Evaluation of Intrinsic Explainable Reinforcement Learning in Remote Electrical Tilt Optimization

Franco Ruggeri[1,2], Ahmad Terra[1,2], Rafia Inam[1,2], and Karl H. Johansson[2]

[1] Ericsson Research, Stockholm, Sweden
{franco.ruggeri, ahmad.terra, rafia.inam}@ericsson.com
[2] KTH Royal Institute of Technology, Stockholm, Sweden
{fruggeri, terra, raina, kallej}@kth.se

Abstract. This paper empirically evaluates two intrinsic Explainable Reinforcement Learning (XRL) algorithms on the Remote Electrical Tilt (RET) optimization problem. In RET optimization, where the electrical downtilt of the antennas in a cellular network is controlled to optimize coverage and capacity, explanations are necessary to understand the reasons behind a specific adjustment. First, we formulate the RET problem in the Reinforcement Learning (RL) framework and describe how we apply Decomposed Reward Deep Q-Network (drDQN) and Linear Model U-Tree (LMUT), which are two state-of-the-art XRL algorithms. Then, we train and test such agents in a realistic simulated network. Our results highlight both advantages and disadvantages of the algorithms. DrDQN provides intuitive contrastive local explanations for the agent's decisions to adjust the downtilt of an antenna, while achieving the same performance as the original DQN algorithm. LMUT reaches high performance while employing a fully transparent linear model capable of generating both local and global explanations. On the other hand, drDQN adds a constraint on the reward design that might be problematic for the specification of the objective, whereas LMUT could generate misleading global feature importance and needs additional developments to provide more user-interpretable local explanations.

Keywords: Explainable Reinforcement Learning, Reinforcement Learning, Artificial Intelligence, Remote Electrical Tilt Optimization, Cellular Networks

1 Introduction

In the last two decades, Reinforcement Learning (RL) has received considerable attention from the telecommunication domain due to its outstanding performance in many applications. Traditional rule-based techniques have been outperformed by RL algorithms in a variety of complex telecommunication use cases, such as edge computing [1], radio resource management [2,3], and Remote

Electrical Tilt (RET) optimization [4–7]. In the RET use case, RL can be used to control the electrical downtilt of multiple antennas in a cellular network in order to optimize specific Key Performance Indicators (KPIs).

Despite the superior performance, state-of-the-art RL algorithms lack explainability and transparency [8, 9], i.e., it is often hard for humans to understand why an RL agent makes a specific decision. This lack of explainability can hinder the adoption of RL for industrial domains [10–12], as users are less likely to use an application that they do not trust, and black-box models may raise ethical issues. In the RET use case, the decision of an RL agent to adjust the downtilt of an antenna influences the experienced communication bandwidth of some users. Therefore, it is desirable to understand the reason behind each decision. Recently, researchers have focused on the development of Explainable Reinforcement Learning (XRL) methods to enhance RL explainability [8,9]. Such methods inspect different parts of the RL framework – namely agent's model, rewards, and environment state – to extract explanations and provide additional insights to the system designer.

The main motivation for this paper is that most XRL methods have not been extensively evaluated in real-world applications, including the telecommunication domain. XRL is, indeed, still a research area in its infancy. A search of the literature has revealed only few studies that have applied XRL to real-world problems [13–16], and many promising XRL methods have not been covered by such studies. Due to this scarcity of results, it is unclear which method best suits a particular problem and what advantages and disadvantages each method provides. There is a need for systematic comparisons of XRL methods that can provide guidelines for the application of XRL.

The purpose of this paper is to empirically evaluate on the RET optimization problem two promising XRL algorithms: Decomposed Reward Deep Q-Network (drDQN) [17] and Linear Model U-Tree (LMUT) [18]. For doing so, we first formulate the RET optimization problem as an RL problem suitable to apply both drDQN and LMUT and describe how to apply them to such a problem. In particular, we consider a simplified use case where only one of the antennas needs to be controlled, which allows checking the explanations provided by the two XRL methods. Second, we evaluate drDQN and LMUT in a simulated Radio Access Network (RAN). Since these two XRL algorithms are intrinsic, i.e., they modify existing RL algorithms to make them more transparent and provide explanations by showing their internal mechanism, the performance may be degraded compared to the original RL algorithm. Thus, the evaluation considers both performance and explainability. The results are discussed thoroughly to highlight the advantages and disadvantages of drDQN and LMUT.

Contributions: The main contributions of this paper are: (i) to present, to the best of our knowledge, the first study that empirically compares drDQN and LMUT on the RET use case; (ii) to formulate the RET optimization problem as an RL problem suitable to apply drDQN and LMUT, as well as to benchmark other XRL methods in the future; and (iii) to present qualitative results of the explanations achievable using drDQN and LMUT in the RET use case.

Outline: Section 2 presents the related work. Section 3 formulates the RET optimization problem as an RL problem suitable to apply the selected XRL methods. Section 4 briefly describes drDQN and LMUT with focus on how to apply them to the RET problem. Section 5 discusses the simulation setup and the results of the simulations. Section 6 summarizes our findings and the ideas for future work.

2 Related Work

The relevant literature for this paper comprises two research areas: RL in RET optimization and XRL. Previous studies have found that RL approaches can outperform traditional rule-based policies hand-crafted by domain experts in the RET use case [4–6]. In particular, Vannella et al. [5] demonstrated how safe RL can outperform rule-based policies in terms of both performance and safety by training the agent offline on real network data collected under the rule-based policy. In [6], Vannella et al. showed that an off-policy contextual bandit algorithm consistently outperforms the rule-based policy in RAN simulations. Buenestado et al. [4] found that RL can tune several parameters of the antenna, among which the downtilt, more efficiently than traditional algorithms.

The literature on XRL has developed several promising methods. According to [8, 9], these methods can be classified into post-hoc and intrinsic methods. Whereas post-hoc methods rely on an auxiliary model to extract explanations from the complex agent, intrinsic (or transparent) methods are RL algorithms that provide explanations by design. Another criterion is given by the scope of the explanation: a global explanation summarizes the general behavior of the agent, while a local explanation refers to a specific agent's decision. Among the intrinsic methods, Liu et al. [18] proposed LMUT as a transparent model to learn mimicking a black-box agent. LMUT is a combination of regression trees and linear regressions. It can provide both global and local explanations. Juozapaitis et al. [17] modified existing RL algorithms by decomposing the reward into a sum of reward components and, consequently, decomposed the agent's model to learn separate value functions while still optimizing the total reward. This idea, known as reward decomposition, was applied, for example, to Deep Q-Network (DQN) [19] to generate drDQN, which is capable of providing contrastive explanations with lightweight post-processing consisting of differences of Q-components. Starting from drDQN, Terra et al. [16] combined the reward decomposition idea with SHAP [20], an established feature attribution method from Explainable AI (XAI), to find correlations between inputs and outputs of the Q-network. The new method, called Both Ends Explainations for Reinforcement Learning (BEERL), was implemented on the RET use case. Regarding post-hoc methods, Hayes and Shah [21] devised an algorithm-agnostic framework to summarize the agent's policy (i.e., state-to-action mappings) in natural language by using user-interpretable communicable predicates. Van der Waa et al. [22] modified the idea in [21] to explain actions in terms of expected consequences, rather than correlations between state and action, through user-

interpretable outcomes describing rewards. More specifically, they proposed to compute the expected outcomes of an action by simulating future steps through a learned model of the environment. Greydanus et al. [23], instead, considered agents learning directly from raw images, in which case the XRL methods mentioned so far can hardly be successful. The authors used perturbation-based saliency maps to highlight pixels with a high impact on the action selection. Iucci et al. [13] applied [17] and [21] to a human-robot collaboration scenario and showed how having both local and global explanations can help to have a complete overview of the agent's behavior.

The existing literature on RL in RET optimization focuses on performance but neglects explainability. An exception is [16], which applied XRL to the RET use case. Inspired by it, we design the reward as a linear combination of KPIs in order to apply drDQN. However, while [16] focuses on developing a new XRL method and implementing it on the RET problem, our work aims to *assess and compare* two existing XRL algorithms, drDQN [17] and LMUT [18], to find their advantages and drawbacks. To be able to assess explanations, we consider a RET problem where only one antenna is controlled. This simple setup serves as a benchmark for XRL and can be used in future work to evaluate other XRL methods. Among the available XRL methods, we decided to restrict the work to *intrinsic* XRL, as it does not suffer from potential inaccuracies of the auxiliary model present in post-hoc methods. Furthermore, to the best of our knowledge, no previous study has applied LMUT to a telecommunication problem.

3 RET optimization

In this section, we describe in detail the RET use case, including the explainability aspects we seek in this work, and formulate it as an RL problem suitable to apply the selected XRL methods.

3.1 System model

We consider a geographical area where B Base Stations (BSs) and U User Equipments (UEs) are deployed. Each BS is equipped with A_b directional antennas. Thus, the set of antennas $\mathcal{A}$ in the cellular network has cardinality $|\mathcal{A}| = B \cdot A_b$. Fig. 1 shows an example with $B = 2$, $A_b = 3$ (three-sectorial BSs), and cells with hexagonal shape. Given a fixed electrical downtilt of the antennas $a_2, ..., a_{|\mathcal{A}|} \in \mathcal{A}$, the problem is to control the electrical downtilt θ of the remaining antenna $a_1 \in \mathcal{A}$, denoted as a hereafter, in order to optimize certain KPIs. The downtilt is defined as the inclination angle of the main lobe of the antenna radiation pattern with respect to its horizontal plane [5] (see Fig. 1).

Two important KPIs for mobile network operators are coverage and capacity, which are conceptually defined as follows. Coverage refers to the area from which a UE can access the cellular network, while capacity refers to the amount of traffic the cellular network can handle simultaneously. Inspired by [16], we use Reference Signal Received Power (RSRP) and Signal-to-Interference-plus-Noise

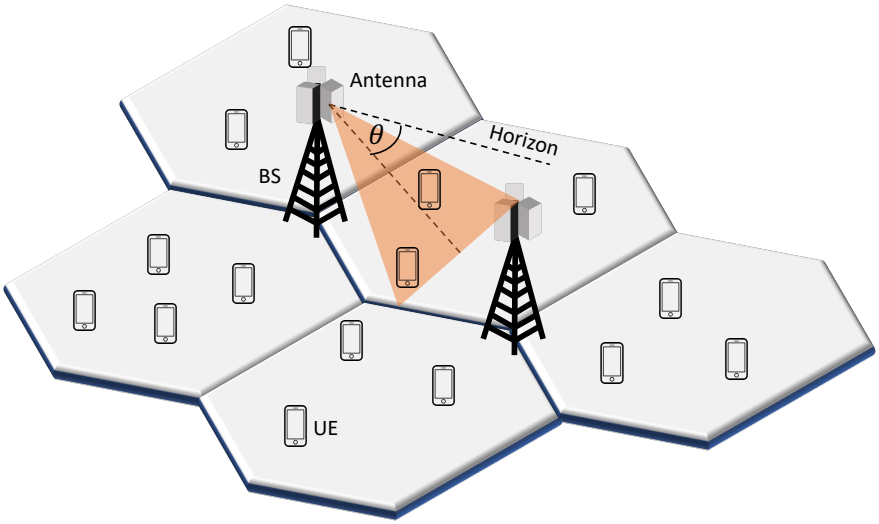


Fig. 1: Abstract representation of the RET optimization problem. There are two BSs, each equipped with 3 directional antennas. The downtilt θ of one antenna needs to be adjusted to optimize coverage and capacity. The downtilt of the other antennas is fixed.

Ratio (SINR)[3] to represent coverage and capacity, respectively. RSRP can be used to assess whether a coordinate in the area of interest is covered by the network, as it considers a reference signal from the antenna that has not been optimized for a specific UE. In contrast, the amount of traffic the network can handle strongly depends on the number of errors and retransmissions and, consequently, on the SINR to each UE. Note that more sophisticated ways to model coverage and capacity have been proposed (e.g., [6, 7]), but are out of the scope of this work.

Explainability is important in the RET use case and represents the focus of this work. When an RL agent adjusts the downtilt θ of the antenna, the performance may degrade for some users and increase for others. Specifically, higher values of the downtilt θ reduce the area covered by the antenna, with the risk of leaving a certain area without coverage , but increase the capacity in the covered area due to a stronger signal. In contrast, smaller values of θ result in a larger area covered but lower capacity due to a weaker signal. Essentially, the problem implies a multi-objective optimization of coverage and capacity with a clear trade-off determined by the downtilt. For these reasons, it is crucial to explain the RL agent's tilt decisions. The specific kind of explanation depends on the XRL method and will be described in Section 4. For example, a local

[3]In this work, SINR refers to the SINR in the data plane.

explanation might highlight which KPI has the most significant impact on an agent's decision.

Before formulating the RET use case as an RL problem, it is worth mentioning why RL is suitable to solve this problem. A cellular network is a dynamical system with intricate dynamics that are difficult to model analytically and accurately. There are plenty of factors to take into account, such as propagation loss of the signal along with fading and shadowing, antenna model, UE mobility, and the BS selection procedure (also known as cell association). To exemplify, when θ decreases (i.e., the signal beam turns up), some UEs previously served by antenna a might be reassigned to another antenna that can serve them better. Such a complexity of the system dynamics is the primary motivation for using model-free RL.

3.2 RL problem formulation

Before applying the XRL algorithms, we need to model the RET optimization problem as an RL problem. We assume a quasi-stationary scenario where the UEs are static for a certain period. Under this assumption, the KPIs vary only if θ varies, i.e., they are functions of θ. We define the average RSRP and SINR per UE as:

$$\overline{RSRP}(\theta) = \frac{1}{U} \sum_{i=1}^{U} RSRP_i(\theta) \tag{1}$$

$$\overline{SINR}(\theta) = \frac{1}{U} \sum_{i=1}^{U} SINR_i(\theta) \tag{2}$$

where $RSRP_i$ and $SINR_i$ are the RSRP and SINR measured by the UE i in dBm and dB, respectively. In the following, we use min-max normalized versions fulfilling $RSRP \in [-1, 1]$ and $SINR \in [-1, 1]$. The objective is to find the optimal downtilt maximizing a linear combination of average RSRP and SINR:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & RSRP(\theta) + w \cdot SINR(\theta) \\ \text{subject to} \quad & \theta_{min} \leq \theta \leq \theta_{max} \end{aligned} \tag{3}$$

where θ_{min} and θ_{max} are preconfigured by domain experts according to safety regulations and specifications of the antenna. The weight w controls the relative importance of RSRP and SINR and is set according to the requirements of the network. This optimization problem can be formulated in the RL framework by defining state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function r. Let $s' = (RSRP', SINR', \theta') \in \mathcal{S}$ be the system state after an action:

$$\mathcal{S} = \{(RSRP, SINR, \theta) \mid (RSRP, SINR) \in [-1, 1]^2, \theta \in [0°, 15°]\} \tag{4}$$

$$\mathcal{A} = \{+\delta, 0, -\delta\}, \ \delta \in \mathbb{R} \tag{5}$$

$$r(s, a, s') = r(s') = RSRP + w \cdot SINR, \ \forall s' \in \mathcal{S} \tag{6}$$

The state includes the normalized versions of RSRP and SINR, and the downtilt θ measured in degrees. The action space contains the three actions corresponding to downtilting (i.e., increasing θ by δ degrees), keeping the same downtilt, and uptilting (i.e., decreasing θ by δ degrees). The reward received for an action depends only on the system state $s' \in \mathcal{S}$ after applying the action. Since the state transitions are deterministic under the assumption of static scenario, the reward $r(s, a)$ associated with an action $a \in \mathcal{A}$ in the state $s \in \mathcal{S}$ is also deterministic. Nevertheless, state transitions and rewards are unknown.

4 XRL algorithms

In this section, we present the selected XRL algorithms, focusing on their application to the RET optimization problem. We discuss only the relevant aspects necessary for the evaluation.

4.1 DrDQN

DrDQN [17] is an intrinsic XRL algorithm that modifies the structure of DQN in order to provide local explanations. Its main idea is to decompose the reward, which normally is a single scalar value, into a vector of semantically meaningful reward components. Following Eq. (6), the reward decomposition for our problem is as follows:

$$r(s, a, s') = r(s') = r_{rsrp}(s') + r_{sinr}(s'),\ \forall s' \in \mathcal{S} \tag{7}$$

$$r_{rsrp}(s') = RSRP' \tag{8}$$

$$r_{sinr}(s') = w \cdot SINR' \tag{9}$$

This reward decomposition is then incorporated into the drDQN model by using one separate Deep Neural Networks (DNNs) for each reward component (differently from DQN, which uses a single DNN). Therefore, in our problem, the structure of drDQN consists of two DNNs, as depicted in Fig. 2. The output of the drDQN model is a Q-vector:

$$\mathbf{Q}(s, a) = [Q_{rsrp}(s, a), Q_{sinr}(s, a)]^T \tag{10}$$

where Q_{rsrp} and Q_{sinr} are the Q-components corresponding to the reward components r_{rsrp} and r_{sinr}. The drDQN agent can be trained in the same way as any other RL algorithm by directly interacting with the system. Like the original DQN, drDQN's objective is to learn the Q-function, with the only difference that the Q-function is a sum of Q-components:

$$Q(s, a) = Q_{rsrp}(s, a) + Q_{sinr}(s, a) \tag{11}$$

By exploiting such a decomposed structure, the agent can provide contrastive explanations by calculating the Reward Difference Explanation (RDX) as fol-

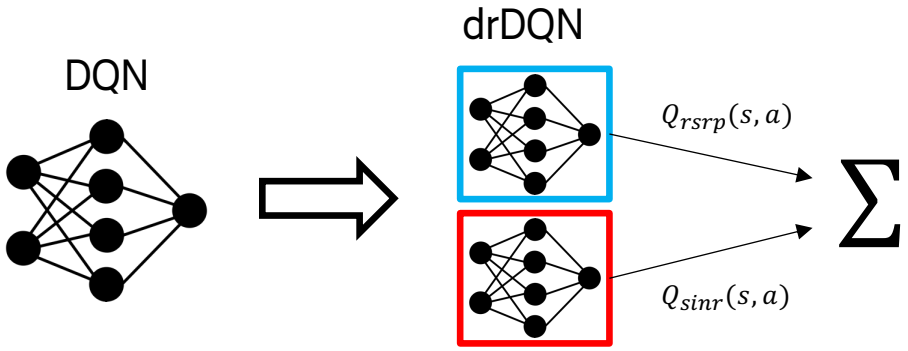


Fig. 2: Structure of drDQN derived from DQN for the RET optimization problem.

lows:

$$\begin{aligned}\boldsymbol{\Delta}(s,a_1,a_2) &= [\Delta_{rsrp}(s,a_1,a_2), \Delta_{sinr}(s,a_1,a_2)]^T \\ &= [Q_{rsrp}(s,a_1) - Q_{rsrp}(s,a_2), Q_{sinr}(s,a_1) - Q_{sinr}(s,a_2)]^T \\ &= \mathbf{Q}(s,a_1) - \mathbf{Q}(s,a_2) \end{aligned} \tag{12}$$

where positive and negative components of $\boldsymbol{\Delta}$ indicate advantages and disadvantages of action a_1 compared to action a_2. For example, $\Delta_{rsrp} > 0$ means that, according to the agent's model, selecting a_1 instead of a_2 benefits the RSRP (or, more specifically, leads to a higher expected discounted r_{rsrp}).

4.2 LMUT

LMUT [18] is an intrinsic XRL method that approximates a black-box model – in our case DQN – with a transparent model in order to provide both global and local explanations. The transparent model, made of $|\mathcal{A}|$ (i.e., number of actions) linear trees[4], is trained in a supervised manner to predict the Q-values produced by DQN. Thus, we train three linear trees corresponding to *downtilting*, *keeping the same tilt*, and *uptilting*. An example of possible LMUT for the RET problem is illustrated in Fig. 3. The training algorithm incrementally (episode by episode, online learning) adds a binary split over a feature every time the model cannot fit the data accurately enough (threshold of Mean Squared Error (MSE)). When a split happens, a leaf becomes a splitting node, and its linear regression is no longer updated. In order to support online learning, the linear regression is trained with Stochastic Gradient Descent (SGD).

[4]A linear tree is an extension of a regression tree where each leaf contains a linear regression instead of a scalar value [18]. It provides more expressiveness than standard regression trees while maintaining transparency.

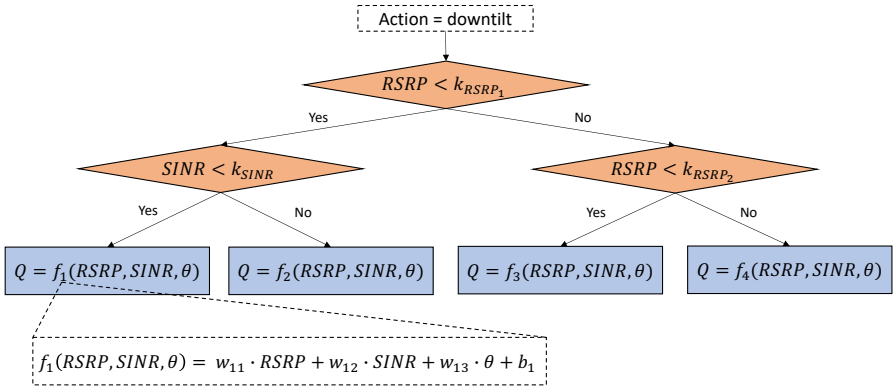


Fig. 3: Example of LMUT for the RET use case. Only the linear tree corresponding to the downtilt action is shown.

LMUT selects the best feature and value for a split according to the criterion of maximum variance reduction of Q-values. In [18], the variance reduction is calculated by considering only the child node with minimum variance, which encourages always splitting on the smallest or greatest value of a feature (left or right child node with one instance and zero variance). The consequence is an unbalanced tree that grows only on one side (left or right) and inevitably becomes large and hard to interpret. We modified this calculation to a weighted sum of the variances in the child nodes, with the weight of each child proportional to the number of instances falling into it. This modification keeps the model small and easy to investigate for explainability, as illustrated in Section 5.

Local explanations LMUT generates local explanations via rule extraction. When the LMUT agent has to select an action in a state $s \in \mathcal{S}$, the Q-value $Q(s, a)$ of each action $a \in \mathcal{A}$ is calculated by using a linear regression in a leaf. Therefore, it is possible to extract rules $A \implies B$, where A is the logical conjunction of the splits from the root to the leaf, and B is the linear regression in the leaf. For example, in the RET problem, a rule can be as follows:

$$RSRP \in (k_{min}, k_{max}) \implies Q(s, a_1) = w_0 + w_1 \cdot RSRP + w_2 \cdot SINR + w_3 \cdot \theta$$

where k_{min} and k_{max} are constants, and w_i are the weights of the triggered linear regression.

Global explanations LMUT provides also a global explanation via feature importance. Let $\mathcal{N}_f$ be the set of splitting nodes in the linear trees using feature f (e.g., RSRP) to split. The global feature importance of f is defined as:

$$\phi(f) = \sum_{n \in \mathcal{N}_s(f)} \phi_n(f) \tag{13}$$

That is, the sum over the feature importance for each splitting node $n \in \mathcal{N}_f$. The feature importance for a single splitting node $n \in \mathcal{N}_f$ is computed as:

$$\phi_n(f) = \left(1 + \frac{|w_{n,f}|^2}{\sum_{j=1}^{F} |w_{n,j}|^2}\right) \cdot \left(var_n - \sum_{c=1}^{C} \frac{I_c}{\sum_{i=1}^{C} I_i} \cdot var_c\right), \forall n \in \mathcal{F}_s \quad (14)$$

where F is the number of features, C is the number of child nodes, and var_i and I_i are the variance and the number of Q-values $Q(s, a)$ stored in node i. The feature importance contains two contributions from the linear regression and the variance reduction. Both contributions come from the splitting nodes.

5 Evaluation

In this section, we describe the simulation setup and present our simulations' results. After that, we discuss the advantages and disadvantages of the evaluated XRL algorithms.

5.1 Simulation setup

We simulated the scenario depicted in Fig. 4 (based on Fig. 1) by using a high-fidelity RAN simulator employing state-of-the-art models. The scenario consists of a sectorized LTE cellular network containing 2 BSs, each with 3 directional antennas, and 1000 UEs whose position was sampled from a uniform random distribution. The geographical area is a square of $16\,000\,\mathrm{m}^2$ divided into squared bins of $1\,\mathrm{m}^2$, which are randomly classified as indoor or outdoor with the same probability. The downtilt θ of the controlled antenna can vary in the range $[0°, 15°]$ with adjustments of $1°$. In contrast, the downtilt of the other antennas is fixed to random values sampled from a uniform distribution in $[0°, 15°]$. According to the assumption mentioned in Section 3, the scenario is static except for the downtilt of the controlled antenna. All the relevant parameters of the simulation are reported in Table 1.

With this setup, the downtilt θ of the controlled antenna can assume 16 different values. Consequently, there are 16 distinct states. Fig. 5 shows rewards and reward components obtained by analyzing the 16 possible configurations with the simulator. Observing the reward curve, we can understand that the optimal policy consists of reaching $\theta = 8°$ in the minimum number of actions and then staying at $\theta = 8°$ for the rest of the time. It is also worth mentioning that, for each episode, we reset θ randomly among the 16 possible discrete configurations.

5.2 Performance

We started by training a DQN agent employing a DNN with two 128-neuron hidden layers. All the relevant hyperparameters are reported in Table 2. The drDQN agent, instead, comprises two DNNs, each with the same hyper-parameters from Table 2 and receiving the same input features (i.e., the complete system state).

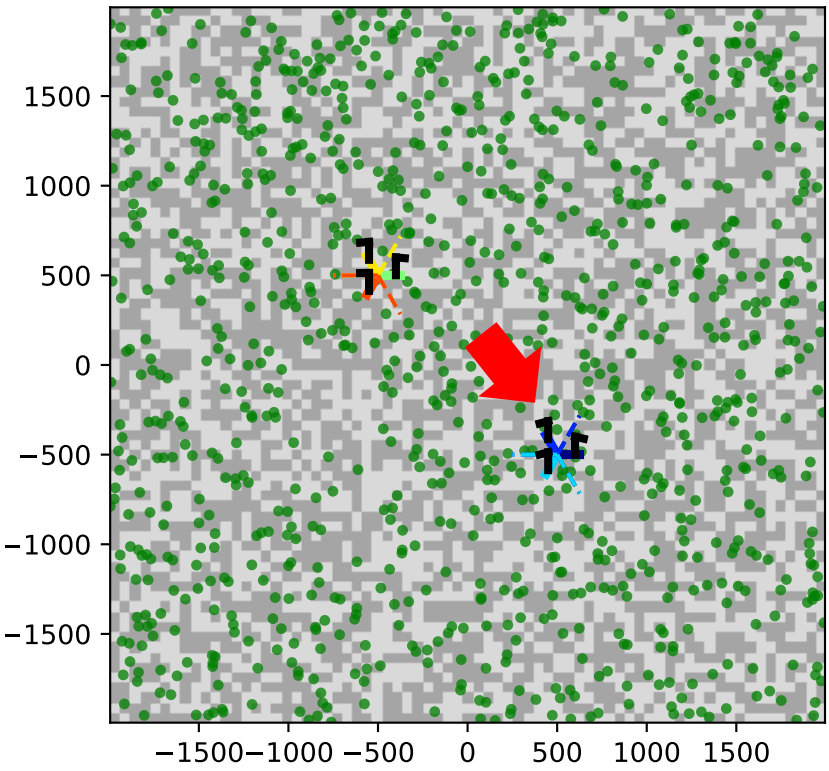


Fig. 4: Simulated scenario. The map is divided in indoor and outdoor bins, colored in dark and light grey, respectively. UEs are represented as green dots, while antennas are drawn in black. The antenna under control is pointed by a red arrow.

The learning curves, shown in Fig. 6, indicate that drDQN and DQN agents converged with a similar trend. With the optimal DQN agent available, we continued by training LMUT. After some tuning, we ended up with the hyperparameters in Table 3. Then, we evaluated the fidelity of LMUT with respect to DQN using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), accuracy, and confusion matrix. Whereas RMSE and MAE measure the difference of the Q-values provided by LMUT with respect to the targets from DQN (regression problem), accuracy and confusion matrix measure the difference between LMUT and DQN in selecting the action (which can also be seen as a classification problem where the ground truth is the action selected by DQN). As shown in Table 4, both RMSE and MAE are small, meaning that LMUT can approximate the DQN model precisely. From Fig. 7, we can observe that LMUT managed to fit the action selection of the DQN agent perfectly. Furthermore,

Table 1: Parameteres of the simulated scenario.

Parameter	Value
Map size	16 000 m^2
Number of BSs (B)	2
Inter-site distance	$\approx$ 1414 m
Number of antennas per BS (A_b)	3
Range of downtilt (θ_{min}, θ_{max})	$[0°, 15°]$
Variation of downtilt per action (δ)	1°
Downlink maximum power	40 W
Antenna model on BS	hv 742215 fitted
Number of UEs (U)	1000
Antenna type on UEs	Isotropic
Propagation model	Okumura–Hata
Fraction of indoor/outdoor	0.5
Radio technology	LTE
Optimization weight (w)	1

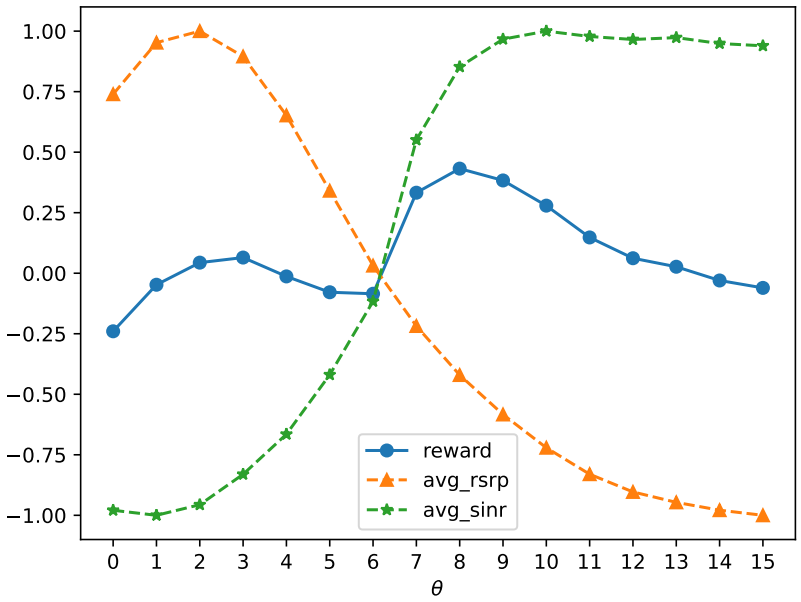


Fig. 5: Rewards and reward components for varying downtilt θ in the simulated scenario.

Table 2: Hyperparameter to train DQN and drDQN.

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Num. hidden layers	2
Num. neurons per layer	128
Activation function	tanh
Num. steps per episode	20
Exploration strategy	ϵ-greedy

Table 3: Hyperparameteres to train LMUT.

Hyperparameter	Value
Min. MSE to split	0.01
Min. instances to split	50
Min. instances per child	10
Min. variance reduction	0.01
Max. tree depth	50
Optimizer (linear regression)	Adam

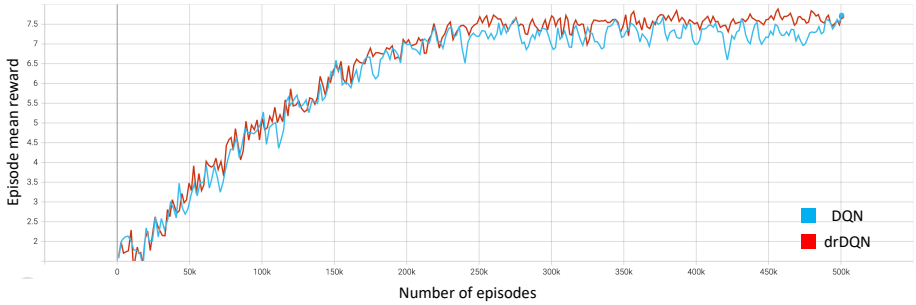


Fig. 6: Learning curves of DQN and drDQN.

this fit was achieved with only a few binary splits, which allows inspecting the model directly as a white box, as depicted in Fig. 8. Such a small model was expected, given the small state space.

The performance of the trained agents is quantified by the episode mean reward during the evaluation phase. As reported in Table 5, both drDQN and LMUT perform optimally like DQN. Considering the duration of an episode (Table 1) and the reward at the optimal downtilt (Fig. 5), it is evident that all three algorithms found the optimal policy. These results help prove the correctness of the implementation, especially in preparation for the explainability evaluation. However, it is worth pointing out that this paper focuses on the explainability aspects of these algorithms rather than their performance. Such satisfactory results in terms of performance were expected, considering the small state space.

5.3 Explainability

Once the drDQN and LMUT agents were adequately trained, we evaluated their explanations. As described in Section 4, drDQN can explain its action selection by providing contrastive explanations through RDX. Fig. 9 shows two of such local explanations. In particular, from Fig. 9c, we can understand why the drDQN agent prefers to downtilt instead of uptilting when $\theta = 2°$. According to the agent's explanation, downtilting outperforms uptilting in terms of expected

Table 4: Fidelity of LMUT with respect to DQN.

Metric	Value
RMSE	0.01
MAE	0.01
Accuracy	1.0
Balanced accuracy	1.0

Table 5: Performance of DQN, drDQN, and LMUT.

Algorithm	Episode mean reward
DQN	7.74
drDQN	7.74
LMUT	7.74

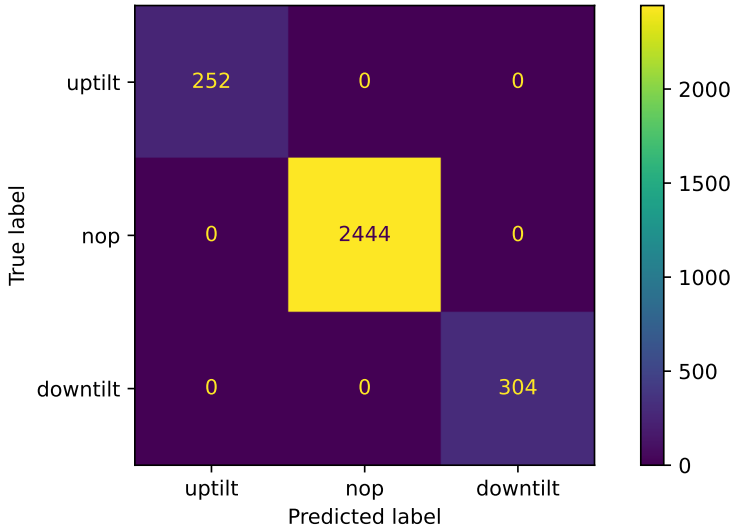
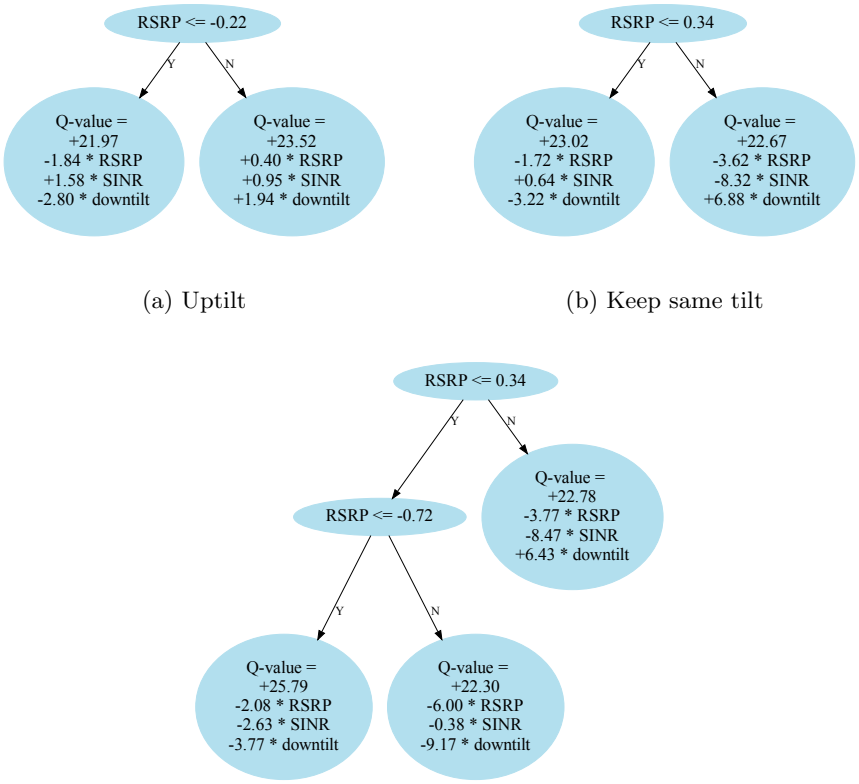


Fig. 7: Confusion matrix of LMUT with respect to DQN.

r_{sinr} but not expected r_{rsrp}, and the agent prefers this action because the benefit is greater than the loss. This explanation can be confirmed by checking Fig. 5. Considering the initial state with $\theta = 2°$, we can observe that downtilting and then following the optimal policy collects a higher r_{sinr} but a lower r_{rsrp}, in comparison with uptilting. We also know that downtilting is optimal because it proceeds towards the optimal configuration ($\theta = 8°$). Similarly, in Fig. 9d we can understand why drDQN decides to keep the same downtilt in $\theta = 8°$. The agent believes that this action, compared to downtilting, leads to a better overall reward thanks to a higher r_{rsrp}, despite the cost in terms of r_{sinr}. Once again, we can verify that the agent's belief is correct by looking at Fig. 5.

LMUT can generate both global and local explanations, as explained in Section 4. Table 7 presents two local explanations for the same cases considered before for drDQN. It is worth highlighting that the rules for the Q-value computation can be directly identified from the LMUT model in Fig. 8, but extracting and listing them is convenient, especially when dealing with a larger model. When $\theta = 2°$, the best action is to downtilt, which is optimal, as already argued



(a) Uptilt

(b) Keep same tilt

(c) Downtilt

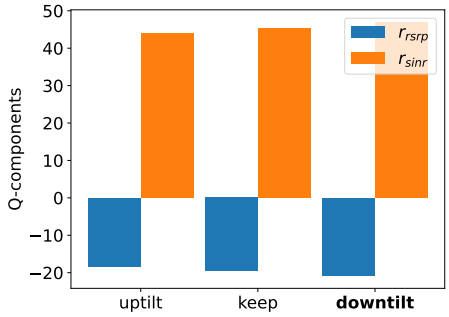
Fig. 8: LMUT structure after training. There is one linear tree for each action.

for drDQN. The triggered rule in this situation is:

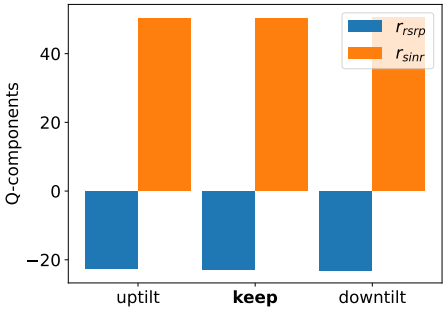
$$RSRP \in (0.34, +\infty) \implies Q = +22.78 - 3.77 \cdot RSRP - 8.47 \cdot SINR + 6.43 \cdot \theta \quad (15)$$

Linear regression models are transparent by design, as their weights indicate the relative contribution of each input feature to the output. Thus, we can observe that, when $RSRP \in (0.34, +\infty)$, the largest impact is caused by the current SINR value. Similarly, when $\theta = 8°$, the best action is to keep the same downtilt (which is, again, optimal), and the greatest contribution comes from θ.

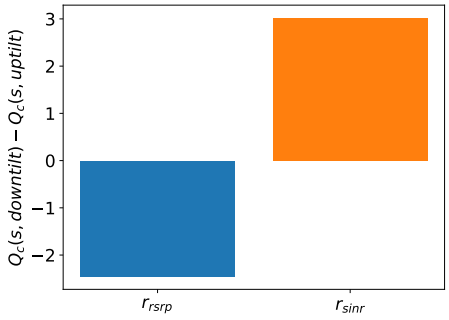
The global explanation provided by LMUT, which indicates the overall importance of $RSRP$, $SINR$, and θ on the LMUT agent's decision-making, is shown in Table 6. Surprisingly, the explanation contains a null value for $SINR$ and θ that, in light of the model in Fig. 8, is fallacious. These features, in fact, have a non-negligible impact on the Q-function through the linear regression on



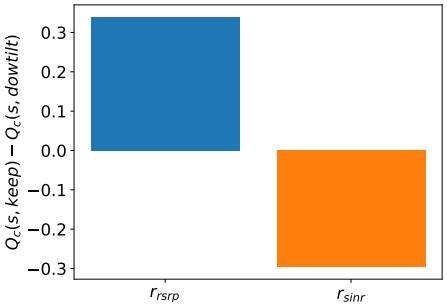
(a) Q-components when $\theta = 2$. The *downtilt* action is selected.



(b) Q-components when $\theta = 8$. The *keep* action is selected.



(c) RDX showing the advantage of *downtilting* over *uptilting* when $\theta = 2$.



(d) RDX showing the advantage of *keeping the same tilt* over *downtilting* when $\theta = 8$.

Fig. 9: Local explanations generated by drDQN for two different states ($\theta = 2$ and $\theta = 8$), including Q-components (a-b) and RDX (c-d).

the leaf nodes. The root of the problem is that Eqs. (13) and (14) considers only the splitting nodes in the regression trees and neglects the leaf nodes altogether.

5.4 Discussion

We now critically discuss the results presented in the previous section to generalize and draw conclusions. We found that both drDQN and LMUT achieve optimal performance in the considered RET scenario. Despite this finding, we warn that LMUT might lose performance compared to deep RL algorithms in more complex RET scenarios with a larger state space (e.g., multiple antennas to control), since linear trees are less expressive than DNNs. In contrast, due to its design and convergence proof, drDQN is expected to always keep the same performance as the state-of-the-art DQN.

Both XRL algorithms have additional requirements compared to standard RL. DrDQN adds a constraint to the reward design, as the reward needs to be a sum of reward components. We argue that this requirement is the most critical

Table 6: Global explanation via feature importance generated by LMUT.

Feature	Importance
$RSRP$	0.82
$SINR$	0.00
θ	0.00

Table 7: Local explanations via rule extraction generated by LMUT for two different states ($\theta = 2°$ and $\theta = 8°$). The normalized version of θ is denoted by $\overline{\theta}$.

State	$RSRP = 1.00$ $SINR = -0.96$ $\theta = 2$ ($\overline{\theta} = -0.73$)	$RSRP = -0.42$ $SINR = 0.85$ $\theta = 8$ ($\overline{\theta} = 0.07$)
Uptilt rule	$RSRP \in (-0.22, +\infty)$ $\Downarrow$ $Q = 0.40 \cdot RSRP + 0.95 \cdot SINR$ $+1.94 \cdot \overline{\theta} + 23.52 = 21.59$	$RSRP \in (-\infty, -0.22)$ $\Downarrow$ $Q = -1.84 \cdot RSRP + 1.58 \cdot SINR$ $-2.80 \cdot \overline{\theta} + 21.97 = 23.89$
Keep rule	$RSRP \in (0.34, +\infty)$ $\Downarrow$ $Q = -3.62 \cdot RSRP - 8.32 \cdot SINR$ $+6.88 \cdot \overline{\theta} + 22.67 = 22.01$	$RSRP \in (-\infty, 0.34)$ $\Downarrow$ $Q = -1.72 \cdot RSRP + 0.64 \cdot SINR$ $-3.22 \cdot \overline{\theta} + 23.02 = \mathbf{24.07}$
Downtilt rule	$RSRP \in (0.34, +\infty)$ $\Downarrow$ $Q = -3.77 \cdot RSRP - 8.47 \cdot SINR$ $+6.43 \cdot \overline{\theta} + 22.78 = \mathbf{22.39}$	$RSRP \in (-0.72, 0.34)$ $\Downarrow$ $Q = -6.00 \cdot RSRP - 0.38 \cdot SINR$ $-9.17 \cdot \overline{\theta} + 22.30 = 23.86$
Best Action	Downtilt	Keep

drawback of drDQN and limits the specification of the optimization objective in the RET use case. Such a constraint has indeed affected our problem formulation in Section 3 and prevented us from considering different reward designs. In contrast, LMUT requires a high-performing RL agent that provides targets for supervised learning, not only for the initial training but also to keep learning while interacting with the environment. As a consequence, this RL agent should use an off-policy algorithm to keep learning from experiences generated by LMUT and providing updated targets for the online training of LMUT.

In terms of model complexity, while drDQN needs to specify the DNN structure before training, LMUT adapts to the problem and grows its structure as little as possible to fit the data. In our RET scenario, LMUT fit the data with a small model, which was easy to inspect directly. In addition, drDQN still employs black-box DNNs, whereas LMUT is fully transparent with only linear models.

The types of local explanations provided by the two XRL algorithms are very different. DrDQN can compare two actions by looking at the composition of future expected rewards (i.e., Q-values). In the RET use case with only three actions (uptilt, downtilt, and keep the same tilt), we argue that contrastive explanations are very user-interpretable, as they concisely answer questions in

the form of "why did you decide to downtilt instead of uptilting?". On the contrary, LMUT cannot provide contrastive explanations. Extracted rules make how the Q-value is computed transparent but do not provide a direct comparison with other actions. From a rule head, a more desirable explanation would be to understand in which sub-range the agent prefers the selected action and what happens outside that sub-range. Furthermore, the idea of reward decomposition from drDQN might be integrated into LMUT to enhance its explainability.

Differently from drDQN, LMUT can also provide a global explanation. However, we showed that this global explanation could be very misleading. The computation completely neglects the linear regressions at the leaf nodes, which contribute to the Q-value computation. Modifying this computation is necessary to make the global explanation more reliable.

6 Conclusions

In this paper, we empirically evaluated two intrinsic XRL algorithms, drDQN and LMUT, in the RET optimization problem. In terms of explainability, we conclude that drDQN can provide intuitive contrastive local explanations which fit perfectly the RET problem. In contrast, even employing a fully transparent model, we found that LMUT needs further research on the extracted explanations, which are less interpretable than drDQN's. Surprisingly, we identified an issue in the global explanation of LMUT, which neglects the critical contribution of leaf nodes. In terms of performance, our results indicate that both drDQN and LMUT can perform well in the presented RET use case, with performance similar to DQN. However, further experiments are needed to test LMUT's performance with larger state spaces.

Future research directions are as follows. The explainability of LMUT might be improved by fixing the computation of the global explanation and by developing more intuitive contrastive explanations extracted from the transparent model. DrDQN might be extended to support more general functions than simple sums. Finally, the complexity of the RET scenario might also be increased.

References

1. X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 6, pp. 4005–4018, June 2019.
2. R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.
3. N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

4. V. Buenestado, M. Toril, S. Luna-Ramirez, J. M. Ruiz-Aviles, and A. Mendo, "Self-tuning of remote electrical tilts based on call traces for coverage and capacity optimization in LTE," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2016.
5. F. Vannella, G. Iakovidis, E. A. Hakim, E. Aumayr, and S. Feghhi, "Remote Electrical Tilt Optimization via Safe Reinforcement Learning," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, (Nanjing, China), pp. 1–7, IEEE, Mar. 2021.
6. F. Vannella, J. Jeong, and A. Proutiere, "Off-Policy Learning in Contextual Bandits for Remote Electrical Tilt Optimization," *IEEE Transactions on Vehicular Technology*, pp. 1–11, 2022.
7. E. Balevi and J. G. Andrews, "Online Antenna Tuning in Heterogeneous Cellular Networks With Deep Reinforcement Learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, pp. 1113–1124, Dec. 2019.
8. A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, Feb. 2021.
9. E. Puiutta and E. M. S. P. Veith, "Explainable Reinforcement Learning: A Survey," in *Machine Learning and Knowledge Extraction* (A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, eds.), vol. 12279, pp. 77–95, Cham: Springer International Publishing, 2020.
10. R. Inam, A. Terra, A. Mujumdar, E. Fersman, and A. Vulgarakis, "Explainable AI – How humans can trust AI," *Ericsson White Paper*, Apr. 2021.
11. European Commission, "On Artificial Intelligence - A European approach to excellence and trust," Feb. 2020.
12. M. T. Ribeiro, S. Singh, and C. Guestrin, ""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*, (San Francisco California USA), pp. 1135–1144, ACM, Aug. 2016.
13. A. Iucci, A. Hata, A. Terra, R. Inam, and I. Leite, "Explainable Reinforcement Learning for Human-Robot Collaboration," in *2021 20th International Conference on Advanced Robotics (ICAR)*, (Ljubljana, Slovenia), pp. 927–934, IEEE, Dec. 2021.
14. K. Zhang, P. Xu, and J. Zhang, "Explainable AI in Deep Reinforcement Learning Models: A SHAP Method Applied in Power System Emergency Control," in *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, (Wuhan, China), pp. 711–716, IEEE, Oct. 2020.
15. L. He, N. Aouf, and B. Song, "Explainable Deep Reinforcement Learning for UAV autonomous path planning," *Aerospace Science and Technology*, vol. 118, p. 107052, Nov. 2021.
16. A. Terra, R. Inam, and E. Fersman, "BEERL: Both Ends Explanations for Reinforcement Learning," *Applied Sciences*, vol. 12, p. 10947, Oct. 2022.
17. Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.
18. G. Liu, O. Schulte, W. Zhu, and Q. Li, "Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees," in *Machine Learning and Knowledge Discovery in Databases* (M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, eds.), vol. 11052, pp. 414–429, Cham: Springer International Publishing, 2019.
19. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013.

20. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
21. B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 303–312, IEEE, 2017.
22. J. van der Waa, J. van Diggelen, K. van den Bosch, and M. Neerincx, "Contrastive Explanations for Reinforcement Learning in terms of Expected Consequences," *arXiv:1807.08706 [cs, stat]*, July 2018.
23. S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding Atari agents," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1792–1801, PMLR, July 2018.