Explainable Deep Reinforcement Learning: State of the Art and Challenges

GEORGE A. VOUROS, University of Piraeus

Interpretability, explainability, and transparency are key issues to introducing artificial intelligence methods in many critical domains. This is important due to ethical concerns and trust issues strongly connected to reliability, robustness, auditability, and fairness, and has important consequences toward keeping the human in the loop in high levels of automation, especially in critical cases for decision making, where both (human and the machine) play important roles. Although the research community has given much attention to explainability of closed (or black) prediction boxes, there are tremendous needs for explainability of closed-box methods that support agents to act autonomously in the real world. Reinforcement learning methods, and especially their deep versions, are such closed-box methods. In this article, we aim to provide a review of state-of-the-art methods for explainable deep reinforcement learning methods, taking also into account the needs of human operators—that is, of those who make the actual and critical decisions in solving real-world problems. We provide a formal specification of the deep reinforcement learning explainability problems, and we identify the necessary components of a general explainable reinforcement learning framework. Based on these, we provide a comprehensive review of state-of-the-art methods, categorizing them into classes according to the paradigm they follow, the interpretable models they use, and the surface representation of explanations provided. The article concludes by identifying open questions and important challenges.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning;**

Additional Key Words and Phrases: Deep learning, deep reinforcement learning, interpretability, explainability, transparency

ACM Reference format:
George A. Vouros. 2022. Explainable Deep Reinforcement Learning: State of the Art and Challenges. *ACM Comput. Surv.* 55, 5, Article 92 (December 2022), 39 pages.
https://doi.org/10.1145/3527448

1 INTRODUCTION

EU ethical guidelines [50] on accountability, technical robustness and safety, oversight, privacy and data governance, non-discrimination and fairness, transparency, and societal and environmental well-being specify the essential aspects to be considered by **artificial intelligence (AI)** and **machine learning (ML)** products and appliances. Concise analysis for accredited adherence of systems to EU ethical principles is an essential prerequisite for the acceptability of deploying

This work was partially supported by the TAPAS H2020-SESAR-2019-2 Project (GA number 892358) Towards an Automated and exPlainable Air traffic management (ATM) System.
Author's address: G. A. Vouros, University of Piraeus, Greece, Gr Lambraki 126, Piraeus, Greece, 18534; email: georgev@unipi.gr.

0360-0300/2022/12-ART92 $15.00
https://doi.org/10.1145/3527448

and using any such application. In addition, the U.S. National Security Commission [49] reports that "Ethical considerations are an important facet of R&D, application, training, protection, and cooperation in AI . . . Developing trustworthy AI systems is essential for operational integrity and adoption. It is closely connected to, and depends on, reliability, robustness, auditability, explainability, and fairness. From the earliest phase, systems should be designed with ethics in mind."

Designing and implementing systems with "deep knowledge" prevails in AI: we may recall the need of incorporating "deep knowledge" about the domains and problems addressed (in contrast to "shallow knowledge") in expert and knowledge-based systems, mitigating brittleness, vulnerability to unforeseen circumstances, lack of ability to reason from "first principles" when needed, and inability to provide informative explanations on the reasoning toward reaching conclusions and prescribing solutions. Deep knowledge is indeed essential for AI systems today with some very emergent, important, and tremendously larger/broader challenges, and with more severe, challenging consequences and inescapable needs. In contrast to interpretable systems that exploit explicit knowledge and whose reasoning mechanisms principles and inner workings are more transparent to humans, today deep knowledge is encoded in hidden layers of multi-layered deep **neural networks (NNs)**, in the form of weights on "synapses" or in models that fit a value function ranging into sets of large number of actions in millions of states. Reaching this point of technological advances allows us to deploy autonomous systems acting in the real world with significantly more advanced levels of perception, decision making, and automation, all integrated in a single loop. These systems have not been instructed by direct advice but have been trained on massive data sampled from the real world. Most importantly, the interpretability of knowledge and reasoning mechanisms has been lost, not only for the domain experts, or for the humans using these systems, but also for AI systems' engineers and developers. This has huge impact on ethical concerns when deploying and using these systems, thus on systems' accountability, liability, safety, and trustworthiness.

The consequences are crucial especially when we aim at higher levels of automation, where models aim to support autonomous agents to act autonomously. The problem is addressed in the literature from different perspectives, depending on the targeted use of explanations: ML experts and system developers need explanations of algorithms' inner workings, whereas domain experts and operators using/working with autonomous systems need to understand how the system reaches specific decisions, especially when the solutions proposed are not intuitive or obvious, understanding models' "whims" (to use an anthropocentric metaphor), objectives, and capabilities.

Toward developing trustworthy and accountable autonomous systems (agents) to support automation in safety-critical real-world settings, this article aims to review state-of-the-art methods regarding interpretability, explainability, and transparency of **reinforcement learning (RL)** methods, with emphasis on their **deep reinforcement learning (DRL)** versions: currently we lack a concise understanding of the qualities of explainable and transparent DRL methods and we have to close this gap. This is important in many real-world complex settings in which DRL agents are deployed, spanning from (co-)driving autonomous vehicles to supporting automation in air traffic management and other cases with time-critical decisions. The challenges to make DRL models interpretable/explainable are great: some of these are "inherited" from ML models in general; however, for DRL methods, some additional subtle and important issues need to be considered, given that DRL models exploit intertwined closed prediction box models and they do solve sequential decision tasks, aiming to increase the expected feature reward with respect to agents' preferences, intended objectives, and constraints that the environment imposes (partial observability, delayed rewards, dynamics, multiple co-existing agents, etc.).

First, agents are expected to provide informative explanations even in cases where humans cannot explain their own way of thinking/acting on solving sequential decision problems,

comprehensively and with sufficient clarity: this is true in cases where it is easier for humans to demonstrate how they apply their knowledge and skills in example cases. For instance, when systems follow a data-driven approach learning from demonstrations, it may be hard to explicate the acquired knowledge. In these cases, providing concise "mainstream" explanations (e.g., explicating their knowledge and reasoning chains for expected rewards in long-term time horizons) may simply fail. Systems must be empowered with sophisticated means that allow them to demonstrate reasoning and problem-solving abilities more intelligently (i.e., taking into account informative factors, providing examples of behavior, or critical aspects of decision making), supporting users to understand systems' rationale at various levels of detail. Let me provide an example on this. We may all have used the "shoe laces" example in our early AI lectures to show how difficult it is for experts to express procedural knowledge on specific sequential tasks and problem solving, and thus how difficult it is for systems to represent in symbolic and explicit ways such knowledge. However, today, providing paradigmatic demonstrations of expert skills and activity may be enough for a system to gain competency in performing such tasks and in solving sequential decision problems. Such a system should explicate facets of the knowledge acquired and must have the ability to provide comprehensive explanation for how solutions are reached along trajectories of states reached, how objectives are achieved, and when/why it fails.

Second, we need to investigate what the principles or paradigms of building an interpretable/explainable DRL system are. Is it enough to add some interpretability/explainability modules to a DRL system (as we would do in any ML system), or do we need to design a system from the early phases to be explainable (as it is conjectured to be the case for building collaborative and ethical systems)? In other words, do we need to design and develop inherently explainable DRL methods? An answer to this question may vary between domains and conditions of systems' deployment, but this decision should not be based on any assumption about a trade-off between accuracy in prediction and interpretability. Belle and Papantonis [58] propose methodological aspects for reaching such a decision for explainable ML, whereas Rudin et al. [57] conjecture that closed boxes are generally unnecessary, given that their accuracy is generally not better than a well-designed interpretable model (whose probability of existence is rather high). However, a DRL method may comprise more than one interpretable models whose functionality is intertwined, and whose interpretability/accuracy may affect final outcomes.

Third, we need to answer "what" information and "how" it should be communicated for providing effective explanations. We may take advantage of research in other scientific fields (social and cognitive sciences, as well as in philosophy) regarding the qualities of "good" explanations. Toward this, we need to consider not only the qualities of the explanation content ("what") but also the modalities that will be used to provide surface representations of that information ("how"), the characteristics of the context in which explanations are provided, and the purpose that they need to serve with respect to constraints and requirements (pragmatic aspects). For example, humans may be acquainted with specific modes and forms of information, using specific nomenclature and/or visual means, also in relation to responsiveness, temporal requirements in certain contexts, cognitive load, and so forth.

Fourth, we need concise experimental protocols for explainability, considering the purposes and contexts of explanations' provision together with human factors, in relation to the explanation problem and the required qualities of explanations. This will allow the community working on explainability to compare different approaches and build a firm background, making conjectures and refuting theories on explainability, proceeding on more advanced results. Although this holds for ML methods in general [56], experimental protocols are tremendously important for DRL methods, which have to support humans to infer faithful models of behavior, objectives, and capabilities of agents, toward developing trustworthiness in real-world problem-solving settings.

Motivated by these challenging issues, in this article we review **explainable deep reinforcement learning (XDRL)** methods. We aim to understand their potential capabilities and limitations to solve interpretability/explainability problems while focusing on the needs of human operators. Operators need to be aware not only of the particular situation the agent faces but also of the role that certain real-world features play in decision making, of the agent's objectives, decision-making capabilities, and preferences to act, of the overall policy, and of the choices made under specific (potentially, critical) circumstances. In general, operators need to be able to understand and judge in an informed way when to trust an agent, when to take control and/or apply appropriate mitigation actions, and when to provide further advice for refining an agent's knowledge. Although surveys on explainable RL exist (e.g., [1, 24]), here we aim at a comprehensive survey focusing on XDRL, providing details about all (according to our knowledge) works describing and providing well-justified insights on state-of-the-art methods, under a common framework, distinguishing clearly between existing paradigms, providing details on evaluating these approaches, also with respect to transparency, and stating emerging and important challenges.

This article is organized as follows. In Section 2, we provide preliminary knowledge and discuss interpretability/explainability aspects that we must consider, as well as explainability desiderata. Section 2 formulates the problems of providing explanations for DRL agents. Based on the definitions provided, Section 3 identifies the functionality of the necessary components of an XDRL method and attempts a categorization of XDRL methods, based on the explanation paradigm they follow and needs they do address. Section 4 provides a comprehensive review of the state of the art methods, and Section 5 concludes the article with final conclusions and challenges ahead. The appendix (supplementary online material) of this article provides a detailed categorization of reviewed methods in three tables using a comprehensive set of characteristics.

2 INTERPRETABILITY, EXPLAINABILITY, AND TRANSPARENCY OF DRL METHODS

This section sets the ground to proceed toward formulating the explainability problems for DRL agents and providing a general framework of XDRL agents in the next section. First, it provides DRL preliminary knowledge, so as to introduce terminology and symbols that we will use in subsequent sections, assuming that the readers are acquainted with DRL methods. It proceeds to clarify the notions of explainability, interpretability, and transparency, which is useful toward understanding the various facets of the problem, and states explainability desiderata. Finally, it provides definitions of the explainability problems for DRL agents.

2.1 DRL Preliminaries

Given a set of states S and a set of actions A available to an agent acting in an environment E, possibly with continuous parameters, and a reward function $r : (S \times A; E) \rightarrow \mathbb{R}$, the goal of the agent is to learn a policy π—that is, a mapping from states to actions or to a distribution over actions—that will maximize its expected cumulative rewards.

Policy and reward models π_θ and r_ϕ are parameterized by variables' vectors θ and ϕ, respectively, and take as input subsets of M, a set of real-world features. A trajectory in time horizon (length) H is defined as a sequence of states s_t and actions a_t, for $t = 1, \ldots, H$. The policy and reward parameters determine the optimal trajectory $\tau_E^{\theta,\phi}$ in the environment E:

$$\tau_E^{\theta,\phi} = \underset{\tau_E^\theta \in TR_E^\theta}{\operatorname{argmax}} \sum_{t=0}^{H} \gamma^t r_\phi(s_t, a_t; E),$$

where γ is the discount factor between 0 and 1 that controls the trade-off between immediate and future rewards, and TR_E^θ denotes the set of all possible trajectories in environment E, given

parameters θ and reward function r_ϕ. Those are the trajectories that can be sampled from the policy π_θ (i.e., $\tau_E^\theta \sim \pi_\theta$).

The goal of DRL methods is, given the reward function r_ϕ, to tune the parameters θ so as to maximize the expected cumulative reward while acting in the environment E—that is, determine θ^*—such that

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{\tau_E^\theta \sim \pi_\theta} \sum_{t=0}^{H} \gamma^t r_\phi(s_t, a_t; E).$$

The agent, depending on the RL method used, may be required to fit a model of the environment E and/or estimate the values of states, or the values of state-action pairs (i.e., the expected cummulative reward from that state, or state and action, respectively), or even (e.g., in case of imitation learning) to learn a model of the reward function, tuning parameters ϕ, given the set of states' features M.

2.2 Explainability, Interpretability, and Transparency

Given the preceding specifications and the models that a DRL method comprises (policy, reward models, etc.—these are detailed in subsequent sections), we need to specify what it means to enhance a DRL method with explainability and interpretability, providing transparency in decision making.

Lipton [39] asserts that explanation is post hoc interpretability. Biran and Cotton [40] (among others, e.g., Rudin et al. [57]) define interpretability as the degree to which one can understand the response of a system. Relevance, of information for relationships contained in the data or learned by a ML model, for a particular audience into a chosen problem, is an important aspect of interpretable models in the work of Murdoch et al. [55]. Explanation is a particular way of obtaining such an understanding, selecting "what" must be communicated and "how" this is presented (using textual and/or visual means, providing "global" vs. "local" or case-specific explanations, at different levels of detail, scale, and granularity, etc.): this usually involves filtering interpretations and realizing surface representations of interpretations, using an explanation logic that exploits interpretable models. A distinction between "what" and "how" is also considered in the work of Puiutta and Veith [1].

To avoid overlaps and circularity and provide a concise definition of terms, close to the definitions from Puiutta and Veith [1], we use the term *explanation* to denote the surface representation of an interpretation provided to the user. The term *interpretation* is used to denote the explanation content provided by an interpretable model. Therefore, *interpretability* is the ability of a system to provide explanations' content exploiting an interpretable model, *explainability* is the ability to provide surface representations of interpretations. *Transparency* is the ability of the system to generate explanations that are understandable in the context of system deployment, with respect to a domain-specific set of constraints: this definition of transparency is close to the first principle (definition) specified by Rudin et al. [57].

More specifically, considering that *explainability* is the ability of systems to provide qualitative understanding of responses and objectives, given environment features from the set M and information provided by interpretable models, and close to our focus on DRL, we need to elaborate and clarify various aspects:

- First, what does *qualitative understanding* mean; why do we require qualitative understanding; and what should be made understandable, and how can this be achieved?
- Then, what is a *system's response* in the context of DRL methods? Do we consider any single such response at any time instance, or sets of responses (maybe aggregated or summarized) toward the evolution of a (sub-)trajectory, reaching certain objectives?

- What *features* from the set of available features in M should be considered for providing an explanation, and how do we determine the importance of features?
- Finally, a critical dimension that should be considered while elaborating on any of the preceding questions is *whose needs* must an XDRL method satisfy?

First, qualitative understanding expresses how humans understand the qualities of an agent's policy and an agent's preferences, abilities, and objectives, being able to provide accurately the rationale for an agent's chosen courses of actions, as the environment evolves within a state space, or for specific responses under certain circumstances. Such an understanding must occur in a human interpretable domain including domain-specific concepts, entities, and environment features that are presented using domain terms and other information modalities that humans are acquainted with while operating in a domain (e.g., sounds or special visualization symbols). Such an understanding is in contrast to providing, for instance, detailed calculations, or detailed numerical assessments of any kind, such as by means of visualizations, at any scale and level of detail. This may of course be useful for understanding the inner workings of any method and model quantitatively, but it cannot serve the purpose of explaining the qualities of an agent's decision making in an explicit way. This requirement for qualitative understanding is more relevant to domain operators who need to understand why an agent takes, or why it does not take a decision in a specific circumstance, at any level of detail, granularity, or scale (we explain these terms further in the next paragraphs). Thus, such an understanding supports humans to generalize beyond individual explanations, allowing them to infer how probable it is for the agent to follow specific modes of behavior, apply specific skills,[1] or take a certain action under specific circumstances.

This is equivalent to supporting humans to infer models on (a) how the agent responds in certain circumstances with respect to its abilities and preferences, (b) the agent's behavior with respect to its objectives, and (c) the agent's behavior with respect to its abilities, preferences, and objectives, without considering "where do the numbers come from." These models can be denoted respectively by $P(\sigma_E^{\theta,\phi}|f(\theta);E)$, $P(\sigma_E^{\theta,\phi}|f(\phi);E)$, or $P(\sigma_E^{\theta,\phi}|f(\theta),f(\phi);E)$, where σ denotes the agent's response to the environment (e.g., an action, a (sub-)trajectory, or any skill), revealing an agent's behavioral choices at different scale and granularity levels, and $f(\cdot)$ provides a representation of those model elements that drive an agent's response at any time point or time period, at a low level. In other words, $f(\cdot)$ can be considered as a "window" to the DRL agent's inner models' aspects. Such functions may, for example, provide Q values, importance/saliency of state features assessed by the models, policy rollouts, and even **Markov decision process (MDP)** components or models' raw weights. To keep the presentation simple, we consider a single function, whose indicated arguments ϕ or θ indicate which of the DRL models' elements f exploits.

At this point, we can elaborate further on the different roles these models play:

- $P(\sigma_E^{\theta,\phi}|f(\theta);E)$ models how humans interpret the agent's activity under various circumstances in E, based on their understanding of agent abilities and preferences. For example, one may infer an agent's preferable mode of behavior in E, considering a subset of parameters M.
- $P(\sigma_E^{\theta,\phi}|f(\phi);E)$ models how humans interpret the agent's activity in the environment, given their beliefs on the objectives targeted in E. This allows humans to infer how the agent's objectives affect its responses. For example, one may infer the agent's mode of behavior given various weighted mixtures of features in M, according to importance.
- Finally, $P(\sigma_E^{\theta,\phi}|f(\theta),f(\phi);E)$ provides a model of how humans interpret an agent's activity based on their beliefs on agent abilities, preferences, and objectives targeted in E. For exam-

[1]Skills comprise options, macro-actions, behavior modalities: skills drive, in a hierarchical way, agent decisions at a low level of detail.

ple, one may infer an agent's mode of behavior given different combinations of capabilities, preferences, and environment features' importance.

Just to give a further example on these models and on their differences, given an apprenticeship learning paradigm, $P(\sigma_E^{\theta,\phi}|f(\theta);E)$ depends mainly on the training examples demonstrated to the agent; $P(\sigma_E^{\theta,\phi}|f(\phi);E)$ depends mainly on the set of features selected to train an agent's reward model, whereas$P(\sigma_E^{\theta,\phi}|f(\theta),f(\phi);E)$ depends on both.

The second question concerns the granularity and scale at which the agent's responses are explained. Whereas *granularity* refers to the level of detail that the agent behavior is considered, *scale* refers to the extent where a specific response is considered. Thus, granularity may refer to the actions performed at any time instance, or, -in a more coarse level, to the trajectories chosen and their aggregated features, or, in even more coarse terms, to the mode of behavior or skill exercised by the agent within a time period. Scale refers to the temporal or to the spatial (in whatever space dimensions) extent in which responses are considered. For instance, one may consider a trajectory within a specific time horizon as a response. Sub-trajectories in more restricted time horizons may be considered, although one may consider trajectories in different spatial extents and dimensions. Thus, depending on the kind of response, $\sigma_E^{\theta,\phi}$ may be instantiated to either (a) $\alpha_E^{\theta,\phi}$, denoting specific actions, (b) $\tau_E^{\theta,\phi}$, denoting specific (sub-)trajectories, or (c) $\zeta_E^{\theta,\phi}$, denoting specific skills, given a set of skills Z, where ζ belongs. For instance, $P(\zeta_E^{\theta,\phi}|f(\theta),f(\phi);E)$ models how humans understand an agent's choices of skills exercised in E, based on their understanding of an agent's preferences and objectives, using the elements provided by $f(\theta),f(\phi)$. Finally, it must be noted that granularity and scale parameters are not included in the denotation, so as to simplify it. For instance, an action is a response given at a specific time instant t and is denoted by $\alpha_{E,t}^{\theta,\phi}$, whereas a trajectory in a time horizon H, from t_0 (now) to t_H, is denoted by $\tau_{E,H}^{\theta,\phi}$, and so forth.

The third question refers to the environment features in M that are being used for providing an explanation of an agent's responses. This is important, as various subsets of features in M may be chosen, or the features in M can be used in different ways to provide an understanding of agent responses. Scale and granularity directly affect how the agent's responses are perceived—for instance, considering scale, a trajectory may be considered in a space of $||M||$ or fewer dimensions, given different projections of the trajectory in subspaces. As a concrete example, consider a trajectory in the 3D space, where $M = \{longitude, latitude, altitude\}$: one may view such a trajectory in any 2D projection, thus considering responses in any subset of features in M. Generally, we may consider the "projection" of responses $p(\sigma_E, S_M)$ in any subset S_M of M. Granularity may impose filtering and/or aggregation of features in the temporal dimension. However, a critical aspect to be considered about features in M is how the agent's responses are affected, if any state of E is projected in a $S_M \subset M$? This is in contrast to the simple response projection mentioned earlier, as it affects how the agent perceives the environment: such "masking" of features and "perturbations" reveal causalities and sensitivities on environment features.

The last question concerns whose needs explainability addresses. As already pointed out, in this article we focus on the needs of human operators, who require to solve domain problems in real-life contexts rather than on the needs of ML and RL experts. This is connected to several aspects considered previously, such as (a) how humans' beliefs and understanding of agents' inner workings are modeled, (b) how and in which terms humans are assumed to perceive and understand the DRL inner workings, (c) in which scale and granularity DRL responses are provided, (d) how sensitivity on features and features' importance is communicated in specific circumstances and

overall, and (e) how, specifically, the policy and/or the objective function are made understandable (also in connection to (b)).

In this article, we do not elaborate on issues related to human factors and modeling of humans, nor on the effectiveness of different surface representations of explanations. There are many works exploring these issues, and this is outside the scope of this article.

2.3 Explainability Desiderata

An agent, to be explainable, needs an interpretable model and an explanation logic. The interpretable model provides explanations' content, whereas the explanation logic selects the modes and the way explanations are surfaced, toward meeting transparency requirements. Interpretable models may be functional parts of the agent—that is, the agent may have the ability to provide responses without them, as we will explain in subsequent sections.

At high levels of automation, the transparent agent must be *trustworthy*, in the sense that operators should be comfortable relinquishing control to it, as pointed out by Lipton [39]. A major factor to that is the *accuracy of predictions and "goodness" of prescribed actions* in terms of solving the identified problems (predictive accuracy), with respect to the interests of stakeholders (e.g., without increasing cost of operations or compromising safety). However this is not sufficient: transparency is crucial, especially when the system's responses do not comply with experts' usual practices or intuition, and in cases where operators need to understand the system's capabilities and limitations.

There are different desiderata for transparency and explainability, regarding human interaction/communication issues, and aspects concerning properties of the interpretable models. This becomes apparent if we consider that transparency requires explainability that meets pragmatic constraints, which in its turn requires interpretable models that provide information toward enhancing humans' understanding of systems' responses, objectives, and capabilities.

Thus, desiderata concern providing explanations that are (a) accurate on the information provided, (b) fitting the purpose and context of system deployment, (c) with respect to *operational requirements* (e.g., on the type of information provided per type of problem instance), *operational constraints* (time constraints and other constraints regarding privacy and security, etc.), and *ethical aspects*.

Toward satisfying these desiderata, the type and amount of information being provided and the way explanations are being surfaced are crucial. Scale and granularity of responses are important aspects. Specifically, concerning the type and amount of information provided, a system may provide a systematic trace of its inner-workings (low scale), with an exhaustive enumeration of causal paths and features being considered per problem instance (low granularity), increasing the complexity of explanations, which may also be affected by the size and complexity of the interpretable model. The ways in which this information is communicated affects transparency and trustworthiness. People need to accomplish tasks with awareness of the problem(s) faced, also with respect to pragmatic constraints and ethical issues (e.g., to sensitivity of information, to potential dual use of the information and system, fairness—making no discrimination against groups of entities), and with respect to any contextual factor that affects acceptability of the system's responses and explanations. For instance, a lengthy explanation regarding a system response to an emergent problem instance (with details known to the explainee) may lower explanations' acceptability and can determine system use in contexts where timely response is necessary, unacceptable. Selecting the most important causes/features that should be included in an explanation, focusing especially on those that are of interest to operators with respect to the problem instance considered, is very important, as evidenced by studies regarding human cognition and explanation effectiveness [41, 42]. This supports effective update of operators' models on the agent's policy, objectives, preferences, and capabilities, relating operators' existing knowledge and agents' interpretations.

Causality and *robustness* are additional features for the system to be trustworthy, and these properties must be conveyed to humans via the explanations provided. Causality requires identifying the input features and causal chains that affect the system's responses, whereas robustness requires that the system performance is not affected considerably with small variations of input features, especially of those that are not important to problem instances: importance of features per problem instance and sensitivity on features are important elements for explanations.

Regarding accuracy, we distinguish predictive accuracy and the accuracy of the information provided by the explanation logic (descriptive accuracy). Accuracy of predictions concerns the ML method used, whereas accuracy of the information provided in explanations concerns the capacity of the interpretable model and of the explanation logic to locate and present the exact models' elements affecting the agent's responses. In cases where the interpretable model is distinct from the models used by the decision-making agent, the interpretable model must replicate the agent's behavior for explainability purposes with *fidelity*: the interpretable model must reproduce the agent's responses accurately, identifying also the important features that affect the agent's responses. Although this may happen with *local fidelity* (i.e., in the vicinity of the problem instance being solved), the interpretable model may provide a *global perspective* of the agent's behavior (e.g., explaining objectives, intended outcomes, preferences, capabilities). These important distinctions are clarified by the problem definitions stated in the next section.

2.4 Problem Formulation

Based on the goal of DRL agents, as specified in Section 2.1, and on the explainability/transparency goals and desiderata, as specified in Sections 2.2 and 2.3, respectively, this section provides definitions for the *model explanation*, *outcome explanation*, and *model inspection* problems for DRL methods, refining those provided in the work of Guidotti et al. [25], so as to connect explainability for DRL methods to explainability of ML prediction methods, allowing a more comprehensive view of explainability in ML. Overall, the model explanation problem aims at providing explanations regarding the overall, global logic behind agents' responses, revealing also global information on agents' preferences and capabilities. The outcome explanation problem is related to explaining responses provided by agents under certain circumstances, specifying the correlation between responses and values of features in M, at a low level of granularity (i.e., regarding certain actions) and scale (i.e., in restricted temporal and spatial extent). Finally, the model inspection problem concerns the provision of information that reveals aspects and properties of the individual models that a DRL method comprises.

Definition 2.4.1 (Model Explanation Problem). Given a DRL method, the *model explanation problem* consists of finding a global explanation $X \in \mathcal{X}$ of behavior, belonging to a human interpretable domain $\mathcal{X}$ through an interpretable model $c_g(f(\theta), f(\phi); E)$, in a specific environment E, using $f(\cdot)$ that provides a representation of policy and objectives models' elements. The global explanation X is provided via an explanation logic $\epsilon_g(c_g)$ that reasons over c_g.

Providing a solution to the model explanation problem aims at updating $P(\sigma_E^{\theta,\phi} | f(\theta), f(\phi); E)$, interpreting the agent's responses at coarse levels of granularity and at large-scale extents, exploiting agent preferences, abilities, and objectives in E. We may say that solutions to the model explanation problem aim to provide global (i.e., independently from specific states) explanations both for the policy and for the objectives of the agent.

Refining Definition 2.4.1 to either the policy or to the objectives models of a DRL method, we have the policy model explanation problem and the objectives model explanation problem.

Definition 2.4.2 (Policy/Objectives Model Explanation Problem). Given a DRL model, the *policy (objectives) model explanation problem* consists of finding an explanation $X \in \mathcal{X}$ of behavior,

belonging to a human interpretable domain $\mathcal{X}$ through an interpretable model $c_{gp}(f(\theta)[f(\phi)]; E)$ for policies ($c_{go}(f(\phi); E)$ for objectives), using a representation of the agent's policy or objective models' elements, provided by $f(\cdot)$. The global policy/objectives explanation X is provided via an explanation logic $\epsilon_g(c_{gp})$ (respectively, $\epsilon_g(c_{go})$) that reasons over c_{gp} (respectively, c_{go}).

It must be noted that in the preceding definition, the policies' interpretable model c_{gp} $(f(\theta)[f(\phi)]; E)$ includes optionally (this is what square brackets denote) a representation of the agent's objectives. We leave this as an option, given that the objectives may be assumed to be known or interpreted separately.

Definition 2.4.3 (Outcome Explanation Problem). Given a DRL method, the *response explanation problem* consists of finding a local explanation $X \in \mathcal{X}$, belonging to a human interpretable domain $\mathcal{X}$ through an interpretable model $c_l(f(\theta), f(\phi), s_t; E)$ at state s_t, using a representation of the agent's policy and objectives models' elements, provided by $f(\cdot)$. The local explanation X is provided via an explanation logic $\epsilon_l(c_l, s_t)$ that reasons over c_l and s_t.

It must be noted that the outcome explanation problem concerns skills and actions that the agent applies at states s_t, as well as (sub-)trajectories that at time t are in state s_t. In any case, the response explanation problem concerns the agent's responses in fine granularity and small scale. Additionally, it may also concern providing local interpretation of the objective function, motivating some responses instead of others.

Definition 2.4.4 (Local Response/Objectives Explanation Problem). Given a DRL method, the *local response/objectives explanation problem* consists of finding an explanation $X \in \mathcal{X}$, belonging to a human interpretable domain $\mathcal{X}$ through an interpretable model $c_{l\sigma}(f(\theta)[f(\phi)], s_t; E)$ for responses (respectively, $c_{lo}(f(\phi), s_t; E)$ for objectives) at state s_t, using representations of models' elements provided by $f(\cdot)$. The local explanation X is provided via an explanation logic $\epsilon_l(c_{l\sigma})$ (respectively, $\epsilon_l(c_{lo})$) that reasons over $c_{l\sigma}$ (respectively, $c(_{lo})$).

Providing a solution to the outcome explanation problem aims at updating a model $P(\sigma_E^{\theta,\phi}|f(\theta), f(\phi); E)$, toward interpreting the agent's responses and objectives at fine levels of granularity and scale, at any state. We may say that any solution to the outcome explanation problem aims at providing local joint explanations both for the fine-grained responses and objectives of the agent, at particular states.

The model inspection problem for DRL methods concerns providing surface representations of DRL internal models' elements that would reveal the agent's internal models' workings and properties, provided via the function $f(\cdot)$.

Definition 2.4.5 (Model Inspection Problem). Given a DRL method, the *model inspection problem* consists of finding a surface representation $r = R([f(\theta)] + [f(\phi)]; E)$ of DRL models' elements, or of models' properties, exploiting representations of models' elements provided by $f(\cdot)$. The notation $[f(\theta)] + [f(\phi)]$ indicates that R exploits either $f(\theta)$, $f(\phi)$, or both, in a joint manner.

In contrast to providing model explanations, the model inspection problem concentrates on the presentation of individual models' elements and properties at low levels of detail, toward inspecting the inner workings of models, without necessarily revealing an understanding of the agent's policy, objectives, or responses.

3 FRAMEWORK AND PARADIGMS FOR XDRL METHODS

Given the preceding stated definitions of the DRL explainability problems, Figure 1 provides the overall architecture of a framework for XDRL methods. In addition to the inner dark box, which

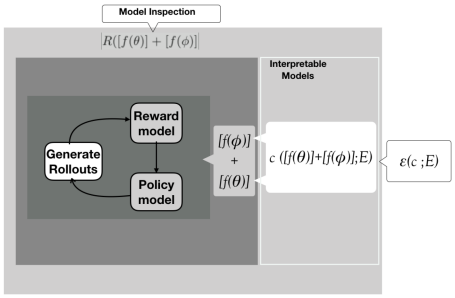

Fig. 1. Overall framework for XDRL.

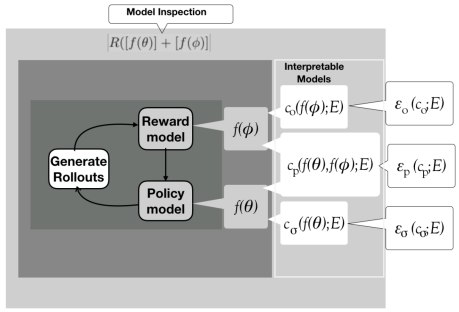

Fig. 2. Framework for XDRL with interpretable models.

provides the main components of any DRL method,[2] Figure 1 presents the basic pipeline for XDRL. DRL models provide interpretations, denoted by $c(\cdot)$, which, moving from dark to lighter areas of the blueprint, are exploited to provide content to an explanation logic. It must be noted that, as defined previously, any such model exploits either $f(\theta)$ or $f(\phi)$, or both, in a joint manner.

The policy π_θ of a DRL agent is represented by the *policy model* with parameters θ (e.g., a deep NN). Given environment states' features in M, this model provides a probability distribution on all actions that can be applied at a state. In conjunction to that model, there can be other models either fitting a value function (e.g., V on states or Q on state-action pairs), or modeling the way the environment evolves, or fitting the reward function. Figures 1 and 2 show only the reward model, placing special emphasis on the objectives of the agent. This model approximates the reward function r_ϕ at any time instance t, given a state-action pair (s_t, a_t). Overall, the DRL method exploits the reward model (possibly in conjunction to other models) to fine-tune the policy model parameters θ, whereas the agent may generate trajectories' samples in E to test the optimality of the policy with respect to the objectives.

To inspect individual models, function $f(\cdot)$ provides a representation of policy model and of objectives model elements. This may solve the model inspection problem, but it does not solve any of the (global policy or objectives) model or (local responses/objective) outcome explanation problems.

Figure 2 shows a refined version of the XDRL framework with interpretable models c exploiting the representations provided by $f(\cdot)$. As is shown, any such interpretable model may be an objectives model $(c_o(f(\phi); E))$, a local responses model $(c_\sigma(f(\theta), s_t; E))$, or a policy model $(c_p(f(\theta), f(\phi); E))$. Any XDRL agent may contain all or any combination of these interpretable models, at the local and/or at the global level.

Explanations are generated by functions ϵ that exploit the interpretable models, maybe in combination with the elements provided by the model inspection components. Figure 2 shows functions providing explanations for objectives $(\epsilon_o(c_o; E))$ or responses $(\epsilon_\sigma(c_\sigma; E))$, or for the policy model $(\epsilon_p(c_p; E))$.

3.1 Interpretable Methods

Some of the models that a DRL agent exploits may be directly interpretable. For instance, the policy model may be a decision tree, or the reward model may be a linear function on features in M. In

[2]This comprehensive DRL blueprint has been suggested by Sergey Levine in their deep reinforcement learning course at https://rail.eecs.berkeley.edu/deeprlcourse/, according to our knowledge.

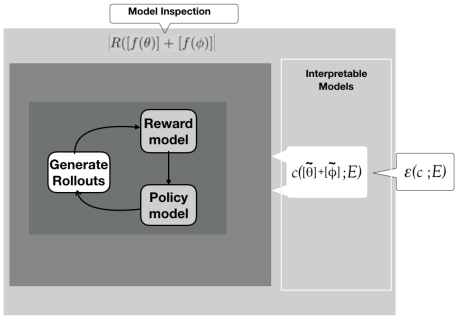

Fig. 3. Overall framework for the interpretable box design paradigm.

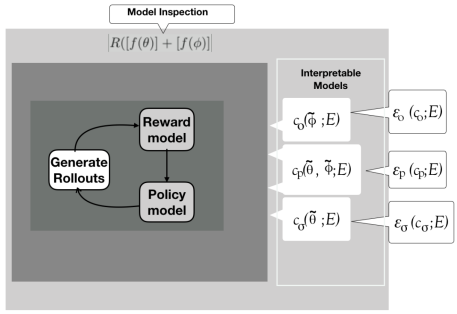

Fig. 4. The interpretable box design paradigm with specific interpretable models.

this latter case, ϕ realizes the weights of the reward function $r_\phi(s_t, a_t; E) = \phi^{\mathrm{T}}\mu(s_t, a_t; E)$, where μ are functions on features in M and action parameters. As functions μ become arbitrarily complex, they hinder the interpretability of the reward model.

Interesting variations of the XDRL framework provided in Figure 1 are frequently met in the literature, where interpretable models c either (a) substitute inner constituent models of the DRL method (thus making the DRL method "inherently" interpretable) or (b) are being trained using the DRL agent and its models as oracles to provide explanation content. These cases, in close relation to definitions provided in the work of Guidotti et al. [25], aim to solve the *transparent box design problem*. In this article, these variations are considered to offer distinct paradigms for building interpretable models, according to what we call the *interpretable box design paradigm*.[3]

Interpretable models, as Figures 3 and 4 show, are parameterized using variables' vectors $\tilde{\theta}$ for policies and $\tilde{\phi}$ for rewards (possibly in conjunction to other models), distinct from the parameters of the DRL models, if separate DRL models exist. It must be noted that interpretable models in these cases do not necessarily exploit DRL models' elements by means of $f(\cdot)$. As a consequence, although model inspection facilities may be provided for the constituent DRL models, these in some cases are totally disconnected from the interpretable models.

Interpretable models using the DRL method as an oracle (the second variation mentioned previously) can be trained either during or after training the DRL models. Although while training interpretable models during training the DRL model may result in instability and inefficiency of the training process, aiming to reach a moving target, it may provide insights to the DRL learning process. The training process may use samples and results (e.g., rewards, state-action values) provided by the deep method, and it may also exploit the models of the DRL method in a direct manner, delving into their inner workings. This results into two distinct paradigms that refine the interpretable box paradigm: the *mimicking* and the *distillation* paradigms, respectively.

It must be noted that although the distillation process has been introduced as an NN model compression process through training a student NN, and has been introduced as a term in the work of Hinton et al. [46], the first work that introduces it for sequential decision making (as a model compression technique, not as a method for building interpretable models) is described in the work of Rusu et al. [45], where the teacher provides samples, each comprising an observation sequence and a vector of state-action Q-values.

[3]We do not use the term *transparent*, as we associate transparency with pragmatic aspects regarding the provision of explanations.

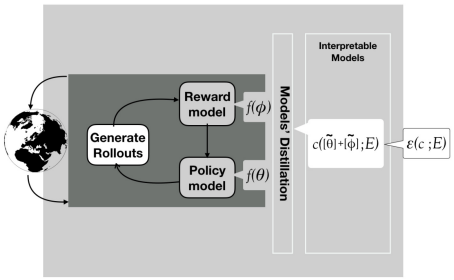

Fig. 5. The overall framework for distillation.

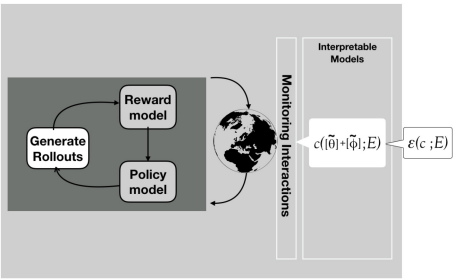

Fig. 6. The overall framework for mimicking.

For DRL methods, the distinguishing line between the distillation and the mimicking processes is not that clear in the literature. Here we consider these as two distinct paradigms for providing interpretability to DRL methods.

3.2 Distillation and Mimicking: Two Distinct Paradigms for Interpretability

The distinct paradigms are shown in Figures 5 and 6. The main difference between the two paradigms concerns the "receptive field" of the interpretation process. As Figure 5 shows, the distillation process gathers knowledge from the trained DRL agent, exploiting any of the constituent DRL models in a *direct* way through a process of transforming DRL models' elements into interpretable models. In contrast, the mimicking process, as Figure 6 shows, monitors the interaction of the RL agent with the environment, and gathers interaction samples, recording agent decisions, state transitions, rewards, and consulting DRL assessed values.

As an important note to the preceding discussion, one may argue that the architecture shown in Figure 5 is more general than the one shown in Figure 1. Actually, the architecture shown in Figure 1 abstracts all XDRL paradigms while providing *exact* interpretations of DRL models. The two paradigms offer interpretable models that are approximations of the DRL constituent models: the distillation paradigm shown in Figure 5 aims at exploiting the knowledge acquired by the DRL models via model inspection facilities, whereas the mimicking method in Figure 6 produces interpretable models without exploiting the inner-working components of DRL agents.

4 XDRL METHODS: STATE OF THE ART

This section aims at a detailed presentation of XDRL approaches. For each of the reviewed approaches, we describe the overall approach and motivating points, assumptions, and important technical details, and we conclude with important aspects with respect to the specific problem it addresses, the paradigm it follows, interpretable models it constructs, required access to constituent closed-box models, and, finally, information on evaluation objectives, methods, and results. In addition to these, the appendix summarizes the XDRL approaches described, focusing on important aspects of their functionality and applicability, using a comprehensive list of characteristics that includes those described previously, and additionally, the XDRL requirements for the DRL methods used, as well as "XDRL–DRL interplay aspects": access to DRL models' information and/or samples required, in an online mode or in an offline mode.

As a first note before delving into XDRL methods, there are many proposals for interpreting deep NN models, mainly through distillation and mimicking approaches. The main emphasis of these approaches is building high-fidelity interpretable models, in comparison to the original models. These approaches differ in several dimensions: (a) the targeted representation (e.g., decision trees in DecText [29], logistic model trees (LMTs) in the work of Dancey et al. [11], or gradient boosting trees in the work of Che et al. [13]), (b) to the different splitting rules used toward learning a

comprehensive representation, (c) to the actual method used for building the interpretable model (e.g., Dancey et al. [11] use the LogiBoost method and Boz [29] proposes the DecText method, whereas the approach presented in the work of Che et al. [13] proposes a pipeline with an external classifier, (d) on the way of generating samples to expand the training dataset. These methods can be used toward interpreting constituent individual DRL models employing (deep) NNs. We do not aim to review these in this article, given the thorough review provided elsewhere (e.g., [25, 55, 57, 58]). However, we delve into XDRL approaches that exploit any of these methods.

4.1 Solving the Model Inspection Problem

Here we provide a succinct review of prominent general methods addressing the DRL model inspection problem.

4.1.1 Why Should I Trust You? Ribeiro et al. [2] propose LIME for explaining the predictions of any classifier in an interpretable and faithful manner. LIME learns an interpretable model locally around the prediction, searching for instances in the vicinity of the original instance through perturbations of uniformly sampled features in subsets of M (i.e., non-zero elements). Emphasis is given to the qualitative understanding of correlations between input features and model responses, aiming to guarantee local fidelity, explaining how the model behaves in the vicinity of the instance being predicted, using a proximity measure. Thus, as LIME addresses the model inspection problem assuming binary vectors for instances' interpretable representations, it contributes to the outcome explanation problem solution, as it provides the importance of features in a local only context.

Building on explanations of individual predictions, Ribeiro et al. [2] propose the SP-LIME method by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a sub-modular optimization problem. SP-LIME provides a global perspective and contributes to solving the model explanation problem.

Experiments performed with simulated users regard (a) the faithfulness of explanations to the model, given a set of gold features that explanations must recover; (b) the trust on predictions, given untrustworthy features in predictions; and (c) the trust on the model, given a set of trustworthy examples covering cases with varying subsets of important features. Evaluations with human subjects regard the accuracy of choosing the best classifier and improving the classifier by eliminating features that are untrustworthy (measuring classification accuracy after rounds of interaction).

LIME provides a generic (model-agnostic) method that contributes to local and global (model) interpretability. In the context of DRL, it can be used for explaining any of the models (e.g., to explain local decisions for selecting an action given an interpretable representation of states' features), although not being designed for that particular task. Functions $f(\cdot)$ provide binary vectors for instances' representations, whereas c_l (for outcome explanation) can theoretically be any interpretable model (a linear model, a decision tree, etc.). Surface representations of instances' interpretable representations (for model inspection, via R), as well as explanation logic ϵ_l for local explainability and ϵ_g for model explainability, are provided using visual means.

Extensions of LIME include Anchor [31] that uses decision rules as local interpretable classifier c_l, and LORE [32] that learns a local interpretable predictor c_l on a synthetic neighborhood using a genetic algorithm.

4.1.2 Shapley Additive Explanations. The **Shapley Additive Explanations (SHAP)** approach aims to provide a unified approach to interpreting model predictions, showing how interpretation methods are related, and when one method is preferable over another. Lundberg and Lee [26] identify the class of additive feature attribution methods with a linear function of binary variables as an explanation model: $g(z) = \phi_0 + \sum_{i=1}^{SIF} \phi_i z_i$, where $z \in \{0,1\}^{SIF}$, SIF is the number of simplified input features, and $\phi_i \in R$.

There is a family of methods—including LIME, DeepLIFT [33], **layer-wise relevance propagation (LRP)** [34], and classic Shapley value estimation methods—that are additive feature attribution, and Lundberg and Lee [26] unify six of the existing methods. They provide theoretical results on the existence of a unique solution in this class of methods with a set of desirable properties that several recent methods in the class lack: *local accuracy*, requiring the explanation model to at least match the output of the original model; *missingness*, representing features presence so as to locate the features with no prediction impact on a local decision; and *consistency*, regarding the relation of the input's contribution and the input's attribution.

Specifically, considering a simplified input x' mapping to the original input x through the mapping function h_x: $x = h_x(x')$, and input $z' \approx x'$, with S to be the set of non-zero indexes in z' (z_S has missing values for features not in S) subject to $h_x(z') = z_S$, SHAP values are the Shapley values of a conditional expectation function of the original model $E[pred(z)|z_S]$, given the model prediction $pred(z)$. SHAP focuses on explaining a prediction $pred(x)$, given a model $pred$ with input x and a simplified input x'. Such local methods ensure that $g(z') \approx pred(h_x(z'))$, whenever $z' \approx x'$, where g is the local interpretable model c_l (for outcome explanation). SHAP, similarly to LIME, provides an interpretable model as a linear function of binary vectors for instances' representations provided by $f(\cdot)$. Surface representations R of instances' interpretable representations for model inspection, as well as explanation logic ϵ_l for local explainability, are provided using visual means.

Based on insights from unifying methods, Lundberg and Lee [26] present methods that show improved computational performance and/or better consistency with human intuition than previous approaches. Evaluation of Kernel SHAP vs. LIME and Shapley sampling values show improvement in terms of learning accurate estimates with fewer evaluations of the original model (sample efficiency). User studies in a limited setting show that SHAP values prove more consistent with human intuition than alternative feature importance allocations represented by DeepLIFT and LIME, which fail to meet the desirable properties identified. Finally, the authors use MNIST digit image classification using a convolutional network to compare SHAP with DeepLIFT and LIME, showing that SHAP is able to explain differences in classifying a digit to a class rather than to a near-class, measuring the probability of classifying instances to classes and changes in log-odds when masking over images.

A concrete example of employing SHAP (actually, a specific instance of LIME, named *Kernel SHAP*) in the context of DRL, is provided in the work of Rizzo et al. [38]. This article uses policy gradient to implement a traffic signal control agent on a signalized roundabout. Kernel SHAP estimates the effect of the road detectors state on the agent selected phases, interpreting the decision taken, in relation to the traffic volumes and the lanes occupancy.

Other methods, such as LRP [34] and neighborhood component analysis (NCA) [8] can be used for addressing the model inspection problem for any of the constituent DRL models, also providing (a) importance of features in M, and (b) reduction of predictors' dimensionality, which can facilitate data visualization and fast classification. Such methods are thoroughly reviewed in the work of Guidotti et al. [25].

Following this short presentation of model inspection general methods, we proceed to review XDRL methods addressing any of the model, outcome explanation, or interpretable box design problems.

4.2 Solving the Policy Explanation Problem

4.2.1 Summarizing Agent Behavior to People with HIGHLIGHTS. With the goal of increasing people's familiarity with the agent's capabilities and limitations, Amir and Amir [43] propose the HIGHLIGHTS and the HIGHLIGHTS-DIV algorithms that choose trajectories highlighting

important and distinct aspects of agent behavior. In doing so, this approach provides an overview of an agent's policy rather than explaining specific responses.

Specifically, the HIGHLIGHTS online algorithm constructs summaries of an agent's behavior by selecting (sub-)trajectories with important states. A state is considered important if taking a "wrong" action in that state (i.e., an action that is not prescribed by the policy) can lead to significant decrease in future rewards. To provide context to the user, for each important state the algorithms extracts a sub-trajectory including neighboring states (before and after the important states) and actions. To highlight behavior in various subspaces in the state space, HIGHLIGHTS-DIV avoids including in the summary (sub-)trajectories with very similar states, if these are equally important.

HIGHLIGHTS has been evaluated by playing the Pac-Man game, based on the observed behavior rather than scores. Q-values in this setting are defined as a weighted function of state feature values. Given different agents trained for different number of episodes, human participants were asked to select the best agent based on the summaries provided, while subjective opinions about the helpfulness of summaries were elicited, given summaries of the same agent. The surface representation of the explanation in the work of Amir and Amir [43] is provided by means of video clips, but in general, other means and information modalities may be used (e.g., as in the work of Van der Waa et al. [48] that is presented in a subsequent section). This evaluation resulted in observations regarding participants' confidence in choosing agents based on HIGHLIGHTS summaries. Experiments show that short summaries may not enable participants to correlate their confidence with their ability to assess the agent's abilities. However, with statistical significance and for very well trained agents, participants preferred summaries generated by HIGHLIGHTS over baselines, and summaries generated by HIGHLIGHTS-DIV over HIGHLIGHTS.

Overall, this approach addresses the policy model explanation problem, demonstrating agent behavior. It is important to point out that this approach bypasses the construction of an interpretable model c for policies, but it gathers explanations' content by selecting examples of the agent's behavior in important states. The representation of importance is realized via the function $f(\cdot)$, which has access to the agent's policy and Q-values.

Huber et al. [64] use a version of HIGHLIGHT-DIV in a combination of global and local explanations for DRL agents. Specifically, they use a version of the LRP [34] method (i.e., LRP with the *argmax* rule [70]) to augment trajectory summaries with saliency maps. This local approach is presented in detail in Section 4.4.11. LRP has been used to show what information is important to the agent toward a specific decision: the approach presented by Huber et al. [64] addresses the policy and the responses explanation problems by combining methods, contributing to our understanding of the role of saliency maps in the context of explaining agents' behavior. Although saliency maps have been shown to improve classification decisions in images (albeit with 60% of correctness in the work of Alqaraawi et al. [68]), their role in RL is not that clear (e.g., as shown in [3]). However, they may provide significant positive effects when combined with other methods (as in the work of Iyer et al. [69]). Empirical evaluation of the proposed combined method aims to investigate (a) the mental model of humans about the DRL agent (via a retrospection task), (b) humans' ability to assess agents performance (via an agent comparison task), and (c) participants' satisfaction with respect to the explanations presented (via answering explanation satisfaction questions). The results of this study reinforce prior findings for HIGHLIGHTS-DIV. Overall, in this study, the choice of states shown to humans was more important than the inclusion of local explanations in the form of saliency maps. For saliency maps, the results reported are mixed: there were no significant differences between the saliency and non-saliency conditions in the study. Showing saliency maps as part of a video is more challenging for humans in comparison to showing them in a still image. As the authors point out "[saliency maps] in RL [add a] layer of complexity as interpretation also requires making inferences regarding how the highlighted regions affect the agent's long-term,

sequential decision-making policy." However, in combination with behavior explanations, saliency maps put the important features in context, although they are not preferable when humans get highlights of agents' behavior.

Extending the HIGHLIGHTS algorithms, Sequeira and Gervasio [44] propose using not only state importance but also various interestingness elements so as to highlight important agents' behavioral aspects, increasing humans' understanding of the agent's capabilities and limitations. Such elements are (in)frequent situations, (un)certain executions, observation minima and maxima, and most likely sequences to maxima, each serving a specific explanation provision purpose. To compute these elements, the proposed framework collects data from the agent's interaction with the environment, such as the number of times the agent has specific observations and estimated transition probabilities, as well as estimated Q-values and V-values. Although there are elements (e.g., frequency) that may provide a good understanding of the agent's behavior characteristics, overall, results show that no single summarization technique—and hence no single interestingness element—provides a complete understanding of every agent in all possible situations of a task. Ultimately, different combinations may be required for agents with distinct capabilities and levels of performance.

4.2.2 Contrastive Explanations for RL in Terms of Expected Consequences. Motivated by findings indicating that contrastive explanations offer intuitive motivations for humans to understand why one performs an action instead of another [42], Van der Waa et al. [48] propose a method for providing contrasting descriptions of rollouts generated by an RL agent's policy and a policy that is driven by humans' (non-experts in RL) queries.

This proposal (a) defines a translation of states and actions in a set of descriptive states' classes and action outcomes, by training binary classifiers, similarly to Guidotti et al. [23], and (b) proposes a method for obtaining a foil policy π_f based on the foil in the user's query, suggesting potential alternative actions in a particular state s and consecutive state-action pairs. This foil policy, based on an update of the Q-function to reflect the user's preferences (and an informed update of the reward function) is contrasted to the fact policy π_t learned by the agent. The explanations are based on generating rollouts of specific length simulating the effects of π_t and of π_f, based on the environment's transition function and starting on a state before the state specified in the user's query. State-action pairs in rollouts are transformed in descriptions that are then used for generating the trajectory contrastive descriptions by instantiating natural language templates.

Hayes and Shah [23] report on a case study using a simple RL setting with a very restricted objective, addressing either the next action of the agent policy or the entire policy. They show that human participants prefer explanations using the policy, as in most cases the next action is evident to the user.

This approach addresses the policy model explanation problem by contrasting rollouts of the agent's policy to foil policy rollouts. The interpretable model c provides qualitative descriptions of rollouts by means of distinct state and action classes. The interpretation process accesses the MDP, which may be fitted by the learning process, the state-action Q-values learned, and the rewards received, via the function $f(\cdot)$. Finally, the explanation logic ϵ uses natural language templates that are instantiated by qualitative (potentially contrasting) descriptions of rollouts.

4.2.3 Establishing Appropriate Trust via Critical States. Motivated toward helping end users build a mental model of an agent's policies and capabilities, Huang et al. [47] propose approaches for computing critical states in a policy-agnostic way, requiring only access to action-value function Q. This proposal is driven by the insight that for many tasks the essence of the policy is captured by the agent's reactions in a few critical states.

A critical state s is one in which it is very important to take a certain action. As defined in the work of Huang et al. [47], $Q^{\pi}(s, a)$ varies greatly across different actions a—that is, $Q^{\pi}(s, a)$ is very high for some actions, but for most of them, this value is mediocre or low. Given the agent's policy, critical states are exposed to end users who can spot false-positive or false-negative critical states (i.e., states that are considered as critical by the agent but not by the humans and vice versa). Inspecting the critical states, the end users may refine their trust to the agent, or gain proper knowledge on the situations where they have to take control. For instance, both false-negative and false-positive critical states indicate that the robot has failed to learn something fundamental about the task. Similarly, if the policy identifies a true-positive critical state but is mistaken about which action is correct in that state, then the end user may not trust the policy.

This approach does not provide an explicit interpretable model but aims to provide a coherent view of the agent's capabilities and limitations so as to establish trust to the agent. Critical states are determined by accessing Q-values via the function $f(\cdot)$. The interaction with human experts concerns (a) spotting false-positive and false-negative states, or incorrect actions in critical states and assessing whether to deploy the agent and (b) deploying the agent operating with the user in the loop, who she is able to take control over whenever needed. A critical issue is whether observing a set of critical states leads participants to develop appropriate trust in high-entropy policies that generated those critical states, using the soft actor-critic (SAC) algorithm. In addition, although we may consider that this approach addresses the policy model explanation problem, it may be hard for the human to generalize and consider how the agent reacts in states that have not been presented as critical. And while the surface representation of states-action pairs is shown to be depicted using visual means, it is harder to do so in settings where the dimensionality of the state-action space is large.

4.2.4 Graying the Black Box: Understanding Deep Q-Networks. The main observation driving the work of Zahavy et al. [53] is that **Deep *Q*-networks (DQNs)** are learning an internal hierarchical model of the domain without explicitly being trained to it. Therefore, the authors propose a methodology and tools to analyze DQN, aiming to identify spatio-temporal abstractions directly from the learned representation in two different ways: first, manually clustering the state space using handcrafted features, and second, computing the **semi-aggregated Markov decision process (SAMDP)**, an approximation of the true MDP learned from data, by means of spatio-temporal abstractions. Visualization tools, including **t-distributed stochastic neighbor embedding (t-SNE)** that exploits directly the collected neural activations of states (a model inspection task), saliency maps, and analysis tools for understanding the common attributes of states' clusters and for analyzing dynamics between clusters, aim to provide facilities toward understanding, debugging, and interpreting the policy model.

Specifically, Zahavy et al. [53] explain the five stages toward building an SAMDP model aggregating states and skills, thus allowing analysis with spatio-temporal abstractions: (0) feature selection; (1) state aggregation, mapping the MDP feature space to the abstracted MDP state space enforcing temporal coherency among trajectories; (2) identification of skills (a.k.a. sub-trajectories) from the data; (3) inference of skill length, the SAMDP reward, and the SAMDP transition probabilities; and finally, (4) selecting the best among the SAMDP candidates, according to evaluation criteria proposed.

Overall, this approach proposes a methodology and abstraction methods addressing the policy model explanation problem for DRL methods, where the policy is learned by the DQN method. The interpretable model c_g is an SAMDP that is devised by consulting manually crafted state features, as well as rewards and neural activations of states provided by $f(\cdot)$. The explanation logic ϵ_g is implemented by t-SNE maps with representative states per cluster, and appropriate t-SNE-enabled

SAMDP visualizations. In conjunction to that, the authors demonstrate how to do (policy) model inspection by processing the network's neural activity using visualizations enabled by t-SNE.

4.2.5 Toward Better Interpretability in DQNs. Annasamy and Sycara [51] propose an interpretable NN architecture for Q-learning, which provides explanations for visual agents. The aim is to provide an understanding of the policy model, visualizing clusters of state representations in attention maps, and using saliency maps.

The proposed model learns a latent representation of states that captures important visual aspects of input images, and an association between representations of states and representations of keys in a key-value store using an attention mechanism. Each key represents an action and an associated Q-value, among N such values—thus, the network learns to associate states to "representatives" of (action,Q-value) pairs, which serve as cluster "exemplars." The attention weights over keys and their corresponding value pairs are used to calculate Q-values. Different types of losses are being linearly combined to train the network: Bellman error, distributive Bellman error, reconstruction error, and diversity (over keys) error. Uncertainty of the attention weights is being used to drive exploration during training, approximating an upper confidence on the Q-values. Through key inversion (which not a process driven by a certain input), the authors attempt to find important aspects of the input space that influence the choice of a particular action-return pair.

As concluded by Annasamy and Sycara [51], with a directed exploration strategy, the proposed model can reach training rewards comparable to the state-of-the-art deep Q-learning models. Specifically, the best agreement between the actions selected using the distributions in the image space and latent space is rather low, but the explanations provide useful insight into the kinds of features extracted by convolutional layers. This possibly suggests, according to the authors, that reconstructions rely heavily on generalizability to unseen keys. However, results suggest that the features extracted by the NN are shallow, and the agent does not model interactions between objects.

Concluding, this approach addresses the interpretable box design problem for deep Q-learning methods, focusing on providing global policy explanations for visual agents. The interpretable model c_g is an attention model associating state embeddings to key-value embeddings. The explanation logic ϵ_g provides keys—state embeddings clusters' visualizations using t-SNE for any specific Q-value, as well as images reconstructed via key-value (i.e., action, Q-value) pair inversion.

4.2.6 Interpretable Off-Policy Evaluation by Highlighting Influential Transitions. **Off–policy evaluation (OPE)** estimates the value of a policy using data collected under another policy. Aiming to enable human experts to assess and analyze the validity of OPE estimates, Gottesman et al. [62] propose a hybrid human-AI system that highlights observations in the data whose removal will have a large effect on the OPE estimate. Their proposal aims (a) to address the needs of real-world applications where the data itself might never be enough, and (b) to involve domain experts in the integration of decision support tools, incorporating their expertise into the evaluation process. In so doing, they formulate a set of rules for choosing which observations to present to domain experts for validation.

Specifically, in this article, the authors propose a framework for using influence functions to interpret OPE and discuss the types of questions that can be shared with domain experts to use their expertise in debugging OPE: influence functions show the impact that the removal of a transition has on state-action values or on the value of the evaluation policy. Based on this framework, they develop computationally efficient algorithms to compute the exact influence functions for several importance sampling estimators, as well as two broad function classes for fitted Q-evaluation (FQE): kernel-based functions and linear functions.

The benefits of influence analysis are demonstrated on a cancer simulator, with the objective to identify limitations in the evaluation process and make evaluation more robust. Furthermore, the authors present results of analysis together with practicing clinicians of OPE for management of acute hypotension from a real intensive care unit (ICU) dataset.

Overall, this approach proposes a framework that addresses OPE independently from any RL method. It is closer to the policy explanation problem, since the influence analysis proposed aims to interpret a policy evaluation method. The method can be applied to parametric and non-parametric fitted Q-evaluation models, regarding continue or discrete states and actions, although the presentation and evaluation considers discrete cases. The method focuses on the influence of specific state transitions to the value of the policy, and thus it can be considered as a method that reveals the contributions of local decisions to the trajectories generated by a policy model. The model exploits trajectories generated by the evaluation policy, although there is not any specific proposal for the explanation logic: different examples use different visualizations of transitions' influences.

4.2.7 Interpretable RL Using Attention Augmented Agents. Mott et al. [52] propose a spatial attention mechanism to visual information in an RL setting. The model enables agents to actively select important, task-relevant information from visual inputs, by sequentially querying and receiving compressed, query-dependent summaries. Actually, the model generates attention maps that can uncover the underlying decision process. By exploiting the attention maps, one can understand how the system solves a task, identifying the type of entities that the agent attends ("what") and spatial locations where the agent focuses attention ("where").

Succinctly, the proposed model works as follows. Observations pass through a recurrent vision core network, producing a "keys" and a "values" tensor, to both of which a spatial basis tensor (encoding information about spatial positions) is concatenated. A query network produces a set of query vectors taking as input the state of an LSTM recurrent network (policy network) from the previous timestep. The softmax of the pixel-wise inner product between each query vector and each location in the keys tensor produces an attention map for the query. The attention map is broadcast along the channel dimension, and it is point-wise multiplied with the values tensor. The result is then summed across space to produce an answer vector. This answer is sent to the LSTM to produce the next LSTM state.

The performance of the proposed method is evaluated across a broad range of ATARI levels. Attention maps are used to visualize which parts of the input are attended. The authors' conjecture that the top-down nature of the attention (i.e., the fact that the attention map is produced by taking as input the LSTM states) provides a large performance gain compared to equivalent, bottom-up attention-based mechanisms. Comparison of the attention maps to alternate methods for visualizing saliency shows that they allow more comprehensive analysis of the information the agent is using to inform its policy. The agent is able to make use of a combination of "what" and "where" queries to select both regions and entities within the input, depending on the task. Experiments show that the agents are able to learn to focus on key features of the inputs, look ahead along short trajectories, and place tripwires to trigger certain behaviors.

This approach proposes a model addressing the interpretable box design problem for DRL methods applied in visual inputs, where the policy is modeled by an LSTM network. The method focuses on providing global policy explanations. The model exploits LSTM states to produce queries, as well as keys and values tensors produced by the visual input—thus, we could say that the query network and the vision core serve as the $f(\cdot)$ to feed the interpretation component. The interpretable model c_g is a soft attention model and the explanation logic ϵ_g is implemented by visualizing the attention maps, showing the original input frames and super-imposing the attention map for each attention head. In addition, saliency maps, visualizing the relative dominance of "what" and "where" parts of the query are being used, to inspect where the agent focuses attention.

4.2.8 Conservative Q-Improvement. Decision trees that represent policies span the state space and may be larger than required, even if they do express the action value function quite accurately. Roth et al. [21] aim to provide an answer to the question "what should be the size of such a decision tree" to make it accurate and interpretable? This work is based on the hypothesis that the size should be that which if increased further it does not increase significantly the estimated discounted future reward of the overall policy.

In doing so, Roth et al. [21] propose the **Conservative *Q* Improvement (CQI)** RL algorithm. CQI learns a policy in the form of a decision tree, in any domain with discrete actions and multi-dimensional state space. In contrast to other RL methods (e.g., [35]) that learn a decision tree policy representation, this approach uses a lookahead approach that predicts which split will produce the largest increase in reward. The method is strictly additive: initialized with a single leaf node, over time, it creates branches and leaf nodes by replacing existing leaf nodes with a branch node and two child leaf nodes. A branch node represents abstract states and indicates actions to be taken using the Q-function. A leaf node is split (and turned into a branch node) only when the expected discounted future reward of the new policy would increase above a dynamic threshold, moderated by the node visit frequency and the dynamic split threshold.

CQI is evaluated in the RobotNav domain, where a robot must navigate in a 2D environment to reach a goal location, avoiding obstacles. It is shown that the method is able to produce substantially smaller trees compared to the method of Pyeatt and Howe [35], with smaller variance and of greater reward. No explanation logic ϵ is proposed by Roth et al. [21], and this article does not evaluate the method in terms of interpretability or explainability.

CQI follows the interpretable box design paradigm: it aims to learn an interpretable policy model c_g in a direct way toward increasing explainability.

4.2.9 Hierarchical Interpretable RL. Shu et al. [28] propose a framework for efficient multi-task RL, training agents to employ hierarchical policies. These policies support an agent to decide when to use a previously learned policy and when to learn a new skill. Each learned task has a corresponding human language description, thus represented in an interpretable way. Through the hierarchical model proposed, one may (a) accumulate tasks progressively from a terminal policy to a top-level policy, and (b) unfold the global policy from top-level to basic actions, using human language descriptions. This approach not only learns the language grounding for visual knowledge and policies but also is trained to utter human instructions as an explicit explanation of its decisions to humans.

More specifically, training progresses at stages, increasing the set of tasks from $\mathcal{G}_0$ to $\mathcal{G}_k$ at stage k, and the learned policies from π_0 for $\mathcal{G}_0$, to π_k for $\mathcal{G}_k$. At stage k, the policy π_k is defined by a hierarchical policy that consists of four sub-policies: a base policy for executing previously learned tasks (the policy at stage $k-1$), an instruction policy that manages communication between the global policy and the base policy (mapping a task $g \in \mathcal{G}_k$ to a base task $g' \in \mathcal{G}_{k-1}$), an augmented flat policy that allows the global policy to directly execute actions instead of using a base task, and a switch policy that decides whether the global policy will primarily rely on the base policy or the augmented flat policy. A stochastic temporal grammar (STG) model is trained on the sequence of policy selections (previously learned skill or new skill) and focuses on priorities among tasks: it interacts with the hierarchical policy through modified switch policy and instruction policy. Learning at each stage is a two-phase curriculum learning task: policies in each phase are learned with an advantage actor-critic (A2C) off-policy method, assuming that the terminal policy π_0 has been trained, with base skill acquisition (learning to use previously learned skills) and novel skill acquisition phases (learning to rely on the augmented flat policy for executing novel tasks).

The approach is evaluated in a two-room environment in Minecraft with 24 tasks in total. Experiments aim mainly to show learning efficiency and generalization abilities. Hierarchical plans of

several tasks generated by global policies learned by the full model are visualized. However, these visualizations are mainly used to show the agent's learning efficiency rather than the provision of explanations.

Concluding, this approach follows the interpretable model design paradigm, training interpretable models that are hierarchical plans, where skills and tasks are described in natural language based on human instructions.

4.2.10 LVIN: Imitation Learning Using the Value Iteration Networks Approach. The **logic-based value iteration network (LVIN)** proposed by Leech [10] combines imitation learning and learns to represent problems as compact **finite state automata (FSA)** with human-interpretable logic states. The aim is to facilitate understanding and manipulation of the learned model, toward adding trust and flexibility to robotics applications trained with LVIN.

LVIN produces a policy that describes desired actions in terms of transitions between human-interpretable logic states. In doing so, the planning problem posed by the decision layer is decomposed into two pieces: a high-level FSA that corresponds to high-level sub-goals the robot follows (described in logic) and a low-level MDP describing the motion of the robot in the physical environment. The FSA learns the rules governing the robot's behavior in terms of interpretable and manipulable logic states, and transitions, called *propositions*. LVIN comprises separate value iteration networks for each FSA state, where learning the state transitions associated with the high-level FSA involves encoding the order of propositions to trigger given the current FSA state, and learning the $P(s'|s, a)$, $s \in S$ and $a \in A$, associated with the low-level MDP. LVIN learns the entire model via an iterative update process, similar to value iteration networks, where the update procedure is encoded in a deep NN.

Evaluation aims to show the success rate (i.e., the percentage of correctly completed tasks) of LVIN compared to a CNN trained with a map as input and the desired action as output. The method has been evaluated in a lunchbox packing and in a cabinet checking problem, with no emphasis on evaluating interpretability or explainability.

LVIN addresses the model explanation problem for discrete actions and in a 2D space. The interpretable global model c_g is the high-level FSA. In this case, the function $f(\cdot)$ provides the "raw" model (deep NN) parameters (i.e., making no interpretation). Since the high-level FSA and the low-level MDP are trained "seamlessly" taking as input the same expert demonstrations, learning the interpretable model can be considered as a mimicking process. LVIN does not propose any specific explanation logic for realizing the function ϵ_g. Abstract policies in terms of logical states and transitions explain low-level policies, assuming that these abstractions are interpretable by humans. As the authors point out, "the current LVIN framework . . . is still somewhat limited by the fact that it requires a hard coded logic oracle for training."

4.2.11 Interpretable DRL with Linear Model U-Trees. Liu et al. [12] introduce **linear model U-trees (LMUTs)** to approximate NN predictions for DRL agents. An LMUT is learned by an online algorithm that is well suited for an active play setting, where the mimic learner observes an ongoing interaction between the DRL and the environment. Their article shows how the interpretable tree structure of an LMUT facilitates analyzing feature influence, extracting rules, and highlighting the super-pixels in image inputs.

U-tree [36, 37] is a classic online RL method that represents a Q-function using a tree structure. In addition to that, continuous U-tree (CUT) [37] dynamically generates a tree-based discretization of the input signal and estimates state transition probabilities by retaining transitions in every leaf node. To strengthen the generalization ability of these representations, as well as their learning efficiency, LMUTs [12] add a linear model to each leaf node. This can be trained either in a batch mode or in an active play setting. LMUT applies stochastic gradient descent to update the linear

models, given some memory of recent input data stored on each leaf node. In the active play setting, the method exploits transitions in the form of (*state*, *action*, *Q-value*, *next-state*, *reward*), where the *action* is decided by the DRL model, together with the "soft" Q-value, whereas the *reward* is provided by the environment.

LMUT builds an MDP from the interaction data between the environment and a deep model. Compared to a linear Q-function approximator, LMUT (corresponding to an action) defines an ensemble of linear Q-function models, one per leaf node and one for each state partition cell. Since each Q-value prediction comes from a single linear model, the prediction can be explained by the feature weights of the model.

This approach has been evaluated toward mimicking a DQN model, and compared against CART, M5 with regression tree, the fast incremental model tree (FIMT) baseline, and FIMT with adaptive filters (FIMT-AF). The evaluation environments include Mountain Car, Cart Pole, and Flappy Birds, all with discrete actions but with discrete or continuous states. Evaluation concerns approximating the Q-values in DQN, reporting on the number of the tree leaves constructed. Furthermore, the sampling efficiency of LMUT is evaluated by computing the correlation and testing error of LMUT, as more transitions are provided. Interestingly, LMUT models are evaluated in terms of the average reward per episodes by performing the tasks (e.g., playing games) in a direct manner: it is reported that LMUT, among other mimic models, achieves performance that is closest to the DQN. As far as interpretability is concerned, the authors propose (a) a way to measure splitting features' "importance" using the total variance reduction of the Q-values, (b) a way to extract and present rules in the form of partition cells, each cell representing a range in specific splitting features, and (c) a way of highlighting pixels (when DRL takes raw pixels as input) while explaining local decisions. However, these options are not evaluated in any setting with human subjects.

Concluding, this is a mimicking approach, where the interpretable model is trained with transitions caused by the interactions of the DRL agent with the environment, also consulting DRL model decisions and values, via the function $f(\cdot)$. The interpretable model represents the agent policy, and as mentioned earlier, it can be exploited in several ways to provide interpretations. The article does not suggest any particular function ϵ implementing an explanation logic.

4.2.12 Policy-Level Explanations for RL. To address the need of explaining sequences of decisions, Topin and Veloso [7] introduce **abstracted policy graphs (APGs)**. APGs are Markov chains of abstract states and are assumed interpretable representations of policies, concisely summarizing them so that individual decisions can be explained in the context of expected future transitions. The authors propose the APG Gen method for generating APGs for deterministic policies given a learned value function and a set of observed (potentially off-policy) transitions.

An APG is a Markov chain over abstract states where edges are transitions induced by actions from the original MDP. The basic assumption here is that if the agent's transitions between grounded states are approximated using a Markov chain between abstract states, then grounded states that are *treated similarly* (and thus can be *treated interchangeably*) are readily identified (i.e., in groups) and the agent's transitions between abstract states can be predicted. Interchangeable ground states grouped into abstract states should have similar future outcomes in terms of rewards, which are assumed known, and the importance of all features for a set of similar states should be low. The importance of a feature is calculated by the feature importance ranking measure (FIRM), as the variance of the conditional expected score of a state s for that feature, with respect to an arbitrary function $g(s)$. This score is the average value of $g(s)$ for all states, where that feature takes a specific value v.

APG Gen [7] uses states' similarity based on actions taken under the policy, to repeatedly divide abstract states along important features, and then computes transition probabilities between them.

Important features can be trivially recorded for each abstract state, allowing for humans to inspect which features affect the agent's decisions, also providing a summary of an abstract state and of the corresponding ground states.

APG is evaluated in an abstraction of a production task for a multi-component item using a number of manufacturing steps, in deterministic and stochastic settings. Policies are computed using value iteration. In providing local explanations, APG is evaluated in terms of the features' prediction accuracy (important vs. not important) as a function of the portion of states sampled: this provides a view on how APG generalizes on features' importance, beyond the states seen. Additionally, n-hop action predictions are evaluated to measure the error caused by assuming that only single actions are performed for a transition between abstract states. This is measured by the action prediction agreement (predicted vs. actual) for increasing time horizon. Finally, the size of "explanations" are measured by the number of abstract states (nodes in APG) against the number of grounded states in the MDP (reported sub-linear). No evaluation with human subjects in simulated or real-world settings are reported.

Concluding, APG Gen is a mimicking approach, with the objective of providing an interpretable model of a learned policy, considering a trained policy model. That model is realized by an APG. APG Gen takes as input transition samples (s, a, s') from the learned policy, the value function, and features' importance. APG is the interpretable model, and local explanations may be provided consisting of the features that are important for a specific state. Topin and Veloso [7] do not propose an explanation function ϵ for providing interpretations' surface representations.

4.2.13 Programmatically Interpretable Reinforcement Learning. **Programmatically interpretable reinforcement learning (PIRL)** is an RL framework [27] designed to generate agent policies that are represented using a high-level, domain-specific programming language. Such programmatic policies are not as performant as those from DRL methods, but they are interpretable and being amenable to verification by symbolic methods. Verma et al. i [27] propose the **neurally directed program search (NDPS)** method for solving the challenging non-smooth optimization problem of finding a programmatic policy with maximal reward, minimizing the distance from an "oracle" DRL policy.

Specifically, considering a **partially observable Markov decision process (POMDP)** setting, a functional language, and given a policy sketch that syntactically defines a set of programmatic policies, the objective is to find a program in this set with maximal long-term reward. To restrict the search, the NDPS algorithm performs directed policy search guided by an oracle policy learned by a DRL method, via local search. NDPS measures the "distance" between the outputs of a programmatic policy and the outputs of the oracle policy, given a set of "interesting" histories that are enriched with additional histories generated by the programmatic policy (a technique that in the work of Verma et al. [27] is called *input augmentation*). Further optimization improves the programmatic policies found (e.g., being shorter).

PIRL is evaluated on the task of learning to drive a simulated car in the TORCS car-racing environment, using as oracle a policy computed by the Deep Deterministic Policy Gradient (DDPG) algorithm: NDPS policies are compared to human-readable policies, also showing that PIRL policies can have smoother trajectories, and better generalization abilities compared to the policies discovered by DRL. No evaluation of the explainability or interpretability of PIRL is provided, with or without human subjects.

Concluding, PIRL with NDPS is a mimicking process, exploiting agents' interactions with the environment to find optimal interpretable programmatic policies that are presented as functional programs.

4.2.14 Understanding Decisions by Interpretable Policy Learning. INTERPOLE is an imitation learning method that jointly models the agent's policy (belief-action mapping) and belief-update process, toward interpreting demonstrated behavior. As such, it can be used to model any "expert" demonstrated policy, without assuming unbiasedness of an agent's beliefs or optimality of policies. In so doing, Hüyük et al. [61] proposed an inherently interpretable imitation learning method, locating factors that contribute to individual decisions, in a language that is readily understood by domain experts, accommodating partial observability and operating offline.

INTERPOLE aggregates sequential observations through the agent's subjective belief-update process (decision dynamics) and probabilistic belief-action mapping for determining actions (decision boundaries). The objective is to model the most likely parameterizations $\theta = (T, O, b_1, \eta, \{\mu_a\}_{a \in A})$, drawn from some prior $\mathbb{P}(\theta)$—that is, the likelihood of θ with respect to a given set of demonstrated observation trajectories $\bar{\mathcal{D}}$ and unobserved state trajectories $\mathcal{D}$: $T(s_{t+1}|s_t, a_t)$ is the state transition probability, $O(z_t|s_t, a_t, s_{t+1})$ is the observation's probability after a transition to a new state, $b_1(s)$ is the probability of being in state s in $t = 1$, the smoothness of transitions between decision boundaries is tuned by the temperature factor η, and $\{\mu_a\}_{a \in A}$ are actions' mean vectors inducing decision boundaries over the belief space. Parameters θ are jointly determined by means of an expectation-minimization (EM)-like algorithm for maximizing $\mathbb{P}(\mathcal{D})$, given that $\mathcal{D}$ is not known. Estimating jointly the parameters θ, INTERPOLE learns the agent's decision and belief dynamics (not the environment dynamics) offering the most plausible explanation of how the agent reasons.

INTERPOLE has been evaluated using simulated and real-world demonstrated trajectories for disease diagnosis, focusing on interpretability, accuracy of learned policies, and subjectivity toward recovering interpretations of behavior even if the agent is biased. For interpretability, INTERPOLE has been evaluated by nine clinicians focusing on the utility of representing possibly subjective action-belief trajectories instead of raw action-observation trajectories, and on understanding policies by means of suboptimal decision boundaries, instead of using a reward function: the majority of clinicians preferred the explanations offered by INTERPOLE. Regarding accuracy, INTERPOLE is evaluated in modeling the belief-update process and the belief-action mapping (policy), learning high-quality models driving the agent's subjective reasoning.

INTERPOLE [61] addresses the policy explanation problem, offering a method for imitating agents' possibly suboptimal policies without assuming unbiasedness beliefs and objective reasoning. The interpretable model c_g comprises decision dynamics and decision boundaries and is specified by $\theta = (T, O, b_1, \eta, \{\mu_a\}_{a \in A})$ without assuming a fully observable setting, given a set of demonstrated action-observation trajectories $\bar{\mathcal{D}}$. The explanation logic ϵ visualizes a belief simplex in low-dimensional spaces, showing decision boundaries and belief trajectories.

4.2.15 Automatic Discovery of Interpretable Planning Strategies. With the aim to assist human experts in decision making, Skirzyński et al. [65] introduced the Adaptive Imitation-Interpretation (AI-Interpret) algorithm for transforming policies into sets of interpretable descriptions by means of decision rules that have a disjunctive normal form (logical program policies), presented as flowcharts.

Specifically, given a set of demonstrated trajectories $\mathcal{D}$ produced by the original policy, a domain-specific language specifying the predicates for constructing the logical program policies, a parameter α defining the threshold at which the discovered policy (represented by a logical formula) is at least better than the original policy, and the maximum rule length (d) of the logical formula, the objective is to find the logical formula that maximizes the number of demonstrations in $\mathcal{D}$ that it can imitate with respect to α and d. AI-Interpret uses the predicates to separate the set of demonstrations into clusters. Clusters (whose number is determined automatically) are

evaluated by means of a heuristic value that shows the similarity of trajectories in the cluster and how representative the cluster is. Doing so, the algorithm can consider increasingly smaller sets of demonstrations (disregarding less important clusters) and employ **logical program policies (LPPs)** [67] in a structured search searching for the simplest logical formula with respect to α and d that imitates most of the demonstrated trajectories. Finally, the formulas are transformed into decision trees and then visualized as flowcharts that people can follow to execute the strategy.

AI-Interpret is evaluated in three types of challenging planning tasks. The central question is whether discovered flowcharts can support decision makers in the process of planning. To achieve that, Skirzyński [65] perform one large behavioral experiment for each of the three types of tasks and a fourth experiment in which they evaluate whether AI-Interpret improves human decision making against conventional training (i.e., by providing feedback). Results show that the proposed approach (a) allows people to largely understand the automatically discovered strategies and use them to make better decisions, and (b) is more effective than conventional training, improving human decision making.

Overall, AI-Interpret [65] addresses the policy explanation problem, discovering policies described by means of logical programs. This is a mimicking approach, as the input it requires are demonstrations from the closed-box RL policy model. The interpretable model c_g is the LPP discovered, and the explanation logic e_g transforms LPP to flowcharts.

4.3 Solving the Objectives Explanation Problem

4.3.1 Enabling Robots to Communicate Their Objectives. Huang et al. [30] conjecture that, end users, to understand and predict what the agent will do, need to have an accurate mental model of the agent's objective function. To reduce the time needed for this otherwise time-lengthy process, the authors propose to model how people infer objectives from observed behavior, for agents to demonstrate behaviors that are maximally informative.

Following an algorithmic teaching approach, Huang et al. [30] define candidate models of human inference regarding a robot's objective function, given the robot's optimal behavior in specific environments: these demonstrations increase the probability of humans to infer the correct objective function. The objective function is assumed to be a linear combination of (possibly complex functions on) environment features in M, weighted by parameters ϕ^*. Features are supposed to be known by humans (although not always the case), who need to learn the true reward parameters. This involves searching for a sequence of environments $E_{1:n}$ such that when the person observes the optimal trajectories in these environments, their updated belief places maximum probability on the correct reward parameters. The challenges tackled to solve this optimization problem are as follows: (a) humans are likely to be approximate in their inference, whereas (b) the agent needs to incorporate a model of this approximate inference, (c) encouraging full coverage of all possible strategies that the agent is capable of adopting. Updates of humans' beliefs for the reward parameters can be modeled either via an inverse RL technique (exact method) or via Bayesian inference (approximate method), modeling how probable humans consider trajectories to be, given their beliefs on the reward function. To address the coverage challenge, a new term is added to the formulation of the optimization problem, ensuring that no extra trajectory examples are selected unless these examples significantly increase the probability of inferring the true reward parameters. Extra examples are chosen to be the best with respect to the best approximate model.

Experimental results evaluate how approximate and exact inference models perform in a driving simulation environment. The aim is the system to provide to participants examples of how the car drives, with the goal of being able to anticipate how it will drive when they ride in it. The analysis of models is done with ideal users and with human subjects. Analysis shows that although not

just any approximate inference model is better than the exact one, choosing a suitable approximate model helps significantly. The user study on coverage concludes that coverage with the right model of approximate inference has a significant advantage over RL models assuming exact inference users.

Concluding, Huang et al. [30] provide methods that address the objective model explanation problem, focusing on teaching the agent's reward function true parameters via examples. The proposed approach explicitly incorporates a model of the human beliefs on reward parameters and aims to provide a view on the true objective function. Although the interpretable model can be considered to be the linear objective function, no explanation logic is provided here.

4.3.2 Explaining Agent Behaviour Through Intended Outcome. Yau et al. [59] aim to interpret decisions by describing agents' intended outcome. This method applies to RL methods that estimate values off/on policy, given discrete states and actions and a deterministic MDP. The approach is demonstrated to off-policy Q-learning and aims to recover the future events that impact the Q-values learned.

This approach learns a *belief map* of discounted expected sum of state visitation $\mathbf{H}$. $\mathbf{H}$ is updated when a Q-value is updated, as follows: $\mathbf{H}(s_t, a_t) = \mathbf{H}(s_t, a_t) + \alpha(1_{s_t,a_t} + \gamma\mathbf{H}(s_{t+1}, argmax_a Q(s_{t+1}, a_{t+1})) - \mathbf{H}(s_t, a_t))$, where $1_{s_t,a_t}$ denotes an indicator function with 1 if the agent is at (s_t, a_t) and 0 otherwise.

Given a deterministic reward function R_{s_t,a_t}, the belief map $\mathbf{H}$ is consistent with Q if $vec(\mathbf{H}(s,a))^T vec(R(s,a)) = Q(s,a), \forall(s,a) \in S \times A$. The authors prove that if $\mathbf{H}$ and Q are zero initialized, then $\mathbf{H}$ and Q are consistent for all iterations of the algorithm in the tabular case. For deep Q-learning methods, there is no guarantee for consistency of $\mathbf{H}$ with Q.

The proposed approach is evaluated in domains with deterministic or stochastic reward functions and with continuous or discrete state-action space. In environments with stochastic rewards and continuous states, new discrete states are added, or states are discretized. The method is demonstrated to work effectively for tabular and deep Q-learning agents.

This is a generic approach that addresses the objective model explanation problem, providing projections of future intended trajectories of agents. The interpretable model is provided by the belief maps, and explanations are provided by means of belief map visualizations showing the value of each intended future state-action pair. The method is limited to low-dimensional state-action spaces.

4.3.3 Distal Explanations for Explainable RL. Madumal et al. [20] emphasize on the role of opportunity chains for explaining agents' behavior and introduce a distal explanation model that can generate opportunity chains as explanations for model-free RL agents. An opportunity chain takes the form of "A enables B and B causes C," where B is the distal event or action, and can inform the explainee about long-term dependencies between events, when certain events enable others.

Madumal et al. [20] proposes distal explanation model that learns opportunity chains using a recurrent NN. As distal explanations by themselves would not make a complete explanation, this approach uses action influence models [16] that approximate the causal model of the environment relative to actions taken by an agent, to get the agent's goals. It further improves upon action influence models (specifying the effects of actions in features) using decision trees instead of structural equations used in another work by Madumal et al. [16] (presented in Section 4.4.7) to represent the agent's policy. The decision tree policy model is trained concurrently with the RL policy model, assuming a model-free algorithm and exploiting state-action samples using an experience replay buffer. The tree is constrained to have a max number of leaves, equal to the agent actions. The same dataset of samples exploited for learning the decision tree model is exploited by a prediction recurrent NN to approximate distal actions and their cumulative rewards.

The accuracy of distal prediction and counterfactuals is evaluated in six benchmark domains using different model-free RL algorithms. Benchmarks have a mix of complexity levels and causal graph sizes (number of actions and state variables). Results show that the distal explanation model is robust and accurate across different environments and algorithms. Furthermore, experiments with humans using RL agents report on task prediction (i.e., understanding of the agent) and subjective explanation satisfaction. Agents are trained to solve (a) an adversarial task, (b) a search and rescue task, and (c) a human-AI collaborative build task, all in Starcraft II. The results obtained for explanation quality show that distal explanations perform substantially better than baselines in human-agent collaborative tasks.

Concluding, this is a mimicking process, where the interpretable models (decision tree and opportunity chains) are trained by exploiting the RL agent's interactions with the environment. The interpretable models comprise a policy model c_g using a decision tree, and a causal model representing the objectives (c_o), as well as the importance of specific responses using opportunity chains. The explanation logic ϵ exploits models to produce explanations in natural language.

4.4 Solving the Outcome Explanation Problem

4.4.1 Reward Decomposition. The approach proposed by Juozapaitis et al. [6] decomposes rewards into sums of semantically meaningful reward types so that actions can be compared in terms of trade-offs among the types, with a focus on explainability. In particular, this work introduces the concept of minimum sufficient explanations for compactly explaining why one action is preferred over another in terms of the reward types.

Specifically, assuming that the MDP formulation incorporates a set of reward components/types C and defining a vector-valued reward function with a component for each $c \in C$, $R_c(s,a)$, the agent objective is to optimize the overall (mixed) reward function $R(s,a) = \sum_{c \in C} R_c(s,a)$. This vector-valued reward allows for defining a vector-valued Q-function, with action-value components $Q_c(s,a)$ for rewards of type c. The overall Q-function is defined to be $Q(s,a) = \sum_{c \in C} Q_c(s,a)$. This results in a straightforward heuristic tabular algorithm updating Q-values in a standard way for each of the components, called *HRA*. The definitions are extended to DRL settings, where each Q_c is parameterized by means of θ_c. This results in DRL algorithms for decomposed reward (dr), drDQN and drD-SARSA.

To gain insight into why an agent prefers action a_1 over a_2 in state s (i.e., $Q(s,a_1) > Q(s,a_2)$), this approach uses the difference explanation (RDX) as the difference of the decomposed Q-vectors. RDX is exploited to find the minimal sufficient explanations MSX^+ and MSX^- for each pair of actions and state, indicating the "critical" positive and negative reasons (in terms of reward types), respectively, for the preference.

Case studies in two environments illustrate the potential of visual explanations in the hands of an RL practitioner. The authors show how decomposed reward methods support spotting certain "bugs" in the agent's action values and even help identify a more subtle issue related to the interaction of the gradient optimizer and the DRL loop.

Concluding, this approach addresses the outcome explanation problem, focusing on explaining local(i.e., in specific states) decisions on actions against other actions, exploiting reward types. The interpretable model c_l takes the form of tuples (MSX^+, MSX^-) per state and applicable action couples, exploiting (decomposed) action values provided by $f(\cdot)$. The explanation logic ϵ_l is implemented by means of bar charts on features.

The reward decomposition approach has been proposed as the basis for a set of decision-making tasks in the work of Pocius et al. [4], in environments that naturally provide multiple reward signals: the goal is to enhance explainability by providing high-level abstractions for sequential tasks.

4.4.2 Autonomous Policy Explanation. Hayes and Shah [23] aim to enable an autonomous agent to reason over and answer questions about its underlying control logic L, independent of its internal representation. First, it learns a domain model of the agent's operating environment from real or simulated demonstrations, exploiting programmer-specified code annotations for actions, and variables (state parameters/attributes) affected by actions. Within this learned domain model, this approach uses statistics computed over data extracted from continued observations to construct a behavioral model that approximates the agent's control logic. The article shows applicability of the proposed approach in tabular Q-learning, deep Q-learning, and hand-coded controller logic.

To explain the policy L given a query q, this work provides an interpretation function c to provide a natural language response *Response*. L is constrained to state parameters and annotated functions. *Response* is constrained to a template-based approach using natural language.

The interpretation function c is defined to be $c = Summarize\_attributes\ o\ Resolve\_states\ o\ Identify\_question$, where *Resolve_state* resolves the question identified by $Identify\_question$ to a set of problem states (context), and *Summarize_attributes* summarizes attributes of problem states into a comprehensive representation. The *Resolve_state* realizes the function $f(\cdot)$, providing a representation of relevant parameters to resolve a question. Depending on the question type (see the following), this may take the form of searching and gathering state parameters from states in a region, or from similar states given a specific state, or from nearby states. Questions concern when an agent will behave in a specific way, how it will behave under specific conditions, and deviations from the expected behavior. Question-answering is template driven using a response resolution algorithm per query type. Depending on the question type, c provides an explanation for a specific response (i.e., locally) at the granularity of policy actions, explaining the control logic in specific contexts (i.e., sets of states). The explanation logic ϵ is implemented by the *Compose_summary* function and is realized by means of Boolean classifiers using communicable predicates, which are similar to STRIPS-style planning predicates, associated with natural language descriptions.

This work allows human collaborators to shape their expectations on agents' behavior without requiring a full understanding of an agent's logic, and facilitates the debugging of aberrant behaviors by explaining differences between the conditions under which particular actions occur. The article demonstrates the applicability of the proposed method within three representative robotics domains with discrete and continuous state parameters, in single/multi-agent settings, using different types of controllers. However, experiments focus on the applicability of the method rather than on the accuracy/fidelity and conciseness of the explanations provided in any specific human-robot collaborative setting.

4.4.3 Transparency Communication for RL. Wang et al. [14], as a continuation of previous work on static models [14], discuss the design of model-based and model-free RL for robots in a human-robot simulation testbed. The goal is to provide communications for the robot's decision-making update process toward interpretability and repairing trust. This is achieved by interpreting components of the robot's decision-making and learning process, assuming that the robot bases its decisions on a POMDP model.

Considering a dynamic POMDP and assuming a model-based RL that updates the POMDP model, the proposed method interprets a static POMDP after each update, informing the teammate about changes. This work focuses on updating the observations model—that is, the probability for the robot to get any observation. Additionally, assuming a model-free RL, Q-values account for changes in a possible model but with no explicit representation of these changes and model. To address this last issue, the authors propose finding a potential POMDP that is consistent with the policy arrived at by the model-free RL. To do that, they compute the optimal policy for that POMDP exploiting the assumption of a piecewise-linear model [15], representing that policy by

means of a decision tree, which is compared to the RL policy. If it matches, this POMDP is used to provide explanations.

This article does not report on the efficacy of explanations with human subjects. As the authors also point out, this must be done with different permutations of explanations' aspects, also providing insight into what aspects should be included in the explanations and how to present them.

Concluding, this approach provides interpretations of POMDP components, with no access to the learning process itself. We can consider that functions $f(\cdot)$ provide updates on agents' beliefs, transition models, or updates in the Q-values, while the interpretation process updates the existing POMDP, or searches for a potential matching POMDP, which serves as the interpretable model. Function ϵ provides explanations on POMDP components' updates using natural language static templates. In addition to revealing information about individual POMDP components, templates can be created informing about the overall model and how the learning arrived at that model, also leveraging the interpretable policy models generated by exploiting the piecewise-linear assumption.

4.4.4 Transparency and Explanation in DRL. Iyer et al. [3] report on the interpretability of DRL networks in visual tasks, proposing the **object-sensitive deep reinforcement learning** (O-DRL) method that (a) incorporates explicit object recognition processing into DRL models, (b) forms the basis for the development of object saliency maps, and (c) can be incorporated in any existing DRL framework such as DQN and A3C. The goal here is to produce intelligible visualizations of agents' state and behavior, focusing on objects saliency (i.e., on the influence of objects features in agents' decisions).

The proposed O-DRL approach uses a deep neural network (CNN) that takes as input object channels and original images to predict Q-values. Object channels represent types of objects recognized in an image and incorporate objects' features. To compute object saliency, this proposal masks the object and computes the Q-values on the new (perturbed) state, and the difference of new values to the Q-values in the original state.

The authors report on experiments with humans, regarding the behavior of a Pac-Man agent. The goals of the experiments were to (a) test whether object saliency maps contain enough information to allow humans to match them with corresponding game scenarios, (b) test whether participants could use object saliency maps to generate reasonable explanations of the behavior of the Pac-Man, and (c) test whether object saliency maps allow participants to correctly predict the agent's next action. Experiments include (a) a matching task, where participants match saliency maps to agents' behavior, and (b) a movement prediction task from video clips with or without saliency maps. Results show that object saliency maps can be linked to corresponding game scenarios by participants (thus, they do provide a basis for visual explanations), but, as results from the prediction task suggest, the exact use of salient maps must be further investigated in conjunction to providing rich contextual information about each situation.

This is an approach that addresses the outcome explanation problem in visual tasks, focusing on the influence of objects in agents' decisions in specific states. The proposed interpretation method is realized by the O-DRL method that exploits Q-values, provided by function $f(\cdot)$ to rank importance of features (image pixels or objects). Interpretable models are realized by pixels' and objects' salience. The explanation function ϵ_I realizes surface representations of interpretations using visual means.

4.4.5 Understanding Agent Actions Using Specific and Relevant Feature Attribution. Recognizing that perturbation-based approaches for RL, such as proposed by Greydanus et al. [66] and the O-DRL approach [3] in Section 4.4.4, tend to produce saliency maps that are not specific to the

action of interest, Puri et al. [63] propose SARFA to generate saliency maps that focus on explaining the specific action decided, balancing between specificity and relevance.

SARFA is a perturbation-based approach for generating saliency maps focusing on capturing the impact of perturbation only on the Q-value of the action to be explained (*specificity*), and downweighting features that alter the expected rewards of actions other than the action explained (*relevance*): specificity is measured by capturing the relative change to the Q-value for the action to be explained with respect to other actions, and relevance by considering how the relative expected reward of taking some actions (other than the one explained) changes with a perturbation. The harmonic mean of these two measures produces the *salience of state features*.

SARFA [63] has been evaluated qualitatively in games (chess, Atari, and Go), showing that in comparison to O-DRL [3] and the approach by Greydanus et al. [66], it produces more focused saliency maps. In addition, human studies on problem-solving chess puzzles show that saliency maps produced by SARFA are less confusing than those produced by Iyer et al. [3] and Greydanus et al. [66]. SARFA is compared to the other approaches using a chess saliency dataset produced by human experts, showing superiority in identifying salient features and robustness in perturbations.

Overall, SARFA [63] addresses the outcome explanation problem, focusing on visual tasks and aiming to identify state features whose salience is specific and relevant to the decided action. SARFA exploits Q-values, provided by $f(\cdot)$, and interpretation is provided by the salience of state features. The explanation is surfaced using visual means, showing the salience of features.

Regarding the power of salient maps to assess the degree to which hypotheses about features of the learned policy correspond to the semantics of RL settings, Atrey et al. [60] suggest that salience maps should be used as an exploratory rather than explanatory tool and propose a methodology grounded in counterfactual reasoning, focusing on issues of subjectivity, unfalsifiability, and cognitive biases on mapping salient features to semantic concepts and behaviours. In addition to that, saliency maps have been used in combination with policy explanation methods in the work of Huber et al. [64], investigating their role when put in the context of agents' strategies (discussed in Section 4.2.1).

4.4.6 Autonomous Self-Explanation of Interactive RL. Fukuchi et al. [22] propose **instruction-based behavior explanation (IBE)** to explain an agent's future behavior within a sufficient temporal extent while the policy is being learned in an interactive RL setting. In IBE, an agent exploits the instructions given by a human expert (which may not indicate the actions to be performed) to explain its own behavior: this happens under the assumption that when the agent receives more rewards, it is likely that it followed the instructions given.

IBE consists of two steps. The first step is estimating the target of the agent's actions by simulation: the target is the change in the environment state after a fixed temporal extent for explanations specified by agents' actions in n steps. The second step is acquiring a mapping from the target to the instructions, through a k-means classifier, to explain the action target based on the instruction signal given by a human expert. The authors report on experiments with humans in a game setting, using two DQN agents, each at a different training stage. The most important finding is that the environment changes must be chosen deliberately for assigning an explanation signal: the parameter n that defines the number of agent's actions to a target state cannot be fixed, whereas predictions in settings of high-complexity must be accurate.

IBE [22] provides a local interpretation method, interpreting the decision of the agent in specific states, given a target state in a *constant* temporal extent of n timesteps. The interpretation function is the IBE method itself, exploiting agents' target and instructions provided by experts. Thus, the RL method is treated as a dark box, where the function $f(\cdot)$ fetches instructions provided by experts, provided as representations of the agent's behavior. The classification of assessed targets

to instructions provides a local interpretable model. The explanation logic is context and domain dependent: examples of visualizations are provided in the work of Fukuchi et al. [22]. As stated in their work [22], prediction of changes is challenging in complex settings, and the temporal extent may need to vary in different contexts, even for the same agent.

4.4.7 Explainable RL Through Causal Lens. Madumal et al. [16] aim to generate explanations using causal chains from action influence graphs, for model-free RL agents. This is accomplished by encoding causal relationships between variables of interest and learning a structural causal model [17] during RL. Action influence models approximate the causal model of the environment relative to actions taken by an agent. This approach uses causal models to generate contrastive explanations for *why* and *why not* questions, and formulates the minimally complete explanations, and minimally complete contrastive types of explanations to be provided from an action influence model.

A critical part of this approach is learning the structural equations—that is, the quantitative influences of actions on variables: assuming that a DAG specifying causal direction between variables is given, structural equations are approximated as multivariate regression models during the training phase of the RL agent, exploiting samples of agent interaction with the environment via an experience reply. Therefore, this is a mimicking process, where interpretable models are structural equations specifying the effects of actions in features, the causes of rewards, and the states resulting in rewards. These local models c_l provide local response and local objectives' explanations. The explanation logic ϵ provides explanations to the two types of questions: "why" and "why not."

This approach was computationally evaluated on six RL benchmark domains using six different model-free RL algorithms with discrete actions. Results indicate that the proposed models are robust and accurate enough to perform task prediction with a negligible performance impact. A human study (120 participants) evaluated the method in task prediction, explanation satisfaction, and trust, showing that the proposed model performs better than the tested baseline, but its impact on trust is not statistically significant.

4.4.8 Self-Supervised Discovery of Causal Features. Shi et al. [5] proposed a self-supervised interpretable framework (SSINet) to explain a trained policy model of vision-based RL agents to locate fine-grained causal features toward gathering state-specific and task-relevant evidence for the agent's decisions.

Specifically, given a trained policy model, the proposed framework learns an explanation model that predicts an attention mask. The basic attention patterns learned are exploited for identifying the relative importance of features and analyzing failure cases. If the generated actions are consistent when the policy takes as input either the state or the attention-overlaid state, the features highlighted by the proposed method are considered to be task relevant and constitute evidence for the agent's decisions. The agent in the work of Shi et al. [5] is pre-trained using model-free RL algorithms including proximal policy optimization (PPO), soft actor-critic (SAC), and twin delayed deep deterministic policy gradient (TD3).

More technically, SSINet uses a U-Net architecture, providing "masked" states through a feature extractor, a mask decoder, and a sigmoid non-linearity. SSINet uses the same feature extractor with the pre-trained agent policy model, thus SSINet trains only the mask decoder. SSINet is trained using a dataset of state-action pairs generated by the policy. Training is "driven" by a mask loss function aiming to satisfy two desiderata: *maximum behavior resemblance* (i.e., the agent's behavior using masked states must be as consistent as possible with that when using original states) and *minimum region retaining* (i.e., attentions must attend to as little information as possible).

SSINet has been evaluated on several Atari 2600 games, as well as on Duckietown, a self-driving car simulator environment. Empirical results verify the effectiveness of the proposed method, and

demonstrate that SSINet produces high-resolution and sharp attention masks to highlight task-relevant information. Special emphasis has been given to explaining why the agent performs well or badly quantitatively, by introducing evaluation metrics that assess the effectiveness of attention masks, in multiple RL algorithms and actor architectures.

Concluding, SSINet proposed in the work of Shi et al. [5] is a self-supervised ML process for vision-based DRL agents. Although it exploits samples from a trained policy model, it does not aim to produce an interpretable model that mimicks or that distills the knowledge acquired in the policy model. Rather, it addresses the outcome explanation problem, providing a local model c_l that identifies salient state features, which are further presented via appropriate visual masking techniques realized by e_l, applied in the original state.

4.4.9 Distilling DRL in Soft Decision Trees. Coppens et al. [18] illustrate how **soft decision trees (SDTs)** [19] can be used as interpretable policy models. SDTs are hybrid classification models of binary trees of predetermined depth, and NNs. Each branching node represents a hierarchical filter that influences the classification of input data. The leaf nodes learn softmax distributions over possible classes using model parameters θ^{SDT}, whereas the overall loss is minimized using mini-batch gradient decent optimization.

Using state-action pairs generated from the policy learned using the actor-critic PPO algorithm, the SDT is trained to classify states to actions. The applicability of the method is evaluated on the Mario AI benchmark: the authors show the fidelity with which SDTs can provide interpretations, and elaborate on the performance of these models when one would consider to use SDTs directly for task execution.

This is a mimicking process (as Coppens et al. [18] also state), despite the article title that characterizes the approach as a distillation process. The model c_l is the SDT, which provides local-only interpretations, classifying states to agent actions. Despite the possibility of examining the learned filters along any decision tree path from the root to leaf, it is questionable whether the SDT offers a fully interpretable model, given the parameterized leaf nodes and the decision tree filtering nodes leading to an action distribution. As noted in the article, since the distributions in the leafs need to generalize and provide an action strategy for multiple state samples, it is to be expected that the resulting action distributions in the SDTs can differ from the NN output of the PPO agent for individual input samples: this results in low fidelity in mimicking the PPO policy. The interesting aspect of this work is that bigger decision trees offer greater agent rewards. However, this comes with the cost of reduced interpretability. The explanation logic ϵ provides surface representations using hitmaps, exploiting the SDT model in spatial settings. The use of SDTs in non-spatial settings is something worth further investigation.

4.4.10 Memory-Based Explainable Reinforcement Learning. Cruz et al. [9] propose a **memory-based explainable reinforcement learning (MXRL)** approach according to which the agent exploits an episodic memory to infer the probability of success and the number of transactions to reach the goal state. Specifically, the agent exploits past interactions with the environment, and by "introspection" can compare the probability of choosing an action against the probability of being successful, thus providing explanations in terms of the necessity to complete an intended task. The interpretable model aims to explain the agent's decision for an action at a specific state, based on past experience, exploiting transitions in successful runs. It is assumed that all transitions have been recorded. This limits the method applicability to small, discrete state—action spaces, such as the grid world in which experiments have been conducted.

In essence, this is a distillation process where the agent distills knowledge regarding the reasons the RL agent favors specific actions for specific states, through past experience gathered in an "interpretation" that applies at the episodic memory. The explanation logic is provided via hitmaps

and diagrams, showing probabilities for choosing actions in states, and probabilities to succeed in reaching a goal state.

4.4.11 Enhancing Explainability of DRL Through Selective LRP. As noted in the work of Miller [42], people usually prefer explanations that focus on selected, specific evidence instead of showing every possible cause of a decision. Based on this insight, Huber et al. [70] aim to adjust the LRP [34] saliency map approach to focus on the parts of the input that are most relevant for the decision-making process of a DRL agent. Their contribution is twofold: (a) their adjustment uses an *argmax* function to follow only the most contributing neurons, filtering out the most relevant information, and (b) they adjust LRP to dueling DQN convolutional layers.

LRP describes a concept that can be applied to any classifier if it fulfills the following two requirements. First, it has to be decomposable into several layers of computation where each layer can be modeled as a vector of real-valued functions. Second, the first layer has to be the input x of the classifier and the last layer has to be the real-valued prediction of the classifier $f(x)$. Any DRL agent fulfills those requirements, considering actions decided as the output: in this case, LRP can be used to address the outcome explanation problem. According to the LRP concept, relevance values R_j^l are assigned to each computational unit j of each layer l in such a way that R_j^l measures the local contribution of the unit j to the prediction $f(x)$. The calculation of R_j^l values follows the LRP concept, if it sets the relevance value of the output unit to be the prediction $f(x)$ and calculates all other relevance values by defining $R_j^l = \Sigma_{k \in \{j \text{ is input to } k\}} R_{j,k}^{l,l+1}$ for messages $R_{j,k}^{l,l+1}$ such that $R_k^{l+1} = \Sigma_{j \in \{j \text{ is input to } k\}} R_{j,k}^{l,l+1}$. Relevance values R_j^l can be computed in a backward pass, starting from the output layer, toward the input layer. This concept has been applied to the dueling DQN architecture in the work of Huber et al. [70], specifying the backward pass through the fully connected and the convolution layers. However, to find the most relevant input neurons contributing to a specific decision, the authors propose the *argmax* rule, defining the messages as follows:

$$R_{j,k}^{l,l+1} = \begin{cases} R_k^{l+1} & \text{if } j = argmax\{w_j a_j\} \\ 0 & \text{otherwise.} \end{cases},$$

where w_i are model weights and a_i are the inputs.

This approach has been evaluated on three Atari 2600 games to verify that the saliency maps generated are more selective than the ones created by alternative LRP methods while still including the information expected from visual explanations: this can be beneficial to the transparency of DRL methods.

Overall, this is an approach that addresses the outcome explanation problem, aiming to provide the most salient features toward specific DRL agents' decisions. The model constructed concerns the local contributions (relevance values) of inputs and neurons to the decision made, whereas explanations are provided visually by means of inputs' saliency maps.

5 CONCLUDING REMARKS

Explainability of closed-box methods that provide agents with the capacity to act autonomously in the real world, and particularly of DRL methods, is rather challenging, albeit emerging: DRL, being successful to prescribe actions and courses of actions in complex settings according to the evolution of the environment, may comprise a number of closed boxes, including models of the environment, of the agent objectives and the agent policy, as well as models that fit states' or state-action pairs' values. Based on the general blueprint of DRL methods, in this article we formulate specific explanation problems: (a) the policy and objectives model explanation problem, aiming to provide interpretation of the overall (global) control logic behind a specific agent's responses

prescribed by the agent's policy and motivated by the agent's objectives, in coarse granularity and large scales; (b) the responses and local objectives explanation problem that aims to interpret agents' timely (local) responses in specific states, also explaining the correlation between the environment's state features with agents' objectives in fine granularity and small scale; and (c) the model inspection problem, providing elements and properties of any DRL individual model.

The article identifies specific paradigms that have been proposed for implementing XDRL methods, proposing a distinction between the distillation and mimicking approaches. It proceeds to provide a review of state-of-the-art methods for XDRL methods, addressing the needs of human operators—that is, of those that take the actual and critical decisions in solving real-world problems. For each state-of-the-art proposal, the article describes the overall objectives and the specific explanation problem that is being addressed, the method used for providing interpretations, the interpretable models constructed according to the DRL explainability paradigm followed, the explanation logic, and the surface representations of explanations.

What can follow as a general and straightforward remark from this review is that although there is a lot of work and progress toward solving XDRL problems, it is very early for conclusions regarding DRL explainability and transparency. We need to understand the possibilities to their full extent in all aspects across the pipeline, and in all different dimensions regarding the explainability desiderata. Indeed, there are many questions that remain unanswered, which present major challenges. Subsequent paragraphs try to point out the major and important ones, according to our view.

Starting from the questions that are stated in Section 1, there is not any work that identifies and provides evidence about the qualities of "good" and objective explanations for RL agents. There is some evidence about the qualities of explanations from other scientific fields (e.g., for an overview in such evidence in social sciences, one may start with the work of Lipton [39]), but we need to bridge these to the effectiveness of RL agents' explanations (for any of the explanation problems stated), also considering the characteristics of the real-world contexts in which agents are deployed. Specifically, although many approaches make specific proposals for interpretation and presentation of explanations, these are made in a fragmentary and ad hoc manner (e.g., exploiting few natural language templates, or hitmaps with specific characteristics) providing mixed results or no evidence on the effectiveness of these explanations (maybe in combination with other modalities) in real-world settings. For vision-based RL, for instance, a feasible explanation approach is to learn t-SNE maps, whereas there are works visualizing important features for the RL agent's decisions using saliency maps: the role of these maps to the effectiveness of providing local explanations needs further investigation, also in combination with other methods that provide contextual and more broad perspectives of agent behavior.

Interpretable ML presents many challenges (e.g., some of them are discussed in the work of Rudin [57]), which are inherited to constituent models of DRL methods. However, RL agents that solve sequential problems need to provide explanations on sequences of decisions and exploit the constituent models' interpretations in intertwined ways. In the following, we present a list of challenges that we consider important to report essential progress in XDRL in particular:

— *Build a toolbox for explaining DRL*: We need as general as possible and advanced tools for the surface representation of explanations from DRL agents. These must identify generic types of information to exploit and must include appropriate configuration functionality to provide information using multiple modalities, allowing users to explore explanations at different levels of detail, granularity, and scale. Such a toolbox would provide explanations' content exploiting specific types of information and transforming information (via projections, filtering, aggregation, and extraction of features, etc.) in comprehensive ways, for any combination of problem dimensions.

— *Be as comprehensive as needed*: We need to realize approaches that interpret and explain the behavior of DRL agents in as much as possible comprehensive ways. Such approaches should address combinations of the explanation problems stated, in a joined way, supporting explaining the overall policy and objectives model, specific responses, indicating features importance in general, refining features' ranking in specific cases, and inspecting the consequences, indicating parts of the state-action space of special interest at different levels of granularity and scale. This will allow synthesizing comprehensive explanations for humans to understand agents' decision making, tailored to specific context of agents' deployment, with respect to users' requirements, constraints, and cognitive aspects.
— *Bridge to theory while answering pragmatic concerns*: We need to bridge properties/qualities of interpretable models and of explanations provided with fundamental theories and facts about human cognition and the qualities of explanations, so as to understand better the mechanisms for providing context-dependent qualitative explanations.
— *Focus on transparency*: There is not any comprehensive XDRL framework that has been developed for providing explanations under pragmatic constraints. Although there are approaches targeting to *transparency*, the term is used more or less equivalently to interpretability. We need methods/frameworks whose explainability capabilities can be tuned to pragmatic constraints, focusing on transparency.
— *Explore the interpretable box design paradigm*: We need to further explore the distinct XDRL paradigms for interpretable box design, also in conjunction to other objectives (e.g., as in the work of Bastani et al. [54]): while the mimicking paradigm provides a solution that somehow bypasses explaining the deep models in a direct way, it adds a layer to the system that is susceptible to additional loss of accuracy in problem solving and raises questions about its fidelity to the control logic of the DRL agent. Models' distillation provides opportunities for generalizing effectively beyond the training examples provided originally to the agent and allow building compact models. However, we need to further explore methods for distilling acquired knowledge in direct ways, and exploiting these models to provide concise explanations' for different purposes and across problem tasks.
— *Develop XDRL in a principled way*: We need principles, methodologies, and tools for designing and developing XDRL agents from early development phases. Although there are methods that follow the interpretable box design paradigm, which is close to that aim, most approaches add explainability layers/modules to agents with advanced DRL methods.
— *Explore XDRL in complex and/or large-scale multi-agent settings*: Solving explainability problems for multi-agent and potentially complex settings does not follow in a straightforward way from solving these problems for individual agents. Interpretability may involve further filtering/aggregation steps in providing explanations' content, whereas we need to advance explainability and transparency in challenging but highly critical real-world multi-agent settings.
— *Set up experimental protocols with emphasis on pragmatic concerns*: We need to set up experimental protocols, methodologies, and widely acceptable indicators regarding how and what one should evaluate to provide evidence regarding humans' qualitative understanding and acceptability of explanations, closely connected to the functionality of DRL agents deployed in real-world settings. Holzinger et al. [56] propose the system causability scale for measuring explanations' quality, which can be a starting point, incorporating issues of transparency.
— *Provide comparative evaluations for interpretability, explainability, and transparency*: Although some kinds of models are considered more interpretable than others, or some explainability methods may provide better explanations than others (whatever this means, but

with respect to the explanation problem addressed), we need a comprehensive comparative evaluation on the quality of explanations that interpretable models can provide, also with respect to pragmatic issues.

REFERENCES

[1] E. Puiutta and E. M. S. P. Veith. 2020. *Explainable Reinforcement Learning: A Survey*. arXiv:2005.06247 (2020).

[2] M. T. Ribeiro, S. Singh, and C. Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD KDD*. ACM, New York, NY, 1135–1144.

[3] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara. 2018. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

[4] R. Pocius, L. Neal, and A. Fern. 2019. Strategic tasks for explainable reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 1 (2019), AAAI-19, IAAI-19, EAAI-20.

[5] W. Shi, S. Song, Z. Wang, and G. Huang. 2020. Self-supervised discovering of causal features: Towards interpretable reinforcement learning. arXiv:2003.07069v2 (2020).

[6] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. 2019. Explainable reinforcement learning via reward decomposition. In *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.

[7] N. Topin and M. Veloso. 2019. Generation of policy-level explanations for reinforcement learning. arXiv:1905.12044 (2019).

[8] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. 2004. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*. 513–520.

[9] F. Cruz, R. Dazeley, and P. Vamplew. 2019. Memory-based explainable reinforcement learning. In *AI 2019: Advances in Artificial Intelligence*. Lecture Notes in Computer Science, Vol. 11919. Springer, 66–77.

[10] T. Leech. 2019. Explainable machine learning for task planning in robotics. Master's Thesis. Massachusetts Institute of Technology, Cambridge, MA.

[11] D. Dancey, Z. A. Bandar, and D. McLean. 2007. Logistic model tree extraction from artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Part B* 37, 4 (2007), 794–802.

[12] G. Liu, O. Schulte, W. Zhu, and Q. Li. 2018. Toward interpretable deep reinforcement learning with linear model U-trees. arXiv:1807.05887 (2018).

[13] Z. Che, S. Purushotham, R. Khemani, and Y. Liu. 2016. Interpretable deep models for ICU outcome prediction. In *Proceedings of the AMIA Annual Symposium*.

[14] N. Wang, D. V. Pynadath, and S. G. Hill. 2016. The impact of POMDP-generated explanations on trust and performance in human-robot teams. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'16)*. 997–1005.

[15] D. V. Pynadath and S. C. Marsella. 2004. Fitting and compilation of multiagent models through piecewise linear functions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*. 1197–1204.

[16] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. 2019. Explainable reinforcement learning through a causal lens. arXiv:1905.10958 (2019).

[17] J. Y. Halpern and J. Pearl. 2005. Causes and explanations: A structural-model approach. Part I: Causes. *British Journal for the Philosophy of Science* 56, 4 (2005), 843–887.

[18] Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni. 2019. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the 2019 IJCAI Workshop on Explainable Artificial Intelligence*.

[19] N. Frosst and G. Hinton. 2017. Distilling a neural network into a soft decision tree. In *Proceedings of the 1st International Workshop on Comprehensibility and Explanation in AI and ML,* Vol. 2071 of the AI*IA Series at CEUR Workshop Proceedings.

[20] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. 2020. Distal explanations for explainable RL agents. arXiv: 2001.10284 (2020).

[21] A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso. 2019. Conservative Q-improvement: Reinforcement learning for an interpretable decision-tree policy. arXiv:1907.01180 (2019).

[22] Y. Fukuchi, M. Osawa, H. Yamakawa, and M. Imai. 2017. Autonomous self-explanation of behavior for interactive reinforcement learning agents. In *Proceedings of the 5th International Conference on Human-Agent Interaction*.

[23] B. Hayes and J. A. Shah. 2017. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 12th ACM/IEEE International Conference on Human-Robot Interaction*.

[24] S. Mohseni, N. Zarei, and E. D. Ragan. 2018. A multidisciplinary survey and framework for design and evaluation of explainable AI systems. arXiv:1811.11839 (2018).

[25] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. 2018. A survey of methods for explaining black box models. *ACM Computing Surveys* 51, 5 (2018), 1–42.
[26] S. M. Lundberg and S. Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 4768–4777.
[27] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. 2018. Programmatically interpretable reinforcement learning. arXiv:1804.02477 (2018).
[28] T. Shu, C. Xiong, and R. Socher. 2018. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *Proceedings of the 6th International Conference on Learning Representation (ICLR'18)*.
[29] O. Boz. 2002. Extracting decision trees from trained neural networks. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*. 456–461.
[30] S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan. 2019. Enabling robots to communicate their objectives. *Autonomous Robots* 43, 2 (2019), 309–326.
[31] M. T. Ribeiro, S. Singh, and C. Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[32] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. 2018. Local rule-based explanations of black box decision systems. arXiv:1805.10820 (2018).
[33] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. arXiv:1605.01713 (2016).
[34] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Muller, and W. Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layerwise relevance propagation. *PLoS One* 10, 7 (2015), e0130140.
[35] L. D. Pyeatt and A. E. Howe. 2001. Decision tree function approximation in reinforcement learning. In *Proceedings of the 3rd International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*.
[36] A. K. McCallum. 1996. Learning to use selective attention and short-term memory in sequential tasks. In *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior (SAB'96)*. 315–325.
[37] W. T. Uther and M. M. Veloso. 1998. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications on Artificial Intelligence (AAAI'98/IAAI'98)*. 769–774.
[38] S. G. Rizzo, G. Vantini, and S. Chawla. 2019. Reinforcement learning with explainability for traffic signal control. In *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC'19)*. 3567–3572.
[39] Z. C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16, 3 (May–June 2018), 31–57.
[40] O. Biran and C. V. Cotton. 2017. Explanation and justification in machine learning: A survey. In *Proceedings of the 2017 IJCAI Workshop on Explainable Artificial Intelligence*.
[41] M. A. de Graaf, B. F. Malle, A. Dragan, and T. Ziemke. 2018. Explainable robotic systems. In *Companion of the ACM/IEEE International Conference on Human-Robot Interaction (HRI'18)*. 387–388.
[42] T. Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267 (2019), 1–38.
[43] D. Amir and O. Amir. 2018. HIGHLIGHTS: Summarizing agent behavior to people. *In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'18)*.
[44] P. Sequeira and M. Gervasio. 2020. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. arXiv:1912.09007v2 (2020).
[45] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. 2015. Policy distillation. arXiv:1511.06295v2 (2015).
[46] G. Hinton, O. Vinyals, and J. Dean. 2014. Distilling the knowledge in a neural network. arXiv:1503.02531 (2014).
[47] S. H. Huang, K. Bhatia. P. Abbeel, and A. D. Dragan. 2018. Establishing appropriate trust via critical states. arXiv:1810.08174 (2018).
[48] J. van der Waa, J. van Diggelen, K. van den Bosch, and M. Neerincx. 2018. Contrastive explanations for reinforcement learning in terms of expected consequences. arXiv:1807.08706 (2018).
[49] National Security Commission on Artificial Intelligence. 2019. *Interim Report*. National Security Commission on Artificial Intelligence.
[50] EU High-Level Expert Group on Artificial Intelligence. 2019. Ethics Guidelines for Trustworthy Artificial Intelligence. Retrieved April 7, 2022 from https://ec.europa.eu/futurium/en/ai-alliance-consultation.1.html.
[51] R. M. Annasamy and K. Sycara. 2019. Towards better interpretability in deep Q-networks. arXiv:1809.05630 (2019).
[52] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende. 2019. Towards interpretable reinforcement learning using attention augmented agents. arXiv:1906.02500 (2019).
[53] T. Zahavy, N. Ben Zrihem, and S. Mannor. 2017. Graying the black box: Understanding DQNs. arXiv:1602.02658 (2017).

[54] O. Bastani, Y. Pu, and A. Solar-Lezama. 2019. Verifiable reinforcement learning via policy extraction. arXiv:1805.08328 (2019).
[55] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. 2019. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* 116, 44 (2019), 22071–22080.
[56] A. Holzinger, A. Carrington, and H. Meuller. 2020. Measuring the quality of explanations: The system causability scale (SCS). *Künstliche Intelligenz* 34 (2020), 193–198.
[57] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. 2021. Interpretable machine learning: Fundamental principles and 10 grand challenges. arXiv:2103.11251v2 (2021).
[58] V. Belle and I. Papantonis. 2020. Principles and practice of explainable machine learning arXiv:2009.11698v1 (2020).
[59] H. Yau, C. Russell, and S. Hadfield. 2020. What did you think would happen? Explaining agent behaviour through intended outcomes. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'20)*.
[60] A. Atrey, K. Clary, and D. Jensen. 2019. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. arXiv:1912.05743 (2019).
[61] A. Hüyük, D. Jarrett, C. Tekin, and M. van der Schaar. 2021. Explaining by imitating: Understanding decisions by interpretable policy learning. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.
[62] Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Celi, Emma Brunskill, and Finale Doshi-Velez. 2020. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. arXiv:2002.03478 (2020).
[63] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. 2020. Explain your move: Understanding agent actions using specific and relevant feature attribution. arXiv:1912.12191 (2020).
[64] T. Huber, K. Weitz, E. André, and O. Amir. 2021. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. arXiv:2005.08874 (2021).
[65] J. Skirzyński, F. Becker, and F. Lieder. 2021. Automatic discovery of interpretable planning strategies. *Machine Learning* 110 (2021), 2641–2683.
[66] S. Greydanus, A. Koul, J. Dodge, and A. Fern. 2018. Visualizing and understanding Atari agents. In *Proceedings of the 35th International Conference on Machine Learning*.
[67] T. Silver, R. A. Kelsey, A. K. Lew, L. P. Kaelbling, and J. Tenenbaum. 2019. Few-shot Bayesian imitation learning with logical program policies. arXiv:1904.06317 (2019).
[68] A. Alqaraawi, M. Schuessler, P. Weiss, E. Costanza, and N. Berthouze. 2020. Evaluating saliency map explanations for convolutional neural networks: A user study. arXiv:2002.00772 (2020).
[69] M. Erwig, A. Fern, M. Murali, and A. Koul. 2018. Explaining deep adaptive programs via reward decomposition. In *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.
[70] T. Huber, D. Schiller, and E. André. 2019. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *Proceedings of KI 2019: Advances in Artificial Intelligence*. 188–202.

Received 2 October 2020; revised 14 March 2022; accepted 16 March 2022