# Explainable Reinforcement Learning: A Survey and Comparative Review

STEPHANIE MILANI, Carnegie Mellon University, USA
NICHOLAY TOPIN, Inpleo Inc., USA
MANUELA VELOSO, J. P. Morgan AI Research, USA
FEI FANG, Carnegie Mellon University, USA

Explainable reinforcement learning (XRL) is an emerging subfield of explainable machine learning that has attracted considerable attention in recent years. The goal of XRL is to elucidate the decision-making process of reinforcement learning (RL) agents in sequential decision-making settings. Equipped with this information, practitioners can better understand important questions about RL agents (especially those deployed in the real world), such as what the agents will do and why. Despite increased interest, there exists a gap in the literature for organizing the plethora of papers—especially in a way that centers the sequential decision-making nature of the problem. In this survey, we propose a novel taxonomy for organizing the XRL literature that prioritizes the RL setting. We propose three high-level categories: feature importance, learning process and Markov decision process, and policy-level. We overview techniques according to this taxonomy, highlighting challenges and opportunities for future work. We conclude by using these gaps to motivate and outline a roadmap for future work.

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

Additional Key Words and Phrases: Explainable reinforcement learning, interpretability, explainability

**ACM Reference format:**
Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. 2024. Explainable Reinforcement Learning: A Survey and Comparative Review. *ACM Comput. Surv.* 56, 7, Article 168 (April 2024), 36 pages.
https://doi.org/10.1145/3616864

N. Topin's work began as a doctoral student at Carnegie Mellon University.
This research was supported in part by NSF grant IIS-2046640 (CAREER) and the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the social policies, either expressed or implied, of the funding agencies. This material is based upon work supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DoD.
Authors' addresses: S. Milani and F. Fang, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213; e-mails: smilani@cs.cmu.edu, feifang@cmu.edu; N. Topin, Inpleo Inc., 48 26th Street, Pittsburgh, PA 15222; e-mail: nicholay.topin@gmail.com; M. Veloso, J. P. Morgan AI Research, 383 Madison Ave, New York, NY 10017; e-mail: manuela.veloso@jpmchase.com.


0360-0300/2024/04-ART168 $15.00
https://doi.org/10.1145/3616864

## 1 INTRODUCTION

In **Reinforcement Learning (RL)**, an agent learns to take actions $a_1, \ldots, a_T$ to maximize accumulated reward through trial and error. To choose actions, agents use a policy $\pi$ that takes as input the current state $s$ of the environment, commonly modeled as a **Markov Decision Process (MDP)**. Agents learn from their experiences with the environment using state-action-reward tuples $s, a, r, s'$, where $r$ is the reward and $s'$ is the state resulting from applying action $a$ in state $s$. This framework has been successfully applied to many sequential decision-making problems, like games [70, 109, 151], robotics [67, 77, 160], and more [114, 142, 169]. However, the difficulty of verifying and predicting the behavior of RL agents often hampers their real-world deployment [24]. This problem is exacerbated when RL is combined with the generalization and representational power of deep neural networks, which is often required to achieve the desired performance on these tasks. These networks often have thousands to millions of parameters, which makes them challenging or impossible for a person to understand. Without an understanding of how the agent works, it is hard for a person to intervene when necessary or trust that the agent will act reasonably and safely.

Recently, there has been increasing interest by researchers in many fields [25, 143] in *explaining* artificially intelligent agents, including those trained using RL, to gain insight into the agent's decision-making process. This increased interest is in part due to large initiatives such as the DARPA Explainable **Artificial Intelligence (AI)** project [59], which evolved from a narrower focus on supervised learning to include broader tasks in AI, including RL. **Explainable Reinforcement Learning (XRL)** both enables RL to be used more broadly and offers unique research opportunities: we can apply techniques from explainable supervised learning and also develop methods that account for and utilize the sequential nature of RL problems.

To organize and understand the plethora of literature, we present an extensive review of the literature in the field of XRL. We focus on papers in which the authors explicitly identify explainability or interpretability as a goal of the work. We primarily consider techniques developed for a *single* agent. We limit our survey to published work, up to and including papers from August 2022. By published, we mean that the paper is included in a thesis, journal, workshop, conference, or book chapter. We exclude arXiv-only papers. To find papers to include in the survey, we searched Google Scholar with many combinations and abbreviations of explainable and interpretable RL. Our stopping criteria for each search term was when there were no further relevant papers in an entire page. We also scoured more popular AI and machine learning conferences and relevant workshops, such as the AAAI Conference on Artificial Intelligence (AAAI), the International Conference on Machine Learning (ICML), and the International Conference on Learning Representations (ICLR). After curating a larger set of candidate papers, we removed papers that did not satisfy our inclusion criteria. For example, we removed papers that used RL to generate explanations for supervised learning. After curating a larger set of candidate papers that satisfy the inclusion criteria, we produced a representative sample of 73 papers in this area.

### 1.1 Contributions

We are not the first to survey XRL techniques; however, we make the following key contributions. First, existing surveys on XRL tend to organize the literature according to existing taxonomies for understanding the explainable machine learning literature, which we discuss more in Section 2.2. Although this organization is reasonable, we believe that it obfuscates the unique challenges and opportunities of RL problems, such as the inclusion of reward and the sequential nature of the problem. We contribute a novel taxonomy for conceptually organizing and understanding the different threads of work in the space of XRL. This taxonomy focuses on the RL problem, with clear ties to key parts of the RL framework. Second, existing surveys on XRL focus on delving deeper

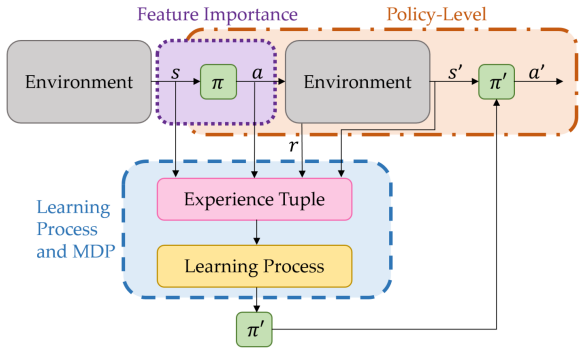


Fig. 1. Proposed XRL taxonomy and its relationship to the RL problem. Explanations may explain individual actions $a$ in terms of the input state $s$ (Feature Importance), influential experiences from training (Learning Process and MDP), and long-term behavior (Policy-Level).

into a more limited set of papers [71, 121, 158] or focus on specific applications [168]. For example, Heuillet et al. [71] discuss 15 XRL papers in detail, and Puiutta and Veith [121] examine 18 papers. Similarly, we identify an exemplar paper in each subcategory to discuss in more detail, resulting in an in-depth look at 10 papers and a broader look at 73. By relating a larger set of work to exemplar papers within our taxonomy, we highlight the distinguishing features within different types of XRL literature. This broader look allows us to make statements about trends in XRL work. Third, we cover a more up-to-date set of papers, up to and including work published by August 2022. As a result, our survey provides a look at more recent work as compared to existing surveys that cover papers up to those published in 2020. The inclusion of more recent work is also reflected in our taxonomy, which accounts for more recent lines of work. Fourth and finally, this combination of new taxonomy and recent papers allows us to propose concrete opportunities for future research, many of which are unique to this work.

Our contributions are summarized as follows:

- *New taxonomy*: We present a new taxonomy for understanding XRL research. We organize techniques according to the part of the RL agent they explain. We propose the following three high-level categories: **Feature Importance (FI)**, **Learning Process and Markov Decision Process (LPM)**, and **Policy-Level (PL)**. FI explanations provide the immediate context for single actions. LPM explanations determine influential experiences from training or other aspects of the environment, which is commonly modeled as an MDP. PL explanations show the long-term behavior of an agent, for example, by summarizing the policy. Figure 1 depicts the high-level categories of the taxonomy and their relationship to the RL framework.
- *Comprehensive review*: We overview the XRL literature, categorizing representative approaches according to our novel taxonomy. For each subcategory, we identify an exemplar paper and discuss it in more detail. We compare and contrast the different types of techniques.
- *Future directions*: We highlight gaps in the literature and outline a roadmap for future work: investigating properties of explanations, developing benchmarks for standardized comparisons, and creating RL-specific explanations.

### 1.2 Article Organization

The article is organized as follows. In Section 2, we provide the necessary background on concepts and notation for understanding RL and explainable machine learning. In Section 3, we explain how

one can assess whether an RL system is interpretable or explainable. We categorize existing papers according to the metrics used to assess their techniques. In Section 4, we introduce our novel taxonomy for organizing and understanding XRL methods. We compare and contrast the different categories to showcase what they do and do not enable. Sections 5 through 7 delve into the techniques that fall into the high-level categories created by our novel taxonomy. Section 5 discusses the FI methods, Section 6 details the LPM techniques, and Section 7 focuses on the PL papers. We discuss the practical applications of the taxonomy for RL researchers and other practitioners in Section 8. In Section 9, we synthesize the gaps in the literature to present a roadmap for future work in XRL. In Section 10, we conclude with a summary of our contributions.

## 2 BACKGROUND AND PRELIMINARIES

In this section, we first provide the necessary background and associated notation for understanding the framework of RL problems. We then introduce important concepts in the field of explainable machine learning. We contextualize these concepts in terms of the subfield of RL.

### 2.1 Reinforcement Learning

In RL, an agent learns how to behave in an environment to maximize a reward signal [139]. The environment is typically defined by an MDP $M = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0, 1]$ is the state-transition function dictating the environment's transition dynamics, $R : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the scalar discount factor that governs the relative importance of future rewards vs immediate rewards. Within this environment, an agent acts based on a policy that specifies the probability for selecting each given action within a state, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. At each timestep $t$, an agent observes a state $s_t$ and chooses an action $a_t$ according to its policy $\pi$. The agent executes $a_t$, receives reward $r_t$ according to $R$, and observes a new state $s'_{t+1}$ according to $T$. To solve an MDP is to find the policy that leads to the greatest expected cumulative discounted reward.

To learn a policy, an agent may keep an estimate of the future cumulative reward from each state, although some methods directly optimize the policy without such an estimate. The action-value for a state corresponds to the expected cumulative discounted future reward when taking the specified action and then following policy from the next state, $Q^\pi(s, a) = \mathbb{E}[r + \gamma Q^\pi(s', a)]$. Similarly, the value function corresponds to the expected cumulative discounted future reward when starting in a given state, $V^\pi(s) = \mathbb{E}\sum_t[\gamma^t r_t]$. $V^\pi(s)$ is the value function that provides the value of state $s$ under policy $\pi$. The goal of RL is to find an optimal policy $\pi^*$ specifying which action to take in each state that results in the highest expected discounted future return $\pi^* = \arg\max_\pi Q^\pi(s, a)$.

In RL, the dynamics $T$ and rewards $R$ of the problem are unknown to the agent. Instead, an agent may additionally approximate the model (i.e., with $\hat{T}$ and $\hat{R}$) during the learning process. These agents are called *model-based* learners [13, 110], whereas agents that directly learn from trial-and-error are called *model-free* learners. For example, these methods may predict the next state and reward resulting from taking an action in a state.

Large or continuous state spaces prevent exactly storing $Q^\pi$ or $\pi$. Instead, many RL methods leverage function approximators: most commonly, neural networks. These networks are often used with factorized states, where a state consists of an assignment of values to specific features, or images. Deep neural networks have taken a central role in RL: their use has resulted in many of the celebrated successes of RL in recent years, including super-human performance on domains previously thought to be out of reach [134, 135].

However, agents that use a non-interpretable function approximator are often challenging, if not impossible, for a person to understand. These models often involve complex non-linear func-

tions and millions of parameters, resulting in not only an agent that is difficult to understand but also one with behavior that is challenging to predict. Indeed, many deep RL agents are susceptible to adversarial examples, which involve perturbing $s$ to induce different behavior [99]. In contrast, people often are unaffected by these adversarial examples, meaning that the underlying mechanisms used by people and deep RL agents likely differ. The lack of understanding of these underlying mechanisms and difficulty predicting these agents suggest the need for techniques to make these agents more explainable and interpretable. For example, a deep neural network may be converted to an interpretable format or be made to produce interpretable outputs.

### 2.2 Explainable Machine Learning

Most work in explainable machine learning focuses on the supervised setting [27], in which a model learns from a labeled dataset $X, Y$ to predict a label $y$ given input data $x$. This area of research is more established than the subfield of XRL, meaning that XRL techniques are often organized and understood using taxonomies and properties from the explainable supervised learning literature [121]. We believe that these properties are useful for understanding and making connections across seemingly disparate techniques in the XRL literature.

The most common taxonomy for organizing explainable machine learning methods groups them into two categories: intrinsic and post-hoc. *Intrinsic* interpretability refers to the direct construction of understandable models, such as small decision trees [101, 126]. The use of intrinsically interpretable models produces a system that is already compliant with interpretability requirements [127]. *Post-hoc* interpretability refers to creating an additional model to explain an existing one. Critically, methods of this type enable either the construction of an intrinsically interpretable model to replace a non-interpretable one or offer explanations of an existing non-interpretable model. These methods are useful for repairing a system to achieve interpretability. Post-hoc and intrinsic interpretability are also useful properties for understanding and organizing XRL techniques. Indeed, we may want to learn an intrinsically interpretable policy or explain it using a number of post-hoc techniques, such as constructing a surrogate model that imitates the non-interpretable policy.

Explainable machine learning techniques differ along a few additional axes, including but not limited to the following. The *expressive power* of an explanation method refers to the structure of explanations the method can produce [111]. For example, an explanation method may produce decision lists or trees to represent a model. In RL, this model may be the policy. To produce already-compliant interpretable models, we must learn the explanation in a form that is considered human interpretable through concrete assessments. Similarly, to repair an existing system, explanations must exhibit an interpretable structure.

Furthermore, an explanation-generating technique may be more or less *algorithmically complex*. Algorithmic complexity measures the required computation time for an algorithm to complete, given an input size. This measurement is important when one is bottlenecked by computation to produce explanations, such as in online settings. Although we do not discuss methods in this work along this axis, it is an important consideration for many real-world applications of explanations.

Explanation methods also differ in terms of the *locality* of the explanation: a local explanation technique produces explanations that explain the prediction for a specific data point; in contrast, a global explanation technique holistically views the machine learning model and its predictions. A local RL explanation explains the immediate action in the context of the state. In contrast, a global explanation depicts the policy behavior over time.

A highly *portable* explanation technique can be used with many different machine learning models. For example, a technique that is indifferent to the underlying model is more portable than one that relies on **Recurrent Neural Networks (RNNs)**. By definition, portable techniques are

Table 1. Important Categories of Metrics for Assessing XRL Techniques

| Metric | Question Answered | Example |
|---|---|---|
| Fidelity | Is the explanation faithful to the underlying explained model? | Model-explanation agreement: $\mathcal{L}(y_m, y_e)$, where $\mathcal{L}$ is a prediction error function (e.g., MSE), $y_m$ is the model prediction, and $y_e$ is the explanation prediction |
| Performance | Can the model do well on the desired task? | Accumulated reward over a $T$-length episode (*return*), $\sum_{t=0}^{T} r_t$ |
| Comprehensibility | Can the end user understand the explanation? | Explanation-audience agreement: $\mathcal{L}(\hat{y}_m, y_a)$, where $\mathcal{L}$ is a prediction error function, $\hat{y}_m$ is the interpretable model's output, and $y_a$ is the audience prediction of that output |
| Preferability | Would the end user use this explanation in practice? | For all use cases, users prefer explanation $e_i$ to all alternatives $e_j \in \mathcal{E}, i \neq j$ |
| Actionability | Does the explanation provide actionable insights for a particular audience? | Audience performance on target task before and after looking at explanations, considering practice effects $\epsilon$, $T'_A - T_A + \epsilon$ |
| Cognitive load | How mentally taxing is the explanation? | Number of seconds required to comprehend the explanation |
| Visualization | What does the explanation look like? | Picture of the generated saliency map |

We list important, non-exhaustive categories of metrics for assessing XRL techniques. For each category, we identify the general question that it answers and provide a grounded example.

often post-hoc methods that are applied to a previously trained underlying model. In RL, this may look like training an additional model to mimic the non-interpretable policy by only querying the non-interpretable policy for action labels of the state. Techniques that are more portable are often less *translucent*. A more opaque method does not require internal access to the underlying model.

All of the aforementioned comparisons can be viewed as properties of different explainable machine learning and, by extension, XRL techniques. In the rest of this article, we try to refer back to many of these properties to help compare and contrast the aggregated papers.

## 3 ASSESSING XRL

Over the years, explainable machine learning researchers have proposed an array of metrics for evaluating techniques in the field [111]. One who designs an XRL method may need to consider potential tradeoffs between different properties of their explanation. Practitioners choosing which XRL methods to deploy also benefit from thorough assessments of these methods: for example, they may prefer an explanations that require very little time to understand. As a result, a few key metrics have emerged for assessing XRL techniques. Table 1 provides a high-level view of these important metrics, including the name of the metric, the general question answered by the metric, and a concrete instantiation of the metric.

In this section, we provide a summary of these key metrics. We define each metric and provide a concrete instantiation of it in the context of XRL, wherever possible. We also comment on the

conditional applicability of some of these metrics. After introducing these metrics, we categorize the literature according to the metrics used for assessment. With this categorization, we identify the most popular metrics for assessing existing XRL techniques. We believe that this perspective is useful for understanding some of the current evaluation shortcomings in XRL.

### 3.1 Metrics for Evaluating XRL Techniques

We review the following metrics: fidelity, performance, comprehensibility, preferability, actionability, cognitive load, and visualizations. Some metrics seek to understand the capability of the interpretable model on the desired task, whereas others seek to assess the understandability of the explanation. Still others seek to understand other useful and important properties of explanations, such as how much cognitive ability it takes for people to understand them and whether they are preferable to other solutions.

*3.1.1 Fidelity.* Fidelity measures how faithful the model explanation is to the explained model. Commonly, the literature operationalizes this metric as *external* fidelity [106], meaning the comparison is between the decision implied by the explanation model and the non-interpretable model. We can also think of this metric as a measure of agreement between the explanation and the underlying explained model.

This metric may look like $\mathcal{L}(y_m, y_e)$, where $\mathcal{L}$ is a proper prediction error measurement (e.g., MSE), $y_m$ is the model prediction, and $y_e$ is the explanation prediction. In the context of RL, $y_m$ may be the action recommended by the non-interpretable policy $\pi(s)$ and $y_e$ may be the action recommended by a surrogate model that approximates the non-interpretable policy $\hat{\pi}(s)$. Then, the metric can be defined as the (potentially weighted by importance) average over time—for example, $\sum_{t=i}^{t=i+k} \mathbb{I}[\pi(s_t) == \hat{\pi}(s_t)]$, where $k$ is the time horizon, if the actions are discrete. Although these evaluations only assess the similarity of output and not the decision-making process, high-fidelity mimicry and prediction is implicated in social intelligence [54] and often does not require an understanding of the exact rationale behind the actions. Fidelity is important to measure when we are producing explanations that are used to interpret a non-interpretable model.

*3.1.2 Performance.* Performance refers to the standard evaluation metric for the task. This metric is important to measure to understand how well we can expect a model to perform when deployed to the target setting. For example, in the episodic RL setting, we may be interested in the the cumulative reward achieved by an interpretable policy $\hat{\pi}$ over the course of the episode, called the *return*. The return is defined as $R^{\hat{\pi}} = \sum_{t=0}^{T} r_t$. We may also incorporate the performance of the non-interpretable policy $\pi$ in this metric by computing $\frac{R^{\hat{\pi}}}{R^{\pi}}$, where the numerator is the return achieved by the interpretable, surrogate policy and the denominator is the return achieved by the non-interpretable policy. In other words, we compare how much of a performance loss we risk if we either use an interpretable policy for deployment or as an explanation.

This metric is particularly valuable when the goal is to deploy the interpretable model instead of the non-interpretable one. In this case, we achieve interpretability by design through the selected model and ensure that it will be successfully applied to the domain of interest by measuring $R^{\hat{\pi}}$. In almost all cases, if there is an interpretable model that achieves high performance on the desired task, we recommend deploying that model rather than trying to explain the non-interpretable one. In fact, there are many machine learning techniques that do not sacrifice accuracy but gain interpretability [31, 173].

*3.1.3 Comprehensibility.* Comprehensibility refers to the ability of the target audience to understand the explanation. Designing tests to assess this ability requires careful consideration [4] and an interpretation of the nebulous term *understanding*. Assessing this metric may include a

self-report for individuals to state whether they comprehend the explanation. However, due to the inconsistency of self-report data [28], we recommend (at the least) checking the reliability of the respondents [164]. Furthermore, researchers may consider utilizing other tests, such as ones that task participants with predicting the output of the explanation (or the explanation itself) prior to seeing the outcome on a set of previously unseen data points.

Denote by $y_a$ the audience prediction of the interpretable model's output $\hat{y}_m$. Then, a simple example of this metric is the explanation-audience agreement, calculated as $\mathcal{L}(\hat{y}_m, y_a)$. The goal is to have $\mathcal{L}(\cdot)$ be close to 0, as it indicates that the audience understands the explanation to the point that they almost always correctly predict its output, even when faced with data points that they could not have memorized.

*3.1.4 Preferability.* End users may prefer to use a specific explanation-generation method compared to alternatives or to no explanation-generation method. A specific instantiation of this metric would be presenting users with a set of different explanations $\mathcal{E}$ and asking them which of the explanations they preferred in the use cases of interest. An explanation $e_i \in \mathcal{E}$ is strictly dominant if for all use cases $u \in \mathcal{U}$ and for all participants $p \in \mathcal{P}$ it is preferred to all other explanations $e_j \in \mathcal{E}, j \neq i$. Note that preferences may be elicited in an explicit or an implicit [57, 144] way. An explicit preference elicitation method asks users to select which of the explanation-generation methods they preferred, whereas an implicit preference elicitation method seeks to understand a user's automatic preferences. Explicit feedback is generally considered to be more accurate [7], but it requires more cognitive effort for study participants to generate. In either case, researchers should carefully design their studies and questions to mitigate potential *subject bias* [113] (e.g., by not recruiting participants in their lab who are aware of their work). For other important considerations in preference elicitation, we refer an interested reader to previous work [120].

A comprehensive assessment of preferability not only assesses preferences but also the reasons behind them. An example of such a reason is trust, as sufficient trust in the model may influence a person's judgment of whether they want to use the explanation-generation method. Explanations that are more preferred should ideally be highly positively correlated with their utility to the user in solving the target task.

*3.1.5 Actionability.* Actionability refers to whether the explanations enable actionable insights for a particular audience in a domain. Actionable insights enable the audience to take meaningful actions based on their understanding of the explanation. This metric is important to assess, especially in application-grounded evaluations, to examine the real utility of explanations for a particular audience on a task. Assessments of this metric may additionally consider factoring in user- and explanation-specific properties to understand how these explanations are useful in different contexts.

A simple example of this metric is the difference in task performance of the intended end users, or audience, before $T_A$ and after $T'_A$ looking at the explanations, accounting for any practice effects $\epsilon$, calculated as $T'_A - T_A + \epsilon$. Practice effects [48] refer to the change in performance from repeated testing and have been demonstrated in a variety of settings [62, 90], so either taking steps to avoid it or explicitly including it in the metric is important to understand whether the explanation actually contributes to a performance improvement. In practical scenarios, like loan applications, an actionable explanation would enable recourse, in which users can understand what they can change to receive a different outcome [65, 108, 147, 154].

*3.1.6 Cognitive Load.* Some explanations require more cognitive resources to comprehend and use. The cognitive resources required for performing a mental task are called the *cognitive load* [2]. One way to measure cognitive load is by recording the number of seconds required for a user to

comprehend an explanation. A naive implementation of this metric would cease the calculation once a user provides a response or otherwise indicates comprehension completion, regardless of the accuracy of the prediction. A more sophisticated implementation may include measurements of user predictive accuracy of the system and user expertise [168], enabling a more nuanced analysis of the tradeoffs of these components. Methods may implicitly attempt to reduce cognitive load by reducing the number of parameters in a model (e.g., measuring the depth of a decision tree). We refer the interested reader to the work in education and psychology on measuring cognitive load [43, 140]. A good explanation helps a user with their target task without adding too much additional cognitive strain.

*3.1.7 Visualizations.* Although not technically a metric, we include visualizations in this category because they are commonly used in place of a quantitative metric for examining the interpretability of an explanation technique. In other words, some papers will provide a visualization of a proposed explanation as a means of demonstrating to the reader that it is understandable. Although visualizations help a reader understand the form of the explanation generated by the explanation method, visualizations alone are not sufficient to measure the comprehensibility of an explanation. For that reason, we refer to visualizations as pseudo-metrics. Note that visualizations do not need to be images: they can include natural language descriptions, logical formulas, and more.

### 3.2 The Most Common Metrics in Existing Work

Equipped with an understanding of the definitions and potential use cases of each of the metrics, we now analyze how commonly they are assessed in the literature. To perform this analysis, we exclude papers that indirectly handle that metric. On one hand, we exclude papers that constrain decision trees to a small maximum depth from the cognitive load category, as it is not the same as explicitly looking at its effect on cognitive load. On the other hand, we include papers that treat parameter counts or tree depth as a proxy metric for cognitive load.

Table 2 categorizes the literature based on the types of metrics used to evaluate the proposed explanation methods. The color of each citation corresponds to the category from Figure 1. Surprisingly, we find that the most common evaluation metric is visualizations (67/73 total papers), which is a pseudo-metric. The second-most commonly utilized metric was performance (56/73 total papers). Given that the goal of XRL is to produce interpretable representations, we find it surprising that metrics like fidelity and comprehensibility are not more popular. The least common evaluation metric is comprehensibility (4/73 total papers), with preferability at a close second (7/73 total papers).

## 4 A NOVEL TAXONOMY FOR XRL

We now turn our attention to our novel taxonomy for categorizing XRL techniques. Existing taxonomies for organizing explainable machine learning literature primarily follow distinctions made in explaining supervised learning systems. Specifically, they focus on post-hoc vs intrinsic interpretability. Although this type of taxonomy serves a valuable purpose in broadly understanding the approaches for enhancing interpretability, it fails to emphasize the unique challenges and opportunities that emerge when applied to RL. In particular, this type of taxonomy does not capture important components of the RL problem, such as action selection and long-term behavior, which are essential for understanding and developing XRL techniques. To address this gap, we propose a taxonomy that prioritizes the RL setting. Different from existing work, we distinguish between the surveyed literature according to the central goals of explanations for RL.

Figure 1 depicts how these categorizations correspond to parts of the RL framework. FI explanations identify the features that affect an agent's action choice $a_t$ for the input state $s_t$. LPM explanations show the past experiences or the components of the MDP that led to the current

Table 2. Common Metrics Used to Evaluate XRL Methods

| Metric | Number of Publications | Publications |
|---|---|---|
| Fidelity | 18 | [22] [32] [39] [60] [61] [67] [83] [84] [92] [98] [102] [103] [105] [115] [133] [146] [163] [167] |
| Performance | 56 | [8] [11] [17] [18] [20] [21] [22] [32] [35] [37] [38] [39] [40] [41] [45] [46] [49] [55] [58] [60] [61] [66] [67] [68] [69] [79] [81] [82] [83] [84] [85] [89] [93] [98] [102] [103] [104] [112] [117] [119] [123] [124] [129] [133] [136] [141] [145] [149] [150] [153] [153] [155] [157] [170] [171] [172] |
| Comprehensibility | 4 | [80] [103] [105] [115] |
| Preferability | 7 | [8] [103] [49] [61] [74] [115] [124] |
| Actionability | 11 | [8] [53] [56] [58] [61] [74] [80] [92] [103] [105] [115] |
| Cognitive load | 12 | [39] [41] [45] [66] [69] [105] [117] [124] [137] [146] [149] [170] |
| Visualization | 67 | [8] [11] [18] [17] [20] [21] [32] [35] [36] [37] [38] [39] [40] [41] [42] [45] [46] [49] [53] [55] [56] [58] [60] [61] [66] [67] [69] [74] [75] [79] [80] [81] [82] [83] [84] [85] [89] [92] [93] [98] [103] [105] [112] [115] [117] [119] [123] [124] [129] [133] [136] [137] [141] [146] [145] [149] [150] [153] [155] [157] [163] [167] [170] [171] |

We categorize the papers discussed in this survey by the metrics used to evaluate them. The color of the citation corresponds to the category from Figure 1 and Table 1: FI explanations, LPM explanations, and PL explanations. Visualizations are the most common pseudo-metric. Performance is the next most common metric. Taken together, these two results show the need for the field to move toward evaluating explanation-specific properties to demonstrate utility and interpretability.

behavior. PL explanations illustrate the long-term behavior of the agent. We believe that this categorization is particularly useful for understanding and organizing XRL literature because each type of explanation communicates a particular aspect of the RL problem.

We posit that this decomposition provides a useful factorization for both understanding the existing literature and developing new techniques. Since a practitioner starts with a goal for using explanations, we prioritize partitioning based on goal. Other aspects, such as the technique used, are used for subcategory division. By using our taxonomy, practitioners can better understand the interplay between their end-use case and the specific parts of the RL problem, enabling them to select explanations that better suit their needs. Similarly, since new techniques should be compared

Table 3. Proposed Taxonomy and Corresponding Publications in XRL

| Category | Subcategory | Number of Publications | Publications |
|---|---|---|---|
| Feature Importance (FI) | Convert Policy to an Interpretable Format | 9 | [17] [18] [20] [39] [60] [83] [102] [150] [170] |
| | Learn an Intrinsically Interpretable Policy | 14 | [37] [38] [40] [45] [68] [69] [93] [104] [117] [124] [136] [145] [149] [171] |
| | Directly Generate Explanations | 21 | [10] [11] [14] [49] [55] [67] [75] [79] [80] [85] [58] [112] [115] [119] [129] [133] [141] [155] [156] [157] [172] |
| Learning Process and MDP (LPM) | Model Domain Information | 8 | [32] [35] [36] [98] [100] [103] [153] [163] |
| | Decompose Reward Function | 7 | [10] [21] [22] [61] [82] [84] [123] |
| | Identify Training Points | 2 | [42] [56] |
| Policy-Level (PL) | Summarize Using Transitions | 6 | [8] [46] [53] [74] [81] [92] |
| | Convert RNN to Interpretable Format | 3 | [41] [66] [89] |
| | Extract Clusters or Abstract States | 4 | [105] [137] [146] [167] |

Category color corresponds to color used in Figure 1. In total, we discuss 73 papers. The most numerous types of explanations fall under the FI category (total papers: 44)—more specifically, those that directly generate explanations (21 papers). This is due to the popularity of saliency maps as an explanatory technique. Indeed, 15 of the 21 papers that directly generate explanations are saliency map techniques. In contrast, LPM and PL explanatory techniques consist of 30 of the 73 total papers. We believe that techniques of these types, especially those that take advantage of and explain the RL-specific components of the problem, are ripe for exploration.

to existing techniques that fulfill their same role, groups that share an explanation goal are suited for organizing related works and experimental comparisons.

For example, if a practitioner wants to investigate the importance of specific features in the agent's decision-making process at a given state, they can use FI explanations. Doing so may enable practitioners to diagnose whether agents are utilizing appropriate features for action selection and adjust their algorithms accordingly. In contrast, if a practitioner wants to understand the agent's interaction with the environment and its learning process over time, they can focus on LPM explanations. If a practitioner wants to summarize the long-term behavior of the agent, they can use PL explanations. Because this decomposition first prioritizes *what* the practitioner wants to understand, it provides more of a use case centered perspective on explanations for RL.

Table 3 depicts the high-level taxonomy and corresponding XRL publications. By decomposing the literature in this way, we discover some interesting findings. We find that the most popular high-level category of explanations is FI (total papers: 44)—specifically, the subcategory of *Directly Generate Explanations* (21 papers). In contrast, there are only 13 papers that fall under the PL category and 17 papers that fall under the LPM category. We believe that this discrepancy is likely

due to the common practice of extending supervised learning techniques to the RL setting, which results in an FI technique by definition. As a result, there are ample opportunities to develop techniques in less commonly studied categories, such as PL explanations.

We now discuss the three high-level categories in turn. In addition to defining each category, we compare and contrast the techniques with one another to showcase what they do and do not enable. With these comparisons, we aim to illustrate the tradeoffs between the different types of explanations and help form a more concrete understanding of the proposed taxonomy.

### 4.1 FI Explanations

FI explanations provide an action-level look at the agent's behavior: for each action, one can query for the immediate context that was critical for making that decision. These techniques answer questions such as "what immediate context influenced the agent to perform that action?" These explanations are local, as they explain individual decisions of the agent. Some techniques guarantee that the explanations capture the information used in the agent's decisions, but others do not. For example, techniques that directly produce decision tree policies necessarily elucidate the features and corresponding values that influence the agent's decision. In contrast, techniques that generate post-hoc *rationalizations*, like some natural language explanations, produce explanations based on a human's decision-making process when solving a task. Furthermore, FI explanations do not provide global behavioral summaries that convey subtask or plan information: they only explain individual action choices. PL explanations, however, provide this information by producing summaries using transitions, model approximations, or similar states.

### 4.2 LPM Explanations

LPM explanations provide additional information about the effects of the training process or the MDP. Some LPM explanations enable understanding of the influential experiences that led to the agent's current behavior. Others describe how the agent acts in terms of the reward or objective. These techniques answer questions like "which training points were most influential on the agent's learned behavior?" and "which objectives is the agent prioritizing?" These explanations may be global or local, depending on what is being explained (e.g., behavior or individual actions).

Many of these techniques generally require extra information or learning during training, so we cannot generate them post-hoc. For example, understanding which objectives the agent is prioritizing require the explanation designer to use domain information to decompose the reward function before any learning occurs. In contrast, many FI and PL explanations do not require additional information and/or learning, so we can generate them post-hoc. These post-hoc techniques can be used to repair an existing non-interpretable system, whereas the techniques that require additional information are interpretable by design.

### 4.3 PL Explanations

PL explanations present summaries of long-term behavior through abstraction or representative examples. These types of explanations are critical for understanding an agent's behavior to evaluate its overall competency. PL explanations answer questions like "how will the agent behave over time?" By definition, these explanations are global.

Because these explanations are summaries, they necessarily exclude details for the sake of interpretability. For example, techniques from Section 7.3 construct summaries of agent behavior by aggregating similar states and showcasing how the agent will behave. They exclude information in the state aggregation techniques by definition of abstraction. In contrast, intrinsically interpretable FI explanations explicitly include all of the information available about how the agent makes choices.

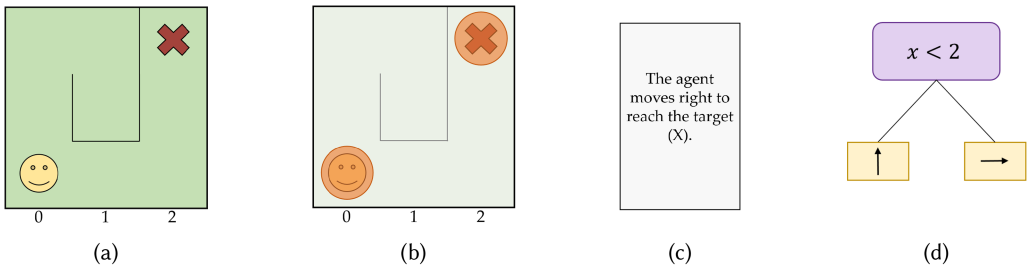


Fig. 2. Examples of different types of explanations. Given a state $s_t$ from a domain (a), many types of explanations can be created, such as an object saliency map (b), a natural language explanation (c), and a decision tree policy (d). Note that the explanations convey different information. For example, the natural language explanation (c) does not contain the counterfactual information of the decision tree policy (d).

## 5 FEATURE IMPORTANCE

Many FI techniques for RL extend those from the explainable supervised learning literature. Viewed in the supervised learning framework, the input data is the state and the label is the action. The objective is then to query the policy $\pi$ or value function $Q^\pi$ to determine which action an agent would take given an input state. Figure 2 depicts an example of different types of explanations that could be generated given a state. These explanations utilize different representations to communicate potentially different information. The object saliency map visually depicts the objects in the environment that contextualize a particular action choice, whereas the natural language explanation does so in terms of natural language. The decision tree policy communicates additional counterfactual information through its well-defined structure.

As mentioned in Table 3, the FI category consists of three subcategories: convert policy to an interpretable format, learn an intrinsically interpretable policy, and directly generate explanations. Some FI techniques repair existing uninterpretable systems through a post-hoc conversion step (Section 5.1). Other techniques directly learn an interpretable policy by design (Section 5.2). Still other methods do not learn an interpretable policy and, instead, generate action explanations in various forms, such as natural language (Section 5.3). These techniques are more similar to those in Section 5.1 because they seek to repair an existing uninterpretable system. We now discuss each of the three types of FI techniques in turn.

### 5.1 Convert Policy to Interpretable Format

We first discuss methods that repair an existing uninterpretable policy to make it interpretable. These methods approximate an existing, performant neural network policy $\pi^*$ with an interpretable *surrogate* model $\hat{\pi}$. To learn $\hat{\pi}$, we can use *imitation learning* [1]—also known as *learning from demonstration* [12]—where the RL policy $\pi^*$ is the *expert* and the imitation learning policy $\hat{\pi}$ is the *learner*. The problem becomes a classification problem: the learner aims to correctly classify which action (label) to take from a state, given examples from the expert. These methods are generally quite portable, as they make little to no structural assumptions on the expert.

VIPER [18]—and its multi-agent extension [107]—is an exemplar technique for extracting decision tree policies from a neural network using imitation learning. It extends the classic DAGGER algorithm [125] for imitation learning by training a sequence of decision trees on data points sampled based on how "critical" they are, measured by $V^\pi(s) - \min_{a \in \mathcal{A}} Q^\pi(s, a)$. More concretely, VIPER runs for $N$ iterations. In each iteration $i \in N$, it samples $M$ trajectories from a dataset $\mathcal{D}_i$ induced by following the decision tree policy trained at the previous iteration. Let $d^{(\hat{\pi}_{i-1})}$ be the

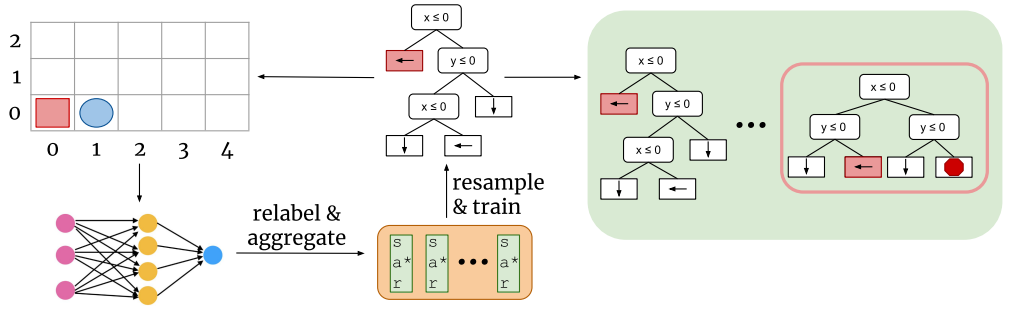


Fig. 3. Overview of the VIPER algorithm. In the VIPER algorithm [18], an *expert* trains a decision tree policy in an iterative fashion. The expert interacts with the environment to collect an initial dataset of state-action pairs. At each iteration, the previously trained decision tree policy interacts with the environment to produce more data samples. The expert relabels the actions, and the relabeled data and the existing data are aggregated into a single dataset. VIPER then resamples data points from this dataset to prioritize critical states and trains a decision tree policy on this resampled data. At the end of the training process, VIPER selects the best decision tree policy from the collection of decision tree policies.

distribution induced by following the decision tree policy $\hat{\pi}$ at the previous iteration $t-1$. Then, the data collection step is $\mathcal{D}_i \leftarrow \{(s, \pi^*(s)) \sim d^{(\hat{\pi}_{i-1})}\}$. Note that the action is relabeled by the expert: $\hat{\pi}_{i-1}(s) \rightarrow \pi^*(s)$. This dataset is then aggregated with all previous data: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$. The complete dataset is resampled according to the aforementioned measure of "criticalness." VIPER then trains a new decision tree policy on the resulting dataset. After all iterations are complete, VIPER chooses the best-performing decision tree policy of the $N$ trained ones. Figure 3 visualizes this process.

Because VIPER makes no assumptions of the structure of the expert—only that the learner has access to it—it is highly portable. At the same time, this property makes VIPER less translucent. If the decision tree is used as an explanation, then it is considered a post-hoc technique; if it is used to replace the non-interpretable policy, then it is considered an intrinsically interpretable technique.

Also inspired by DAGGER, PIRL [150] locally searches over interpretable programmatic policies by minimizing the distance between the deep RL computed expert policy $\pi^*$'s outputs and the programmatic policy $\hat{\pi}$'s outputs on heuristically chosen states. The main idea is to find a program $\hat{\pi}$ that closely imitates the expert. Other work that similarly learns programmatic policies through imitation learning [17] demonstrates the desirable properties of this representation, including interpretability, verification, and robustness, through case studies. Jhunjhunwala [83] distills deep RL policies into Q-value-based trees with nodes that encode more complex boundary expressions than standard decision trees. Zhang et al. [170] derive $\hat{\pi}$ from $\pi$ with genetic programming. TripleTree [20] expands the information used in the tree-learning algorithm. For example, the algorithm uses the variance in value estimates in their splitting criteria. Other work [39] extracts an oblique decision tree [161] using the rate of information gain as the splitting criteria for the purpose of power system control. Liu et al. [102] introduces decision trees with linear models at the leaves and proposes to use stochastic gradient descent to update their values. Still other work [60] distills non-interpretable policies into soft decision trees [52], which are decision trees with sigmoid activations rather than Boolean operators.

Although imitation learning offers a promising solution when a high-performing expert is available to be queried, it faces some unique challenges. For example, an imitation learning policy may not be robust to novel situations, such as changes in the task and surrounding environment. For a more extensive discussion of challenges with imitation learning, we refer the interested reader to

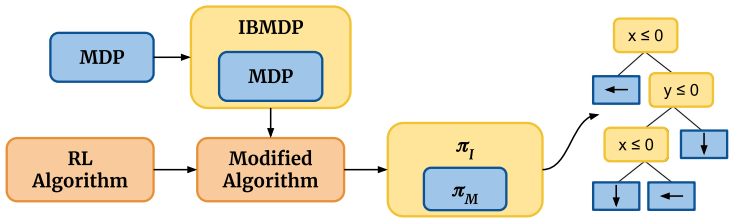


Fig. 4. Overview of the CUSTARD algorithm for learning interpretable decision tree policies. CUSTARD [146] constructs an augmented MDP, called an *IBMDP*, that contains expanded action and state spaces. These additional components correspond to the tests and outcomes, respectively, of growing a decision tree. The resulting policy takes the form of a decision tree while still enabling the use of a neural network for training.

a previous survey of the literature [76]. However, as robust imitation learning techniques advance, we expect even more developments in post-hoc XRL.

Importantly, the interpretable policies produced by these methods may be used as either inherently interpretable policies for deployment (given that they achieve relatively high performance) or explanations of the underlying non-interpretable policies. If used as an explanation, practitioners should be careful to understand the instances where the interpretable model and the underlying neural network deviate. Additionally, in this case, *fidelity* becomes a more important metric than performance.

### 5.2 Learn an Intrinsically Interpretable Policy

We now overview methods that directly learn intrinsically interpretable policies. By *directly*, we mean that the policies are learned in an interpretable format, without any conversion step. These methods produce interpretable models *by design*; however, they require augmentations to the training procedure. We primarily distinguish techniques by the *form* of the learned policy. Because techniques that utilize decision trees are so numerous, we construct two categories to represent the different forms: decision trees and other representations.

We first discuss methods that learn decision tree policies. CUSTARD [145] is an exemplar technique for directly producing an intrinsically interpretable policy in the form of a decision tree. Figure 4 depicts an overview of the CUSTARD algorithm. CUSTARD achieves intrinsic interpretability by constructing the MDP such that it naturally produces a tree-structured policy.

This augmented MDP, called an *IBMDP*, contains the actions from the original MDP and introduces actions for constructing a decision tree (i.e., choosing a feature to branch a node). Let $\mathcal{A}_M$ represent actions from the original MDP and $\mathcal{A}_I$ represent the actions introduced by the new formulation. Then, each action $a_i \in \mathcal{A}_I$ consists of a tuple $\langle c, v \rangle$, where $c$ refers to the chosen feature and $v$ is the threshold value for that feature. This action corresponds to the test in the decision tree $c \leq v$ that controls the branching. In other words, the new actions $\mathcal{A}_I = \{f_1, \ldots, f_N\} \times \{v_1, \ldots, v_K\}$ correspond to decision tree tests, where $f_n$ corresponds to the chosen feature $n$ (of the $N$ total features) and $v_k$ corresponds to the selected value $k$ (of $K$ possible values). Therefore, the full action space for the augmented MDP is $\mathcal{A} = \mathcal{A}_M \cup \mathcal{A}_I$.

To reflect the partitioning of the decision space, we need a way to store the outcomes of these tests. The natural place to store these outcomes is in the state such that the agent is constrained to only use the knowledge from these partitions in its decision-making process. Consequently, the augmented state space contains additional features that correspond to the bounds on the original state space's features as chosen by the agent. Let $u_{f_n}$ and $l_{f_n}$ refer to the upper and lower bounds of feature $f_n$ for all $n \in N$. Then, the new features $\mathcal{S}_I = \langle l_{f_1}, \ldots, l_{f_N}, \ldots, u_{f_1}, \ldots u_{f_N} \rangle$ correspond

to these resulting upper and lower feature bounds. For example, suppose an agent chooses feature $f_n$ and value $v_k$. If the true value for that feature is less than or equal to the chosen value $v_k$, the new upper bound for that feature $u_{f_n}$ becomes the minimum of $v_k$ and the existing upper bound. Similarly, if the true value for that feature is greater than the chosen value $v_k$, the new lower bound $l_{f_n}$ becomes the maximum of $v_k$ and the existing lower bound. Therefore, the full state space for the augmented MDP is $\mathcal{S} = \mathcal{S}_M \cup \mathcal{S}_I$, where $\mathcal{S}_M$ is the state space for the original MDP.

To ensure valid correspondence to a decision tree policy, the policy only receives $\mathcal{S}_I$ as input during training: $\pi : \mathcal{S}_I \rightarrow \mathcal{A}$. Training the agent in this augmented MDP means that the agent directly learns a decision tree policy for the original environment while still using neural networks during training. Although this algorithm requires modifications to the training process, the changes are relatively minimal, making this technique portable to a variety of RL algorithms.

Another technique for permitting gradient-based training involves augmenting the structure of the tree itself. Because decision trees are not differentiable, Rodriguez et al. [124] propose to learn a soft decision tree [52], called a *differentiable decision tree* [138], in which sigmoid activation functions replace the Boolean decisions in classic decision trees. At the cost of performance, this tree can then be discretized to form a standard decision tree policy that approximates the original, soft decision tree.

Other techniques that learn decision tree policies do so without gradients. Most commonly, these training procedures utilize a modified genetic algorithm [37] or evolutionary approach. For example, one technique extracts information from the environment, such as locations of objects, and uses these features to directly learn a tree with an evolutionary approach [38].

In addition to choosing the algorithm for learning an interpretable structure, we must also choose the structure that we want to learn. In the aforementioned cases, we learned classic decision trees. However, decision trees have drawbacks: they produce only axis-parallel partitions, cannot handle approximate reasoning, and are non-differentiable. Alternative policy representations mitigate these issues. Trees with algebraic expressions represented by nodes [69, 93] and non-linear decision trees [45] can represent complex functions because they are more expressive than the standard Boolean comparisons. Utilizing automatons (e.g., through Tsetlin machines [149]) enables the use of more general graph structures. Including conceptual embeddings [40] permits learning interpretable intermediate representations. Fuzzy controllers [68] enable approximate reasoning because they permit truth values of variables to range between 0 and 1, as in fuzzy logic [166]. Certain logic formulations [136, 171] and tree structures [117] permit differentiable learning. Despite their ability to resolve the issues with classic decision trees, most of the aforementioned approaches have not been evaluated with user studies to understand their actionability, comprehensibility, or preferability.

All of the techniques in this section enjoy the benefit of producing an intrinsically interpretable policy. Although decision trees are generally considered interpretable, their qualities may not fit a particular application (e.g., one that necessitates approximate reasoning). Other representations risk more initial cognitive load from a user to understand, so they are more applicable when their long-term benefit outweighs the cost of the initial scaffolding for understanding explanations [47]. Furthermore, these alternative representations ought to be evaluated more thoroughly to understand when (and whether) they provide more useful and understandable explanations.

### 5.3 Directly Generate Explanation

The final set of FI techniques do not learn an interpretable policy; instead, they generate action explanations from a non-interpretable policy. Explanations of this type sometimes require auxiliary components to produce, meaning that it is only possible to repair an existing non-interpretable system to produce these explanations if we have access to these components and the required data.

Table 4. Example Queries and Responses for a Natural Language Explanation Technique

| Template Query | Example Query | Template Response | Example Response |
|---|---|---|---|
| When do you <action>? | When do you *pick up the widget*? | I <action> in <region>. | I *pick up the widget* when *my camera detects a widget and the widget is in reach.* |
| What will you do when <disjunctive normal form state region description>? | What will you do when *a person is near you*? | I <action> when <state region description>. | I *perform no action* when *a person is near me.* |

This table depicts example queries and responses for a specific natural language explanation technique [67]. The left-most column provides two examples of template queries that a user can make to the system. The middle-left column provides an instantiated example of each template. The middle-right column shows template responses to the queries, and the right-most column shows an instantiated example of each templated response.

These explanations most commonly take the forms of natural language and saliency maps. Natural language explanations linguistically describe the agent's decision-making process, whereas saliency maps are images that topographically highlight regions considered important for an agent's decision. There are also a few other types of explanations that cannot be categorized as one of the aforementioned types. We discuss each of these high-level types of explanations in turn.

*Natural Language.* We distinguish natural language explanations by their means of production. Some work leverages templates for the agent to fill in [67]; other work allows the agent to generate free-form explanations (Table 4). Some techniques [49, 156] leverage human-provided action-explanation pairs to rationalize agent behavior. We identify the work by Hayes and Shah [67] as the exemplar for this section.

In the work by Hayes and Shah [67], an explanation is a function $g : I \times L \rightarrow N$, where $I$ is a natural language inquiry, $L$ is software control logic, and $N$ is a natural language response. The software control logic $L$ refers to the underlying control logic that governs the behavior of a robot or autonomous agent. This logic may be embedded in modern machine learning methods, making it challenging to directly interpret. For examples of inquiries $I$ and responses $N$ enabled by this method, please refer to Figure 4. Given this formulation, they introduce a set of question templates $\mathcal{T}$ and binary classifiers to characterize aspects of the state $C$. These binary classifiers are also known as predicates [51]. For example, a predicate may ask whether a robot is in a certain region or not.

The high-level process of producing an explanation is decomposed into four distinct parts. First, identify the question being asked, which maps the inquiry to a template $I \rightarrow \mathcal{T}$. Second, resolve the question template to relevant states, which maps the identified template and the underlying software control logic to a subset of relevant states $\mathcal{T} \times L \rightarrow \hat{L}$ such that $\hat{L} \subseteq L$. Third, concisely summarize the relevant attributes of these states $\mathcal{T} \times L \rightarrow C^n$. Finally, compose summaries into natural language to express to a person $C^n \rightarrow N$.

Papers that fix a response structure typically enjoy more clarity than those that do not. However, it requires an adherence to a pre-specified template, which must be defined to repair an existing non-interpretable system. Recent advances in natural language processing through large language models [23, 26, 44] may enable us to move toward more free-form explanation-generation methods without sacrificing clarity. Furthermore, there is some question about the validity of some of these types of explanations. Some techniques produce post-hoc rationalizations that produce plausible but not valid explanations. The goal of explanations is not to increase trust in the system but to

Table 5. Saliency Maps and Their Associated Categorization

| Type | Publications |
|---|---|
| Gradient | [75] [157] |
| Perturbation | [10] [58] [119] [129] |
| Object | [55] [80] |
| Attention | [11] [79] [85] [112] [133] [141] [172] |

Most techniques utilize some form of attention. Interestingly, papers that generate explanations using saliency maps are more numerous than natural language and "other" forms of explanations combined. The popularity of such techniques may be in part due to both their popularity for interpretable machine learning in general and the common use of images as input to RL algorithms.

allow the user to make a choice about trusting the system [127]. Human-like explanations that are plausible but not valid may increase agent trust, but the trust is unfounded without grounding the explanations to the agent's decision-making process.

*Saliency Maps.* We categorize saliency map techniques using the decomposition in the work of Atrey et al. [14] (Table 5). For example, Greydanus et al. [58] construct saliency maps by perturbing the input image describing the state with Gaussian blur, then measuring changes in the policy after removing information from an area. For more details, we refer the reader to the work of Atrey et al. [14]. Saliency is viewed as insufficient for explaining RL, as the conclusions drawn from it are highly subjective [14]. Indeed, the majority of claims in the surveyed papers either propose an *ad-hoc* explanation of agent behavior after observing saliency or develop an *a priori* explanation of behavior evaluated using saliency. Although these techniques may be useful as a debugging tool, we desire more reliable explanations for high-stakes deployment scenarios.

*Other.* There exist other forms of explanation, including characteristic functions [155], which assign values to every possible subset of features, and counterfactual states [115], which illustrate the minimal change needed to induce a different action selection $a'_t$. To produce counterfactual states, one can train a deep generative model. The query state $s_t$ is first encoded to a latent representation. We can then move in the latent space to increase the probability of performing $a'_t$ in a way that is amenable to creating realistic outputs. To highlight changes from the original state to the counterfactual state, we can apply saliency map techniques [58] to highlight the changes.

This area is ripe for exploration, especially given the recent interest in multimodal machine learning [15]. In this setting, we may want custom local explanations that reflect different modalities. Additionally, given that this area produces novel forms of explanations by design, we should take care to evaluate these explanations on user-specific metrics, like comprehensibility and actionability. In particular, it would be interesting to see which forms of explanations are preferred in multimodal settings and whether a unimodal explanation suffices for a multimodal task.

## 6 LEARNING PROCESS AND MDP

LPM explanations reveal the influence of other parts of the MDP (e.g., $R$ and $T$) or the interactions with the MDP during learning on the agent's behavior. As shown in Table 3, this category consists of three subcategories. Most commonly, the agent reveals its learned transition dynamics $\hat{T}$ (Section 6.1), referred to as model domain information. Alternatively, techniques decompose the reward function to produce explanations in terms of the agent's objectives (Section 6.2), which we refer to as decompose reward function. Other techniques determine training points that influenced the learned policy (Section 6.3), known as identify training points.

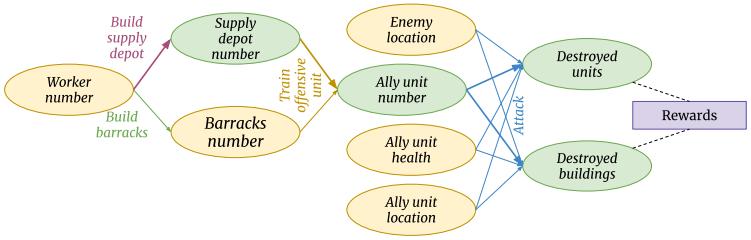


Fig. 5. Example action influence graph in a Starcraft II game environment. The action influence model provides a backbone for producing complete action explanation: the causal change from that action to any future reward that it can receive [103]. For example, the causal chain for action *build supply depot* is depicted by the bolder edges. Each colored edge corresponds to a unique type of action (e.g., blue arrows correspond to the attack action). Each node corresponds to one of the nine state variables.

Some of these methods are interpretable by design, whereas others repair existing systems. Most of these techniques fall under the former category. They commonly involve specifying additional structure to enable decomposed reward representations, causal models of the agent's influence on the environment, and more. Techniques that repair existing systems typically assume access to the training data to select important points after training.

### 6.1 Model Domain Information

Approaches in this category learn models $\hat{T}$ that approximate $T$ and generate explanations based on $\hat{T}$. For a recent survey of techniques that learn $\hat{T}$ not for the purpose of explainability, we refer the interested reader to the work of Moerland et al. [110]. These techniques primarily differ in terms of the questions these generated explanations answer.

We identify the work on integrating causal models with RL as our exemplar [103]. This work uses causal models to generate contrastive explanations for *why* and *why not* questions. They introduce action influence models $A_M = (\mathcal{S}_a, \mathcal{F})$ (see an example of an action influence graph of a Starcraft II game environment [152] in Figure 5), which are based on (pre-specified, potentially learned [153]) structural causal models [64]. Structural causal models decompose the world with random variables of two types: exogenous (external) and endogenous (internal). The potential relationships between these variables can be described with a set of structural equations.

Action influence models define a signature $\mathcal{S}_a$ that consists of the set of exogenous variables $\mathcal{U}$, the set of endogenous variables $\mathcal{V}$, and the set of actions $\mathcal{A}$, and a function $\mathcal{R}$ denoting the range of values for every variable. An action influence model is a tuple $\mathcal{S}_a, \mathcal{F}$, where $\mathcal{S}_a$ is the signature and $\mathcal{F}$ is a set of structural equations—one for each *unique* action set that influences $X \in \mathcal{V}$. The causal effect on $X$ from applying an action $A$ is defined by a function $F_{X,A}$, and the set of reward variables $X_r \subseteq \mathcal{V}$ are defined by the set of sink nodes. In other words, action influence models are structural causal models except that each edge is associated with an action.

In this work, the structural equations are learned during the RL process. It is assumed that the structure of the graph is pre-specified, but it could also be learned [153]. This structure consists of a directed acyclic graph specifying causal direction between variables. The structural equations are learned as multivariate regression models.

To generate explanations for "why?" questions, this work sets the desired goal to achieve as the predecessor nodes of the rewards. We can traverse the causal graph to obtain explanations. Let $\overrightarrow{X}_r$ be the vector of reward variables reached by following the causal chain of the instantiated graph $M_{\overrightarrow{\mathcal{V}} \leftarrow \overrightarrow{S}}$ to sink nodes. Let $\overrightarrow{X}_h$ be the vector of variables of the head node of action $a$. Let $\overrightarrow{X}_p$ be

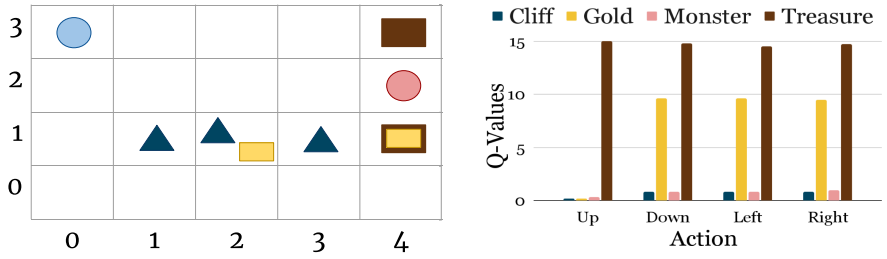


Fig. 6. Treasure-hunting scenario (left) and its corresponding reward decomposition (right). In this setting, a robot (blue circle) tries to find treasure (brown rectangle with gold center) while avoiding cliffs (blue triangle) and monsters (pink circle). As a result, this task has a clear semantic decomposition of reward: falling off cliffs gets a high penalty, encountering monsters gets a small penalty, finding an empty treasure chest (brown rectangle) gives a small reward, finding gold (gold rectangle) gives a higher reward, and discovering a full treasure chest (brown rectangle with gold center) offers the highest reward. The right figure depicts the resulting reward decomposition for the HRA [148] algorithm. Intuitively, treasure is the most important reward component for all actions.

the vector of variables that are immediate predecessors of any variable in $\overrightarrow{X}_r$ within the causal chain. The corresponding values of these variables under the instantiated graph are $\overrightarrow{x}_r$, $\overrightarrow{x}_h$, and $\overrightarrow{x}_p$. Then, the produced explanations are in the form of $(\overrightarrow{X}_r = \overrightarrow{x}_r, \overrightarrow{X}_h = \overrightarrow{x}_h, \overrightarrow{X}_p = \overrightarrow{x}_p)$.

To answer "what-if" questions, Rupprecht et al. [129] learn a generative model of the environment, which enables the probing of agent behavior in novel states created by an optimization scheme that induces certain actions in agents. For example, one can optimize for states where the agent expects to perform poorly, then visualize such states to observe the agent's behavior. To contrast between a selected action $a_t$ and alternative ones $a'_t$, Cruz et al. [35, 36] use episodic memory to explain decisions with success probability—that is, "$a_t$ gives an 85% probability of task success compared to 38 % of $a'_t$." Lin et al. [100] provide action preference explanations in terms of human-provided features. Yau et al. [163] provide explanations in terms of intended outcome: the agent projects predicted future trajectories from its current state and proposed action. To understand correlative context for failure cases, Chen et al. [32] learn a sequential latent environment model that can be used to generate a semantic mask to understand the surrounding context. Other work [98] learns a graph neural network [19] policy to capture the structure of the environment and later extract rules.

Explanations can address a plethora of potential questions a user may have. Clearly defining these questions and the ability of the techniques to address them is important for users to understand an explanation's capabilities. For example, we do not want users to assume causal relationships when the explanation only provides correlations.

### 6.2 Decompose Reward Function

By conveying aspects of the reward that contributed to the agent's behavior, an explanation provides information in terms of the agent's objectives. Most work in this area focuses on presenting interpretable views of $Q^\pi(s, a)$.

Before the agent starts learning, the explanation designer can use reward decomposition [10, 84] to split $R(s, a)$ into a set of additive terms with semantic meaning: $R(s, a) = \sum_{c \in C} R_c(s, a)$, where $C$ is the set of reward types. $Q^\pi(s, a)$ is similarly decomposed as $Q^\pi(s, a) = \sum_{c \in C} Q^\pi_c(s, a)$. With this decomposition, one can inspect actions in terms of tradeoffs between the types. Figure 6 shows an example reward decomposition chart that results from the semantic decomposition of the reward

function in a treasure-hunting domain. Intuitively, the treasure is the most important component of the reward function.

However, this approach cannot be used with *any* RL algorithm. As mentioned in the original paper [84], some algorithms do not produce $Q_c^\pi$ with values that are accurate to the greedy execution policy, which yields explanations that violate intuition. These approaches adjust the value of each $Q_c^\pi$ toward $r_c + \gamma \max_{a'} Q_c^\pi(s', a')$. Instead, for the reward components to take on the correct values, we need to use algorithms that train each $Q_c^\pi$ toward the greedy policy by first computing $a^+ = \arg\max_{a'} \sum_{c \in C} Q_c^\pi(s', a')$ to update each component $Q_c^\pi(s, a)$ toward $r_c + \gamma Q_c^\pi(s', a^+)$. As a result, explanation designers can only construct XRL algorithms of this type by design.

To contextualize this one-step (local) reward decomposition, one technique [123] utilizes hierarchical RL [16] to analyze the effect of hierarchical goals on the interpretability of these explanations. More concretely, they employ a two-level hierarchy: the top level learns a sequence of goals and the bottom level learns how to satisfy these goals [91]. The objective is then to use these hierarchical goals to have an explicit understanding about the objective of the agent while also understanding the concrete action selections using reward decomposition.

Other work [21] similarly presents explanations using a Q-value decomposition; however, they decompose the task into subtasks to visualize per-subtask Q-values as a heat map over the environment. Although this technique offers a way to repair an existing system by generating explanations, these visualizations are more challenging to interpret and are not evaluated with end users.

To answer "what-if" questions with respect to specific actions, Bica et al. [22] model an expert's reward function in terms of preferences over proposed alternative outcomes. They aim to recover the unknown reward function of an expert $R^*(h_t, a_t) = w^* \cdot \phi(h_t, a_t)$ by learning $R(h_t, a_t)$, where $h_t$ is a realization of the history of observations and actions until timestep $t$ and $\phi$ is the feature map. To define the feature map, they use counterfactual reasoning to explain the expert's behavior in terms of the tradeoffs associated with alternative outcomes: $\phi(h_t, a_t) = \mathbb{E}[Y_{t+1}[a_t]|h_t]$, where $\mathbb{E}[Y_{t+1}[a_t]|h_t]$ is the potential outcome from taking an action. Along similar lines, other work [82] aims to learn a reward function from human feedback by first preprocessing reward functions into simpler but equivalent functions.

Reward-based explanations may help laypeople understand how reward-based objectives influence the decision-making process of an RL policy [10]. However, we require further explanatory steps to produce actionable explanations. For example, a decomposition of the reward function could be combined with an approach from Section 6.3 to illustrate which experiences impact each of the $Q_c^\pi$ estimates. Ideally, this combination also reveals the experiences to remove to affect the estimates of these different components. An example technique that yields more actionable explanations of this form is EDGE [61], which learns to identify which timesteps contributed to the final reward to identify policy errors and weaknesses.

### 6.3 Identify Training Points

These techniques identify important training points, or transition tuples, that influence the agent's learned behavior. Influence commonly refers to a change in some function value (e.g., $Q^\pi(s, a)$).

We identify the work on interpretable off-policy evaluation [56] as the exemplar for this type of work. Off-policy evaluation in RL refers to estimating the value of a given evaluation policy $\pi_e$ using data collected from a different behavior policy $\pi_b$. This work identifies points that are *influential* for value estimates with the insight that highly influential transitions should be presented to human experts to verify whether the data points contain mistakes, such as human measurement errors. Figure 7 depicts the high-level human-in-the-loop process enabled by interpretable off-policy evaluation. Given access to trajectories and the computation required to compute these influence functions, this technique makes a system interpretable by design in this context.

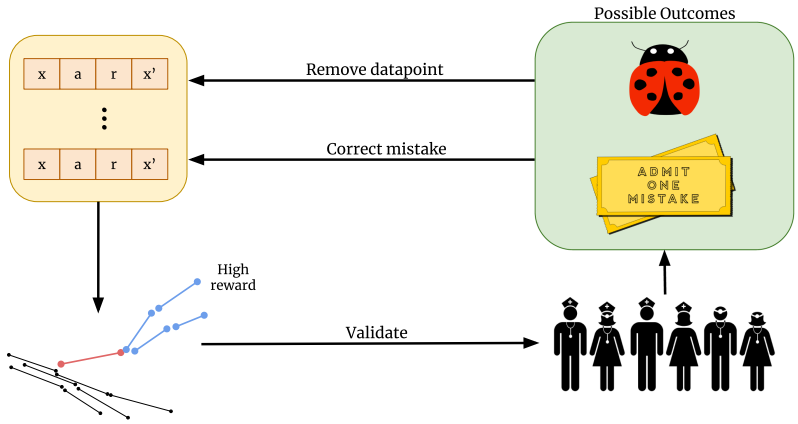


Fig. 7. Use cases of identifying influential transitions in a medical scenario. When training data is human-generated, there can often be mistakes in labeling or measurements. To help correct these cases, interpretable off-policy evaluation [56] identifies and extracts influential transitions. Experts—in this case, doctors—can inspect and validate these transitions. One use case is identifying and removing an influential, buggy measurement from the dataset. Another use case is identifying and correcting temporal misalignment present in the data. In general, this work can be used as a decision support tool for injecting human expertise in evaluating OPE methods for noisy and confounded observational data.

To evaluate influence, the authors propose a few measurements, including total influence and individual influence. The total influence $I_j$ of a transition $\tau^{(j)}$ is the change in value estimate $\hat{v}$ if that transition was removed: $I_j \equiv \hat{v}_{-j} - \hat{v}$, where $\hat{v}_{-j}$ is the value estimate of the dataset when $\tau^{(j)}$ is removed. The individual influence of a trajectory refers to the change in the Q-estimate $q(x^{(i)}, a^{(i)})$ after the removal of a trajectory $\tau^{(j)}$, defined as $I_{i,j} \equiv \hat{q}_{-j}(x^{(i)}, a^{(i)}) - \hat{q}(x^{(i)}, a^{(i)})$. With these influence functions, we can compute other metrics of interest, such as the normalized absolute value of the influence of a trajectory.

These influence functions can be efficiently computed to identify whether there exist any significantly influential transitions. The authors present a few methods to compute the influence functions, including kernel-based and linear least squares versions of fitted Q-evaluation [96]. We refer the interested reader to the paper to understand the precise implementation of these algorithms. If these algorithms do not identify influential trajectories, then OPE seems reliable. If they do, then the trajectories can be presented to domain experts, such as doctors, to identify potential problems with the data, like temporal misalignment and buggy measurements. After correcting these mistakes, we can rerun interpretable off-policy evaluation to check if there still exist any highly influential trajectories.

Other work [42] selects and records states with associated importance weights that correspond to different decisions in a learned policy. Equipped with this knowledge, one can understand the most influential transition tuples for a policy's individual decisions. However, unlike the interpretable off-policy evaluation work, it is unclear how to amend the training process to remove undesirable behaviors based on this information alone.

## 7 POLICY-LEVEL

A less-studied but important category of explanations is those which describe an agent's longer-term behavior. We discuss three subcategories, as mentioned in Table 3: summarizing behavior in the context of experiences encountered by the agent during training (Section 7.1), converting

RNNs to interpretable forms (Section 7.2), and summarizing behavior according to similar states (Section 7.3).

### 7.1 Summarize Using Transitions

We can explain an agent's behavior in terms of the experiences encountered by the agent during training. With these explanations, human operators know which experiences influence the agent's behavior, providing more information to adjust the behavior if desired. These methods are distinct from those described in Section 6.3 because the goal is to describe the agent's *long-term* behavior based on influential *trajectories*, not individual transition tuples. These trajectories are combined to create a summary of agent behavior.

We identify the work by Amir et al. [9] as the exemplar technique of this type. This work selects trajectories based on the state importance, defined as $I(s) = \max_a Q^\pi(s,a) - \min_a Q^\pi(s,a)$, and diversity, which is considered by identifying the most similar state $s_s$ to a state $s$ currently included in the summary and comparing $I(s_s)$ with $I(s)$. The more important state and its corresponding trajectory is used in the summary. The summary therefore consists of a set of important (and potentially diverse) trajectories.

Other methods largely differ from one another due to the different selection criteria for choosing the trajectories or transition tuples. Huang et al. [74] select states where the chosen action has a much higher Q-value than another action. Dinu et al. [46] use Align-RUDDER [118] to identify key events in trajectories. Align-RUDDER performs a contribution analysis to redistribute rewards to these key events. Jacq et al. [81] present a new type of MDP in which a default behavior is chosen unless an agent chooses to act. This formulation enables us to see the states in which the agent chooses to take control by overriding the default action. These transitions are considered important.

To investigate the effects of different summary-extraction models on policy reconstruction, Lage et al. [92] compare approaches for choosing subsets of $s$, $a$ pairs that characterize the agent's behavior. They find that the best type of summary-extraction model is highly subjective, which motivates personalizing user models for summarization.

The aforementioned techniques help people understand agent behavior in important states. However, these techniques may not be helpful if there is a distribution shift [87], which is common in many practical applications. Distribution shifts can occur for a variety of reasons, including sim-to-real transfer [130]. To help users understand how an agent will behave under distribution shifts, Frost et al. [53] propose to provide users with behavior across a wider trajectory distribution. Instead of showing trajectories along optimal paths, they propose to start in off-distribution states, which are those obtained by initially following the policy and then following an alternate one. These trajectories are then shown to people as counterfactual trajectories.

These metrics for importance tend to highlight *extreme* policy behavior: states in which the agent risks performing very poorly compared to its best action. This property is useful for high-stakes, real-world scenarios. However, these techniques are not helpful for understanding how these behaviors fit together. This problem is exacerbated when scaling the environment complexity, as the trajectories may be so widely temporally spaced that the interpretation is not useful.

### 7.2 Convert RNN to Interpretable Format

In RL, policies are commonly represented with RNNs [128]. RNNs are a class of neural networks that permit cycles, which is amenable for modeling sequential or time series data. This property means that their internal memory encodes features of the observation history critical for decision making [132, 159], making them a good choice to model policies for RL. Although the RNN itself is not interpretable by design, converting the RNN to an interpretable representation enables us

to repair the existing system. Because of the existing relationship between finite state representations and RNNs [30, 116], finite state representations are a popular choice for this interpretable representation.

One work [89] converts RNN policies to a finite state representation called a *Moore machine* [88], that is nearly equivalent to the original RNN. This process involves an additional quantization network [162], called a *quantized bottleneck network*, that encodes the memory states and observation vectors encountered during the RNN operation. These networks are autoencoders [72] with a quantized latent representation with the goal of discretizing a continuous space. After training two quantized bottleneck networks, one for the observed features and one for the states, these networks are inserted into the original RNN. The observed-feature quantized bottleneck network is inserted between the units of the RNN that compute the features and the nodes to which those units are connected, and the state quantized bottleneck network is inserted between the output and input of the recurrent network block. To more closely match the performance of the previous RNN, the authors propose to fine-tune the network on the original rollout data of the RNN.

This procedure results in a Moore machine network, from which a classic Moore machine can be extracted and minimized. However, this minimization process can merge semantically distinct states, obfuscating a deeper understanding of the finite state machine. As a result, Danesh et al. [41] propose a more interpretable reduction technique to preserve key decision points of the policy.

Note that this family of techniques only applies in cases where the agent contains an RNN, which makes them less portable than techniques that make no assumptions on the architecture of the underlying agent. To mitigate this issue, Hasanbeig et al. [66] construct a deterministic finite automaton to capture these sequential dependencies in an interpretable representation without reliance on an RNN policy.

Completely removing the RNN from the agent permits a more interpretable structure. However, the agent is changed through the minimization process, yet it is treated as equivalent to the original agent. In other words, the agent acts *like* it is following the resulting finite state machine, but it is not internally behaving this way. This misalignment is potentially problematic: it may influence users to believe that the agent will behave one way when it will not. As a result, assessment of these techniques should include measures of how well these explanations support users in correctly understanding the model.

### 7.3 Extract Clusters or Abstract States

Sometimes it is invaluable to understand the overall behavior of the agent in terms of how it will act when it encounters similar states. One advantage of this approach is that people only need to understand the similarity metric for aggregating states and the model for summarizing the policy. XRL methods that produce summarizations using these clustered or abstract states are similar to those for generating *abstractions* for RL. We refer the interested reader to recent work [3] on abstraction-generation methods.

TLdR [137] is an example of such a summarization technique. It summarizes policies for stochastic shortest path problems by identifying landmarks, which are defined as propositional formulas that must be satisfied for an agent to complete a goal. TLdR identifies landmarks and their relative ordering to assemble as a graph to present to a user. The technique assumes access to a model $\mathcal{M}$ with a state space that can be specified by a set of propositional fluents $F$, where $|S| = 2^{|F|}$, and a set of goal states $G$ that can be described by a subset of propositions $\mathcal{G}$. These landmarks, called *policy landmarks*, are defined for a given model, fluent set, and policy: a policy landmark is defined from $s_0$ to the goal set if and only if the formulas in the tuple and their corresponding ordering can be satisfied by every execution trace from the original state to the goal set. Because the landmarks

can be viewed as a form of temporal abstraction, the possible transitions between landmarks are displayed as a graph depicting how an agent would reach a goal state from the starting state.

Abstract policy graphs [146] are Markov chains of abstract states that summarize a policy in terms of expected future transitions. CAPS [105] proposes to first cluster states using decision trees, then make an abstract policy graph using these clustered states. Each node of the abstract states is described with natural language. Zahavy et al. [167] do not directly construct graphs; instead, states are embedded into a space, where the agent treats states located close to one another in this space similarly. A human operator can identify clusters and relationships between these clusters through manual inspection.

These techniques rely on different assumptions, such as the binarization of features [146]. Future work could investigate relaxing assumptions to produce more general techniques. Furthermore, we may desire different semantic graph structures, such as hierarchical graphs, where the high-level graph describes the subtask transitions and the lower-level graphs describe the lower-level control policies.

## 8 PRACTICAL APPLICABILITY OF THIS TAXONOMY

To organize the XRL literature, we have proposed a taxonomy that helps elucidate the nuances of the RL decision-making process, allowing practitioners to apply this understanding in real-world scenarios. This taxonomy, divided into three high-level categories, provides a robust framework to comprehensively assess the RL problem and comprehend the agent's actions and choices. We give examples of how this taxonomy can be used to choose an explanation category for XRL use cases.

In a safety-critical system, it is vital to comprehend the agent's prioritization during decision making. For example, in assessing the use of RL to control self-driving cars [86], practitioners, insurance agents, and other stakeholders may wish to understand what the agent was prioritizing when a crash happens. This goal corresponds to identifying the impact of features on the agent's decision making, so FI explanations are needed. Practitioners faced with problems of this type are guided to the first category of our taxonomy (Section 5). This category of explanations is suitable not only for investigating negative outcomes but also for understanding whether agents are employing the appropriate environmental factors to make choices. For example, these explanations would also be appropriate to determine whether the agent is factoring in the proximity of pedestrians or speed-limit signs when choosing its speed. Therefore, any questions about the relationship between state features and action selection suggest using explanations from this category.

When applying RL in a setting where the agent adapts to users, practitioners may want to know how the agent changes due to user feedback and preferences. An example of this type of application is using RL in recommender systems [5], where the goal is to help companies serve suggestions to consumers in a more user-centered way. In this setting, practitioners seek explanations about the effects of different reward signals, domain models, and interactions with users. Unlike the previous example, the representation or content of individual states is not key to answering practitioners' questions about the agent. Instead, explanations from the second category of our taxonomy (Section 6) are suitable. LPM explanations could enable practitioners to improve the system's ability to learn from user interactions and provide more personalized recommendations over time. It could also help identify cases where the agent is trying to negatively manipulate preferences to make them easier to satisfy [29].

When RL is applied to real-world domains, training directly in that domain may be prohibitively expensive or potentially unsafe. As a result, the agent is trained on a simulated domain, which is designed to be similar to the real-world domain but may differ in subtle ways. Given that such RL systems are trained in simulation, the agent may exploit unrealistic domain dynamics. In

this case, it is critical that the human operator agrees with the overall approach of the agent, not just the immediate next step, which requires understanding the entire policy. Industrial scenarios, like datacenter cooling [95], may necessitate explanations of this type. Given the desire to identify and understand behaviors that span multiple actions, PL explanations are best suited for this use case. These explanations correspond to the third category of our taxonomy (Section 7). Explanations from the other two categories are insufficient for this use case: FI explanations focus on single-timestep behaviors, and LPM explanations focus on why an agent has learned a given policy rather than what the policy does.

In summary, our proposed taxonomy offers a comprehensive and structured approach to understanding and interpreting the decision-making process of RL agents. By employing this taxonomy, RL practitioners can start with a question about an RL system and narrow their set of considered methods to those that can answer this question.

## 9 A ROADMAP FOR XRL

We have explained, summarized, and contextualized the existing work in XRL. We now turn our attention to the future. Here, we synthesize the gaps in the XRL literature to discuss opportunities for future research. We focus on the following directions: investigating the properties of explanations for RL, standardizing evaluation through benchmarks, and focusing on techniques that leverage the unique facets of the RL problem to provide correct and coherent explanations.

### 9.1 Assessing Key Properties of XRL Techniques Using Appropriate Metrics

We believe it is important to investigate properties of explanations for RL and to standardize these evaluations through operationalized metrics. Some metrics, such as preferability and actionability, necessitate user studies to properly evaluate; however, only 16 of the 73 surveyed papers include user studies. Furthermore, as identified in Table 2, papers most commonly evaluate techniques using visualizations and task performance measures, which means that claims of interpretability are (typically) not well founded. These metrics must reflect the goals of the task, so we recommend that researchers carefully consider how to operationalize these metrics before evaluating their methods.

Unfortunately, some studies give an incomplete picture of the utility of the proposed explanations. For example, it is not enough to evaluate whether people *believe* explanations are useful. Focusing only on perceived utility risks selecting for explanations that match users' domain knowledge or preconceived beliefs about agent reasoning. To avoid this pitfall, we must also evaluate whether explanation methods *are* useful and accurate in explaining the agent's behavior. Likewise, we caution against substituting explanations with ad-hoc rationalizations. Instead of using a subjective post-hoc interpretation of a behavior and corresponding explanation, explanations ought to reflect the agent's underlying decision-making process [14]. In other words, rather than trying to increase user trust by obfuscating an agent's true decision-making process, it is better to instead faithfully represent the decision-making process. In doing so, we can ensure that any increased trust the user may have in the system as a result of these explanations is well founded.

We now discuss what we believe are good practices for choosing metrics for assessing explanations. It is challenging to provide specific recommendations for every problem because every setting consists of context- and user-specific challenges. However, we believe that there is a difference between establishing a type of explanation as generally interpretable and evaluating an existing interpretable explanation.

*9.1.1 Assessing Existing Explanations.* We first discuss assessing explanations that are more well established as being understandable. For example, consider decision trees. Decision trees are

generally considered to be an interpretable model family, as long as they are not too large. If the goal is to replicate existing findings, then measuring decision trees using established metrics from previous work is reasonable. If the goal is to produce new insights to a problem and one can only choose from a subset of metrics for evaluating the explanation, then we recommend choosing a different set of evaluation metrics.

We now discuss a few examples using decision tree policies as our exemplar explanation. To produce new insights, one can examine more domain-specific assessments, such as whether decision tree structured policies produce more actionable explanations on the target domain than those of a different form. Furthermore, one can explicitly examine the tradeoffs between cognitive load and actionability for the domain of interest. We can also move toward understanding how particular aspects of the human participants may influence their ability to utilize the type of explanation for a problem domain. In these studies, we suggest accounting for influential characteristics of the users, such as domain expertise, to evaluate the *relevance* of the explanations [50].

*9.1.2 Assessing New Types of Explanations.* When proposing a new form of explanation, we recommend performing a user study to evaluate if the explanation is practical and understandable. Measuring comprehensibility tells us whether the explanation is understandable at all, whereas measuring the associated cognitive load provides insight into how challenging the explanation is to interpret. In general, new proposed explanations ought to be evaluated with metrics, such as those discussed in Section 3, that demonstrate the utility of the new explanation.

### 9.2 Comparing Techniques through Benchmarks

There is no singular benchmark for evaluating explanations for RL. The standard domains used in the literature vary widely, spanning toy control [171], games [115], real healthcare data [22], and more. The usage of such disparate domains makes comparing evaluations challenging, especially when high-quality explanations are often context-dependent [6]. To make progress toward concrete and context-dependent tasks within XRL, we believe it is critical to evaluate the quality of explanations using benchmarks with standardized environments and metrics. Although traditional machine learning benchmarks suffer from important issues, such as construct validity with respect to the problem being addressed [122], we believe that benchmarks, and competitions based off of these benchmarks [34, 63], can propel research forward if appropriate scope and focus is provided to an important, carefully selected problem. To that end, we propose two types of benchmarks for XRL research: *type-based* and *objective-based*.

*9.2.1 Type-Based.* Type-based benchmarks aim to produce high-quality explanations of a *certain* type, such as decision tree policies, which are evaluated in a standardized set of environments with common metrics. The goal of this type of benchmark is to understand the tradeoffs between explanation generation methods while controlling external factors such that we can attribute any differences to the methods or explanations. If possible, these benchmarks should vary the complexity of the problem. For example, when comparing decision trees, it should be possible to represent $\pi^*$ with a tree in (at least) the simple environments (e.g., Cartpole) to ensure that the techniques can reliably learn high-performing decision tree policies. These benchmarks will permit evaluation of explanation-specific properties, such as decision tree depth, in addition to more general metrics, like performance and fidelity.

Existing benchmarks of this type are designed for the supervised learning [73] or time series prediction [78] settings. These benchmarks are a great start for evaluating different types of explanations for RL. For example, we can cast the supervised learning problem as a one-step RL problem with a classification-based reward. However, we still require tasks that involve making decisions over time, so MDP-based tasks are still necessary for evaluating explanations in the context of RL.

*9.2.2 Objective-Based.* The purpose of an objective-based benchmark is to produce high-quality explanations of *any* type for a particular task. The goal of this type of benchmark is to understand the *best*, or most useful, explanation(s) for a specific, important task. Indeed, recent work [6] indicates the importance of grounding explanations with their real-world application. This type of benchmark requires clearly defining the goals of the explanations and metrics to ascertain whether these goals have been achieved. These goals may be more general, such as answering causal questions, or more specific, like providing summaries of suggested treatment plans by an assistive medical AI. Metrics may include the predictability of a system by users with and without providing explanations or perceived user comprehensibility of provided explanations. At present, we are not aware of any such benchmarks of this type.

*9.2.3 Considerations for Type-Based and Objective-Based Benchmarks.* Existing RL benchmarks [34, 94, 97, 165] may be repurposed for both benchmarks, as long as they are reworked to include an interpretability focus. At the same time, most existing benchmarks for RL focus on two main areas: simulated robotics and games. Given that a goal of making RL more interpretable is to expand the possible real-world applications of machine learning, we encourage researchers to exercise creativity in imagining the tasks for these benchmarks. We also encourage researchers to consider broader impacts and whether research for that task *ought* to be done before proposing benchmarks in that field, as labor hours and financial contributions help legitimize potential harmful applications.

In both types of benchmarks, we require human-in-the-loop evaluations to properly compare techniques against one another. To our knowledge, only one benchmark provides a principled means of doing such an evaluation [131]; however, it is done in the context of learning from hard-to-define reward functions, not interpretability. As a result, another critical research direction is developing a principled approach for human-in-the-loop evaluations [47] of XRL techniques. Given the interest in deploying RL on real-world problems, we encourage benchmark designers to consider recruiting more specialized workers when evaluating explanation quality (e.g., healthcare workers for healthcare tasks). If the cost to deploy these user studies is prohibitive, another interesting avenue of research is simulating parts of user studies [33], such as filtering out explanation types to present to users before conducting the study.

These benchmarks are complementary: the results of both are interesting and useful to the other. In tackling the objective-based benchmarks, researchers may introduce novel techniques that can be added to the set to evaluate in type-based benchmarks to better understand the tradeoffs. Because type-based benchmarks supply a means of evaluating the tradeoffs between explanation-generation methods of a certain type, their results can be used to help select the techniques used in the objective-based benchmarks. After sufficient evaluation, the objective-based benchmark provides either a means of repairing an existing, non-interpretable system or making a system interpretable by design.

### 9.3 Investigating RL-Specific Explanations

As shown in Table 3, a large portion of XRL techniques stem from the supervised learning literature. In particular, FI techniques mirror the standard interpretable classification setup by providing local explanations of single actions $a_t$ (the label) in the context of the immediate state $s_t$ (the input data to be classified). Although these explanations are useful for elucidating the immediate context of local decisions, we can leverage important components of the RL problem to produce more actionable and informative explanations.

We believe a fruitful direction of research is one that leverages these critical components. For example, an agent's objectives can be conveyed in terms of reward (Section 6.2). Likewise, to communicate the effects of an agent's actions, a causal model can be learned from the agent's

interactions with the environment (Section 6.1). Crucially, supervised learning style explanations convey short-term behaviors, so they can be augmented with explanations that depict how an agent will act over time (Section 7).

Furthermore, we encourage researchers to provide concrete descriptions of what their methods enable. These descriptions can consist of the part of the RL process that the method explains, what questions the technique addresses, and more. We desire transparency around the affordances of explanations to prevent user misunderstanding. For example, it can be misleading to provide explanations phrased as causal relationships for techniques that only explain correlations. These descriptions also enable practitioners to understand whether the techniques can help repair an existing system to be interpretable in the way that is best suited for their needs. The same is true for enabling practitioners to select systems for their use case that are interpretable by design.

## 10 CONCLUSION

We presented an extensive summary of XRL techniques in the context of our novel taxonomy. We compared representative approaches of each type and discussed their applicability and critiques. In doing so, we identified which techniques enable the construction of an interpretable system by design and which enable repair of an existing non-interpretable system. We identified gaps in the current literature and concluded by suggesting three major avenues of research to drive the field forward.

## ACKNOWLEDGMENTS

We thank Zhicheng Zhang for his comments and feedback.

## REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*.

[2] Ashraf Abdul, Christian von der Weth, Mohan Kankanhalli, and Brian Y. Lim. 2020. COGAM: Measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[3] David Abel. 2022. A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397* (2022).

[4] Dickson Adom, Jephtar Adu Mensah, and Dennis Atsu Dake. 2020. Test, measurement, and evaluation: Understanding and use of the concepts in education. *International Journal of Evaluation and Research in Education* 9, 1 (2020), 109–119.

[5] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys* 55, 7 (2022), 1–38.

[6] Kasun Amarasinghe, Kit T. Rodolfa, Sérgio Jesus, Valerie Chen, Vladimir Balayan, Pedro Saleiro, Pedro Bizarro, Ameet Talwalkar, and Rayid Ghani. 2022. On the importance of application-grounded experimental design for evaluating explainable ML methods. *arXiv preprint arXiv:2206.13503* (2022).

[7] Xavier Amatriain, Josep M. Pujol, and Nuria Oliver. 2009. I like it. . . I like it not: Evaluating user ratings noise in recommender systems. In *Proceedings of the International Conference on User Modeling, Adaptation, and Personalization*. 247–258.

[8] Dan Amir and Ofra Amir. 2018. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. 1168–1176.

[9] Ofra Amir, Finale Doshi-Velez, and David Sarne. 2018. Agent strategy summarization. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. 1203–1207.

[10] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. 2019. Explaining reinforcement learning to mere mortals: An empirical study. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

[11] Raghuram Mandyam Annasamy and Katia Sycara. 2019. Towards better interpretability in deep Q-networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*, Vol. 33. 4561–4569.

[12] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57, 5 (2009), 469–483.

[13] Christopher G. Atkeson and Juan Carlos Santamaria. 1997. A comparison of direct and model-based reinforcement learning. In *Proceedings of the International Conference on Robotics and Automation*, Vol. 4. IEEE, Los Alamitos, CA, 3557–3564.
[14] Akanksha Atrey, Kaleigh Clary, and David Jensen. 2020. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep RL. In *Proceedings of the 8th International Conference on Learning Representations*.
[15] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 2 (2018), 423–443.
[16] Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13, 1 (2003), 41–77.
[17] Osbert Bastani, Jeevana Priya Inala, and Armando Solar-Lezama. 2022. Interpretable, verifiable, and robust reinforcement learning via program synthesis. In *Proceedings of the International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. 207–228.
[18] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable reinforcement learning via policy extraction. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS '18)*. 2494–2504.
[19] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).
[20] Tom Bewley and Jonathan Lawry. 2020. TripleTree: A versatile interpretable representation of black box agents and their environments. *CoRR abs/2009.04743* (2020).
[21] Benjamin Beyret, Ali Shafti, and A. Aldo Faisal. 2019. Dot-to-Dot: Explainable hierarchical reinforcement learning for robotic manipulation. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '19)*.
[22] Ioana Bica, Daniel Jarrett, Alihan Hüyük, and Mihaela van der Schaar. 2021. Learning "what-if" explanations for sequential decision-making. In *Proceedings of the 9th International Conference on Learning Representations*.
[23] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).
[24] Henrik Brink, Joseph Richards, and Mark Fetherolf. 2016. *Real-World Machine Learning*. Simon & Schuster.
[25] David A. Broniatowski. 2021. *Psychological Foundations of Explainability and Interpretability in Artificial Intelligence*. Technical Report NISTIR 8367. NIST.
[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christophe Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.
[27] Nadia Burkart and Marco F. Huber. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70 (2021), 245–317.
[28] Donald T. Campbell and Thomas D. Cook. 1979. *Quasi-experimentation*. Rand McNally, Chicago, IL.
[29] Micah D. Carroll, Anca Dragan, Stuart Russell, and Dylan Hadfield-Menell. 2022. Estimating and penalizing induced preference shifts in recommender systems. In *Proceedings of the International Conference on Machine Learning*. 2686–2708.
[30] Mike Casey. 1996. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation* 8, 6 (1996), 1135–1178.
[31] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K. Su. 2019. This looks like that: Deep learning for interpretable image recognition. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS '19)*. 8930–8941.
[32] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. 2022. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 6 (2022), 5068–5078.
[33] Valerie Chen, Nari Johnson, Nicholay Topin, Gregory Plumb, and Ameet Talwalkar. 2022. Use-case-grounded simulations for explanation evaluation. *arXiv preprint arXiv:2206.02256* (2022).
[34] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. 2020. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. 2048–2056.
[35] Francisco Cruz, Richard Dazeley, and Peter Vamplew. 2019. Memory-based explainable reinforcement learning. In *AI 2019: Advances in Artificial Intelligence*. Lecture Notes in Computer Science, Vol. 11919. Springer, 66–77.

[36] Francisco Cruz, Richard Dazeley, and Peter Vamplew. 2020. Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario. *arXiv e-prints arXiv:2006.13615* (2020).
[37] Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable AI for policy-making in pandemics. *arXiv preprint arXiv:2204.04256* (2022).
[38] Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable pipelines with evolutionary optimized modules for reinforcement learning tasks with visual inputs. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 224–227.
[39] Yuxin Dai, Qimei Chen, Jun Zhang, Xiaohui Wang, Yilin Chen, Tianlu Gao, Peidong Xu, Siyuan Chen, Siyang Liao, Huaiguang Jiang, and David Wen-Zhong Gao. 2022. Enhanced oblique decision tree enabled policy extraction for deep reinforcement learning in power system emergency control. *Electric Power Systems Research* 209 (2022), 107932.
[40] Yinglong Dai, Haibin Ouyang, Hong Zheng, Han Long, and Xiaojun Duan. 2022. Interpreting a deep reinforcement learning model with conceptual embedding and performance analysis. *Applied Intelligence* 53, 6 (2022), 6936–6952.
[41] Mohamad H. Danesh, Anurag Koul, Alan Fern, and Saeed Khorram. 2021. Re-understanding finite-state representations of recurrent policy networks. In *Proceedings of the International Conference on Machine Learning*. 2388–2397.
[42] Giang Dao, Indrajeet Mishra, and Minwoo Lee. 2018. Deep reinforcement learning monitor for snapshot recording. In *Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA '18)*. IEEE, Los Alamitos, CA, 591–598.
[43] Krista E. DeLeeuw and Richard E. Mayer. 2008. A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load. *Journal of Educational Psychology* 100, 1 (2008), 223.
[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[45] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. 2022. Toward interpretable-AI policies using evolutionary nonlinear decision trees for discrete-action systems. *IEEE Transactions on Cybernetics*. Early access, June 23, 2022.
[46] Marius-Constantin Dinu, Markus Hofmarcher, Vihang P. Patil, Matthias Dorfer, Patrick M. Blies, Johannes Brandstetter, Jose A. Arjona-Medina, and Sepp Hochreiter. 2022. XAI and strategy extraction via reward redistribution. In *Proceedings of the International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. 177–205.
[47] Jonathan Dodge, Andrew Anderson, Roli Khanna, Jed Irvine, Rupika Dikkala, Kin-Ho Lam, Delyar Tabatabai, Anita Ruangrotsakun, Zeyad Shureih, Minsuk Kahng, Alan Fern, and Margaret Burnett. 2021. From "no clear winner" to an effective Explainable Artificial Intelligence process: An empirical journey. *Applied AI Letters* 2, 4 (2021), e36.
[48] John J. Donovan and David J. Radosevich. 1999. A meta-analytic review of the distribution of practice effect: Now you see it, now you don't. *Journal of Applied Psychology* 84, 5 (1999), 795.
[49] Upol Ehsan, Brent Harrison, Larry Chan, and Mark Riedl. 2018. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 1st AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society.*
[50] Upol Ehsan and Mark O. Riedl. 2020. Human-centered explainable AI: Towards a reflective sociotechnical approach. *arXiv preprint arXiv:2002.01092* (2020).
[51] Richard E. Fikes and Nils J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2, 3-4 (1971), 189–208.
[52] Nicholas Frosst and Geoffrey Hinton. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784* (2017).
[53] Julius Frost, Olivia Watkins, Eric Weiner, Pieter Abbeel, Trevor Darrell, Bryan Plummer, and Kate Saenko. 2022. Explaining reinforcement learning policies through counterfactual trajectories. *arXiv preprint arXiv:2201.12462* (2022).
[54] Oliver Genschow, Sophie Klomfar, Ine d'Haene, and Marcel Brass. 2018. Mimicking and anticipating others' actions is linked to social information processing. *PLoS One* 13, 3 (2018), e0193743.
[55] Vikash Goel, Jameson Weng, and Pascal Poupart. 2018. Unsupervised video object segmentation for deep reinforcement learning. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS '18)*. 5683–5694.
[56] Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Anthony Celi, Emma Brunskill, and Finale Doshi-Velez. 2020. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. *arXiv preprint arXiv:2002.03478* (2020).
[57] Aiden P. Gregg, Beate Seibt, and Mahzarin R. Banaji. 2006. Easier done than undone: Asymmetry in the malleability of implicit preferences. *Journal of Personality and Social Psychology* 90, 1 (2006), 1.
[58] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. 2018. Visualizing and understanding Atari agents. In *Proceedings of the 35th International Conference on Machine Learning*. 1792–1801.
[59] David Gunning and David W. Aha. 2019. DARPA's explainable artificial intelligence program. *AI Magazine* 40, 2 (2019), 44–58.

[60] Wei Guo and Peng Wei. 2022. Explainable deep reinforcement learning for aircraft separation assurance. In *Proceedings of the 4th Digital Avionics Systems Conference*.
[61] Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. 2021. EDGE: Explaining deep reinforcement learning policies. *Advances in Neural Information Processing Systems* 34 (2021), 1–15.
[62] Yunfei Guo, Peiduo Liu, and Xiting Huang. 2019. The practice effect on time-based prospective memory: The influences of ongoing task difficulty and delay. *Frontiers in Psychology* 10 (2019), 2002.
[63] William H. Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noburu Kuno, Stephanie Milani, Sharada Prasanna Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, Manuela Veloso, and Phillip Wang. 2019. The MineRL competition on sample efficient reinforcement learning using human priors. *arXiv:1904.10079* (2019).
[64] Joseph Y. Halpern and Judea Pearl. 2005. Causes and explanations: A structural-model approach. Part II: Explanations. *British Journal for the Philosophy of Science* 56 (2005), 889–911.
[65] Keegan Harris, Valerie Chen, Joon Sik Kim, Ameet Talwalkar, Hoda Heidari, and Zhiwei Steven Wu. 2021. Bayesian persuasion for algorithmic recourse. *arXiv preprint arXiv:2112.06283* (2021).
[66] Mohammadhosein Hasanbeig, Natasha Yogananda Jeppu, Alessandro Abate, Tom Melham, and Daniel Kroening. 2021. DeepSynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, Vol. 2. 36.
[67] Bradley Hayes and Julie A. Shah. 2017. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI '17)*. IEEE, Los Alamitos, CA, 303–312.
[68] Daniel Hein, Alexander Hentschel, Thomas Runkler, and Steffen Udluft. 2017. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence* 65 (2017), 87–98.
[69] Daniel Hein, Steffen Udluft, and Thomas A. Runkler. 2019. Interpretable policies for reinforcement learning by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*.
[70] Johannes Heinrich and David Silver. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121* (2016).
[71] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. 2020. Explainability in deep reinforcement learning. *arXiv:cs.AI/2008.06693* (2020).
[72] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[73] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A benchmark for interpretability methods in deep neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS '19)*. 9737–9748.
[74] Sandy H. Huang, Kush Bhatia, Pieter Abbeel, and Anca D. Dragan. 2018. Establishing appropriate trust via critical states. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '18)*. IEEE, Los Alamitos, CA, 3929–3936.
[75] Tobias Huber, Dominik Schiller, and Elisabeth André. 2019. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI '19)*. 188–202.
[76] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys* 50, 2 (2017), 1–35.
[77] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. 2021. How to train your robot with deep reinforcement learning: Lessons we have learned. *International Journal of Robotics Research* 40, 4-5 (2021), 698–721.
[78] Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. 2020. Benchmarking deep learning interpretability in time series predictions. *Advances in Neural Information Processing Systems* 33 (2020), 6441–6452.
[79] Hidenori Itaya, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi, and Komei Sugiura. 2021. Visual explanation using attention mechanism in actor-critic-based deep reinforcement learning. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN '21)*. IEEE, Los Alamitos, CA, 1–10.
[80] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. 2018. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 1st AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*.
[81] Alexis Jacq, Johan Ferret, Olivier Pietquin, and Matthieu Geist. 2022. Lazy-MDPs: Towards interpretable reinforcement learning by learning when to act. *arXiv preprint arXiv:2203.08542* (2022).

[82] Erik Jenner and Adam Gleave. 2022. Preprocessing reward functions for interpretability. *arXiv preprint arXiv:2203.13553* (2022).
[83] Aman Jhunjhunwala. 2019. *Policy Extraction via Online Q-Value Distillation*. Masters Thesis, University of Waterloo.
[84] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. 2019. Explainable reinforcement learning via reward decomposition. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence Workshop on Explainable Artificial Intelligence*.
[85] Woo Kyung Kim, Youngseok Lee, and Honguk Woo. 2022. Mean-variance based risk-sensitive reinforcement learning with interpretable attention. In *Proceedings of the 2022 5th International Conference on Machine Vision and Applications (ICMVA '22)*. 104–109.
[86] B. Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 6 (2021), 4909–4926.
[87] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the International Conference on Machine Learning*. 5637–5664.
[88] Zvi Kohavi and Niraj K. Jha. 2009. *Switching and Finite Automata Theory*. Cambridge University Press.
[89] Anurag Koul, Sam Greydanus, and Alan Fern. 2018. Learning finite state representations of recurrent policy networks. *arXiv preprint arXiv:1811.12530* (2018).
[90] James A. Kulik, Chen-Lin C. Kulik, and Robert L. Bangert. 1984. Effects of practice on aptitude and achievement test scores. *American Educational Research Journal* 21, 2 (1984), 435–447.
[91] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS '16)*. 1–9.
[92] Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. 2019. Exploring computational user models for agent policy summarization. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.
[93] Mikel Landajuela, Brenden K. Petersen, Sookyung Kim, Claudio P. Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F. Pettit, and Daniel Faissol. 2021. Discovering symbolic policies with deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. 5979–5989.
[94] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. 2021. URLB: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191* (2021).
[95] Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, M. K. Ryu, and Greg Imwalle. 2018. Data center cooling using model-predictive control. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS '18)*. 1–10.
[96] Hoang Le, Cameron Voloshin, and Yisong Yue. 2019. Batch policy learning under constraints. In *Proceedings of the International Conference on Machine Learning*. 3703–3712.
[97] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. 2020. SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:2011.07215* (2020).
[98] Yixin Lin, Austin S. Wang, Eric Undersander, and Akshara Rai. 2022. Efficient and interpretable robot manipulation with graph neural networks. *IEEE Robotics and Automation Letters* 7, 2 (2022), 2740–2747.
[99] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748* (2017).
[100] Zhengxian Lin, Kim-Ho Lam, and Alan Fern. 2020. Contrastive explanations for reinforcement learning via embedded self predictions. *arXiv preprint arXiv:2010.05180* (2020).
[101] Zachary C. Lipton. 2018. The mythos of model interpretability. *ACM Queue* 16, 3 (2018), 31–57.
[102] Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. 2018. Toward interpretable deep reinforcement learning with linear model U-trees. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 414–429.
[103] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*.
[104] Francis Maes, Raphael Fonteneau, Louis Wehenkel, and Damien Ernst. 2012. Policy search in a space of simple closed-form formulas: Towards interpretability of reinforcement learning. In *Proceedings of the 15th International Conference on Discovery Science*. 37–51.
[105] Joe McCalmon, Thai Le, Sarra Alqahtani, and Dongwon Lee. 2022. CAPS: Comprehensible abstract policy summaries for explaining reinforcement learning agents. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 889–897.

[106] Andreas Messalas, Yiannis Kanellopoulos, and Christos Makris. 2019. Model-agnostic interpretability with Shapley values. In *Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems, and Applications (IISA '19)*. IEEE, Los Alamitos, CA, 1–7.

[107] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles Kamhoua, Evangelos E. Papalexakis, and Fei Fang. 2022. MAVIPER: Learning decision tree policies for interpretable multi-agent reinforcement learning. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[108] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38.

[109] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[110] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. 2020. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712* (2020).

[111] Christoph Molnar. 2019. Interpretable Machine Learning. Retrieved September 2, 2023 from https://christophm.github.io/interpretable-ml-book/

[112] Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. 2019. Towards interpretable reinforcement learning using attention augmented agents. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS '19)*. 12329–12338.

[113] Anton J. Nederhof. 1985. Methods of coping with social desirability bias: A review. *European Journal of Social Psychology* 15, 3 (1985), 263–280.

[114] Thanh Thi Nguyen and Vijay Janapa Reddi. 2023. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems* 34, 8 (2023), 3779–3795.

[115] Matthew L. Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. 2021. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence* 295 (2021), 103455.

[116] Christian W. Omlin and C. Lee Giles. 1996. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM* 43, 6 (1996), 937–972.

[117] Rohan Paleja, Yaru Niu, Andrew Silva, Chace Ritchie, Sugju Choi, and Matthew Gombolay. 2022. Learning interpretable, high-performing policies for autonomous driving. *arXiv:2202.02352* (2022).

[118] Vihang P. Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M. Blies, Johannes Brandstetter, Jose A. Arjona-Medina, and Sepp Hochreiter. 2020. Align-RUDDER: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108* (2020).

[119] Michele Persiani and Thomas Hellström. 2022. The mirror agent model: A Bayesian architecture for interpretable agent behavior. In *Proceedings of the 4th International Workshop on EXplainable and TRAnsparent AI and Multi-Agent Systems (EXTRAAMAS '22)*.

[120] Alina Pommeranz, Joost Broekens, Pascal Wiggers, Willem-Paul Brinkman, and Catholijn M. Jonker. 2012. Designing interfaces for explicit preference elicitation: A user-centered investigation of preference representation and elicitation process. *User Modeling and User-Adapted Interaction* 22, 4 (2012), 357–397.

[121] Erika Puiutta and Eric M. S. P. Veith. 2020. Explainable reinforcement learning: A survey. *arXiv preprint arXiv:2005.06247* (2020).

[122] Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. 2021. AI and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366* (2021).

[123] Finn Rietz, Sven Magg, Fredrik Heintz, Todor Stoyanov, Stefan Wermter, and Johannes A. Stork. 2022. Hierarchical goals contextualize local reward decomposition explanations. *Neural Computing and Applications*. Published online, May 12, 2022.

[124] Ivan Dario Jimenez Rodriguez, Taylor W. Killian, Sung-Hyun Son, and Matthew C. Gombolay. 2019. Optimization methods for interpretable differentiable decision trees in reinforcement learning. *arXiv preprint arXiv:1903.09338* (2019).

[125] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 627–635.

[126] Cynthia Rudin. 2014. Algorithms for interpretable machine learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1519–1519.

[127] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* 16 (2022), 1–85.

[128] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning Internal Representations by Error Propagation*. Technical Report. Cognitive Science at UC San Diego, La Jolla, CA.

[129] Christian Rupprecht, Cyril Ibrahim, and Christopher J. Pal. 2020. Finding and visualizing weaknesses of deep reinforcement learning agents. In *Proceedings of the 8th International Conference on Learning Representations*.

[130] Andrei A. Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. 2017. Sim-to-real robot learning from pixels with progressive nets. In *Proceedings of the Conference on Robot Learning*. 262–270.

[131] Rohin Shah, Cody Wild, Steven H. Wang, Neel Alex, Brandon Houghton, William Guss, Sharada Mohanty, Anssi Kanervisto, Stephanie Milani, Nicholay Topin, Pieter Abbeel, Stuart Russell, and Anca Dragan. 2021. The MineRL BASALT competition on learning from human feedback. *arXiv preprint arXiv:2107.01969* (2021).

[132] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-Walk: Learning to walk over graphs using Monte Carlo tree search. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS '18)*. 1–12.

[133] Wenjie Shi, Zhuoyuan Wang, Shiji Song, and Gao Huang. 2020. Self-supervised discovering of causal features: Towards interpretable reinforcement learning. *arXiv preprint arXiv:2003.07069* (2020).

[134] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbreener, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[135] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[136] Zhihao Song, Yunpeng Jiang, Jianyi Zhang, Paul Weng, Dong Li, Wulong Liu, and Jianye Hao. 2022. An interpretable deep reinforcement learning approach to autonomous driving. In *Proceedings of the IJCAI Workshop on Artificial Intelligence for Autonomous Driving*.

[137] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. 2020. TLdR: Policy summarization for factored SSP problems using temporal abstractions. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling*.

[138] Alberto Suárez and James F. Lutsko. 1999. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 12 (1999), 1297–1311.

[139] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[140] John Sweller, Paul Ayres, and Slava Kalyuga. 2011. Measuring cognitive load. In *Cognitive Load Theory*. Springer, 71–85.

[141] Yujin Tang, Duong Nguyen, and David Ha. 2020. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*.

[142] Miguel Tejedor, Ashenafi Zebene Woldaregay, and Fred Godtliebsen. 2020. Reinforcement learning application in diabetes blood glucose control: A systematic review. *Artificial Intelligence in Medicine* 104 (2020), 101836.

[143] Erico Tjoa and Cuntai Guan. 2019. A survey on Explainable Artificial Intelligence (XAI): Towards medical XAI. *arXiv preprint arXiv:1907.07374* (2019).

[144] R. Watson Todd and Punjaporn Pojanapunya. 2009. Implicit attitudes towards native and non-native speaker teachers. *System* 37, 1 (2009), 23–33.

[145] Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. 2021. Iterative bounding MDPs: Learning interpretable policies via non-interpretable methods. *arXiv preprint arXiv:2102.13045* (2021).

[146] Nicholay Topin and Manuela Veloso. 2019. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*.

[147] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 10–19.

[148] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. 2017. Hybrid reward architecture for reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*. 5398–5408.

[149] Varun Ravi Varma. 2021. *Interpretable Reinforcement Learning with the Regression Tsetlin Machine*. Ph.D. Dissertation, University of Gronigen.

[150] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*. 5045–5054.

[151] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[152] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony

Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. Starcraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).

[153] Sergei Volodin. 2021. *CauseOccam: Learning Interpretable Abstract Representations in Reinforcement Learning Environments via Model Sparsity*. Technical Report. Ecole Polytechnique Federale de Lausanne.

[154] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31 (2017), 841.

[155] Stephan Wäldchen, Sebastian Pokutta, and Felix Huber. 2022. Training characteristic functions with reinforcement learning: XAI-methods play Connect Four. In *Proceedings of the International Conference on Machine Learning*. 22457–22474.

[156] Xinzhi Wang, Schengcheng Yuan, Hui Zhang, Michael Lewis, and Katia Sycara. 2019. Verbal explanations for deep reinforcement learning neural networks with attention on extracted features. In *Proceedings of the 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN '19)*.

[157] Laurens Weitkamp, Elise van der Pol, and Zeynep Akata. 2018. Visual rationalizations in deep reinforcement learning for Atari games. In *Proceedings of the 30th Benelux Conference on Artificial Intelligence*. 151–165.

[158] Lindsay Wells and Tomasz Bednarz. 2021. Explainable AI and reinforcement learning—A systematic review of current approaches and trends. *Frontiers in Artificial Intelligence* 4 (2021), 48.

[159] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. 2010. Recurrent policy gradients. *Logic Journal of the IGPL* 18, 5 (2010), 620–634.

[160] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. 2022. DayDreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176* (2022).

[161] Bin-Bin Yang, Song-Qing Shen, and Wei Gao. 2019. Weighted oblique decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5621–5627.

[162] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-Sheng Hua. 2019. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7308–7316.

[163] Herman Yau, Chris Russell, and Simon Hadfield. 2020. What did you think would happen? Explaining agent behaviour through intended outcomes. *Advances in Neural Information Processing Systems* 33 (2020), 18375–18386.

[164] Chong-Ho Yu. 2010. Reliability of Self-Report Data. Retrieved September 2, 2023 from https://www.creative-wisdom.com/teaching/WBI/memory.shtml

[165] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*. 1094–1100.

[166] Lotfi A. Zadeh. 1988. Fuzzy logic. *Computer* 21, 4 (1988), 83–93.

[167] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. 2016. Graying the black box: Understanding DQNs. In *Proceedings of the 33rd International Conference on Machine Learning*.

[168] Amber E. Zelvelder, Marcus Westberg, and Kary Främling. 2021. Assessing explainability in reinforcement learning. In *Explainable and Transparent AI and Multi-Agent Systems*. Lecture Notes in Computer Science, Vol. 12688. Springer, 223–240.

[169] Dongxia Zhang, Xiaoqing Han, and Chunyu Deng. 2018. Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems* 4, 3 (2018), 362–370.

[170] Hengzhe Zhang, Aimin Zhou, and Xin Lin. 2020. Interpretable policy derivation for reinforcement learning based on evolutionary feature synthesis. *Complex & Intelligent Systems* 6 (2020), 1–13.

[171] Li Zhang, Xin Li, Mingzhong Wang, and Andong Tian. 2021. Off-policy differentiable logic reinforcement learning. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 617–632.

[172] Qiyuan Zhang, Xiaoteng Ma, Yiqin Yang, Chenghao Li, Jun Yang, Yu Liu, and Bin Liang. 2021. Learning to discover task-relevant features for interpretable reinforcement learning. *IEEE Robotics and Automation Letters* 6, 4 (2021), 6601–6607.

[173] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. 2019. "Why should you trust my explanation?" Understanding uncertainty in LIME explanations. In *Proceedings of the International Conference on Machine Learning AI for Social Good Workshop*. 1–9.

Received 15 October 2022; revised 22 May 2023; accepted 3 August 2023