Exploring the Reliability of SHAP Values in Reinforcement Learning

Raphael C. Engelhardt[1][0000−0003−1463−2706], Moritz Lange[2][0000−0001−7109−7813], Laurenz Wiskott[2][0000−0001−6237−740X], and Wolfgang Konen[1][0000−0002−1343−4209]

[1] Cologne Institute of Computer Science, Faculty of Computer Science and Engineering Science, TH Köln, Gummersbach, Germany
{Raphael.Engelhardt,Wolfgang.Konen}@th-koeln.de
[2] Institute for Neural Computation, Faculty of Computer Science, Ruhr-University Bochum, Bochum, Germany
{Moritz.Lange,Laurenz.Wiskott}@ini.rub.de

Abstract. Explainable artificial intelligence (XAI) is an increasingly important research field, fueled by the need for reliability and accountability in applications. For reinforcement learning (RL), achieving explainability is particularly challenging because agent decisions depend on the context of a trajectory, which makes data temporal and non-i.i.d. In the field of XAI, Shapley values and SHAP in particular are among the most widely used techniques. In this work, we investigate how SHAP performs in explaining RL models, especially in multidimensional action spaces that other XAI-for-RL methods struggle with. In particular, we make three contributions: (1) We investigate how design choices of the SHAP approach affect SHAP accuracy for RL models. We investigate the size of the so-called background data that is utilized to represent absent features, as well as the selection method with which the background data is formed. We find that SHAP for RL requires only modest amounts of background data and that clustering is preferred over sampling as a selection method. (2) Additionally, we analyze how SHAP-based feature importance relates to overall agent performance (return). We find that while feature importance is often correlated to agent performance, notable exceptions occur, especially for environments that are sensitive or fragile in the sense that small changes in actions may lead to catastrophic failure. However, since a significant correlation is found in the majority of the investigated environments, SHAP proves to be a valuable XAI tool for RL with multidimensional, continuous actions. (3) Illustratively, we show the time evolution of SHAP values and caution against misinterpreting sharp changes therein.

Keywords: Reinforcement learning · Explainability · Shapley values · SHAP · XAI.

1 Introduction

The issue of explainability in artificial intelligence (XAI) has been of increasing importance during the last years and was often cited [1, 2] as one of the main challenges when applying AI to real-world scenarios, especially in safety-critical fields. As a consequence, a variety of methods have been developed with the goal of increasing insights into opaque AI models. One of these methods is the SHAP framework [3], rooted in mathematical game theory [4].

XAI for reinforcement learning (RL), and in particular SHAP for RL, can be more challenging than XAI for supervised machine learning (ML). This is due to the temporal and non-i.i.d. nature of the data in RL, where decisions depend on the state of the environment. In the context of RL, agents with multidimensional actions pose a particular challenge for XAI, and this article examines in particular what contribution SHAP can make to this challenge.

1.1 Shapley Values

Named after Lloyd Shapley, Shapley values give a solution to the problem of fairly distributing a given payout among the cooperative players of a game [4]. In short, the Shapley value corresponds to a player's marginal contribution to the possible coalitions or the expected performance gain when said player joins a coalition. Given a game with a set $\mathcal{N}$ of $n = |\mathcal{N}|$ cooperative players and a function v assigning a value to each coalition, player j's added contribution to coalitions $\mathcal{S}$ is given by

$$\phi_j = \frac{1}{n} \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{j\}} \binom{n-1}{|\mathcal{S}|}^{-1} \left(v(\mathcal{S} \cup \{j\}) - v(\mathcal{S}) \right) \tag{1}$$

It can be mathematically shown that Shapley values are the only method with a variety of desirable properties (efficiency, symmetry, dummy, and additivity) that lead to a payout distribution that can be called "fair" [5]. For the exact computation, 2^{n-1} values of coalitions containing a specific player must be compared to 2^{n-1} values of coalitions without the player, hence the cost is exponential in the number n of players.

1.2 Shapley Values for ML – SHAP

In the transition from game theory to machine learning, the prediction of an ML model takes the place of the value function, while the input features take the role of the single players. For large numbers of features, the exact computation of Shapley values suffers from combinatorial explosion and is generally not feasible. The framework SHAP [3] offers a variety of different approximation methods (one of them being KernelSHAP, which will be explained in more detail in Sect. 4).

The framework has seen a remarkable success, has been expanded with different approximation methods optimized for certain ML models as well as visualization tools, and has often been cited as the go-to approach for model-agnostic explainability of ML models.

When used to explain ML decisions, the first examples are typically the explanation of single decisions (think of the often-used example of a denied bank-loan) or the discovery of general trends in classification examples. When applied to RL, the prediction of the ML model is the action performed by the agent in the environment, while the features are the observables accessible to the agent. The words "features" and "observations" can be used interchangeably; the terms "prediction" and "action" are also used as synonyms in the following. In this setting, the SHAP values are the contribution of each feature towards the model's prediction, so that the RL agent's action is the sum of its average action[3] and the SHAP values of all observables. We note in passing, that taking the RL agent's action as the value function v in Eq. (1) for the SHAP procedure is not the only possibility. Alternatively, one could also take the episode return as a possible value function v. However, since this has higher computational demands, it was not considered in this work, but left for future work instead.

Compared to supervised ML classification and regression tasks, where the data can usually be assumed to be i.i.d., RL has structural differences. Given the interaction between agent and environment, RL data are usually non-i.i.d.: Decisions depend on the state of the environment and the overall success is determined by a sequence of profitable actions. In addition, complex environments require the agent to perform multidimensional actions at each timestep. As a result, each action dimension has its own set of SHAP values. Aggregating this multitude of values and ensuring the meaningfulness of SHAP for RL is a non-trivial task.

1.3 Contributions

The main contributions of this work are summarized as follows: (1) We empirically test the effect of the quantity of background data[4] and the method of background data selection on the computed SHAP values, and relate the results to the computation time. This evaluates the robustness of the approximation method and might help practitioners to better find the appropriate compromise between precision and computation time. (2) We expand the known definition of SHAP feature importance to the case of multidimensional actions and evaluate the computed feature importance on the RL task. (3) We interpret the time evolution of SHAP values throughout the episode in the context of the agent's actions.

In Sect. 2, we discuss the related work on SHAP for RL. Section 3 describes the RL environments we use as benchmark. In the three follow-up Sect. 4, 5, 6, we describe our experiments and discuss results regarding the three main objectives given above. Finally, we draw a conclusion and hint at possible future work in Sect. 7.

[3] When using KernelSHAP, the average action is the average model prediction on the background data (see Sect. 4.1)

[4] The background data are used by KernelSHAP to fill in data for features that are absent in the currently investigated feature coalition. This will be explained in more detail in Sect. 4.1.

2 Related Work

With the success of AI and ML, which was often made possible with the help of complex deep learning models, the last years have seen a growing interest in XAI [6–9]. Important explanation techniques for ML in general (mostly classification and regression) are post-hoc explanations, like SHAP [3], LIME [10], or LRP [11], and self-explainable models like linear models or decision trees (DT) [12].

In the following, we focus on explainability for RL, which is often more challenging than for supervised ML classification or regression. This is due to the reasons already mentioned in Sect. 1.2 (multiple steps contribute to overall return; multiple actions or even multiple agents make it harder to find out which of the model outputs is responsible for reaching a high performance). Explainability in RL has been the topic of several reviews [2, 13, 14]. According to the review of Hickling et al. [13], the most common XAI approaches in RL are similar to the general ML case: either DTs as explainable surrogates for more complex DRL models [15–17] or post-hoc explanations via SHAP [5, 18–20] (or LIME or LRP).

DTs are often used to mimic simple (but not trivial) DRL agents (e.g. less than 10 inputs, single-dimensional action space) as was shown in [15–17, 21, 22]. If DTs are successful, they deliver explanations through human-understandable rules. However, for more complex environments (e.g. the MuJoCo environments studied in this work with $8 - 27$ observables and multidimensional, continuous actions), it can be difficult or impossible to find simple DTs and large DTs are no longer interpretable.

In those cases, many works resort to SHAP-based post-hoc explanations [13], as they can be aggregated across multiple actions, or evaluated for many input dimensions or for multi-agent RL (MARL). Heuillet et al. [5] use SHAP for MARL environments where the contribution of a specific agent to the global reward is measured via Shapley values. Their task is not to explain the actions of a single DRL agent, but to evaluate each agent's contribution. Another interesting approach using SHAP in conjuction with RL is the one described by Sequeira and Gervasio [23]. To the goal of gaining more insights into trained DRL agents, the authors compute different human-inspired "interstingness dimensions", e.g., "confidence" and "riskiness" from interaction data obtained by evaluating RL agents in their respective environment for multiple episodes. Among other analyses, they use the SHAP framework to investigate how features influence the different "interestingness dimensions" on a global level or to better understand local sudden changes throughout the episodes. Rizzo et al. [19] use SHAP to explain a single agent for a traffic light control system. SHAP values indicate which inputs are important in certain states. This is similar to our approach, but they use SHAP only for a single-dimensional action space and for illustrating the decisions made at certain timesteps. Zhang et al. [20] use RL and SHAP for power system control. Liessner et al. [18] study a simple car control environment where an agent follows a street lane and has to obey certain speed limits. They present a so called RL-SHAP diagram where the agent's inputs are placed on

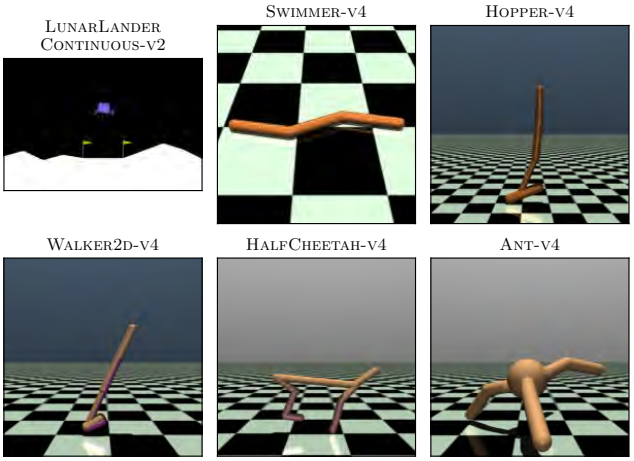

Fig. 1. Renderings of the benchmark environments

the y-axes, color-coded by the SHAP values, while the x-axis shows the distance traveled. This method provides great visual understanding for this simple problem, but is harder to apply to more complex problems with multidimensional action spaces.

3 Benchmark Environments

We conduct our experiments using a variety of RL environments from the Gymnasium [24] suite (see Table 1 and Fig. 1), in particular LunarLanderContinuous and a set of MuJoCo tasks. In LunarLanderContinuous the goal is to operate the main and lateral engines of the lander in order to safely land in a predefined area of the lunar surface. The MuJoCo tasks all have the goal of fast directed locomotion of simulated robots in various different shapes and therefore with different numbers of joints and actuators. All these environments share properties relevant for our investigations: the observations are multidimensional and continuous, as are the actions (corresponding to thrust of the engines in case of LunarLanderContinuous or torque applied to the joints of MuJoCo robots).

For the training of DRL agents, we rely on Stable Baselines3 [25]. Agents are trained using either TD3 [26] or TQC [27] in order to achieve state-of-the-art (as reported on https://huggingface.co/sb3) high-performance and low-fluctuation results. We used the hyperparameters from RL Baselines3 Zoo [28].

4 Experiment 1: Dependency of KernelSHAP on Background Data

The research question of this experiment is whether the size and selection procedure (sampling or clustering) of the background dataset are critical for the

Table 1. Environments used as benchmark

Environment	observation dim.	action dim.	DRL agent	$\overline{R}$: Performance in 100 episodes
LunarLanderCont.-v2	8	2	TQC	278.54 ± 29.29
Swimmer-v4	8	2	TD3	353.50 ± 2.41
Hopper-v4	11	3	TQC	3659.89 ± 6.30
Walker2d-v4	17	6	TD3	4470.01 ± 13.47
HalfCheetah-v4	17	6	TQC	12 098.48 ± 107.42
Ant-v4	27	8	TD3	5966.68 ± 809.69

accuracy of the SHAP values, especially in high-dimensional environments. To clarify the meaning of "background data", we briefly summarize the KernelSHAP approximation method.

4.1 KernelSHAP and Background Data

The KernelSHAP method approximates the features' impact by observing the output of the model when switching a feature from "absent" to "present". To compute the SHAP values of an observation $\mathbf{x}$ in J-dimensional space, first a number M of coalitions are sampled. Each coalition is represented by a vector $\mathbf{z} \in \{0,1\}^J$, where 0 means that the feature j is absent and 1 that the feature is part of the coalition. A transformation function h translates these encodings $\mathbf{z}$ to valid inputs for the model: while present features keep their actual value $h(z_j) = x_j$, absent features are replaced by values drawn randomly from the background data, $h(z_j) \prec \mathbf{B}_j$. This is the point at which the background data $\mathbf{B}$ come into play. They can be understood as matrix with J columns (features) and N_b rows. N_b is the size of the background dataset. $\mathbf{B}$ is constructed once prior to all KernelSHAP computations, by either sampling or clustering from a larger reservoir (see Sect. 4.2 for details). In the last step, a linear model

$$g(\mathbf{z}) = \phi_0 + \sum_{j=1}^{J} \phi_j z_j \tag{2}$$

is fitted to minimize the squared differences $(f(h(\mathbf{z})) - g(\mathbf{z}))^2$ for all M coalitions, where $f(h(\mathbf{z}))$ is the model output for a given input $h(\mathbf{z})$. The squared differences are weighted by the SHAP kernel (Theorem 2 in [3]), which assigns higher weights to coalitions with few and to coalitions with many present features. The coefficients of the linear model g are the SHAP values ϕ_j. As a consequence, the computed SHAP values are not only a function of the model, but also of the background data (as well as stochasticity).

It is recommended to reduce the size of large background datasets by sampling or clustering. To empirically study the impact of the background dataset's size and the effect of sampling or clustering, we propose the following experiment.

4.2 Experimental Setup

First, the background dataset is filled with samples from 10 episodes of the RL agent, leading to 10 000 samples for the MuJoCo environments and about 1700 for LunarLanderContinuous, where each sample is a point in the observation space. This dataset is then reduced to size N_b either by sampling or using the KMeans algorithm to produce N_b cluster centers. Based on these N_b background data, we now compute SHAP values for a fixed set of $N_e = 1000$ samples drawn from data logged during a number of different evaluation episodes. The consistency of SHAP values computed with background data of various sizes is qualitatively visualized by the dependency plots in Fig. 2 and quantitatively evaluated in Fig. 3 by computing the root mean squared error (RMSE), measuring the difference between SHAP values with $N_b \leq 100$ background data and approximately "true" SHAP values based on a much larger dataset with $N_b = 1000$ background data as reference. The SHAP values corresponding to each action dimension are normalized by the standard deviation of the actions σ_a to make them comparable across the action dimensions. The RMSE is then computed across all features j and actions a.

This approach is tested on the variety of simulated control tasks with multi-dimensional, continuous observation and action spaces from Sect. 3. We run tests with increasing number of background data points $N_b \in \{1, 5, 10, 20, 50, 100, 1000\}$ and two different selection methods (sampling or KMeans-clustering). For better statistics each run is repeated five times.

4.3 Robustness of KernelSHAP

Our results show a remarkable robustness of KernelSHAP. Visually inspecting the resulting dependency plots (Fig. 2), we note that only the values based on 1 background sample stand out. Differences between other distributions are often hardly noticeable to the naked eye. The RMSE of the respective set of SHAP values w.r.t. the one based on the largest set of background data ($N_b = 1000$) gives a quantitative measurement. The linear arrangement of measurements in the log-log-plots of Fig. 3 suggests a power-law-relationship between the number of background samples and the error. The parameters of this law seem to be rather consistent across different RL-tasks. This evaluation also shows, that clustering leads to noticeable smaller errors than sampling. The results show little variance across multiple repetitions, as indicated by the very small error bars, and the qualitative results are consistent across all investigated benchmark tasks.

For all environments, when using selection method *sampling*, the power-law relationship follows a $1/\sqrt{N_b}$ power-law remarkably well. This can be understood from the law of large numbers: If a measurement with i.i.d. fluctuations is repeated N times, the error in all averages shrinks by a factor of $1/\sqrt{N}$. The number N_b of background data puts an upper bound on the number of i.i.d. samples. With *clustering* we get a higher power law because the cluster centers are better representatives of the underlying data distribution.

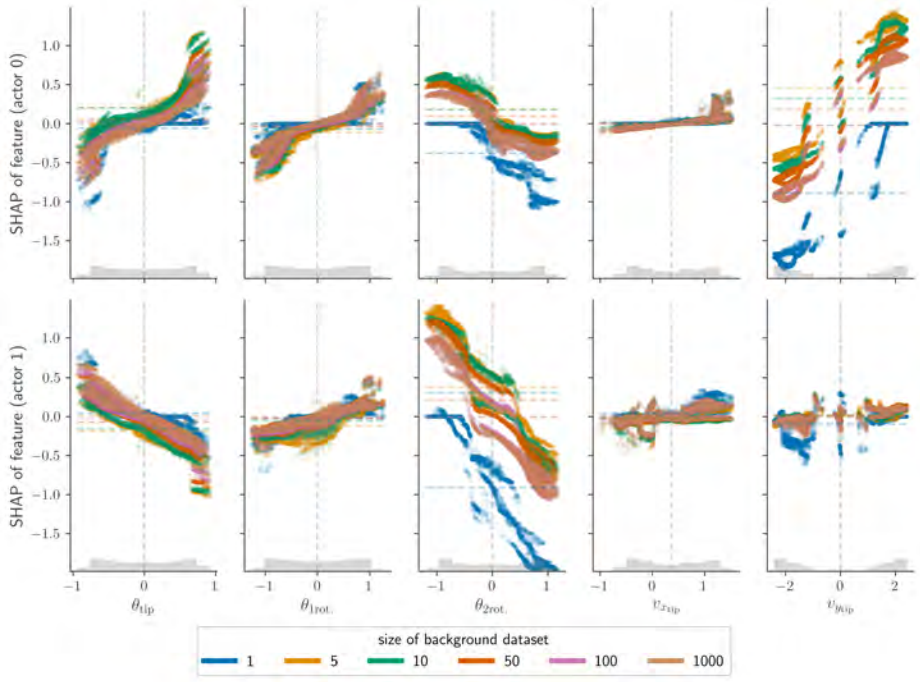

Fig. 2. SHAP dependency plots of 5 out of the 8 features of Swimmer, computed using differently sized background data (encoded by color) obtained by sampling. Colored dotted lines signify the average SHAP value of each feature. The histograms on the abscissa show the features' distribution.

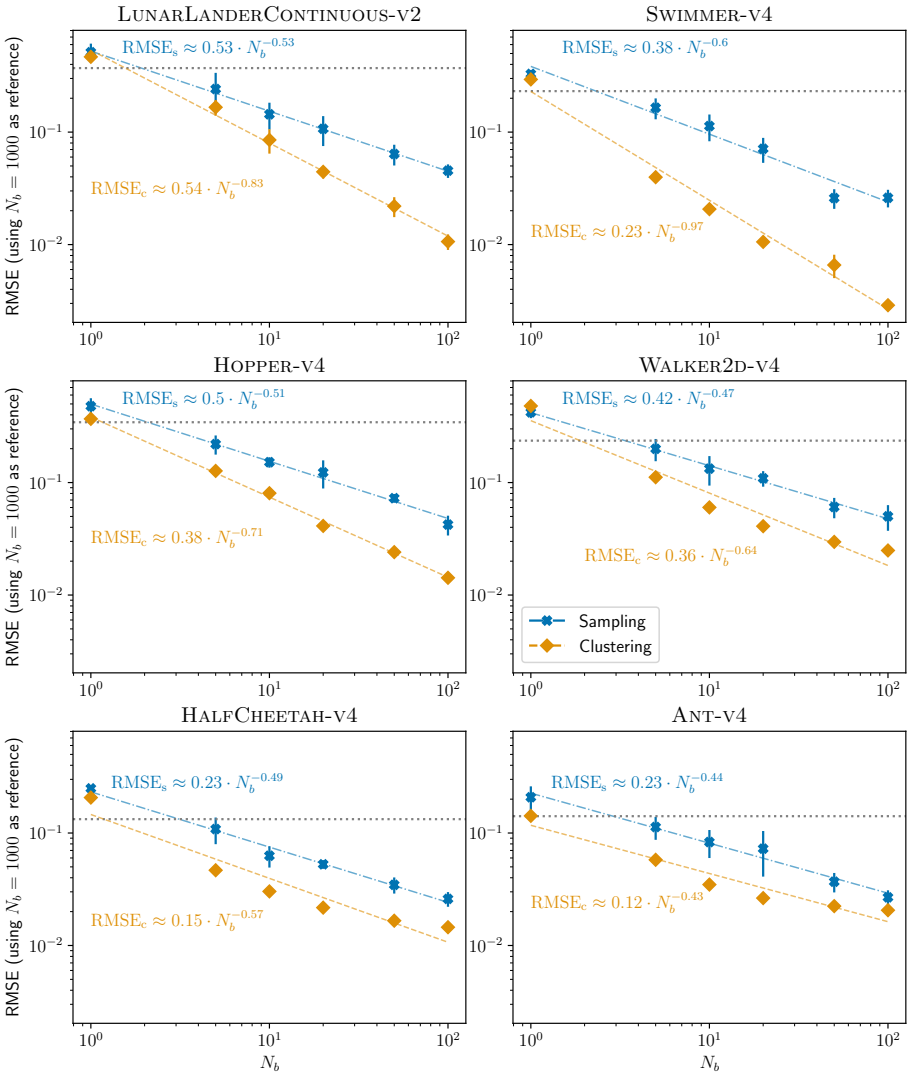

Fig. 3. RMSE of SHAP values computed with different numbers N_b of background samples (using SHAP computed on the largest sample $N_b = 1000$ as reference) and two dataset reduction methods: sampling (dash-dotted blue line) and clustering (dashed orange line). The error bars mark $\pm 1\sigma$ of five repetitions. The dotted horizontal line marks the threshold described in the main text.

The horizontal line in each plot gives an idea of the upper tolerable limit for the error. As a measure, we here use the standard deviation of the SHAP values, normalized by σ_a. If the error is substantially smaller than the horizontal line, the given number N_b of background data should be sufficient. Figure 3 shows that this is the case for $N_b \geq 5$.

The rapidly decreasing gains in precision when adding more samples to the background dataset is especially relevant when put in context of increased computational costs. Figure 4 shows the process time for computing SHAP values of $N_e = 1000$ samples based on differently sized background data. When searching for a trade-off between precision and computation time, this increase in compute should carefully be taken into consideration, especially in conjunction with the rapidly decreasing error. Since the overhead of KMeans clustering is negligible compared to the computational costs of SHAP value approximation (tens of seconds vs hours), this method of background data reduction is generally preferable, given the smaller errors.

In general, the outcome of this experiment is quite surprising: Even for the environments with many observation and action dimensions (meaning that the estimation of marginal distributions for the "missing" features requires high-dimensional integrals), a relatively small number of samples or cluster centers is sufficient to reach reasonable accuracy.

5 Experiment 2: Empirical Evaluation of SHAP-based Feature Importance

SHAP is a method rooted in game-theory we use for attributing a certain action difference to the single elements of the observation vector (features). The action difference is the difference between the actual action of the agent, given an observation, and the average action. Based on this attribution, the importance of the single features can be defined. In the following experiment, we investigate how the computed global feature importance correlates to the feature importance in the RL task.

5.1 Generalized Feature Importance

Although the SHAP value for feature j and action a measures how this specific feature influences this specific action, it is not a priori clear whether the importance for a single action implies a similar importance for the RL performance of the agent (overall return from a RL episode). Molnar [7, Chap. 9.6.5] suggests establishing a connection between SHAP values for a dataset with N instances and feature importance as

$$\mathrm{FI}_{j,a} = \frac{1}{N} \sum_{n=1}^{N} |\phi_{j,a}^{(n)}|, \tag{3}$$

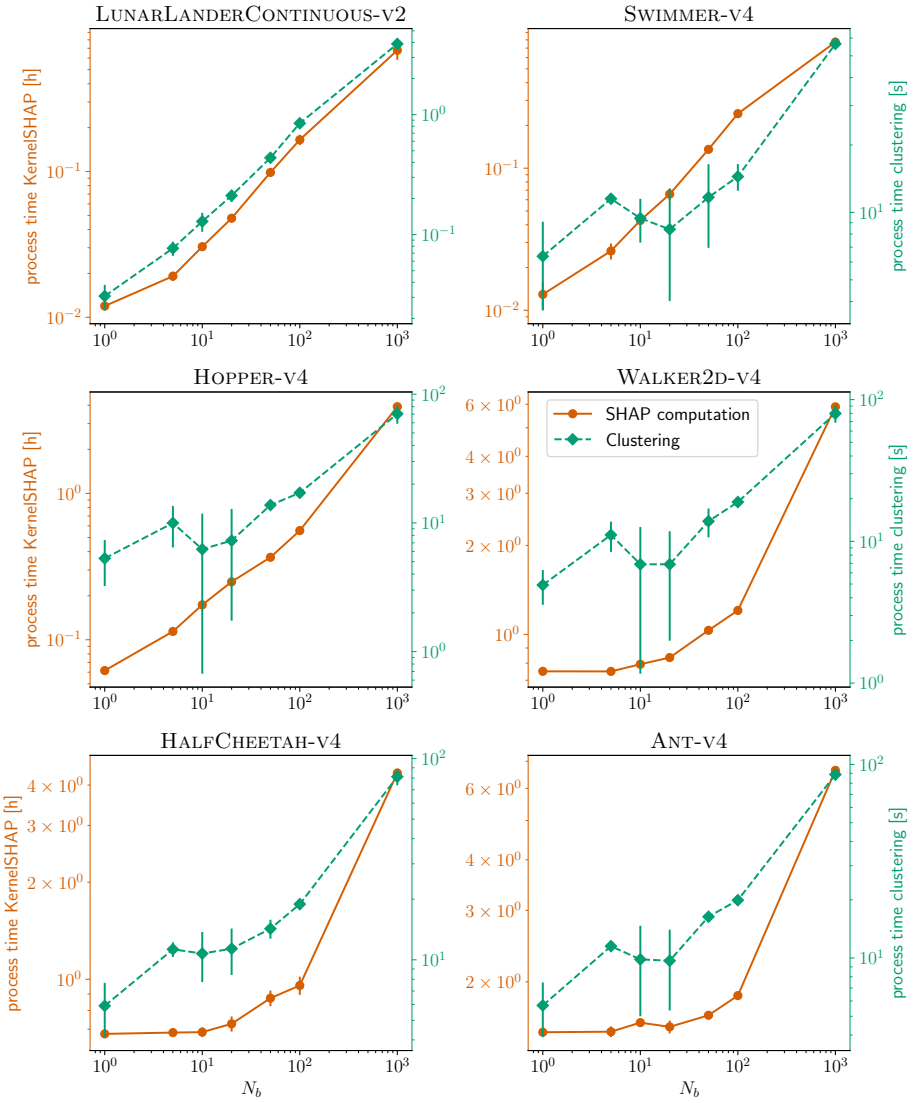

Fig. 4. Computational costs ($\mu \pm 1\sigma$ across five repetitions) of KernelSHAP computing SHAP values of $N_e = 1000$ samples, based on differently sized background dataset N_b and the computational overhead of clustering with N_b cluster centers. Note that the costs of KernelSHAP shown on the left y-axis are measured in hours, while the costs for clustering shown on the right y-axis are measured in seconds. Thus, clustering costs are generally negligible. The costs for sampling instead of clustering are in the order of magnitude 10^{-4} s to 10^{-3} s and therefore completely negligible.

given by averaging the absolute SHAP values $|\phi_j|$ of a feature j over the instances. Since in more complex RL tasks the agent has to perform multidimensional actions, the definition of a global feature importance is not obvious. The importance of a single feature can be very different for different elements of the action vector, as can be seen for example in Fig. 2, where the feature $v_{y_{\text{tip}}}$ has the biggest impact on the first dimension of the action vector, while having only a marginal effect on the second one (rather flat distribution in the SHAP dependency plot). In addition, the different elements a_i of the action vector $\mathbf{a} \in \mathbb{R}^A$ can have very different ranges. This would have a strong impact on the associated SHAP values according to their definition based on the difference between the actual action and average action. To mitigate these problems, normalizing the feature importances for each dimension by the standard deviation of the actions along the specific dimension and averaging them seems a natural extension of Eq. (3) to multidimensional actions. We therefore generalize feature importance to the case of multidimensional actions by defining

$$\mathrm{FI}_j = \frac{1}{A} \sum_{i=1}^{A} \frac{\mathrm{FI}_{j,a_i}}{\sigma(a_i)} \tag{4}$$

for a task with A action dimensions. Is the feature importance FI_j correlated to the change in agent performance (cumulative reward) when feature j is removed from the observations? This research question is investigated in the following experiment.

5.2 Experimental Setup

We assess the research question, using the six different RL tasks described in Table 1, with the following procedure:

1. The agents are evaluated in their respective environment for 10 episodes.
2. A KernelSHAP explainer is set up using $N_b = 1000$ background data samples.
3. SHAP values are computed for $N_e = 1000$ samples drawn randomly from 10 different evaluation episodes.
4. The feature importances FI_j are computed according to Eq (4).
5. The agent is evaluated again for 100 episodes, this time being "blinded" w.r.t. observation j. Observation j is substituted by its average value of the evaluation samples from step 3. The resulting average return of the agent blinded w.r.t. observation j is denoted by $\overline{R}_{\setminus j}$.

5.3 Performance Drop vs. Feature Importance

Plotting $\overline{R}_{\setminus j}$, the performance of the agent blinded w.r.t. observable j, against feature importance FI_j in Fig. 5 shows the general correlation between the two measurements. Table 2 contains the results in succinct form.

Table 2. Summary of the correlation between SHAP feature importance and performance drop of partially-blinded agent

Environment	Pearson r	R^2
LunarLanderCont.-v2	−0.829	0.687
Swimmer-v4	−0.909	0.826
Hopper-v4	−0.0364	0.001 33
Walker2d-v4	−0.687	0.472
HalfCheetah-v4	−0.530	0.281
Ant-v4	−0.557	0.310

The results show, with the notable exception of Hopper, a correlation between a feature's importance in predicting an action and the agent's performance when that feature is absent. While this correlation is especially prominent in the "simpler" environments Swimmer and LunarLanderContinuous, more complex environments show a weaker correlation: a general trend is still visible, but there are many examples where features with lower FI lead to stronger decreases in performance and vice versa. The results suggest interpreting the FI as computed by SHAP with caution.

The environment Hopper stands out, as apparently every single feature is crucial to the agent's success. Omitting any feature leads to almost the same drastic decrease in performance. The feature importance therefore has almost no correlation with $\overline{R}_{\setminus j}$. To investigate whether the crucial role of every single observable is a property of this trained agent or an intrinsic property of the environment, training was repeated with partially-blinded TQC agents. During training and evaluation of each of these agents, one observable is set to zero. Since such experiments require training multiple agents ex novo, and are therefore rather time-consuming, we performed this experiment only for the outstanding case of Hopper. Figure 6 shows the relation between FI_j, $\overline{R}_{\setminus j}$, and $\overline{R}_{\setminus j}^{(\mathrm{retrain})}$, the performance of agents newly trained without the specific feature j. When training new agents partially blinded ab initio w.r.t. one observable, the performance increases notably. While for no feature the performance reaches the fully-observable threshold, in most cases (except for θ_{thigh}, θ_{torso}, and ω_{foot}) the performance lies at least around 3000. The feature importance of the agent, trained with all observables accessible, cannot be expected to correlate with the performance of newly-trained, partially-blinded agents.

By visually inspecting episode renderings[5] corresponding to low feature importance and low performance for the partially-blinded agent (points in the lower left part of the plots in Fig. 5), one common behavior emerges: These episodes are always characterized by catastrophic failure of the agent falling over (e.g., the ant falling on its back) or by premature termination due to reaching a state

[5] The renderings of the five best and five worst episodes can be accessed on the Github repository https://github.com/RaphaelEngelhardt/xai_shap4rl.

defined by the environment as "unhealthy" (one or more observables leaving a predefined range). The hidden observables are apparently crucial for keeping the simulated robot in a safe state, even if they have been assigned a relatively low feature importance.

It should also be noted, that the process of omitting a feature can usually not be applied iteratively: Omitting several of the features *together*, that hidden individually have little impact on performance, often leads to a complete breakdown of performance.

It is also worth noting that omitting some features has a greater impact on the *consistency* of the agent's performance than others. This is most prominent in the case of Swimmer, where omitting a feature generally leads to a very consistent performance. If the agent is blinded w.r.t. feature θ_{tip} instead, it can play either very successful or very unsuccessful episodes. This is shown in Fig. 7 where the distributions of returns are represented by violin plots and point clouds. Omitting feature θ_{tip} leads to a strongly bimodal distribution. This can be explained with a peculiarity of the Swimmer environment: For coordinated movement, the rear rotor has to make a movement to the opposite side of the front tip. In absence of the front tip angle θ_{tip}, the agent still observes the front tip's angular velocity ω_{tip}, which assumes the value 0 at either side, but the agent has to guess which side it is. If it guesses correctly, it receives a good return, otherwise the movement comes to a complete standstill.[6]

6 Interpretation of SHAP Time Dependency in RL

An advantage of using SHAP as XAI method in RL is that SHAP assigns a contribution of each feature to each action dimension at each timestep of an episode. This provides a rich dataset which can be used to gain insights. As an example, Fig. 8 shows the time dependencies of the SHAP values for each feature and action dimension (the two rotors in case of Swimmer). This has some similarity to the RL-SHAP diagram introduced in [18].

We see for example that $v_{y_{\text{tip}}}$ has a large contribution to rotor 0, but not to rotor 1. Likewise, $\theta_{2\text{rot.}}$ has a large contribution to rotor 1, but a much smaller contribution to rotor 0. It might be tempting to relate steep falls or rises in the SHAP value (e.g. for $v_{y_{\text{tip}}}$ around timesteps 10 and 25, respectively) to drastic changes in importance, that is, $v_{y_{\text{tip}}}$ initially appears to have a major positive effect on rotor 0 up to timestep 10 and thereafter a major negative effect. However, with this example, we would like to point out that such an interpretation is wrong. A SHAP value has always to be seen in connection with the ML prediction that it models, in our case the variable *action* 0 shown in the first row and column of Fig. 8. If this target value exhibits a jump, the overall SHAP values will also show the same jump. As a consequence, a high-contributing SHAP value needs to have a similar jump. The right interpretation

[6] Example videos of the best and the worst Swimmer episodes are accessible on the repository https://github.com/RaphaelEngelhardt/xai_shap4rl as supplementary material.

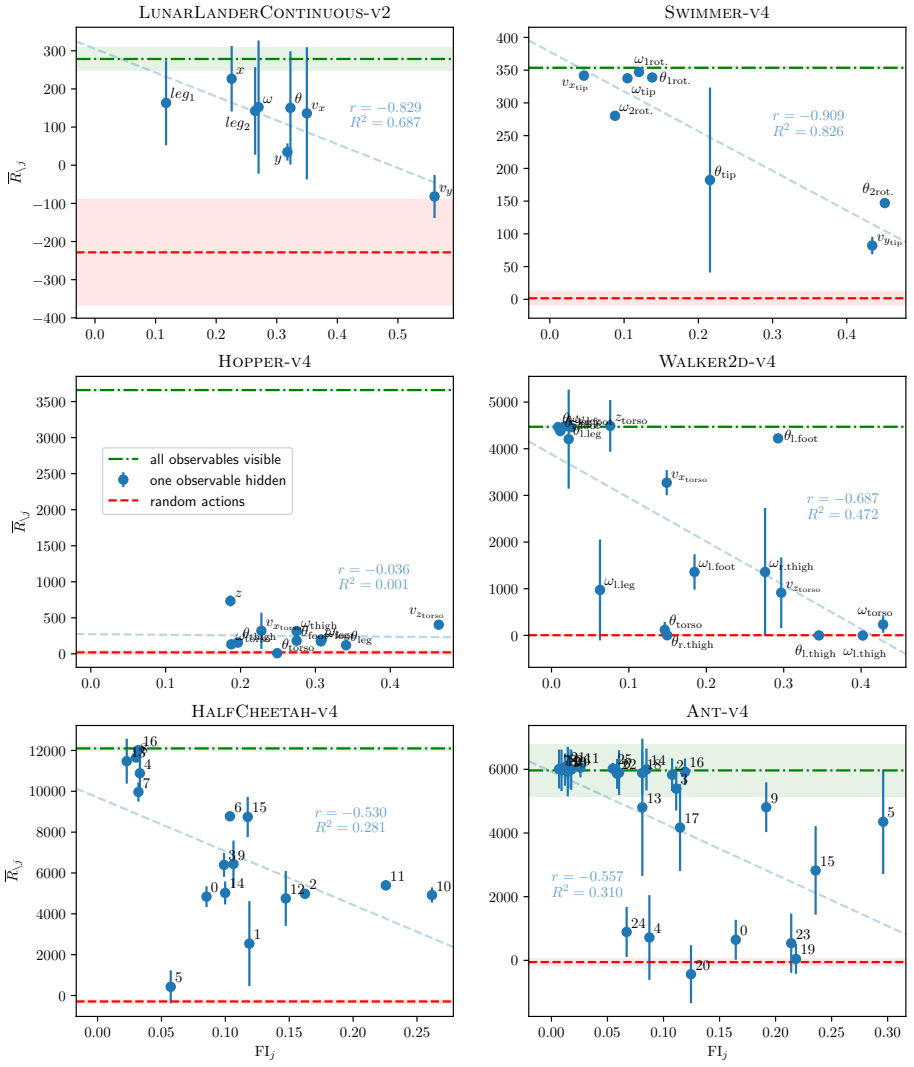

Fig. 5. Relation between SHAP-based feature importance (abscissa) of an observable and cumulative reward of an agent blinded w.r.t. said observable (ordinate). Error bars show $\pm 1\sigma$ of the 100 evaluation episodes. The dash-dotted green horizontal line denotes the average performance of the agent when all observables are fully accessible, the green shaded area shows $\pm 1\sigma$ over the 100 evaluation episodes. Analogously, the dashed red line and area mark the lower baseline, i.e. the return of an agent that acts randomly at each timestep. The dashed blue line is a linear fit to the data with correlation coefficient r and coefficient of determination R^2.

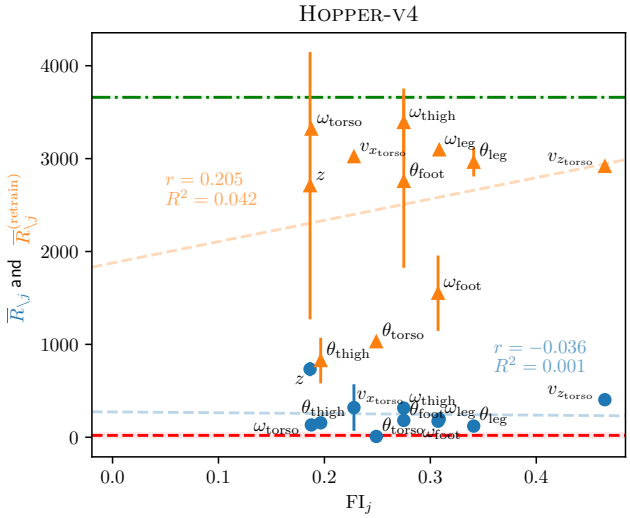

Fig. 6. Version of Fig. 5 (Hopper), but with extended results: The orange triangles show the achievable returns when retraining partially-blinded agents from scratch.

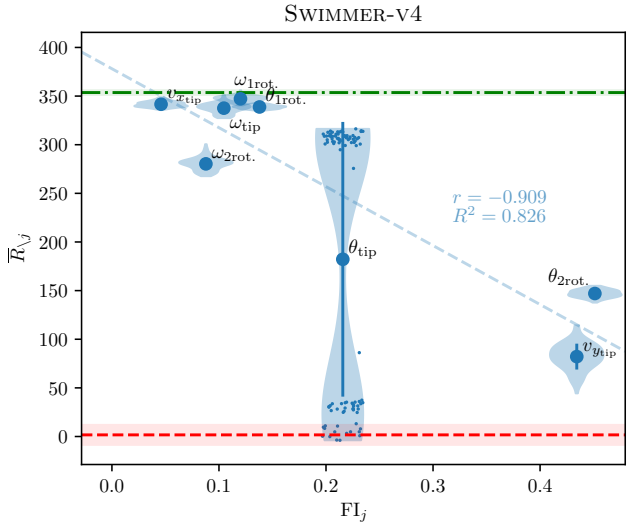

Fig. 7. Version of Fig. 5 (Swimmer) with added distributions of returns. While the distributions are generally centered, leading to small standard deviations, omission of θ_{tip} leads to a bimodal distribution.

is: Whatever the target value of *action* 0 is, feature $v_{y_{\text{tip}}}$ has a large contribution to it of the *same* sign.

7 Conclusion and Outlook

This work analyzes the reliability of SHAP values as a XAI method for complex RL environments with multidimensional actions. A positive aspect of SHAP is its applicability as XAI method also in the case of multidimensional actions, whereas interpretable DTs are hard or impossible to build in such cases.

We have examined the SHAP accuracy as a function of background data size and found it to be surprisingly robust even if only a small size is used in order to reduce computational costs. We found KMeans clustering to be the preferable background data selection method.

Furthermore, we have generalized the SHAP-based feature importance to RL of multidimensional actions. While the SHAP value measures the contribution of a feature to a specific action, the feature importance expresses the importance of the feature in general, regardless of the action dimension. Given this generalized feature importance, we have investigated how well this importance is correlated to the agent's performance (the return). We showed that often there is a clear correlation, while also exceptions exist, most notably in the case of Hopper, where every left-out feature leads to drastic performance breakdowns, irrespective of whether it had high or low feature importance.

Currently, we can only point out these two distinctive cases; finding the reason for these distinct behaviors is left for future research. A possible reason might be SHAP's inability to handle interactions between features. Inspecting the cases with surprising breakdown, we can speculate that this happens more likely for unstable environments with a higher sudden-failure probability (Hopper, can fall down, irreversible) than for more stable environments (Swimmer, cannot fall).

We believe that these findings make an important contribution to the reliability of SHAP-based explainability in RL. The results presented in this work also contain useful insights for XAI practitioners in RL. In the future, we plan to investigate the reasons why SHAP-based importance sometimes does not correlate with agent performance. Furthermore, we plan to examine alternative value functions for SHAP, e.g. episode returns as outlined in Sect. 1.2.

Acknowledgments. This research was supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

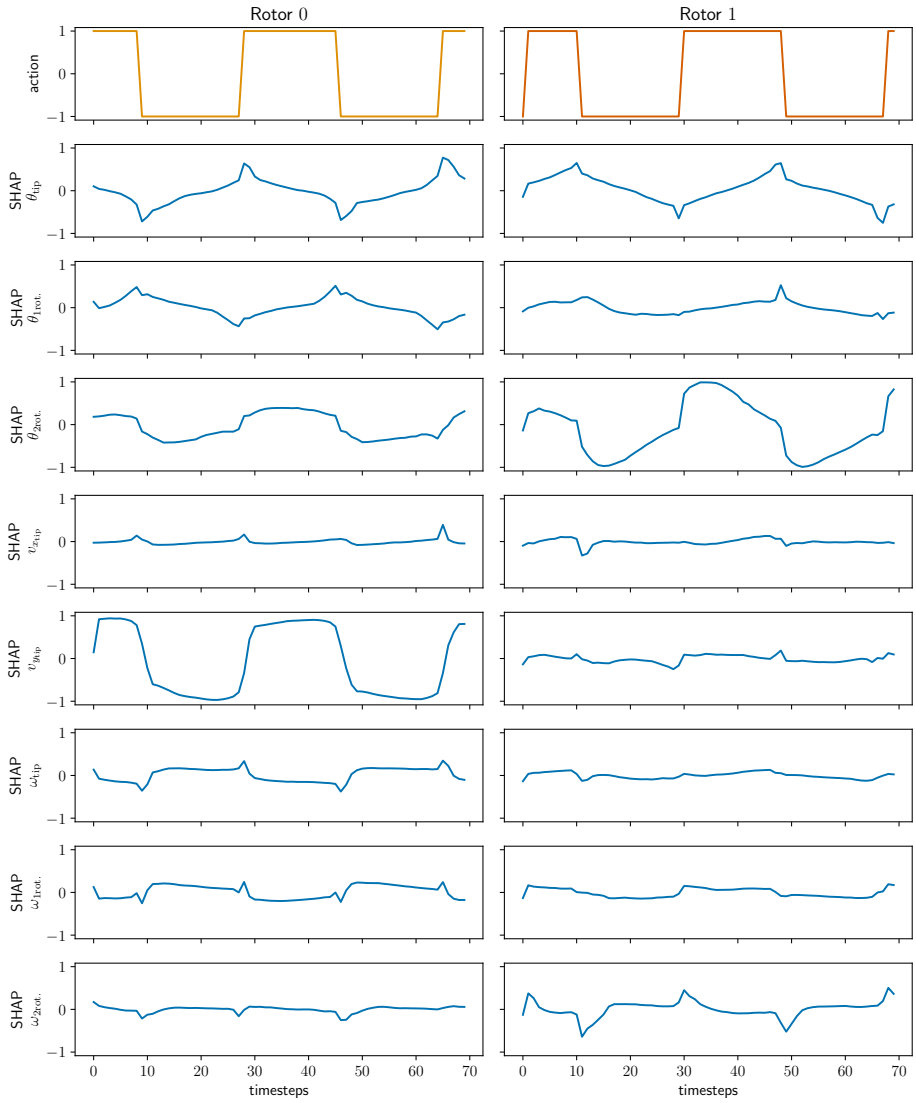

Fig. 8. Time series of roughly two oscillations of Swimmer. Shown is the evolution of actions (first row) and SHAP values of the eight observables (rows $2-9$) for the two rotors (columns).

References

1. Das, A., Rad, P.: Opportunities and challenges in explainable artificial intelligence (XAI): A survey. arXiv preprint (2020). https://doi.org/10.48550/arXiv.2006.11371
2. Vouros, G.A.: Explainable deep reinforcement learning: State of the art and challenges. ACM Comput. Surv. **55**(5) (2022). https://doi.org/10.1145/3527448
3. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc., Red Hook, NY, USA (2017)
4. Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games (AM-28), Volume II. pp. 307–318. Princeton University Press, Princeton, USA (1953). https://doi.org/10.1515/9781400881970-018
5. Heuillet, A., Couthouis, F., Díaz-Rodríguez, N.: Collective explainable ai: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values. IEEE Computational Intelligence Magazine **17**(1), 59–71 (2022). https://doi.org/10.1109/MCI.2021.3129959
6. Holzinger, A.: From machine learning to explainable ai. In: 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA). pp. 55–66. IEEE (2018). https://doi.org/10.1109/DISA.2018.8490530
7. Molnar, C.: Interpretable Machine Learning. 2 edn. (2022), https://christophm.github.io/interpretable-ml-book, last accessed 2024/03/15
8. Nauta, M., Trienes, J., Pathak, S., Nguyen, E., Peters, M., Schmitt, Y., Schlötterer, J., van Keulen, M., Seifert, C.: From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. ACM Comput. Surv. **55**(13s) (2023). https://doi.org/10.1145/3583558
9. Samek, W., Montavon, G., Lapuschkin, S., Anders, C.J., Müller, K.R.: Explaining deep neural networks and beyond: A review of methods and applications. Proceedings of the IEEE **109**(3), 247–278 (2021). https://doi.org/10.1109/JPROC.2021.3060483
10. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939778
11. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015). https://doi.org/10.1371/journal.pone.0130140
12. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019). https://doi.org/10.1038/s42256-019-0048-x
13. Hickling, T., Zenati, A., Aouf, N., Spencer, P.: Explainability in deep reinforcement learning: A review into current methods and applications. ACM Comput. Surv. **56**(5) (2023). https://doi.org/10.1145/3623377
14. Wells, L., Bednarz, T.: Explainable AI and reinforcement learning – a systematic review of current approaches and trends. Frontiers in Artificial Intelligence **4**, 550030 (2021). https://doi.org/10.3389/frai.2021.550030

15. Dhebar, Y., Deb, K., Nageshrao, S., Zhu, L., Filev, D.: Toward interpretable-ai policies using evolutionary nonlinear decision trees for discrete-action systems. IEEE Transactions on Cybernetics **54**(1), 50–62 (2024). https://doi.org/10.1109/TCYB.2022.3180664
16. Ding, Z., Hernandez-Leal, P., Ding, G.W., Li, C., Huang, R.: CDT: Cascading decision trees for explainable reinforcement learning. arXiv preprint (2020). https://doi.org/10.48550/arXiv.2011.07553
17. Liu, G., Sun, X., Schulte, O., Poupart, P.: Learning tree interpretation from object representation for deep reinforcement learning. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 19622–19636. Curran Associates, Inc., Red Hook, NY, USA (2021)
18. Liessner, R., Dohmen, J., Wiering, M.: Explainable reinforcement learning for longitudinal control. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence ICAART. vol. 2, pp. 874–881. SciTePress, Setúbal, Portugal (2021). https://doi.org/10.5220/0010256208740881
19. Rizzo, S.G., Vantini, G., Chawla, S.: Reinforcement learning with explainability for traffic signal control. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). pp. 3567–3572. IEEE (2019). https://doi.org/10.1109/ITSC.2019.8917519
20. Zhang, K., Zhang, J., Xu, P.D., Gao, T., Gao, D.W.: Explainable ai in deep reinforcement learning models for power system emergency control. IEEE Transactions on Computational Social Systems **9**(2), 419–427 (2022). https://doi.org/10.1109/TCSS.2021.3096824
21. Engelhardt, R.C., Oedingen, M., Lange, M., Wiskott, L., Konen, W.: Iterative oblique decision trees deliver explainable rl models. Algorithms **16**(6), 282 (2023). https://doi.org/10.3390/a16060282
22. Custode, L.L.: Evolutionary Optimization of Decision Trees for Interpretable Reinforcement Learning. Ph.D. thesis, Università degli studi di Trento (2023). https://doi.org/10.15168/11572_375447
23. Sequeira, P., Gervasio, M.: Ixdrl: A novel explainable deep reinforcement learning toolkit based on analyses of interestingness. In: Longo, L. (ed.) Explainable Artificial Intelligence. pp. 373–396. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-44064-9_20
24. Towers, M., Terry, J.K., Kwiatkowski, A., Balis, J.U., Cola, G.d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Shen, A.T.J., Younis, O.G.: Gymnasium (2023). https://doi.org/10.5281/zenodo.8127026
25. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021)
26. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1587–1596. PMLR (2018)
27. Kuznetsov, A., Shvechikov, P., Grishin, A., Vetrov, D.: Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5556–5566. PMLR (2020)
28. Raffin, A.: RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo, last accessed 2024/03/15