inspired by hierarchical task networks in both AI and cognitive psychology. This encoder

In D. Barner, N.R. Bramley, A. Ruggeri and C.M. Walker (Eds.), *Proceedings of the 47th Annual Conference of the Cognitive Science Society*

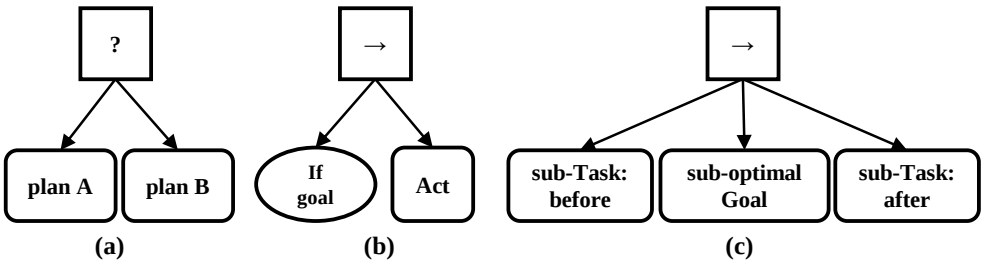

Figure 1: Illustration of the components of a behavior tree. (a) Plan A and plan B are connected by selecting the node. (b) When the sub-optimal goal is judged to be successful, the action is executed, otherwise it switches to plan B. (c) Plan A is divided into the early stage, the target period and the late stage. If any one of them fails, it will lead to the failure of the plan.

identifies latent subgoals within decision sequences, such as the transition from "positioning" to "executing a spin" in a bowling strategy, and constructs BTs that explicitly map these subgoals to modular branches. To prevent cognitive overload, a novel cognitive pruning algorithm simplifies the resulting trees by removing redundant nodes while preserving decision-critical paths—a process guided by chunking theory and validated against human working memory limits. The efficacy of this approach is measured through a mental effort metric, which quantifies the cognitive load required for users to interpret policies, combining objective task performance with subjective assessments like NASA-TLX scores. Empirical evaluations across six benchmark environments, including Atari games and multi-agent StarCraft II scenarios, demonstrate MMA's superiority over state-of-the-art XRL methods.

Behavior Tree Policy

At the core of human decision-making lies the concept of mental models—dynamic, internal representations that simulate how systems operate, predict outcomes, and guide adaptive behavior. They enable individuals to decompose complex tasks into hierarchical subgoals, infer causal relationships between actions and consequences, and dynamically adjust strategies when faced with disruptions. This hierarchical decomposition mirrors the structure of behavior trees, where high-level goals are recursively divided into subtasks until atomic actions are reached. BTs formalize this cognitive process, providing a visual and logical scaffold that aligns with how humans naturally organize knowledge, which can be represented as directed rooted trees:$BT = (Root, Act, Con, Seq, Sele)$. *Root* is the root node of the tree, from which the execution of the tree begins. *Act* is the set of action nodes, representing atomic actions or tasks that the agent can perform. *Con* is the set of conditional nodes (indicated by ovals graphically), representing conditions that determine whether certain actions should be executed. *Seq* is the set of sequence nodes (indicated by arrows graphically), representing nodes that execute their children in order until one succeeds. *Sele* is the set of selector nodes (indicated by question marks graphically), representing nodes that execute their children in order until one succeeds.

The alignment between BTs and mental models extends to their handling of uncertainty and failure. Human cognition inherently prepares for contingencies—a phenomenon described as counterfactual reasoning, where individuals anticipate alternative outcomes and devise backup plans. In a BT, this is operationalized through selector nodes that dynamically switch between branches based on environmental feedback. Consider a medical diagnosis scenario: a clinician might first pursue a likely hypothesis (e.g., "if symptoms suggest infection, prescribe antibiotics") but switch to testing for rare conditions if the patient's condition deteriorates. A BT encoding this logic would explicitly represent both the primary strategy and its alternatives, making the decision-making process transparent and auditable. This transparency is critical for fostering trust, as users can trace not only what the system does but why it pivots between strategies, mirroring the introspective nature of human reasoning. We define the RL dataset as a set of trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, where τ_i denotes the i-th trajectory of the dataset expressed as $\tau_i = \left(x_1^{(i)}, \ldots, x_L^{(i)}\right)$, where $x_t^{(i)} = (s_t^{(i)}, a_t^{(i)}, r_t^{(i)})$. N represents the total number of trajectories. When faced with a task, humans typically prepare both a primary plan (Plan A) and a contingency plan (Plan B). We hypothesize that this phenomenon arises because plan A incorporates a sub-optimal goal x_A in addition to the ultimate objective of mission success, while plan B does not:

$$task(D) = Sele[PlanA(D_{\{A\}}), PlanB(D_{\{-A\}})],$$

where $D_{\{A\}}$ denotes a set of trajectories containing x_A: $D_{\{A\}} = \{\tau_i\} \in D, x_A \in \tau_i$. $D_{\{-A\}}$ is the complement of $D_{\{A\}}$:$D_{\{-A\}} = D - D_{\{A\}}$. For plan A, its process is divided into three sub-tasks: how to achieve the sub-optimal goal, sub-optimal goal detection, and how to utilize the sub-optimal goal:

$$PlanA(D_{\{A\}}) = Seq[task(D_{\{A\}}^{before}), O_A, task(D_{\{A\}}^{after})],$$

where $D_{\{A\}}^{before}$ and $D_{\{A\}}^{after}$ represent the two parts of $D_{\{A\}}$ truncated from x_A: $D_{\{A\}}^{before} = \{\tau_i'\}, \tau_i' = (x_1^{(i)}, \ldots, x_A^{(i)})$ and $D_{\{A-1\}}^{after} = \{\tau_i'\}, \tau_i' = (x_{A+1}^{(i)}, \ldots, x_L^{(i)})$. The critical transition point occurs when situational developments indicate that the

sub-optimal goal cannot be achieved, prompting the activation of Plan B :

$$O_A = Seq[Con(s_A), Act(a_A)].$$

For plan B, it cannot be divided into sub-tasks due to the lack of x_A. We solve it by treating plan B as a task that does not contain x_A:

$$PlanB = task(D_{\{-A\}}).$$

This assumption is grounded in well-established principles from hierarchical decision-making frameworks in reinforcement learning and control theory. In trajectory optimization frameworks (Todorov & Li, 2005), sub-goals along a trajectory are often used to simplify complex planning problems, supporting the idea that sub-optimal intermediate steps are meaningful abstractions. These methods highlight that subgoals are an effective way to structure decision-making hierarchically. We express it in terms of BTs, as shown in figure.1. Obviously, a BT can be systematically generated by iteratively decomposing the task and the dataset. This process continues until the tasks are no longer divisible or the tree reaches a predefined depth threshold *mathcalB*. After reaching the depth threshold, we select the trajectory with the highest reward to build a subtree of sequential execution without subdividing the data set. This hierarchical approach ensures that each sub-task is manageable and aligns with the overall objective. This alignment not only enhances transparency but also empowers users to collaborate with AI systems as intuitive partners, bridging the gap between machine efficiency and human intuition.

Methodology

The MMA framework operationalizes cognitive science principles through a three-stage pipeline designed to transform opaque reinforcement learning policies into interpretable BTs that resonate with human reasoning. At its core, MMA integrates a mental model encoder to distill hierarchical subgoals from trajectory data, a cognitive pruning algorithm to simplify the tree while preserving decision-critical nodes, and a mental effort metric to evaluate the interpretability of the resulting policies. This process ensures that the generated BTs not only replicate the performance of the original RL agent but also align with the structural and temporal logic of human mental models.

Mental Model Encoder

By analyzing the specific criteria and conditions that govern the transition from Plan A to Plan B, we can better understand the underlying mechanisms driving the observed phenomenon. The first stage, mental model encoding, begins by analyzing trajectories generated by the RL agent to identify latent subgoals—key decision points that segment the task into cognitively salient phases. The most important part of MMA is to discover sub-optimal goals x_A from the data set D. Drawing inspiration from hierarchical task networks in AI and the cognitive theory of goal decomposition, the encoder employs a reconstruction-based approach to cluster trajectory segments into meaningful units. For instance, in an autonomous driving task, trajectories might be partitioned into subgoals like "lane merging," "speed adjustment," and "collision avoidance," each corresponding to a reusable BT subtree. The encoder achieves this by training a weak autoencoder to compress trajectory data into a latent space, where reconstruction errors highlight transitions between subgoals. These transitions are then mapped to BT nodes, recursively decomposing the task until atomic actions (e.g., "steer left," "accelerate") are reached. This hierarchical decomposition mirrors how humans chunk complex tasks into manageable steps, ensuring the resulting tree reflects natural problem-solving strategies. Figure.2 illustrates how to discover it.

Even the goal is implicit or not directly observed, it can still be encoded in a latent space, as demonstrated by representation learning techniques (Bengio, Courville, & Vincent, 2013). The paper's methodology could be extended to handle such cases by using an encoder to project the input trajectory into a latent space: $C_v = Encoder(\tau_v)$. $C_v \in \mathbb{R}^{L\times c}$ refers to the encoder output feature corresponding to $\tau_v \in D$, where c denotes a latent dimension. In contrast, we adopt a weak decoder consisting of two fully-connected layers: $\tau'_v = 2 - FC(C_v)$. It is essential that the decoder should not be overly powerful, as this can lead to a situation where its performance is independent of the encoder's ability to encode input. In the extreme case, a strong decoder comprised of multiple deep layers can generate the input data accurately even from random noise that contains no information about input data. Then we minimize reconstruction loss while training. The reconstruction loss $\mathcal{L}$ is defined as $L2$ loss between τ_v and τ'_v:

$$\mathcal{L}_v = \frac{1}{N}\sum_{v=1}^{N} ||\tau_v - \tau'_v||_2^2. \tag{1}$$

Masked Autoencoder masks parts of the input images (e.g., patches of pixels) and trains the model to reconstruct the masked areas. This approach helps in learning robust feature representations without the need for labeled data. In contrast, we seek factors that destabilize the reconstructed sequence. We use masks M in testing rather than in training. In order to test the impact of a single factor x_t on the output, we use the mask $M_t = \{m_1, \ldots, 0, \ldots, m_L\}$, whose corresponding position m_t will be set to 0. For $\tau_v^i = \tau_v \cup M_i$, we can get a loss $\mathcal{L}_v^i$, where $i = [1, \ldots, L]$. We can then construct an impact matrix I. Its rows represent policy-transition tuples x and its columns represent trajectories τ. Its value represents the impact of the x_i^v on the τ_v, which is the reconstruction loss $I(x_i^v, \tau_v) = \mathcal{L}_v^i$. The larger the value, the greater the impact of policy-transition tuples on the trajectory. If the value is 0, it means that this trajectory does not contain this policy-transition tuple.

We assume that $D' = \{\tau_i\}_{i=1}^{U}$ is a sub-dataset of D, where U denotes the total number of trajectories in it. We can measure the impact of a certain policy-transition tuple x_i on D' and

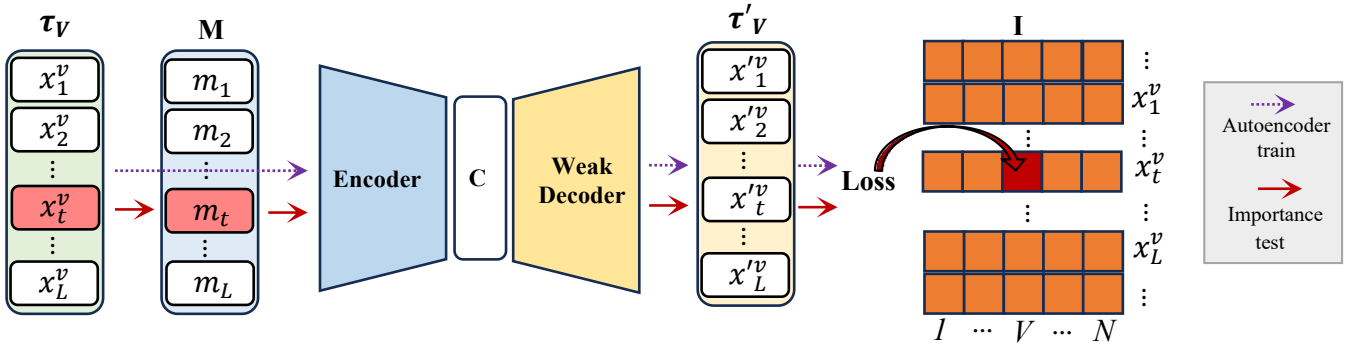

Figure 2: Illustration of proposed MMA. Purple arrow represents the process of training. Red arrow depicts the process of test. 'Weak Decoder' represents two fully-connected layers.

find the sub-optimal goal through this formula:

$$argmax_{x_i} \quad \frac{\sum_j^U I(x_i,\tau_j)}{\sum_j^U Q(j)} \tag{2}$$

$$s.t. \quad \tau_j \in D', \quad Q(j) = \begin{cases} 1 & if I(x_i,\tau_j) > 0, \\ 0 & otherwise. \end{cases} \tag{3}$$

Cognitive Pruning

To prevent cognitive overload, the second stage applies cognitive pruning, a heuristic algorithm inspired by the chunking principle of working memory. By removing unnecessary branches, pruning improves generalization, simplifies the model structure, enhances interpretability, and reduces the risk of overfitting. The algorithm iteratively removes nodes that contribute minimally to policy fidelity or introduce redundant logic. For example, redundant condition checks (e.g., repeated "obstacle detection" nodes in adjacent subtrees) are merged, while branches leading to low-reward outcomes are trimmed. The pruning process is guided by a dual objective: maximizing policy coverage (the BT's ability to replicate the original RL agent's decisions) and minimizing cognitive load (measured by tree depth and node count).

The tree is first grown to its full size BT and then pruned back BT'. Nodes are removed if they do not provide significant power to make decision. This method typically involves evaluations E to determine which branches to prune. First, we set a balance metric between scale and evaluation. For example, we stop pruning when BT's evaluation drops to λ: $E(BT') - \lambda E(BT) < 0$. We adopt a breadth-first approach, i.e. pruning layer by layer starting from the root node. Because the subtrees of high-level nodes are dense, pruning them can quickly help reduce the size without destroying the balance. Removing the subtrees of higher-level nodes can significantly reduce the overall size of the tree , as these nodes tend to have dense subtrees. This approach can quickly reduce the complexity of the model.

Crucially, the algorithm preserves nodes that align with human mental schemas, such as contingency plans ("Plan B" branches) or critical subgoals identified during encoding. Within the same layer, our pruning order is from left to right. Because plan B is always on the right side of the node, its deletion priority must be higher than the original plan. This balance ensures the BT remains both interpretable and effective, adhering to the cognitive limit of approximately four hierarchical levels suggested by Cowan's (2001) working memory model.

Evaluation

We design an evaluation to measure the relationship between BT policy π_{BT} and RL policy π. In the offline case, we use the sampled dataset D to approximately represent the RL policy. If the sampling distribution is reasonable, the closeness between BT and the data set is approximately equal to the relationship between strategies. We define the distance between a policy and a trajectory as an aggregation of the distance between a policy and each transition tuple in the trajectory. Concretely, the policy trajectory distance can be expressed as:

$$d(\pi_{BT},\tau_i) = LSE(d(\pi_{BT},x_1^{(i)}),\ldots,d(\pi_{BT},x_L^{(i)})),$$

where $d(\pi_{BT},x)$ denotes the policy-transition tuple distance and LSE denotes the log-sum-exp function. We use the average l_2 distance between the BT policy action and the trajectory action:

$$d(\pi_{BT},x) = \mathbb{E}_{a'\sim\pi_{BT}(\cdot|s)}[||a'-a||_2].$$

We hope that π_{BT} can be closer to the trajectory with the higher return. The simplest choice would be to take the weighted mean of the policy-transition tuple distances. However, returns that are too high or too low can bring abnormal weights, causing the strategy to deviate from most trajectories. We sort the trajectories in descending order of their returns: $i < j \Rightarrow R(\tau_i) > R(\tau_j)$. We weight $d(\pi_{BT},\tau)$ by exponential scoring method:

$$E(\pi_{BT},D) = \sum_{i=1}^{N} \delta^{(i-1)} e^{-d(\pi_{BT},\tau_i)},$$

where $\delta < 1$ denotes the percentage by which the score decreases for each subsequent rank.

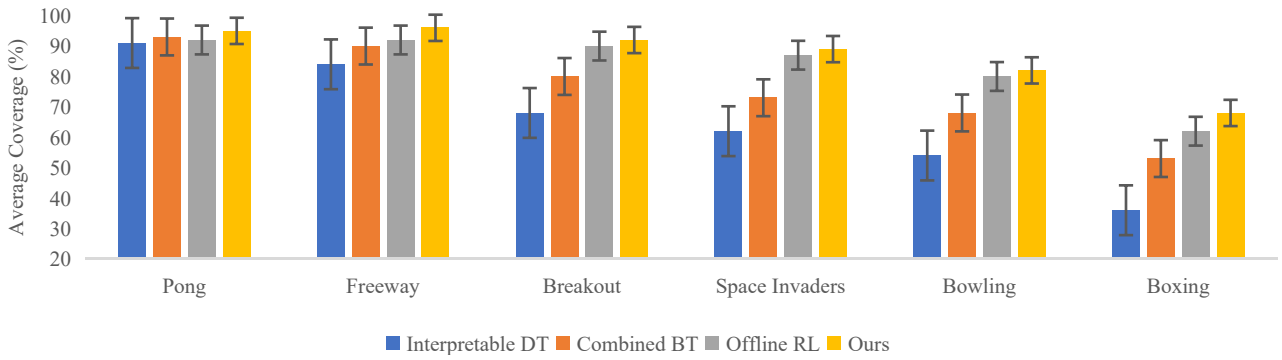

Figure 3: Average coverage over 20 trajectories, with standard error. Error bars correspond to the 95% confidence interval.

Table 1: The average rewards comparison of four different XRL algorithms in 6 environments.

	Pong	Freeway	Breakout	Space Invaders	Bowling	Boxing
DQN	17.9	29.1	386.2	1911	41.2	74.8
Interpretable DT	17.1	27.9	373.1	1801	33.9	47.1
Offline RL	18.7	29.7	381.3	1874	40.4	**70.9**
Combined BT	17.6	28.4	377.5	1825	37.3	62.9
Ours	**18.9**	**30.0**	**381.5**	**1887**	**40.7**	**70.9**

Experiment

Our goal in this section is to demonstrate that a BT trained using MMA can deduce new strategies to improve human understanding, by smartly recombining behaviors seen in the RL data. Several metrics and methods have been proposed to assess the interpretability of RL models, each with its own strengths and focuses: **Coverage** (how well the explanation model approximates the RL model), **Stability** (performance of explanations under slight variations) and **Cost-effectiveness** (The ratio of what humans gain to what they pay in the process of understanding the model).

Setup

We conduct our experiments on Atari games, a standard benchmark for RL (Brockman et al., 2016). The games included in this experiment are purposefully chosen to exhibit a gradual increase in complexity. The selected games consist of Pong, which involves 3 possible actions, Freeway with 3 actions, Breakout with 4 actions, Space Invaders with 4 actions, Bowling with 6 actions, and Boxing with 16 actions. All the atari agents have the same recurrent architecture. The input observation is an image frame, preprocessed by gray-scaling, 2x down-sampling, cropping to an 80×80 square and normalizing the values to [0, 1]. The policies used to guide the BT training are generated by the Deep Q-Network(DQN) model (Mnih et al., 2015). The DQN method is a prominent algorithm in the field of reinforcement learning RL that combines deep neural networks with Q-learning. In every environment, we collect trajectories of number $N = 100$. Episodes terminate when the maximum number of timesteps $L = 20$ is reached. The size of a policy-transition tuple varies depending on the environment. We split the training data into 80% for training and 20% for validation. We compare to three explainable baselines that also do not assume access to online interactions: (1) Offline RL. (Hong, Levine, & Dragan, 2023) achieved success on learning from a dataset of human-human interactions by offline RL. We replace human data with data collected through DRL. (2) Interpretable DT. (Stephanie et al., 2024) learn DT policies by model compression and imitation learning. Their approach also works in multi-agent set. (3) Combined BT. (Wan et al., 2024) obtains the BT policy by combinatorial learning, whcih is the SOTA algorithm.

Results and analysis

Experiment I: Explanation Coverage If there is a perfect policy, it can fit every trajectory, $d(\pi^*, \tau) = 0$. Coverage is the ratio of the evaluations of explainable policy π and perfect policy: $Coverage(\pi, D) = E(\pi, D)/E(\pi^*, D) = (\sum_{i=1}^{N} \delta^{(i-1)} e^{-d(\pi_{BT}, \tau_i)})/\sum_{i=1}^{N} \delta^{(i-1)}$. We report the average coverage over 20 trajectories in Figure.3. Our method outperforms baselines in all environments. Notably, in the game "Pong," all methods perform similarly, suggesting that the correct policy may be simple enough for less sophisticated algorithms to capture. The most significant performance differences emerge as task difficulty increases. Other tree-based algorithms exhibit a clear downward trend in performance, particularly the Interpretable DT. In contrast, our approach shows a slower decline and maintains its performance advantage. These results show that our method generate reasonable BT policies close to source RL policy.

Table 2: Statistical results of Cost-Effectiveness. Values in the table are the average of 10 participants in all scenarios. Understanding Score is calculated by the percentage of correct answers from the single-choice questions. Mental Effort Score is aggregated from the mental effort questionnaire. Cost-Effectiveness Ratio is calculated as follows: Understanding Score / Mental Effort Score

	Understanding Score (%)	Mental Effort Score (lower is better)	Cost-Effectiveness Ratio
Interpretable DT	70	50	1.40
Offline RL	65	60	1.08
Combined BT	75	45	1.67
ours	**85**	**40**	**2.13**

Experiment II: Explanation Stability The explanation models are run directly and interact with environment . Evaluation of all algorithms is performed based on the average episodic reward computed 1,000 episodes, each with different random environment seeds. Table.1 represents the average rewards of 4 explanation models and the source DQN model in all environment. Our method demonstrates superior performance across various environments, particularly as the task difficulty increases. While simpler environments like Pong show negligible differences between methods, more complex environments highlight the robustness and efficiency of our approach. Notably, even when compared to the neural network-based method (Offline RL), which boasts strong representational capabilities, our method performs comparably except in the most challenging scenario, "*Boxing*". Our method consistently outperforms tree-based methods (Interpretable DT and Combined BT) in all environments. This evaluation highlights the robustness and efficiency of our method across varying levels of task complexity, demonstrating its superiority in more demanding environments while maintaining competitive performance in simpler tasks.

Experiment III: Cost-effectiveness To explore the cost-effectiveness of different model interpretability methods by assessing the ratio of human understanding to the mental effort required in comprehending model decisions. We provide participants with a brief overview of the model and the context of each decision. After reviewing each explanation, participants will complete an understanding assessment (Single-choice questions to evaluate factual understanding). After completing the understanding assessment, participants will fill out the mental effort questionnaire, NASA-TLX (Larraga García, Ruiz Bejerano, Gutiérrez Martin, et al., 2023), to get a comprehensive measure of cognitive load. As shown in table.2, the experimental results support our hypothesis that our method has the highest cost-effectiveness ratio. This suggests that our method strikes the best balance between providing understandable explanations and requiring minimal mental effort. Combined BT and Interpretable DT follow, with Offline RL being the least cost-effective due to its higher complexity and cognitive demands. These findings are significant as they highlight the importance of developing interpretability methods that are not only accurate but also user-friendly, enabling better human understanding with less cognitive strain. This balance is crucial for the practical application of machine learning models, especially in fields where interpretability and user interaction are critical.

Multi Agent Task

To better illustrate the explanation we get in multi agent tasks, we introduce tasks with stronger decision-making logic, SMAC (Samvelyan et al., 2019). In these scenarios, both sides deploy multiple unit types. In such scenario, a winning strategy that we generated entails the deployment of focused fire, which involves ordering units to tactically coordinate their attacks, systematically eliminating enemy units in a sequential manner. It is crucial to emphasize the avoidance of overkill, wherein units expend excessive force, surpassing the necessary damage required for eliminating their targets. When we judge that the number of surviving allies is below a certain value, we will turn to Plan B. The plan B is strategic coordination of unit positioning to enable multi-directional attacks. It's important leveraging the terrain to regain tactical advantages can prove instrumental in repelling the enemy. If the firepower is below a certain value and is not enough to turn defeat into victory, we need to fight for a draw by devise new strategies that safeguard surviving allies from enemy attacks. The last strategy at this time is that the remaining allies are scattered across the map and maintain a safe distance from each other to minimize or negate any potential damage. This method ensures the dispersion of enemy damage, minimizing its impact on individual units.

Conclusion

MMA advances the vision of human-centered AI, where systems not only perform tasks but also communicate their reasoning in ways that resonate with human intuition. This is critical for applications like medical diagnosis, where clinicians must validate AI recommendations, or autonomous vehicles, where passengers need transparency to trust emergency maneuvers. By bridging the cognitive gap between machines and humans, MMA paves the way for safer, more collaborative AI ecosystems.

References

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bodria, F., Giannotti, F., Guidotti, R., Naretto, F., Pedreschi, D., & Rinzivillo, S. (2023). Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 1–60.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Dai, W. (2024). Safety evaluation of traffic system with historical data based on markov process and deep-reinforcement learning. *Journal of Computational Methods in Engineering Applications*, 1–14.

Han, G., Choi, J., Lee, H., & Kim, J. (2023). Reinforcement learning-based black-box model inversion attacks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 20504–20513).

Hong, J., Levine, S., & Dragan, A. (2023). Learning to influence human behavior with offline reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 36094–36105). Curran Associates, Inc.

Iovino, M., Förster, J., Falco, P., Chung, J. J., Siegwart, R., & Smith, C. (2023). On the programming effort required to generate behavior trees and finite state machines for robotic applications. In *2023 ieee international conference on robotics and automation (icra)* (pp. 5807–5813).

Iyer, R., Li, Y., Li, H., Lewis, M., Sundar, R., & Sycara, K. (2018). Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 aaai/acm conference on ai, ethics, and society* (pp. 144–150).

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness* (No. 6). Harvard University Press.

Khatri, P. (2023). The gaming experience with ai. In *Research anthology on game design, development, usage, and social impact* (pp. 14–30). IGI Global.

Larraga García, B., Ruiz Bejerano, V., Gutiérrez Martin, Á., et al. (2023). Measuring cognitive load in a simulation game. *Congreso Anual de la Sociedad Española de Ingeniería Biomédica.*.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Prystawski, B., Li, M., & Goodman, N. (2024). Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, *36*.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., ... Whiteson, S. (2019). The StarCraft Multi-Agent Challenge. *CoRR*, *abs/1902.04043*.

Seo, J., Kim, S., Jalalvand, A., Conlin, R., Rothstein, A., Abbate, J., ... Kolemen, E. (2024). Avoiding fusion plasma tearing instability with deep reinforcement learning. *Nature*, *626*(8000), 746–751.

Sreedharan, S., Soni, U., Verma, M., Srivastava, S., & Kambhampati, S. (2021). Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. In *International conference on learning representations.*

Stephanie, M., Zhicheng, Z., Nicholay, T., Zheyuan, S., Ryan, Charles, K., Evangelos, E. P., & Fei, F. (2024). Interpretable multi-agent reinforcement learning with decision-tree policies. In D. W. A. Silvia Tulli (Ed.), *Explainable agency in artificial intelligence* (chap. 5). Boca Raton: CRC Press.

Tasse, G. N., Jarvis, D., James, S., & Rosman, B. (2023). Skill machines: Temporal logic skill composition in reinforcement learning. In *The twelfth international conference on learning representations.*

Todorov, E., & Li, W. (2005). A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, american control conference, 2005.* (pp. 300–306).

Wan, K., Liu, Y., Liu, H., & Xu, X. (2024). Unraveling explainable reinforcement learning using behavior tree structures. In *Icassp 2024 - 2024 ieee international conference on acoustics, speech and signal processing (icassp)* (p. 6465-6469). doi: 10.1109/ICASSP48485.2024.10446357

Yang, S., Mao, X., Lu, Y., & Xu, Y. (2022). Towards a behavior tree-based robotic software architecture with adjoint observation schemes for robotic software development. *Automated Software Engineering*, *29*(1), 31.

Zhou, G., Tian, W., Buyya, R., Xue, R., & Song, L. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artificial Intelligence Review*, *57*(5), 124.