Use Bag-of-Patterns Approach to Explore Learned Behaviors of Reinforcement Learning

Gulsum Alicioglu, Bo Sun[*]
Department of Computer Science, Rowan University, Glassboro, NJ, (USA)

Abstract
Deep reinforcement learning (DRL) has achieved state-of-the-art performance, especially in complex decision-making systems such as autonomous driving solutions. Due to their black-box nature, explaining the DRL agent's decision is crucial, especially for sensitive domains. In this paper, we use the Bag-of-Pattern (BoP) method to explore the learned behaviors of DRL, where we can find high-frequent rewarded and non-rewarded behaviors along with low-frequent rewarded and non-rewarded behaviors. This exploration helps us to identify the effectiveness of the model in completing the given tasks. We use the Atari Learning Environment, the Pong game, as a test-bed. We extracted learned strategies and common behavior policies using the most frequent BoP created for each state. Results show that the agent trained with Deep Q-Network (DQN) has adopted a winning strategy by playing in a defensive mode and focusing on maximizing reward rather than exploration. The agent trained with Proximal Policy Optimization (PPO) algorithm has lowered performance by showing more variational behavior to explore states and takes frequent up and down actions to prepare incoming shots from the opponent.

Keywords
Reinforcement learning, XAI, bag of patterns, deep q-network, proximal policy optimization.

1. Introduction

DRL has often been used by complex systems including Atari games [1], autonomous vehicles [2], and healthcare systems [3] because of its capability to resolve complex decision-making problems. However, due to the difficulty of explaining their decision-making process, RL is still considered a black-box and needs explanations on how the agent works to gain the trust of users and to develop more robust agents [4, 5]. To make RL policies more interpretable, the field of explainable RL (XRL) has emerged. The XRL mainly focuses on adopting explainable artificial intelligence (XAI) methods to provide post hoc and intrinsic explanations for RL decision-making process. XRL extends RL explanations by including human interactions [6, 7] that directly manipulate the agent's ability and providing interactive visualizations [8, 9] to make RL policies transparent [10]. The majority of the XRL research focuses on explaining the agent's decisions based on individual states locally [5, 9]. However, local explanations do not explain how the agent makes decisions and overall behavior of the RL agents. Due to the sequential nature of the RL and limitations of local explanations, XRL research [11-14] has begun to consider global explanations to understand the agent's policy. In this study, we contribute to the field of XRL by extracting the learned behaviors of two DRL agents, trained by Deep Q-Network [1] and Proximal Policy Optimization [15], from a time-dependent sequential patterns using Multivariate Bag-of-Pattern [16]. The proposed method eliminates the limitations of local explanations by summarizing winning strategies adopted by an RL agent. Unlike the other traditional XAI methods, the proposed approach captures the temporal dynamics of RL by highlighting the recurring high- and low-frequent rewarded and not-rewarded patterns over time. The rest of the paper is organized as follows: Section 2 provides a related work in the field of XRL. Section 3 covers the methodology

Late-breaking work, Demos and Doctoral Consortium, colocated with The 2nd World Conference on eXplainable Artificial Intelligence: July 17–19, 2024, Valletta, Malta
*Corresponding author
alicio87@rowan.edu (G. Alicioglu), sunb@rowan.edu (B. Sun)
https://orcid.org/0000-0002-1385-1934 (G. Alicioglu)

of the use of multivariate BoP for RL. Section 4 presents the current results and section 5 concludes the paper with future directions of the work.

2. Related Work

The initial efforts for interpretation of the DRL include the use of t-Distributed Stochastic Neighbor Embedding (t-SNE) for neural activations to identify the similarities of states [17]. The research of Mnih et al. [17] visualize the representations in the last hidden layer for DQN model using t-SNE by coloring state values. Their results indicated that t-SNE groups states based on their cumulative reward similarities [17]. With the advancement of the field of XAI, current XRL research, as seen in [5, 9, 18, 19], focuses on explaining a single decision of an RL agents using saliency maps to highlight the relevant features that contribute to the decision. For example, Greydanus et al. [9] applied perturbation-based saliency maps by adding noise to the observation to identify relevant features that cause the agent to take action [9]. Weitkamp et al. [18] uses Grad-CAM to highlight the activation map on observations for Atari games. Iyer et al. [19] modified saliency maps and introduced object saliency maps that highlight objects that have influence on the agent's decisions rather than pixels. Huber et al. [5] evaluates and compares perturbation-based saliency maps using sanity checks, input degradation and run-time metrics. The study of Huber et al. [5] measures the usability and effectiveness of saliency maps that identify the decision-making of an RL agent.

However, the recent studies [20, 21] emphasize that explaining a single decision does not give insights into the overall behavior and temporal dynamics of an RL agent. Moreover, saliency maps provide subjective explanations [4] and require additional tools [21] to evaluate the agent's behaviors. Another limitation of using saliency maps or more general instance-level explanations is that requirement of analyzing the decisions of RL agents for each observation [4, 22]. To tackle these limitations, XRL researchers develop global XAI methods to explain the learned behaviors [11], extract strategy summaries [12] and list a series of logical rules [13, 14] of DRL agents. The most popular approach is to use decision trees to extract a set of rules for policies using Verifiability via Iterative Policy Extraction (VIPER) [13]. Osbert et al. [13] introduced VIPER that transforms pre-trained policies into decision tree policies to perform imitation learning and make RL policies interpretable. Another popular approach, introduced by Liu et al. [14], is Linear Model U-trees (LMUTs) to perform imitation learning for neural network policy predictions. LMUTs [14] represent Q functions, used for learning optimal policies, as decision trees. While these methods provide logical rules for RL policies, using interpretable models such as decision tree to perform imitation learning do not explain the actual RL policy and are not robust for unseen scenarios [4]. Amir and Amir [11] introduced a heuristic approach called HIGHLIGHTS to summarize learned strategies by DRL agents using trajectories based on state importance [11]. This method focuses on selecting sub-trajectories that represent a summary of a learned behavior. However, selecting important states based on significant decrease in the future rewards because of selected action highlight only the extreme policy behavior rather than generalization [22]. Septon et al. [12] combines HIGHLIGHTS method [11] with reward decomposition for DRL explanations, however their results indicate that HIGHLIGHTS method does not contribute to the explanations due to efficiency of reward decomposition. Recent research emphasizes that current XRL research is insufficient to explain RL decision-making process due to its black-box nature [4, 10, 22] and complexity and requires additional tools [21] and combination of methods. Local explanations lack of temporal dynamics of DRL agents in explaining the learned behavior of agents. global explanations through imitation learning [13, 14] lack of generalization and is not robust for unseen scenarios since they focus on explanations on approximated models. Moreover, heuristic approaches [11] provides subjective solutions and lacks empirical evaluation. Therefore, we focus on sequential-based explanations by extracting recurring patterns and capturing temporal dynamics of RL agents to overcome the limitations of local and global XAI methods. The proposed approach explores time-wise sequences of BoPs for high- and low- frequent rewarded and non-rewarded behaviors of DRL agents. The method explains the overall strategy adopted by RL agents to win the game and shows the effectiveness of behavior of good and bad performing agents.

3. Method

3.1. RL Model and data collection

Reinforcement learning (RL) trains an agent to take actions in an environment with the goal of achieving maximum rewards [23]. RL is modeled as a Markov decision process (MDP), represented as a tuple M = {*S, A, p, r, γ*}, where *S* denotes the state space, *A* is the set of actions, *p : S x A x S*→ [0, 1] is the state transition function, *r : S x A* → R denotes the reward function and γ denotes the discount factor [4, 10]. The main objective of an RL agent is to learn a policy (π*) that maximizes the expected return. We trained two DRL agents using two different RL algorithms: DQN [1] and PPO [15] from Stable Baselines [24] in OpenAI Gym [25] to deal with environments. For the DQN agent, we use the same hyperparameters settings as in Mnih et al. [1]. For the PPO agent, we used the default hyperparameter setting from Stable Baselines except learning rate (set as 0.00025) and entropy coefficient (set as 0.01). We also modified natureCNN [17] feature extractor architecture for our PPO model. The new feature extractor architecture has 3 convolutional layers (*Conv2D (32, 8), Conv2D (64, 4), Conv2D (128,2), Flatten (512)*). We trained both models for 25 million time steps for the Pong game in the Atari Learning Environment [26]. There are two players in the pong game, one represents an agent and controls the right paddle, the other player represents the computer (the environment) and controls the left paddle. If one of the players fails to catch the ball, the other player gets one point, and the game ends when one of the players reaches 21 points. The agent receives rewards of 1 for scoring a point, -1 for failing to catch the ball, and 0 for otherwise. The agents observe the last 4 frames (84x84x4 input images) and make predictions to choose an action. After training phase, we played the pong game for each model one ***episode*** that includes all time steps from starting game to ending game (any player reaches 21 points) and collected:

- *time steps* (starting 0 to until the episode ends),
- *actions* (0 *(noop)*, 1 *(fire)*, 2 *(up)*, 3 *(down)*, 4 *(up fire)*, 5 *(down fire)*)
- *reward* (-1, 0, 1),
- x and y coordinates of the ball,
- x and y coordinates of the paddles controlled by an agent and an opponent,
- *events* (score, miss, hit) for each episode.

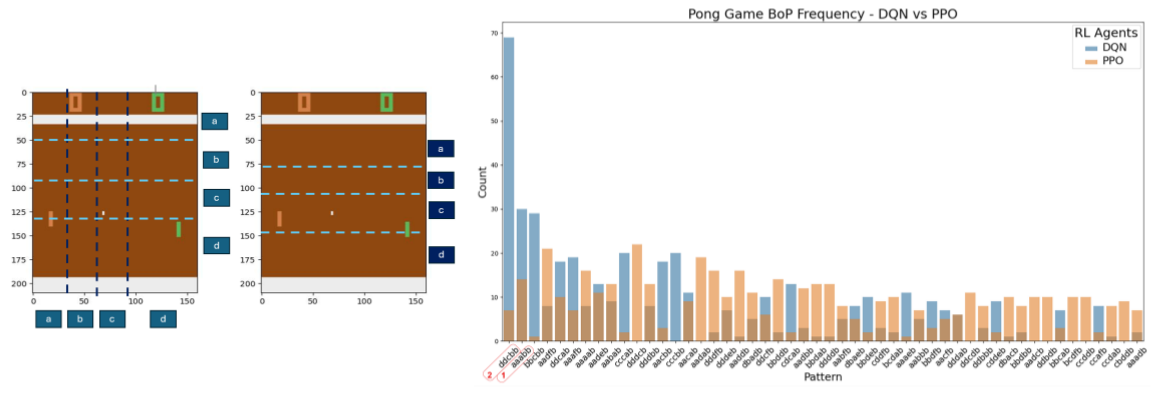
3.2. Interpretation using Multivariate Bag-of-Patterns

Bag-of-Pattern approach [27] is developed to extract and capture high-level information from univariate time series data [28]. The BoP approach uses sliding window technique [29] to extract subsequences from univariate time series and convert them into its Symbolic Aggregate ApproXimation (SAX) [27] representations (i.e., a word/letter). The parameters of BoP include window length (the number of time steps) *l*, word length (the length of symbolic representation) *w*, window step *s*, and alphabet size (e.g., a, b, c, …) α. Multivariate BoP [16] extends BoP method to capture the relationships between multiple time series for multivariate variables.

3.2.1. Extracting BoPs and Initial Exploration

The policy of the RL includes the state of the ball and paddle position, the action of the agent and the reward to encourage actions that contribute to winning. We converted image-based observations of RL into time-series sequences. Then we applied multivariate BoP to convert these changes into an alphabetic letter-based index. For multivariate BoP, we set window length (*l*), word length (*w*), and window size (*s*) as 1 to create a wording index for each time step. The alphabet size is set as 4 (*a, b, c, d*) for the coordinates of the ball and the agent, and the reward feature as seen in Figure 1. Since the number of actions is six, we set alphabet size as 6 (*a, b, c, d, e, f*) to represent actions noop, fire, up, down, up fire and down fire individually. BoP assigned letters:

- ball coordinates (x-axis): "a" from 0 to 29, "b" from 30 to 60, "c" from 61 to 90, "d" from 91 to 160.
- ball coordinates (y-axis): "a" from 0 to 51, "b" from 52 to 94, "c" from 95 to 136, "d" from 137 to 190.
- agent's paddle coordinates (y-axis): "a" from 29 to 76, "b" from 77 to 110, "c" from 111 to 144, "d" from 145 to 190.
- actions: "a" for noop, "b" for fire, "c" for up, "d" for down, "e" for up fire, "f" for down fire.
- reward: "a" for -1 reward, "b" for 0 reward, "d" for +1 reward.



(a) **(b)**

Figure 1: (a) A visual illustration of example for pattern "ccceb" at time step = 4, for DQN agent. The frame on the left side represents the intervals for the coordinates of the ball. The frame on the right side represents the intervals for the position of the agent paddle on the y-axis (right side). **(b)** The most frequent 50 BoPs for both agents in an overlay bar chart.

After creating a word for each feature, we concatenate those generated words to create a 5-lettered pattern (such as "ccceb" as seen in Figure 1.a) that represents game features as one for each time step. The first two letters correspond to the ball coordinates ("cc"), the third letter indicates the paddle position of the agent on the y-axis ("c"), the fourth letter indicates agent's action ("e") and the last letter indicates the current reward ("b"). The x-axis of the paddle for the agent is fixed at 140 by the game itself. The top-left corner of the paddle is considered to determine the exact coordinate of the agent on the y-axis. After playing one episode for each RL agent, we obtained the results and collected the data. The DQN agent won the game with an 8-21 final score, and the PPO agent lost the game with a 21-10 final score. To compare behavioral differences and make in depth exploration for both agents, we visualized the most frequent 50 patterns extracted from multivariate BoPs in an overlay bar chart (Figure 1.b). In the overlay bar chart, overlapping bars that represent the occurrence of BoPs for DQN and PPO indicate common patterns. For example, the BoP "aaabb", highlighted and labeled as 1 in Fig. 1.b, recurred 30 times for DQN and 14 times for PPO agent during an episode.

3.2.2. Extracting Pathways and Learned Behaviors

To capture the temporal dynamics of the RL agents' behaviors, we traced back six time steps starting from high frequent common patterns such as the most frequent BoP "ddcbb" from Fig 1.b labeled as 2. As a result, we explored five common recurring ball movements that are represented as first two letters indicating the x- and y- coordinates of the ball, such as "dd" in the BoPs for both DQN and PPO agents. We define these repeated ball movements as **pathways (PW)**. To focus on agents' behaviors and strategy for incoming shots, the starting point of PWs are set as the midpoint of the game board. Figure 2 shows visual representations of pathways that are created by overlapping time-wise consecutive frames. The first frame is pinned to display the movement paddles controlled by the agent and the environment. The pathways are read from left to right; the ball movement is taken from halfway towards the RL agents. The representation of the letter indicates the direction for incoming shots. For example, since we read the PWs from left to right, we do not expect to see letter "a" or "b" in the first pattern since it indicates x-coordinate of the ball and "a" and "b" represent values up to 60 on the x-axis. The first letter of the patterns may start from "c" and goes to "d" that indicates the ball goes left to the right on the x-axis. The average length of the pathway sequence is 6 time steps, depending on the speed of the ball to reach the agent's paddle. Pathways are named according to their first appearance during an episode. ***Pathway 2,*** explored from the most frequent BoP "ddcbb" in Fig. 1.b by tracing back 6 time steps, is the most recurring pathway for both agents (Figure 2.b). To generalize agent's behavior for each pathway, we created a 1-dimensional nested lists to store sequences of BoPs. The general structure of nested lists of patterns (***P***):

_P_ = [Pathway index, play index, [Ball coordinate], [Agent's position], [Agent's action], Reward]

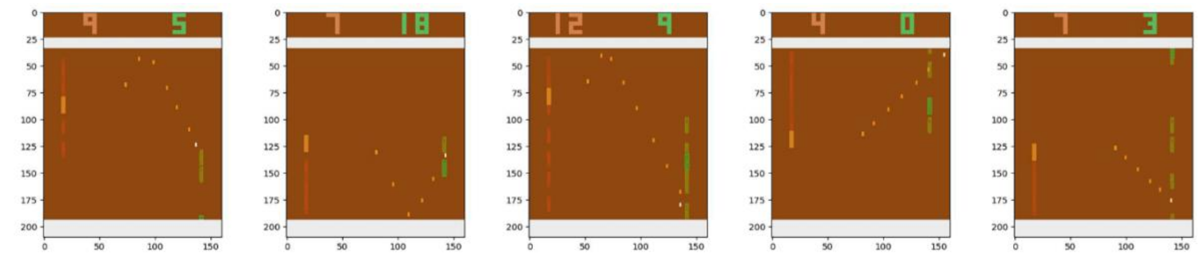

a. Pathway 1 **b**. Pathway 2 **c**. Pathway 3 **d**. Pathway 4 **e**. Pathway 5

Figure 2: A visual representation of common pathways that show the temporal dynamics of the Pong game. Pathways 1, 3 and 5 are visual representation taken from the PPO agent, pathways 2 and 4 are visual representation taken from the DQN agent.

Play index refers to the occurrence of each pathway during an episode. The nested list shows an example of a winning case (reward is 1) for the DQN agent for pathway 2, play index 3 with BoPs:

P = [PW 2, 3, [cc, dd, dd, dd, dd, dd], [c, c, c, c, c, c], [a, b, a, a, b, b], 1]

We applied numerosity reduction for repeated consecutive patterns and we stored reward as it is, e.g., -1, 0, 1 to reduce pattern complexity. Applying numerosity reduction for repeated consecutive frames will reduce the use of storage for representing a pattern list of an episode for a game play. This will remove the memory usage limitations when dealing with high-dimensional large data. The numerosity reduction simplifies BoP analysis by identifying recurring sequences of the repeated patterns such as the ball follows a route "dd" in a state and the agent stays at the region "c" during the next five time steps in *P**. The compressed list representations present the temporal dynamics of the RL agent's behavior and ease higher-level comparison rather than analyzing each observation and action individually. Numerosity reduction and compressed nested lists of sequences of observations and actions contribute generalization for the behavior of an agent for unseen scenarios. The pattern after numerosity reduction is applied is:

P* = [PW 2, 3, [cc, dd_5], [c_6], [a, b, a_2, b_2], 1]

4. Results

The most frequent and dominant pattern is "***ddcbb***", labeled as 2 in Figure 1.b, repeated 68 times per episode for DQN model. "**dd**" indicates that the ball is in the bottom right side of the frame, "**c**" indicates the agent's position, "**bb**" indicates agent takes action "b" (fire) with a reward of "b" indicating 0. Another finding from the BoPs is that the DQN model has high frequent less variational patterns indicating the agent learned a behavior (*a winning strategy*) and repeats this strategy to win the game. However, the PPO agent has low frequent more variational patterns indicating that the agent does not adopt any single strategy and execute stochastic actions to explore more states. Due to intense exploration and not adopting a strategy, the PPO agent failed to win the game. To verify the usability of multivariate BoPs, we played additional episodes for both agents and applied a computational approach that automatically identifies recurring patterns and provides a sequence of nested lists for each pathway and play index. The computational approach applies numerosity reduction to reduce pattern complexity. Table 1 shows detailed explanation of pathways in terms of the number of occurrences during each episode for both DRL agents, scoring and missing points per pathway, and a brief pathway definition. Pathways that appeared only once for each model are removed from the list. Pathways are created to capture the temporal dynamics of both RL agents and highlight the effectiveness of both models.

Table 1
A detailed explanation of pathways extracted from most frequent Bag-of-Patterns.

Pathway		DQN Agent		PPO Agent		Definition
		Freq.	Score/Miss	Freq.	Score/Miss	
PW 1	Episode 1*	1	1 miss	10	2 score/3 miss	Ball bounces back from upper-right side
	Episode 2*	2	1 miss	9	1 score/4 miss	
PW 2	Episode 1	**20**	**18 score/1 miss**	**16**	**3 score/5 miss**	Ball bounces back from down-right side
	Episode 2	**26**	**20 score/4 miss**	**20**	**8 score/3 miss**	
PW 3	Episode 1	1	Rebound~	8	1 score/8 miss	Ball bounces back from upper-left side
	Episode 2	1	Rebound	4	3 miss	
PW 4	Episode 1	**8**	**3 score/4 miss**	10	1 score/3 miss	After restart, ball goes diagonally up
	Episode 2	**11**	**1 score/4 miss**	8	1 score/1 miss	
PW 5	Episode 1	1	Rebound	**12**	**3 score/4 miss**	After restart, ball goes diagonally down
	Episode 2	1	Rebound	**11**	**2 score/5 miss**	

**Episode 1: DQN Agent (8 – 21), PPO Agent (21 – 10). Episode 2: DQN Agent (10 – 21), PPO Agent (21 – 12).* Green color indicates the agent's score, orange color indicates the opponent's score. ~ Rebound indicates that the agent catches the ball but does not receive any score or miss.

Table 1 shows that PW 2 is the most recurring movement pattern for both agents. While the DQN agent scores 18 and 20 points out of total scores of 21 in PW2, the PPO agent scores 3 and 8 points for episodes 1 and 2, respectively. According to BoPs occurred in PW 2, the ball follows a route corresponding to "cd" and "dd" regions in a state (see Figure 1 to recall regions), the agent stays at position "c" as results of actions are "a" (*noop*) and "b" (*fire*) to win the game. Since the actions noop and fire do not change the actual position of the agent on frame, we can group actions noop and fire as "ab", i.e., 'stay still'. Pathway 2 (Fig. 2.b) indicates that the DQN agent anticipates the ball movement and keeps its position still to prepare for incoming shots. We call this learned winning strategy "*defensive mode*". This strategy focuses on winning the game by maximizing rewards rather than exploring the states. From all play indexes for pathway 2, we generated a compact representation of BoP for "winning strategy" for the DQN agent. The general pattern for winning strategy derived from pathway 2 for the DQN model is:

PW 2 – winning strategy: [index, [cc, dd], [c], [ab], 1]

The general pattern shows the high-frequent rewarded behavior of the DQN agent. The DQN agent has high-frequent rewarded behaviors from PW 2 and 4, and low-frequent non-rewarded behaviors from PW 1, 3, and 5 that indicates the effectiveness of the DQN agent in PWs 2 and 4. Since the PPO agent does not adopt a winning strategy and focuses on the exploration, it only gets 3 scores from PW 2 with low-frequent rewarded behavior compared to the DQN. Figure 2.c and 2.e justifies the exploration of the PPO agent that takes frequent *up* and *down* actions to adjust its paddle position for incoming shots. However, the DQN agent chooses to exploit staying at a certain position and wait rather than exploration. Therefore, BoP analysis indicates that the DQN agent has a “repeated” behavior that leads to winning and the PPO agent has a “hesitated” behavior that causes failure. The second most recurring movement appeared on PW 4 for the DQN agent and PW 5 for the PPO agent. PW 4 and PW 5 appear when the game restarts after one of the players gets a point in an episode. While the DQN agent has high-frequent rewarded behaviors only in PWs 2 and 4, the PPO agent has a more scattered behavior in PWs. The PPO agent gets scores from PW 1, 2, and 5. Due to stochasticity of PPO algorithms, the agent focuses on exploration of the states and does not repeat a certain behavior often. We set the number of total time steps as 25 million in the training phase to make comparison between both models. While the total time steps allow a DQN agent to adopt a winning strategy, the PPO agent may require having more time steps to learn a strategy. The PPO agents lost most of the points in PW 3, occurring when the ball bounces back from the upper-left side. The agent failed to anticipate the movement of the ball and could not prepare for an incoming shot. We modified the default feature extractor, natureCNN [17], in our PPO algorithm, while keeping it for the DQN algorithm. One of the failure reasons of the PPO agent in PW 3 is that the agent could not extract the relevant features from the observations to adjust its position to hit the ball.

5. Conclusion

The study uses Bag-of-Pattern as an XAI method to explain the behavioral differences between the agents of DQN and PPO algorithms by capturing temporal dynamics of the agent and the environment. The proposed method provides sequence-based explanations to highlight the learned behaviors adopted by RL agents to achieve the goal. The multivariate BoP identified high- and low-frequent rewarded and non-rewarded patterns and revealed the effectiveness of good and bad performing agents. The results indicate that the DQN agent has adopted a winning strategy and scored 18 points out of 21 in pathway 2 by following the learned high-frequent rewarded behavior called “***defensive mode***”. The proposed method discovered that “defensive mode” forces the ball to follow the same route (pathway 2), so the DQN agent stays at the same position. We also presented a general representation of PW 2 for the DQN agent. The PPO agent has more variation in terms of encountered pathways and actions taken. The PPO algorithm itself has stochasticity and the agent of it focuses on exploration more rather than learning a strategy. The agent takes frequent up and down actions indicating that hesitating to find a location for the paddle to catch the ball. Future work includes testing the proposed approach in other RL domains such as robotic tasks. We aim to quantify the current results and create a generalized approach to apply BoP for explaining RL models.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. A. Riedmiller, Playing Atari with Deep Reinforcement Learning, ArXiv (2013). doi:abs/1312.5602.

[2] L. Wang, J. Liu, H. Shao, W. Wang, R. Chen, Y. Liu, S.L. Waslander, Efficient reinforcement learning for autonomous driving with parameterized skills and priors, arXiv (2023).

[3] M. Fatemi, T.W. Killian, J. Subramanian, M. Ghassemi, Medical dead-ends and learning to identify high-risk states and treatments, Adv. in Neural Inf. Proc. Sys., 34, 4856-4870, (2021).

[4] S. Milani, N. Topin, M. Veloso, F. Fang, Explainable Reinforcement Learning: A Survey and Comparative Review, ACM Comput. Surv. 56(7), (2024).

[5] T. Huber, B. Limmer, E. Andre, Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents, Frontiers in Artificial Intelligence 5 (2021).
[6] M. Sridharan, B. Meadows, B, Towards a Theory of Explanations for Human - Robot Collaboration, KI - Künstliche Intelligenz, (2019). Doi: 10.1007/s13218-019-00616-y
[7] S.H. Huang, D. Held, P. Abbeel, A.D. Dragan, Enabling robots to communicate their objectives. Autonomous Robots 43, 309-326, (2019). doi: 10.1007/s10514-018-9771-0
[8] J. Wang, L. Gou, H.W Shen, H. Yang, Dqnviz: a visual analytics approach to understand deep q-networks. IEEE Trans. Visual. Comput. Graph. 25, 288-298, (2018).
[9] S. Greydanus, A. Koul, J. Dodge, A. Fern, Visualizing and Understanding Atari Agents. ArXiv, (2017). doi: abs/1711.00138.
[10] L. Wells, T. Bednarz, Explainable ai and reinforcement learning—a systematic review of current approaches and trends. Frontiers in artificial intelligence, 4, 550030, (2021).
[11] D. Amir, O. Amir, Highlights: Summarizing agent behavior to people, In Proc. of the 17th international conference on autonomous agents and multi-agent systems (AAMAS), (2018).
[12] Y. Septon, T. Huber, E. Andre, O. Amir, Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents, In Int. Conf. on Practical Apps of Agents and Multi-Agent Systems, 320-332, (2023).
[13] B. Osbert, P. Yewen, S.L. Armando, Verifiable Reinforcement Learning via Policy Extraction. In NeurIPS, (2018).
[14] G. Liu, O. Schulte, W. Zhu, Q. Li, Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees, ECML/PKDD, (2018).
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv,* (2017). *doi: arXiv:1707.06347*.
[16] P. Ordoñez, T. Armstrong, T. Oates, J. Fackler, U.C. Lehman, Multivariate methods for classifying physiological data. In: Proceedings of SIAM International Conference on Data Mining, Workshop on Data Mining Medicine and HealthCare (DMMH 2013). 2013.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Belle-mare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518, 529-533, (2015).
[18] L. Weitkamp, E.V. Pol, Z. Akata, Visual Rationalizations in Deep Reinforcement Learning for Atari Games, BNCAI, (2018). doi:10.1007/978-3-030-31978-6_12.
[19] R.R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, K.P. Sycara, Transparency and Explanation in Deep Reinforcement Learning Neural Networks, ACM Conference on AI, Ethics, and Society, (2018).
[20] W. Guo, X. Wu, U. Khan, X. Xing, EDGE: Explaining Deep Reinforcement Learning Policies, Neural Information Processing Systems, (2021).
[21] A. Atrey, K. Clary, D.D. Jensen, Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning, Intl. Conference on Learning Reps, (2019).
[22] G. A. Vouros, Explainable Deep Reinforcement Learning: State of the Art and Challenges, ACM Comput. Surv. 55(5), (2022). doi: https://doi.org/10.1145/3527448.
[23] R. S. Sutton, A.G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
[24] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, N, Stable-Baselines3: Reliable Reinforcement Learning Implementations, *J. Mach. Learn. Res., 22*, 1-8 (2021).
[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, *arXiv, (2016), doi: arXiv:1606.01540*.
[26] M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: an evaluation platform for general agents, J. Artif. Intell. Res. 47, 253–279, (2013).
[27] J. Lin, Y. Li, Finding Structural Similarity in Time Series Data Using Bag-of-Patterns Representation. Int. Conference on Statistical and Scientific Database Management, (2009).
[28] Y. Benyahmed, Y., A. RazakHamdan, S. Mastura, S. Abdullah, Bag of Patterns Representation Technique of Constructed Detection Temporal Patterns for Particular Climatic Time Series, (2016).
[29] T. Palpanas, M. Vlachos, E.J. Keogh, D. Gunopulos, W. Truppel, Online amnesic approximation of streaming time series. 20th Intl. Conference on Data Engineering, 339-349, (2004).