Self-Interpretable Reinforcement Learning via Rule Ensembles

Yue Yang*
Monash University
Melbourne, Australia
yue.yang1@monash.edu

Fan Yang*
Monash University
Melbourne, Australia
fan.yang1@monash.edu

Yu Bai
Monash University
Melbourne, Australia
ybai0041@student.monash.edu

Hao Wang†
Monash University
Melbourne, Australia
hao.wang2@monash.edu

ABSTRACT

Current reinforcement learning (RL) models, often functioning as complex 'black boxes,' obscure decision-making processes. This lack of transparency limits its applicability in critical real-world applications where clear reasoning behind algorithmic choices is crucial. To tackle this issue, we suggest moving from neural network or tabular approaches to a rule ensemble model, which improves decision-making clarity and adapts dynamically to environmental interactions. Instead, our method constructs additive rule ensembles to approximate the Q-value in reinforcement learning using orthogonal gradient boosting (OGB) combined with a post-processing rule replacement technique. This method enables the model to provide inherent explanations through the use of rules. Our study sets a theoretical foundation for rule ensembles within the reinforcement learning framework, emphasizing their capacity to boost interpretability and facilitate the analysis of rule impacts. Experimental results from seven classic environments demonstrate that our proposed rule ensembles match or exceed the performance of representative RL models such as DQN, A2C, and PPO, while also providing self-interpretability and transparency.

KEYWORDS

Interpretable Reinforcement Learning, Rule based Model

ACM Reference Format:
Yue Yang*, Fan Yang*, Yu Bai, and Hao Wang†. 2025. Self-Interpretable Reinforcement Learning via Rule Ensembles. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

In the field of reinforcement learning (RL), algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), and Advantage Actor Critic (A2C) have driven significant progress in handling complex sequential decision-making tasks [13, 16, 17, 22]. However, these models often lack interpretability, as their complex inner workings are not transparent, making it difficult to trace and understand how decisions are made [11, 19]. This lack of transparency can lead to significant concerns in critical real-world applications, such as in healthcare, finance, and autonomous driving sectors, where understanding algorithms' decision-making processes are crucial. The inability to provide clear explanations hinders the use of advanced RL techniques in scenarios demanding high levels of transparency and accountability [12, 28, 29]. In tabular-based methods, Q-values represent expected rewards for specific state-action pairs [27]. But it is challenging to discern why a particular action was selected, as the Q table does not provide direct insight into the rationale behind these Q-values. Deep neural networks are infamously known for their lack of interpretability, despite efforts to improve it, making these models prone to mistrust and often unsuitable for improving human understanding of the domain being modeled [12, 25]. Similarly, when a Q-value approximation is generated from a black box model, such as a deep neural network, the challenge of interpreting decisions remains, as illustrated in Figure 1. Imagine you are an agent deciding whether to move up or down to escape, where a wrong choice could mean life or death. A typical black-box model would offer a cold, numerical Q-value or approximation, without any explanation for why that value is assigned to that action, leaving you without direct insight into the reasoning behind it. In contrast, a rule-based model would provide decisions along with a clear explanation. For each action, the model details the rules that contributed to the Q-value, explaining why it

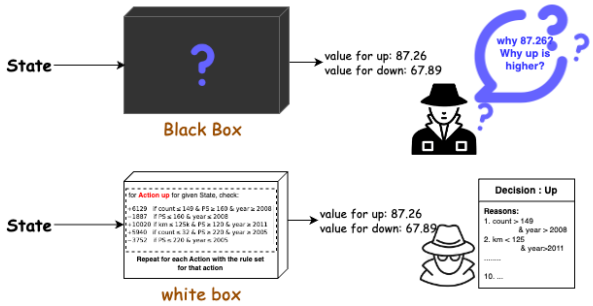

Figure 1: Black Box Decision Making and White Box Decision Making.

*Equal contribution; †Corresponding author.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

suggests moving up or down based on the current state. This transparency approach allows you to understand the reasoning behind the recommendation, enabling you to make a more informed and confident decision.

To address these challenges, our work proposes a fundamental transition from traditional table or neural network-based approaches to a rule ensemble model for reinforcement learning. Rule-based systems, appreciated for their simplicity and clarity, use a sequence of straightforward if-then rules that are easy to interpret and validate [2, 6, 18, 23, 26]. This transparency is especially valuable in scenarios where understanding the reasoning behind decisions is just as essential as the outcomes themselves. Our approach aims to blend the self-interpretable structure of rule-based models with reinforcement learning, focusing on learning and refining rules instead of a Q-table or neural network. This method bridges the gap between sequential decision-making and user transparency, providing clear and interpretable decisions.

We propose reinforcement additive rule ensembles that create a rule set for each action to approximate the Q-value by aggregating the outputs of multiple simpler models—each tied to a specific decision rule—into a cohesive predictive framework. In the ensemble, each rule contributes a weighted value for every state-action pair, collectively forming the overall decision metric based on rule fulfillment. In this paper, we begin by detailing the theoretical foundation of additive rule ensembles in the context of reinforcement learning. By deriving the score from the ensemble of decision rules rather than from a neural network or table, we retain the framework's ability to perform expected reward proximation for each state while significantly improving the self-interpretability of the output. Our method allows us to examine the individual contributions of each rule, providing insights into what factors are considered most critical by the model across different states-action pairs. Through extensive experiments in simulated classic RL environments, we demonstrate that rule ensembles can match or even surpass the performance of benchmark RL models, including Q-tabular, DQN, A2C and PPO, while providing the added value of self-interpretability.

2 RELATED WORK

In the field of RL, **post hoc interpretability methods** are widely used to understand the decision-making behavior of agents. These methods aim to provide explanations for model behavior by analyzing trained RL models. Common post hoc techniques include generating attention maps or heatmaps to show what the agent focuses on in a specific state, as well as local sensitivity analysis based on input-output relationships (e.g., LIME, SHAP4RL), to explain Q-value estimation or policy output. These techniques help understand why an action is chosen in a specific state to some extent [11, 14, 19].

However, post hoc methods in RL have significant limitations. First, since these explanations are generated after model training, they are detached from the actual decision-making process, lacking causality, and cannot clearly indicate the agent's internal decision logic. This separation makes the explanations difficult to verify and trust, especially in high-risk applications like healthcare or autonomous driving [1, 3, 12]. Second, these methods often rely on visualizing complex high-dimensional representations, making the results unstable and subjective. Different tools or observers may derive different interpretations, affecting consistency and generalizability [5, 29]. Additionally, post hoc methods lack real-time adaptability, making it difficult to provide reliable and consistent explanations when dealing with new environments or complex tasks [1, 15].

3 PRELIMINARY

3.1 MDP and Q-Learning

A Markov decision process (MDP) [27] is encapsulated by the tuple (S, A, P, R, γ), where:

- S denotes the complete set of states, potentially either countable or uncountable,
- A represents the complete set of actions available,
- $P : S \times A \to P(S)$ is the transition kernel that governs the dynamics of the state transitions,
- $R : S \times A \to P(\mathbb{R})$ specifies the distribution over immediate rewards received after actions,
- γ, a factor within $(0, 1)$, quantifies the discounting of future rewards.

In this setting, choosing any action $a \in A$ while in any state $s \in S$ leads to the next state according to the transition probability $P(\cdot|s, a)$ and accrues an immediate reward as dictated by $R(\cdot|s, a)$. Additionally, for purposes of mathematical convenience and to ensure model regularity, it is assumed that S is a compact subset of $\mathbb{R}^r$, possibly extending infinitely, and A comprises a finite number of actions M, specifically $A = \{a_1, a_2, \ldots, a_M\}$. The rewards are constrained within the bounds $[-R_{\max}, R_{\max}]$, ensuring that all rewards $R(\cdot|s, a)$ remain uniformly bounded across all states s and actions a.

A policy $\pi : S \to P(A)$ assigns to each state $s \in S$ a probability distribution over actions, denoted by $\pi(\cdot|s)$. The value function associated with a policy π, denoted by $V^\pi : S \to \mathbb{R}$, is defined as the total expected discounted reward accrued by executing actions in accordance with policy π, starting from a given initial state. Specifically,

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right]. \tag{1}$$

While decision makers have the authority to define the policy π, the transition kernel P and the reward function R are inherent characteristics of the environment and are typically not known to the decision makers.

Applying the law of iterative expectation, the value function under any policy π is represented as:

$$V^\pi(s) = \mathbb{E}\left[Q^\pi(s, a) \mid a \sim \pi(\cdot|s)\right], \quad \forall s \in S, \tag{2}$$

where $Q^\pi(s, a)$, the action value function, calculates the expected total discounted rewards starting from state s and taking action a, as follows:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s0 = s, a0 = a\right] \tag{3}$$

$$= r(s, a) + \gamma \mathbb{E}\left[V^\pi(s') \mid S' \sim P(\cdot|s, a)\right], \tag{4}$$

with $r(s,a)$ being the expected immediate reward for choosing action a in state s. We first define an operator P^π by

$$(P^\pi Q)(s,a) = \mathbb{E}\left[Q(s',a') \mid s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')\right], \quad (5)$$

and define the Bellman operator T^π by

$$(T^\pi Q)(s,a) = r(s,a) + \gamma \cdot (P^\pi Q)(s,a). \quad (6)$$

The primary objective of reinforcement learning is to discover the optimal policy that maximizes the cumulative reward through dynamic learning from the acquired data. To define optimality, we refer to the optimal action-value function Q^* as follows:

$$Q^*(s,a) = \sup_\pi Q^\pi(s,a), \quad \forall (s,a) \in S \times A, \quad (7)$$

where the supremum is taken over all possible policies. Moreover, for any action-value function $Q : S \times A \to \mathbb{R}$, we define the greedy policy π_Q such that for any state $s \in S$, $\pi_Q(\cdot|s)$ satisfies:

$$\pi_Q(a|s) = \begin{cases} 1 & \text{if } Q(s,a) = \max_{a' \in A} Q(s,a'), \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Given Q^*, the optimal policy π^* is any policy that is greedy with respect to Q^*. It has been established that $Q^* = Q^{\pi^*}$.

3.2 Orthogonal Gradient Boosting for Additive Rule Ensembles

3.2.1 Additive Rule Ensembles. **Additive rule ensembles** are a type of probabilistic models [10] that predict the mean of a target variable $Y \in \mathbb{R}$, given an input variable $X \in \mathbb{R}^d$ as $\mathbb{E}[Y \mid X = x] = \mu(f(x))$, where $\mu : \mathbb{R} \to \mathbb{R}$ is an inverse link function mapping the output of $f(x)$ to the target variable Y, and $f : \mathbb{R}^d \to \mathbb{R}$ is an affine linear combination of k boolean query functions:

$$f(x) = \sum_{i=1}^{k} \beta_i \phi_i(x). \quad (9)$$

In (9), β_i ($0 \le i \le k$) is the weight of the i-th boolean query function $\phi_i : \mathbb{R}^d \to \{0,1\}$, and ϕ_i is a conjunction or product of c_i boolean propositions:

$$\phi_i(x) = \prod_{j=1}^{c_i} p_{i,j}(x), \quad (10)$$

where $p_{i,j} \in \left\{\mathbf{1}\left(sx^{(j)} \le x_l^{(j)}\right) : j \in [d], l \in [n], s = \pm 1\right\}$ is a threshold function (we denote $[x] = \{1, \ldots, x\}$).

In (9), each term can be represented as a rule in the form of 'IF…THEN …', where the query ϕ_i defines the condition of the rule, and the weight β_i is the rule consequent.

3.2.2 Orthogonal Gradient Boosting. Boosting is an iterative approach for learning additive models [7, 9, 21] based on minimizing the *empirical risk*

$$R_\lambda(f) = \sum_{i=1}^{n} l(f(\boldsymbol{x}_i), y_i)/n + \lambda \|\beta\|^2/n,$$

given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where l is a loss function can be derived as deviance function, i.e., shifted negative log likelihood, and measures the cost of predicting $f(\boldsymbol{x}_i)$ while the true value is y_i, yielding a sequence of models $f^{(0)}, f^{(1)}, \ldots, f^{(k)}$.

When using boosting to learn additive rule ensembles, in the t-th iteration, a query is selected from a set of queries $\boldsymbol{Q}$ to maximize some objective functions:

$$q_t = \arg\max\{\text{obj}(\phi; f^{(t-1)}) : \phi \in \boldsymbol{Q}\},$$

and then added into the rule ensembles. Commonly used boosting objective functions include the gradient boosting objective (GB) [9], gradient sum objective (GS) [7], extreme gradient boosting objective (XGB) [4], etc. Traditionally, boosting only calculates the weight of the newly-added query: $\beta^{(t)} = \arg\min_\beta \hat{R}_\lambda(f^{(t-1)} + \beta\phi_t)$. The risk of the model can be further reduced by recalculating the weights of all queries, which is the *corrective weight update method*:

$$\boldsymbol{\beta}_t = \arg\min\{R_\lambda(\boldsymbol{\Phi}_t \boldsymbol{\beta}) : \boldsymbol{\beta} = (\beta_1, \ldots, \beta_t) \in \mathbb{R}^t\},$$

where $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t]$ is the $n \times t$ query matrix with the output vectors of all selected queries as columns. However, the previous boosting objective functions do not fully utilize the advantage brought by corrective weight update since they do not consider the weight correction step during the selection of the queries. Yang et al. [30] proposed a novel boosting objective function correctly identifying the query, such that the subspace spanned by all selected queries is closest to the ideal gradient descent corrective weight update subspace. More specifically, they derive the Orthogonal Gradient Boosting objective function $\text{obj}_{\text{ogb}} = |\boldsymbol{\phi}^T \boldsymbol{g}_\perp| / \|\boldsymbol{\phi}_\perp\|$, where $\boldsymbol{g}_\perp$ and $\boldsymbol{\phi}_\perp$ are the projection of $\boldsymbol{g}$ and $\boldsymbol{\phi}$ onto the orthogonal complement of the range of $\boldsymbol{\Phi}_{t-1}$.

4 METHODOLOGY

4.1 Rule Based Model for Q-Learning

In an MDP framework adapted with additive rule ensembles for Q-learning, each action $a \in A$ is associated with a specific ensemble model $f_a(s)$ for state $s \in S$, predicting the Q-function as

$$Q(s,a) = \mu(f_a(s)),$$

where μ is the inverse link function.

Each ensemble $f_a(s)$ for an action a predicts the expected return directly as $Q(s,a)$. This prediction is the output of a linear combination of features (defined by boolean query functions) weighted by coefficients specific to each action. Thus, we can simplify the previous function as $Q(s,a) = f_a(s)$. The function $f_a(s)$ is defined as

$$f_a(s) = \sum_{i=1}^{k_a} \beta_{a,i} \phi_{a,i}(s),$$

with $\beta_{a,i}$ as coefficients and $\phi_{a,i}$ as Boolean query functions specific to action a. The policy π is then derived to maximize the expected return by selecting actions that maximize the Q-value, expressed as $\pi(s) = \arg\max_a Q(s,a)$. The learning process involves iterative updates to the coefficient vector $\boldsymbol{\beta}_a$ and the queries $\phi_{a,i}$ based on observed rewards and state transitions, aiming to refine the ensembles for optimal decision-making in complex or high-dimensional state spaces.

4.1.1 Action Selection. In our rule based reinforcement learning algorithm, action selection, as shown in Figure 2,is governed by a

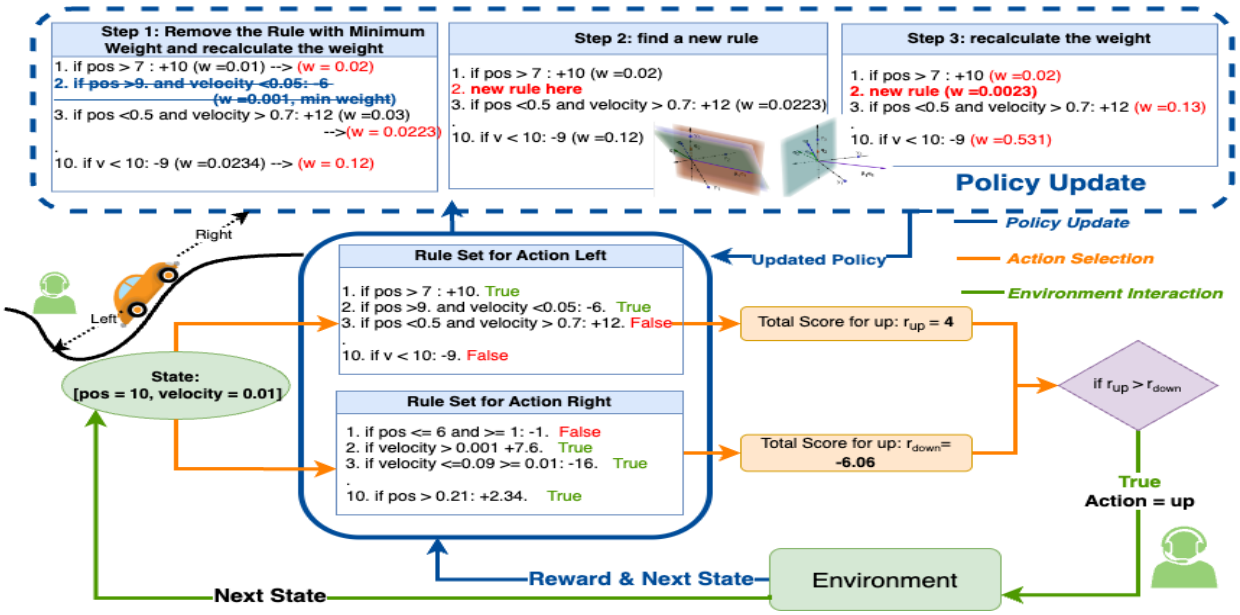

Figure 2: Rule Based Q-Learning.

policy π derived from the Q-values predicted by individual ensemble models for each action. The policy

$$\pi(s) = \arg\max_a Q(s,a) = \arg\max_a f_a(s)$$

is formulated to maximize the expected return. and $f_a(s)$ predicts the Q-value for action a in state s. For each decision step, the system evaluates $f_a(s)$ for all actions, choosing the one with the highest value to optimize immediate expected returns. We also implement the epsilon decay to balance the explore and exploit.

4.1.2 Policy Update. In the our proposed rule based reinforcement learning algorithm, a rule based policy $\pi_{\vec{w},\vec{q}} : S \times A \to \mathbb{R}$ is utilized to approximate the optimal action-value policy π^*, where $\vec{w}, \vec{q}$ denotes the weights and the queries for rule based model. Like all other value based Q learning algorithm, RBQL employs the technique of experience replay. At each timestep t, the transition (S_t, A_t, R_t, S_{t+1}) is stored in the replay memory M, and a minibatch of independent samples is drawn from M to train the additive rule ensemble via orthogonal gradient boosting. This approach helps mitigate the issue of strong temporal correlations in the trajectory of the MDP, aiming to produce uncorrelated samples which enhance the accuracy of the gradient estimation for additive rule ensemble model. We also use a target model $f_{(w,q)^{target}}$ with parameters $\vec{w}^{target}, \vec{q}^{target}$ (current estimate of the parameters). By utilizing independent samples $\{(s_i, a_i, r_i, s_i')\}_{i\in[n]}$ from the replay memory (where s_i' represents the next state following s_i and a_i instead of s_{i+1} to avoid notation collision with another independent sample in the state space), the parameter $\vec{w}^{target}, \vec{q}^{target}$ of the additive rule ensemble is updated by computing the target $Y_i = r_i + \gamma \max_{a\in A} f_{(w',q')^{target}}(s_i', a)$, comparing it with the Bellman optimality operator, and updating $\vec{w}, \vec{q}$ by the gradient of the loss function:

$$L(w,q) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - f_{(w,q)}(s_i,a_i)\right]^2.$$

The parameters $(\vec{w},\vec{q})^{target}$ are updated once every $\mathcal{T}_{\text{target}}$ steps by setting $(w,q)^{target} = (w,q)$, effectively keeping the target network fixed for $\mathcal{T}_{\text{target}}$ steps before updating it to the current weights of the additive rule ensemble. To further elucidate the importance of the target network, let us initially disregard it and set $(w,q)^{target} = (w,q)$. Employing the bias-variance decomposition, the expected value of the loss function $L((w,q))$ is expressed as:

$$\mathbb{E}[L((w,q))] = \left\|f_{(w,q)} - Tf_{(w,q)}\right\|_\sigma^2 + \mathbb{E}\left[\left(Y_1 - (Tf_{(w,q)})(s_1,a_1)\right)^2\right],$$

where $\left\|f_{(w,q)} - Tf_{(w,q)}\right\|_\sigma$ represents the mean-squared Bellman error (MSBE) under the distribution σ, and the second term captures the variance of Y_1, and T is the update operator.

Next, to solve this problem, we incorporate a target model as described before which yields the expectation:

$$\mathbb{E}[L(w,q)] = \left\|f_{(w,q)} - Tf_{(w,q)^{target}}\right\|_\sigma^2 \tag{11}$$

$$+\mathbb{E}\left[\left(Y_1 - (Tf_{(w,q)^{target}})(s,a)\right)^2\right], \tag{12}$$

where the variance of Y_1 is independent of (w,q). Therefore, minimizing $L(w,q)$ closely approximates the solution to the optimization problem:

$$\min_{(w,q)\in(\mathbb{W},\mathbb{Q})}\left\|f_{(w,q)} - Tf_{(w,q)^{target}}\right\|_\sigma^2, \tag{13}$$

where (w,q) is the parameter space.

Let F be the functional space of rule based functions defined on $S \times A$. In the k-th iteration of the algorithm, let $\hat{f}$ be the current estimate of π at time k. We can update our policy by updating f by solving the following minimization problem:

$$f = \arg\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{f}(S_i, A_i) \right)^2 = T\hat{f}, \tag{14}$$

where n is the mini batch size, and T is the update operator.

Detailed Model Updates for Rule Based Model

To update the additive rule ensembles for each action, we adopt the post processing replacement step provided by Shalev-Shwartz et al. [24]. To train and update the model for action a, we use the part of dataset where action a is taken. Initially, we use OGB to train an additive rule ensemble with k rules using the mini batch data for each action. Afterwards, at each iteration where a model update for an action is needed, we take the replacement step. In a rule replacement step, first, we remove a rule according to some criteria, resulting in $k-1$ rules. For instance, we can remove the rule whose weight's absolute value is minimum. After the removal, the weights of the left rules are recalculated. Then, a new query which maximizes the OGB objective function is added into the rule ensemble, and the weight vector is recalculated again. This replacement procedure repeats until the risk value does not decrease anymore or a maximum number of repeating is exceeded.

Algorithm 1 Rule Based Q-Learning

Initialize replay memory D to capacity N
Initialize action-value function $f^{(0)} = 0$
Initialize target action-value function $\hat{f} = 0$
for episode $= 1, M$ **do**
  Initialize sequence s_1
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a f_a(s)$
    Execute action a_t in emulator and observe reward r_t and $done_t$ to check if the episode is terminated
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random minibatch $D^{(t)}$ of transitions $(s_j, a_j, r_j, s_{j+1}, done_j)$ from D
    Set $y_j = r_j + (1 - done_j)\gamma \max_{a'} \hat{f}(\phi_{j+1}, a'; \theta^-)$
    for all action **do**
      if $f^{(t-1)} = 0$ **then** Use OGB to get $f_k^{(t)}$ to fit $D^{(t)}$
      else Update $f_k^{(t-1)}$ to $f_k^{(t)}$ using $D^{(t)}$ by the rule replacement steps
      end if
    end for
    Every C steps reset $\hat{f} = f_k^{(t)}$
  end for
end for

4.2 Theoretical Analysis

Theorem 1. *Let $\pi^*(s) = \arg\max_a Q^*(s,a)$ be the optimal policy that maximizes the Q-function for any state s in a Markov Decision Process. If $f^*(s)$ represents the function mapping state s to the value $Q^*(s, \pi^*(s))$, then $f^*(s) = Tf^*(s)$.*

Algorithm 2 Orthogonal Gradient Boosting

Input: dataset $(\boldsymbol{x}_i, y_i)_{i=1}^n$, number of rules k
$f^{(0)} = 0$
for $t = 1, \ldots, k$ **do**
  $\boldsymbol{g}_t = \left(\partial l(f^{(t-1)}(\boldsymbol{x}_i), y_n) / \partial f^{(t-1)}(\boldsymbol{x}_i) \right)_{i=1}^n$
  $\boldsymbol{\phi}_t = \arg\max_\phi |\boldsymbol{g}_{\perp t}^T \boldsymbol{\phi}| / \|\boldsymbol{\phi}_\perp\|$
  $\boldsymbol{\beta}^{(t)} = \arg\min_{(\beta_1,\ldots,\beta_t)\in\mathbb{R}^t} \hat{R}_\lambda(\sum_{j=1}^t \beta_j \phi_j)$
  $f^{(t)}(\cdot) = \beta_1^{(t)}\phi_1(\cdot) + \cdots + \beta_t^{(t)}\phi_t(\cdot)$
end for
Output: $f^{(k)}$

Algorithm 3 Rule Replacement Steps

Input: dataset $(\boldsymbol{x}_i, y_i)_{i=1}^n$, original rule ensemble with k rules $f_k^{(0)}$, maximum iteration number T
for $t = 1, \ldots, T$ **do**
  Find the index of the smallest weight absolute value $r = \arg\min_{j\in 1,\ldots,k} |\beta_j^{(t-1)}|$
  $\tilde{\boldsymbol{\beta}}^{(t-1)} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{k-1}} \hat{R}_\lambda(\sum_{i\in[k]-\{r\}} \tilde{\beta}_i^{(t-1)} \phi_i)$
  $\tilde{f}_{k-1}^{(t-1)}(\cdot) = \sum_{i\in[k]-\{r\}} \beta_i^{(t-1)} \phi_i(\cdot)$
  $\boldsymbol{g} = \left(\partial l \left(\tilde{f}_{k-1}^{(t-1)}(\boldsymbol{x}_i), y_n \right) / \partial \tilde{f}_{k-1}^{(t-1)}(\boldsymbol{x}_i) \right)_{i=1}^n$
  Find the query $\phi_k^{(t)} = \arg\max_\phi \|\phi^T \boldsymbol{g}_\perp\| / |\phi_\perp|$
  $\boldsymbol{\beta}^{(t)} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^k} \hat{R}_\lambda(\sum_{i\in[k]-\{r\}} \beta_i\phi_i + \beta_r\phi_r^{(t)})$
  Let $\delta = \hat{R}_\lambda(f^{(t)}) - \hat{R}_\lambda(\sum_{i\in[k]-\{r\}} \beta_i^{(t)}\phi_i + \beta_r^{(t)}\phi_r^{(t)})$
  if $\delta > 0$ **then** $f_k^{(t)} = \sum_{i\in[k]-\{r\}} \beta_i^{(t)}\phi_i + \beta_r^{(t)}\phi_r^{(t)}$
  else break
  end if
end for
Output: $f_k^{(t)}$

Proof. Assume $\pi^*(s) = \arg\max_a Q^*(s,a)$ is the optimal policy such that for any state s, $Q^*(s, \pi^*(s)) = \max_a Q^*(s,a)$. Then, by definition, the function $f^*(s)$ which is intended to represent the optimal action value function as:

$$f^*(s) = Q^*(s, \pi^*(s))$$

satisfies the condition:

$$f^*(s) = \max_a Q^*(s,a).$$

The update mechanism for f ensures that adjustments are made only when they result in an improvement. This is achieved by minimizing the mean squared error (MSE) between the observed outcomes and the predicted values, corresponding to minimizing the following loss function:

$$L(w,q) = \frac{1}{n} \sum_{i=1}^{n} \left[Y_i - f_{(w,q)}(s_i, a_i) \right]^2.$$

Since $f^*(s) = \max_a Q^*(s,a)$, it follows that $L(w^*, q^*)$ attains the minimum value possible within the system. Thus, no other pair (w, q) can yield a lower loss, making (w^*, q^*) the optimal parameter set, which implies that $f^*(s) = Tf^*(s)$.

□

Theorem 2. *Let $R(f)$ be the expected squared loss function, where the expectation is with respect to an arbitrary distribution over $X \times Y$. Let $\lambda > 0$ be a scalar, f is an additive rule ensemble with k rules, $\bar{f}$ has k_0 rules, if*

$$k + 1 \geq k_0(1 + 16/\lambda^2), \tag{15}$$

and assume that R is $(k + 1 + k_0, \lambda)$-sparsely-strongly convex. Additionally, let τ be an integer such that

$$\tau \geq \frac{\lambda(k + 1 - k_0)}{2\beta} \log\left(\frac{R(0) - R(\bar{f})}{\epsilon}\right). \tag{16}$$

Then if the fully corrective boosting (Algorithm 2) is run for k iterations and its last predictor is provided as input for the post-processing replacement procedure (Algorithm 3), which is then run for τ iterations, then when the procedure terminates at time t (which may be smaller than τ), we have

$$R(f^{(t)} - R(\bar{f}) \leq \epsilon.$$

Proof. See Theorem 2.9 in [24] for proof. □

Theorem 3. *Assume there exists a reference function $\hat{f}$ which is the optimal solution for Tf_{target}, where $\hat{f} = Tf_{target}$ for all $s \in S$ and $a \in A$. Let f be updated iteratively towards this target function Tf_{target}. We have f converges to $\hat{f}$.*

Proof. We first calculate the risk for rule ensembles for $\hat{f}$ and f_k based on the loss functions:

$$R(\hat{f}) = [\hat{f}(s,a) - Tf_{\text{target}}(s,a)]^2,$$

and

$$R(f_k) = [f_k(s,a) - Tf_{\text{target}}(s,a)]^2.$$

Based on Theorem 2, when (15) and (16) are satisfied, we have

$$\begin{aligned} R(f_k) - R(\hat{f}) &\leq \epsilon, \\ [f_k(s,a) - Tf_{\text{target}}(s,a)]^2 - [\hat{f}(s,a) - Tf_{\text{target}}(s,a)]^2 &\leq \epsilon, \\ (f_k - \hat{f})(-2Tf_{\text{target}}(s,a) + f_k + \hat{f}) &\leq \epsilon, \\ (f_k - \hat{f})(f_k - \hat{f}) &\leq \epsilon, \\ \Rightarrow (f_k - \hat{f}) &\leq \epsilon. \end{aligned}$$

□

5 EXPERIMENTS AND RESULT

We evaluated the performance of our rule-based reinforcement learning model across seven distinct RL environments: MountainCar-v0, Cliffwalk, Blackjack, Taxi [8, 27], and three variations of Postman in grid sizes of 4x4, 5x5 and 7x7. To thoroughly assess the effectiveness of our proposed model, we conducted a comprehensive comparison against several well-known benchmark algorithms, including Q-Tabular methods, DQN, PPO, and A2C with Stable-BaseLine3 [20]. To illustrate the interpretability of our proposed model, we use the Cliffwalk and Mountain Car environments as examples to provide a detailed analysis. These examples demonstrate the explainability and transparency of our method by breaking down the model's actions across scenarios, offering clear insights into the agent's behavior and making the decision-making process more self-interpretable than benchmark reinforcement learning methods.

5.1 Performance

5.1.1 Environments Description. In this paragraph, we provide a brief overview of the settings for each environment.

MountainCar-V0: The agent controls a car in a valley and must build up enough momentum by moving back and forth between two hills to reach the flag at the top of the right hill. The challenge lies in the car's weak engine, which requires strategic movement to gain enough speed.

Cliffwalk: The agent navigates a gridworld along the edge of a cliff. The objective is to reach a goal while avoiding falling off the cliff, which results in a penalty. The challenge lies in balancing risk and reward while moving through.

Blackjack-V1: The agent plays a simplified version of Blackjack, where the goal is to maximize the score without going over 21. The agent must decide whether to 'hit' or 'stick' based on the current hand and visible dealer's card.

Taxi: The agent operates as a taxi driver in a gridworld, tasked with picking up passengers at one location and delivering them to a destination. The agent must plan its route efficiently to minimize the number of moves.

Postman (4x4, 5x5, 7x7): In this custom environment, the agent acts as a postman, tasked with picking up and delivering items at different locations within a grid. The agent must plan routes for picking up and dropping off items, navigating larger grids in the 5x5 and 7x7 environments.

5.1.2 Performance Analysis. Based on the results presented in Table 1, when the number of rules was set to 10, our proposed method demonstrated performance comparable to other established algorithms. The results, averaged over 10 runs with standard deviation, provide reliable and consistent evaluation across all tested environments. In most cases, our algorithm achieved performance comparable to that of benchmark RL methods, including PPO. Notably, in environments like Mountain Car, Blackjack, CliffWalk and Postman, our rule-based model even outperformed benchmark approaches. In scenarios like Taxi, although our model performed comparably to PPO, it did not outperform benchmark methods. This is because the fixed number of rules limits its ability to capture effective strategies in highly complex situations. This highlights the trade-off between interpretability and precision, where the simplicity and transparency of a rule-based model come at the cost of reduced flexibility in highly complex environments.

5.1.3 Ablation Study. In our ablation study, we explored the impact of increasing the number of rules and adjusting the model update frequency on both performance and interpretability. Specifically, we tested the MountainCar-v0 environment with different rule settings, using 5, 10, 20, and 50 rules per action. As shown in Figure 3, the model with 10 rules per action achieved the best reward as the model

Table 1: Performance of RBQL and Benchmarks.

	Mountain-Car	Cliff-Walk	Taxi	Blackjack	Postman4	Postman5	Postman7
Q-tabular	−148.99 ± 9.04	**-13.00** ± 0.00	**10.60** ± 3.26	**0.464** ± 0.05	**34.83** ± 1.58	**35.75** ± 0.11	28.23 ± 4.89
DQN	**-147.51** ± 11.44	**-13.00** ± 0.00	3.27 ± 3.25	**0.464** ± 0.12	22.95 ± 10.78	33.84 ± 1.05	27.26 ± 5.54
A2C	−157.24 ± 37.12	−14.62 ± 0.80	–	0.455 ± 0.10	−90.28 ± 9.70	15.79 ± 21.60	2.87 ± 4.37
PPO	−150.4 ± 40.5	−101.31 ± 46.27	−64.14 ± 0.16	0.455 ± 0.19	24.52 ± 2.14	−58.11 ± 28.70	−91.58 ± 11.01
RBQL(OURS)	**-147.65** ± 9.45	**-13.00** ± 0.00	−21.49 ± 5.01	**0.464** ± 0.06	**34.25** ± 1.02	32.51 ± 0.54	**28.60** ± 1.12

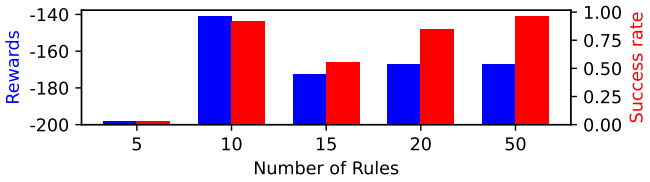

Figure 3: Rewards and success ratios of models with different number of rules for MountainCar-v0.

with 50 rules provided the best win rate. This result implies that with 10 rules per action, the model has enough flexibility to capture the key dynamics of the environment without being overly simplistic. Using fewer than 10 rules (such as 5) may not provide enough coverage to handle the complexity of the environment, leading to suboptimal performance. On the other hand, with too many rules (such as 20 or 50), the model may start to overfit the training data, capturing noise or irrelevant patterns and reducing its ability to generalize to new scenarios. When the number of rules is very high, the model may still experience overfitting; however, it has the potential to learn a diverse policy that could guarantee success with wasted movements. This result from the ablation study indicates a balance between the number of rules and the model performance. It suggests that increasing the number of rules does not always lead to better results, as too many rules can introduce complexity and overfitting, while too few rules may limit the model's ability to generalize across different situations.

5.2 Case Study for Interpretability

Commonly used RL approaches, whether using Q-tabular methods or neural network-based models, offer the values represents the expected payoff for taking an action in a given state. However, these approaches lack transparency in explaining why a particular value is assigned or how the decision-making process is influenced by different aspects of the environment. In contrast, our model not only achieves performance levels on par with these classical approaches but also offers clear self-interpretability by the model. Our framework allows us to understand why certain actions are preferred and how specific rules or features influence the outcome. In this section, we will use Blackjack and Mountain-car from OpenAI Gym as example to show the interpretability of our proposed RL algorithm.

5.2.1 Blackjack-V1. In the OpenAI Gym Blackjack-v1 environment, the player attempts to get as close to a hand value of 21 without exceeding it. The player makes decisions based on their current hand and the dealer's visible card, choosing whether to 'hit' (take

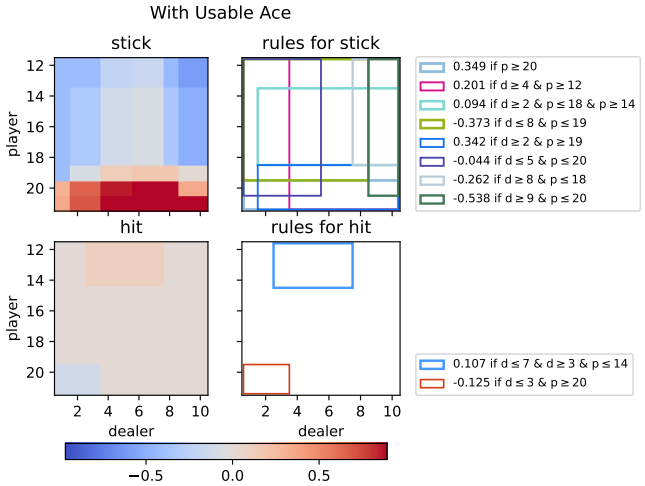

Figure 4: Rule Visualization for Blackjack with Ace.

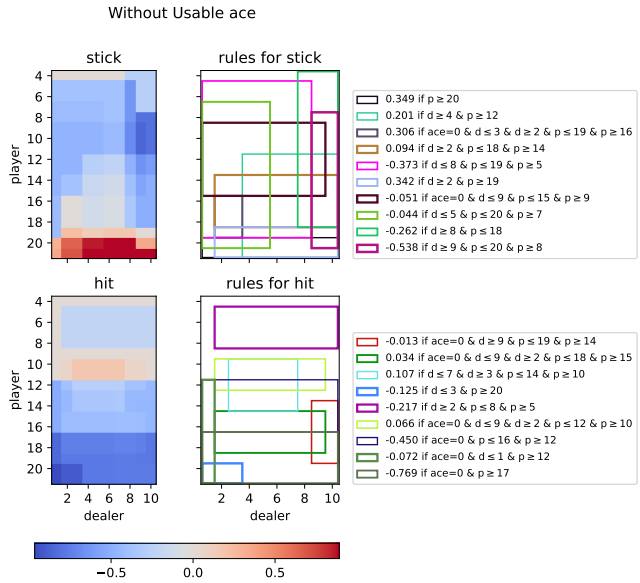

Figure 5: Rule Visualization for Blackjack w/o Ace.

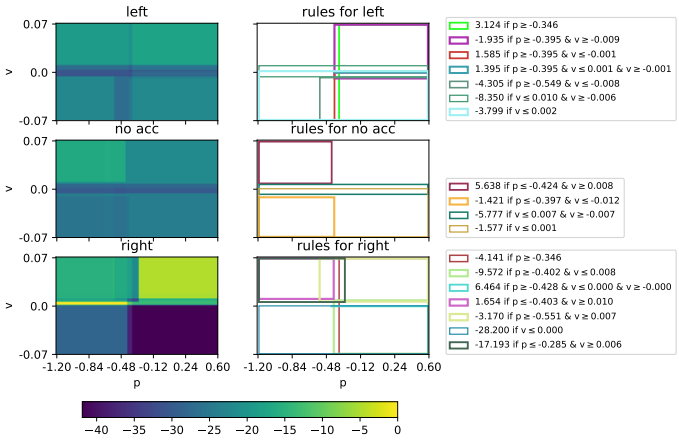

Figure 6: Rule Visualization for Mountain-Car.

another card) or 'stand' (keep their hand). The game incorporates a simple reward structure: +1 for a win, -1 for a loss, and 0 for a tie. A basic strategy for this environment involves maximizing the expected return by learning when to hit or stand based on probabilities. The strategy typically considers factors like the player's current hand value and the dealer's visible card, aiming to balance risk and reward by minimizing the chance of going bust while capitalizing on favorable situations to outplay the dealer.

Rule Examples: ***1***. *Positive Encouragement to Stick (+0.349) if* $p \geq 20$ A hand with a total of 20 (or more with an Ace) is an extremely strong hand, just 1 point away from the best possible total of 21. At this point, the chance of improving the hand by hitting is very low. ***2***. *Negative Incitement to Stick with Ace (-0.538) if* $d \geq 9$ $p \leq 20$*:* The negative value (-0.538) reflects that standing with 20 against a strong dealer card like 9 might lead to a loss or a push. Given the high probability that the dealer can reach a total of 20 or more, the model suggests that taking the risk of hitting could provide a better chance of securing a stronger hand. ***3***. *Positive Encouragement to Hit with Ace (+1.107) if* $d \leq 7$ *and* $d \geq 3$ $p \leq 14$*:* In Blackjack, when the player's hand totals 14 and includes an Ace (which can be counted as either 1 or 11), and the dealer's visible card is a 7 to 3, the player should hit because the hand is considered a soft 14. The key advantage is that if the player hits and draws a higher card that would normally cause the hand to exceed 21, the Ace can switch its value to 1, effectively preventing a bust. ***4***. *Negative Incitement to Hit with Ace (-0.125) if* $d \leq 3$ $p \geq 20$*:* For palyer, a hand value of 20 is already very strong, just 1 point away from the maximum hand value of 21. The chance of improving this hand by hitting is extremely low because almost any card drawn will result in a bust (exceeding 21). ***5***. *Negative Incitement to Stick without Ace (-0.538) if* $d \geq 9$ *and* $p \leq 20$ $p \geq 8$*:* Negative encouragement to avoid standing in this scenario reflects the fact that standing with a hand of 8 to 20, without an Ace, leaves the player in a vulnerable position when the dealer is showing a 9 or higher, as the dealer has a high probability of winning or pushing with a stronger hand. *6. Strong Negative Incitement to Hit without Ace (-0.769) if* $p \geq 17$ *:* The main reason for not hitting on $p \geq 17$ without an Ace is that the risk of busting is far greater than the chances of improving the hand. Standing gives the player a strong enough chance to compete or win, especially if the dealer busts, while hitting increases the likelihood of losing.

5.2.2 Mountain-Car-v0. In the Mountain Car simulation, the car must utilize momentum gained from oscillating back and forth to overcome gravitational forces that inhibit direct ascent. The agent's goal is to navigate the car to the flag located at $p \geq 0.5$ using the least amount of time, hence the reward structure of -1 for each timestep taken until the goal is reached. As shown in Figure 6, we visualize our policy on the state space of the Mountain Car environment (x-axis: position, y-axis: velocity). For any given state, we can easily identify the rules that contribute to the value of each action and obtain explanations for why these rules are applied to the state by interpreting the rules themselves.

Rule Examples: ***1***. *Positive Encouragement to Left (+3.1242) if* $p \geq -0.346$ *and (+1.5851) if* $p \geq -0.395$ *and* $v \leq -0.0005$*:* Supports moving left to subtly adjust the car's positioning or to slightly increase backward momentum, facilitating a strategic setup for the upcoming rightward push. If the car is currently moving to the left, the benefits gets higher. ***2***. *Strong Negative Incitement to Left (-8.3498) if* $v \leq 0.009$ *&* $v \geq -0.006$*:* Significantly penalizes leftward actions when the car has moderate forward velocity. This prevents unnecessary leftward moves that could destabilize the car's approach to the goal or waste energy by moving opposite to the desired direction. ***3***. *Positive Encouragement to No Action (+5.638) if* $p \leq -0.424$ *and* $v \geq 0.008$*:* In the Mountain Car problem, applying no action when the car has high velocity and is on the high left side is a strategic choice to preserve momentum and conserve energy. At this point, gravity will help the car descend, and its existing speed is sufficient to carry it across the valley without additional acceleration. ***4***. *Negative Incitement to No Action (-5.777) if* $v \leq 0.007$ *and* $v \geq -0.007$*:* This scenario discouraging the agent from taking no action when the car's velocity is very close to zero. In this case, the car is not making progress either up or down the hills. If the car remains idle without action, it will fail to gain the necessary momentum to eventually reach the goal. ***5***. *Negative Incitement to Right (-28.2) if* $v \leq 0$*:* In the Mountain Car problem, when the velocity is negative (i.e., the car is moving left), attempting to move right would counteract the natural movement and slow the car down. This is inefficient because the car needs to build up momentum by swinging back and forth between the two hills.

6 CONCLUSION AND LIMITATIONS

In conclusion, we have proposed a novel, self-interpretable reinforcement learning algorithm that leverages rule ensembles to guide decision-making. Our approach not only achieves performance comparable to representative reinforcement learning methods but also provides a clear and transparent understanding of the decision-making process. By combining domain knowledge with rule-based strategies, our model offers enhanced interpretability without sacrificing effectiveness. This interpretability is particularly valuable in applications where understanding and trust in the model's behavior are critical. We will explore the scalability of our method in more complex environments and refine the balance between interpretability and performance.

REFERENCES

[1] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems* 31 (2018).

[2] John A Bernard. 1988. Use of a rule-based system for process control. *IEEE Control Systems Magazine* 8, 5 (1988), 3–13.

[3] Gabriele Brunner, Yang Liu, David Pascual, and Roman Richter. 2020. On Identifiability in Transformers. In *International Conference on Learning Representations (ICLR)*.

[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[5] Yannick Coppens, Konstantinos Efthymiadis, Tom Lenaerts, and Ann Nowé. 2019. Distilling Deep Reinforcement Learning Policies in Soft Decision Trees. In *IJCAI Workshop on Explainable Artificial Intelligence (XAI)*.

[6] Arun Das and Paul Rad. 2020. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371* (2020).

[7] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. 2010. Ender: a statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery* 21, 1 (2010), 52–90.

[8] Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research* 13 (2000), 227–303.

[9] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[10] Jerome H Friedman and Bogdan E Popescu. 2008. Predictive learning via rule ensembles. *The annals of applied statistics* 2, 3 (2008), 916–954.

[11] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. 2021. Explainability in deep reinforcement learning. *Knowledge-Based Systems* 214 (2021), 106685.

[12] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608* (2018).

[13] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[14] Zhenyu Li, Tian Yang, and Yu Cao. 2021. Explainable Reinforcement Learning: Problems, Methods, and Applications. *Comput. Surveys* (2021).

[15] Yang Liu, Xinzhi Wang, Yudong Chang, and Chao Jiang. 2022. Towards Explainable Reinforcement Learning Using Scoring Mechanism Augmented Agents. In *Knowledge Science, Engineering and Management*. Springer, 576–589.

[16] Volodymyr Mnih. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[17] Volodymyr Mnih. 2016. Asynchronous Methods for Deep Reinforcement Learning. *arXiv preprint arXiv:1602.01783* (2016).

[18] Xiangyu Peng, Mark Riedl, and Prithviraj Ammanabrolu. 2022. Inherently explainable reinforcement learning in natural language. *Advances in Neural Information Processing Systems* 35 (2022), 16178–16190.

[19] Erika Puiutta and Eric MSP Veith. 2020. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*. Springer, 77–95.

[20] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[21] Robert E Schapire. 1990. The strength of weak learnability. *Machine learning* 5, 2 (1990), 197–227.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[23] Magne Setnes, Robert Babuska, and Henk B Verbruggen. 1998. Rule-based modeling: Precision and transparency. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28, 1 (1998), 165–169.

[24] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. 2010. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization* 20, 6 (2010), 2807–2832.

[25] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. 2020. Explainable deep learning models in medical image analysis. *Journal of imaging* 6, 6 (2020), 52.

[26] Eduardo Soares, Plamen P Angelov, Bruno Costa, Marcos P Gerardo Castro, Subramanya Nageshrao, and Dimitar Filev. 2020. Explaining deep learning models through rule-based approximation and visualization. *IEEE Transactions on Fuzzy Systems* 29, 8 (2020), 2399–2407.

[27] Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book* (2018).

[28] Lindsay Wells and Tomasz Bednarz. 2021. Explainable ai and reinforcement learning—a systematic review of current approaches and trends. *Frontiers in artificial intelligence* 4 (2021), 550030.

[29] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. 2019. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *Natural language processing and Chinese computing: 8th cCF international conference, NLPCC 2019, dunhuang, China, October 9–14, 2019, proceedings, part II 8*. Springer, 563–574.

[30] Fan Yang, Pierre Le Bodic, Michael Kamp, and Mario Boley. 2024. Orthogonal Gradient Boosting for Simpler Additive Rule Ensembles. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1117–1125.