A Survey on Interactive Reinforcement Learning: Design Principles and Open Challenges

Christian Arzate Cruz
The University of Tokyo
Tokyo, Japan
arzate.christian@gmail.com

Takeo Igarashi
The University of Tokyo
Tokyo, Japan
takeo@acm.org

ABSTRACT

Interactive reinforcement learning (RL) has been successfully used in various applications in different fields, which has also motivated HCI researchers to contribute in this area. In this paper, we survey interactive RL to empower human-computer interaction (HCI) researchers with the technical background in RL needed to design new interaction techniques and propose new applications. We elucidate the roles played by HCI researchers in interactive RL, identifying ideas and promising research directions. Furthermore, we propose generic design principles that will provide researchers with a guide to effectively implement interactive RL applications.

Author Keywords

Interactive Machine Learning; Interactive Reinforcement Learning; Human-agent Interaction.

CCS Concepts

•**General and reference** → *Surveys and overviews;*

INTRODUCTION

Machine learning (ML) is making rapid advances in many different areas. However, standard ML methods that learn models automatically from training data sets are insufficient for applications that could have a great impact on our everyday life, such as autonomous companion robots or self-driving cars. Generally speaking, to enable their use in more complex and impactful environments, ML algorithms have to overcome further obstacles: how to correctly specify the problem, improve sampling efficiency, and adapt to the particular needs of individual users.

Interactive ML (IML) proposes to incorporate a human-in-the-loop to tackle the aforementioned problems in current ML algorithms [33, 40, 41]. This configuration intends to integrate human knowledge that improves and/or personalizes the behavior of agents. Reinforcement learning (RL) is one ML paradigm that can benefits from an interactive setting [93]. Interactive RL has been successfully applied to a wide range of problems in different research areas [10, 108, 56]. However, interactive RL has not been widely adopted by the human-computer interaction (HCI) community.

In the last few years, the two main roles that HCI researchers have played in IML are (1) designing new interaction techniques [90] and (2) proposing new applications [112, 19]. To enable HCI researchers to play those same roles in interactive RL, they need to have adequate technical literacy. In this paper, we present a survey on interactive RL to empower HCI researchers and designers with the technical background needed to ideate novel applications and paradigms for interactive RL.

To help researchers perform role (1) by describing how different types of feedback can leverage an RL model for different purposes. For instance, the user can personalize the output of an RL algorithm based on the user's preferences [2][60], facilitate learning in complex tasks [51], or guide exploration to promising states in the environment [34].

Finally, we help researchers perform role (2) by proposing generic design principles that will provide a guide to effectively implement interactive RL applications. We pay special attention to common problems in multiple research fields that constrain the performance of interactive RL approaches; solving these problems could make interactive RL applications effective in real-world environments.

Scope of this Paper

The literature on RL that could potentially be applied to interactive RL is extensive. For that reason, we focus on works that have been successfully applied to RL methods that use a human-in-the-loop. We further narrow the reach of this survey by considering only works from three research areas, robotics, HCI, and Game AI, published between 2010 and 2019. Among these papers, we picked those that covered all the main research directions in the three research areas. We chose these three fields because they use interactive RL systems in similar ways, enabling us to propose generic design rules for interactive RL that will be useful for a wider audience.

Some surveys have covered larger topics, such as IML in general [5] or IML for health informatics [40]. Other surveys have focused on particular design dimensions, such as types of user feedback [60] or the evaluation of results [9]. Surveys have also been done on particular subjects within the interactive RL research area, such as safe RL [34], the agent alignment

ISBN XXX-X-XXXX-XXXX-X/20/XX.
DOI: https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

problem [60], and inverse RL [115]. In contrast, our survey focuses only on approaches that can be applied to an interactive RL setting.

These constraints allow us to construct a detailed analysis for HCI researchers. Additionally, our survey methodology is favorable for constructing design principles for interactive RL that are generic enough to be applied in physical or simulated environments.

REINFORCEMENT LEARNING (RL)

The *reinforcement learning* (RL) paradigm is based on the idea of an agent that learns by interacting with its environment [97, 46]. The learning process is achieved by an exchange of signals between the agent and its environment; the agent can perform actions that affect the environment, while the environment informs the agent about the effects of its actions. Additionally, it is assumed that the agent has at least one goal to pursue and – by observing how its interactions affect its dynamic environment – it learns how to behave to achieve its goal.

Arguably, the most common approach to model RL agents is the *Markov decision process* (MDP) formalism. An MDP is an optimization model for an agent acting in a stochastic environment [84]; that is defined by the tuple $\langle S, A, T, R, \gamma \rangle$, where S is a set of states; A is a set of actions; $T: S \times A \times S \rightarrow [0,1]$ is the *transition function* that assigns the probability of reaching state s' when executing action a in state s, that is, $T(s' \mid s,a) = P(s' \mid a,s)$; R: $S \times A \rightarrow \mathbb{R}$ is the *reward function*, with $R(s,a)$ denoting the immediate numeric reward value obtained when the agent performs action a in state s; and $\gamma \in [0,1]$ is the discount factor that defines the preference of the agent to seek immediate or more distant rewards.

The reward function defines the goals of the agent in the environment; while the transition function captures the effect of the agent's actions for each particular state.

INTERACTIVE REINFORCEMENT LEARNING

In a typical RL setting, the RL designer builds a reward function and chooses/designs an RL method for the agent to learn an optimal policy. For a survey of the literature on RL, we refer the reader to [11, 83, 35].

In contrast, an interactive RL approach involves a human-in-the-loop that tailors specific elements of the underlying RL algorithm to improve its performance or produce an appropriate policy for a particular task. One of the first approaches that incorporated a human-in-the-loop in an RL algorithm is the Cobot software agent [43]. This interactive method implements a crowd-sourced environment where multiple users work together to create the reward function for a virtual agent. However, a formal introduction to the term interactive RL was later proposed in [105].

Figure 1 presents a generic interactive RL architecture and shows the interaction channels that allow the user to guide the low-level components of an RL algorithm. The foundation of an interactive RL approach is an RL-based agent that learns by interacting with its environment according to a specific RL algorithm. Next, the user observes and evaluates the resulting

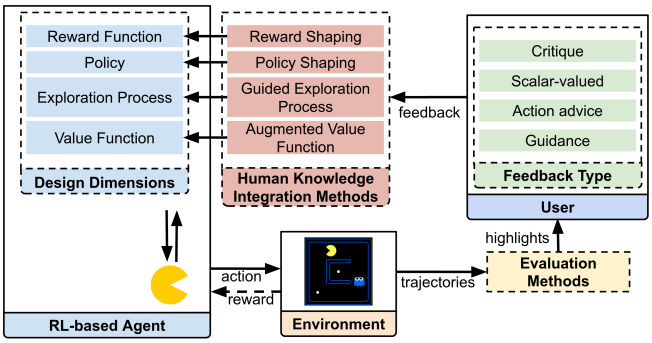


Figure 1. The interactive RL architecture.

agent behavior and gives the agent feedback. Human feedback can be delivered through multiple human knowledge integration methods to a specific element (design dimension) of the underlying RL algorithm.

The interactive RL paradigm has the advantage of integrating prior knowledge about the task the RL-based agent is trying to solve. This concept has proven to be more effective than automatic RL algorithms [33, 53]. Moreover, human feedback includes prior knowledge about the world that can significantly improve learning rates [29]. Arguably, any RL algorithm will more quickly find a high-earning policy if it includes human knowledge. However, it is still unclear how to design efficient interactive RL methods that generalize well and can be adapted to feedback from different user types.

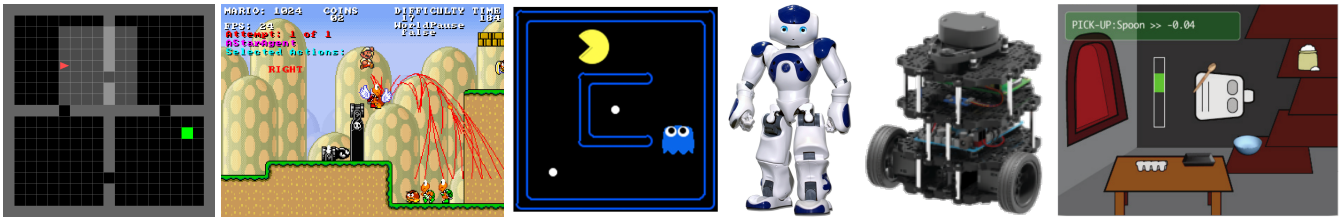
In the rest of this paper, we will detail all the elements of interactive RL that we have introduced so far.

INTERACTIVE RL TESTBEDS

In Figure 2, we present examples of the most common testbeds in interactive RL research. The *GridWorld* platform is a classic environment for testing RL algorithms. The small state-space size of this testbed allows for fast experiments, and its simplicity does not demand specialized knowledge from the users. These characteristics make GridWorld popular among researchers in the human-computer interaction (HCI) field [56, 58, 57]. The *Mario AI benchmark* [48] is a clone of Nintendo's platform game named *Super Mario Bros.* This game is a testbed with a large state-space that offers the conditions needed to test interactive RL algorithms aimed to personalize bot behaviors and create fast evaluation techniques. Similarly, we have the *Pac-Man* game that can reduce the size of its state-space to make it more manageable. The *Nao robot* and the *TurtleBot* are popular platforms in robotics that are perfect for testing natural ways to interact with human users. The *Sophie's kitchen* [103] platform is designed as an online tool for interactive RL that explores the impact of demonstrating uncertainty to the users. Finally, the main focus of the OpenAI Gym platform [17] is designing and comparing RL algorithms.

Generally speaking, these are the classifications of testbeds for each research area included in our survey:

- **Robotics:** GridWorld environments (physical and virtual), OpenAI Gym, soccer simulation, Nao robot, Nexi robot, TurtleBot, shopping assistant simulator.



(a) (b) (c) (d) (e) (f)

Figure 2. Selected testbeds for interactive RL. (a) GridWorld. (b) Mario AI benchmark. (c) Pac-Man. (d) Nao Robot. (e) TurtleBot. (f) Sophie's kitchen.

- **Game AI:** Pac-Man, Mario AI benchmark, Street Fighter game, Atari emulator.
- **HCI:** Pac-Man with small state-space, GridWorld environments.

Applications

We can find applications for interactive RL, such as teaching robots how to dance [74], creating adaptable task-oriented dialogue systems [89], and learning the generation of dialogue [66]. Furthermore, there is the opportunity to adapt current automatic RL applications to an interactive setting, such as procedural content generation for video games [49], generating music [44], and modeling social influences in bots [45].

DESIGN GUIDES FOR INTERACTIVE RL

This section provides HCI researchers with the foundations to design novel interaction techniques for interactive RL.

Design Dimensions

In this section, we analyze how human feedback is used to tailor the low-level modules of diverse RL algorithms; we call this form of interaction between users and RL algorithms *design dimensions*. Furthermore, the summary of selected works in Table 1 presents a concise and precise classification that helps to compare different design dimensions.

Reward Function

All works classified in this design dimension tailor the reward function of the RL-based algorithm using human feedback. The main objectives of this feedback are to speed up learning, customize the behavior of the agent to fit the user's intentions, or teach the agent new skills or behaviors.

Designing a reward function by hand is trivial for simple problems where it is enough to assign rewards as 1 for winning and 0 otherwise. For example, in the simplest setup of a GridWorld environment, the agent receives -1 for each time step and 1 if it reaches the goal state. However, for most problems that an agent can face in a real-world environment, the selection of an appropriate reward function is key — the desired behaviors have to be encoded in the reward function.

The reward design problem is therefore challenging because the RL designer has to define the agent's behaviors as goals that are explicitly represented as rewards. This approach can cause difficulties in complex applications where the designer has to foresee every obstacle the agent could possibly encounter. Furthermore, the reward function has to be designed to handle trade-offs between goals. These reward design challenges make the process an iterative task: RL designers alternate between evaluating the reward function and optimizing it until they find it acceptable. This alternating process is called *reward shaping* [81]. Ng et al. present a formal definition of this concept in [81].

Reward shaping (RS) is a popular method of guiding RL algorithms by human feedback [81, 110]. In what is arguably the most common version of RS, the user adds extra rewards that enhance the environmental reward function [102, 10, 53] as $R' = R + F$, where $F : S \times A \times S \to \mathbb{R}$ is the *shaping reward function* [81]. A hand-engineered reward function has been demonstrated to increase learning rates [81, 102, 10, 85, 28, 31, 30, 55, 107, 73, 37, 63, 7, 32]. In contrast, some approaches use human feedback as the only source of reward for the agent [108, 54, 12].

Using RS is also beneficial in sparse reward environments: the user can provide the agent with useful information in states where there are no reward signals from the environment or in highly stochastic environments [110]. Another advantage of RS is that it gives researchers a tool with which to better specify, in a granular manner, the goals in the current environment. That is, the computational problem is specified via a reward function [95].

On the other hand, we need to consider the *credit assignment problem* [96, 4]. This difficulty emerges from the fact that, when a human provides feedback, it is applied to actions that happened sometime in the past — there is always a delay between the action's occurrence and the human response. Furthermore, the human's feedback might refer to one specific state or an entire sequence of states the bot visited.

Reward hacking is another negative side effect resulting from a deficient reward design [50, 39, 8, 87]. This side effect can cause nonoptimal behaviors that fail to achieve the goals or intentions of the designers. Usually, these kinds of failure behaviors arise from reward functions that do not anticipate all trade-offs between goals. For example, Clark et al. presented an agent that drives a boat in a race [21]. Instead of moving forward to reach the goal, the agent learned that the policy with the highest reward was to hit special targets along the track.

Design Dimension	Testbed	Interaction	Initiative	HKI	Feedback	Algorithms
Reward Function	Robot in maze-like environment [10]	FE	Passive	RS using HF + ER	Critique	DQL [67, 77]
	Navigation simulation [12]	GUI	Passive	Advantage Function	Critique	DAC
	Sophie's Kitchen game [105, 103]	GUI	Passive, Active	RS using HF + EF	Critique	QL [109], HRL
	Bowling game [108]	GUI	Passive	RS + HF	Scalar-valued	DQL
	Shopping assistant, GridWorld [76]	GUI	Active	Active IRD	Queries	Model-based RL
	Mario, GridWorld, Soccer simulation [85]	Coding	Passive	PBRS	Heuristic Function	QL, QSL, QL(λ), QSL(λ)
	Navigation simulation [102]	VC	Passive	RS using HF + ER	AcAd	SARSA [86], SARSA(λ)
	Atari, robotics simulation [20]	GUI	Active	RS using HF	Queries	DRL
Policy	GridWorld, TurtleBot robot [71]	GUI, GC	Passive	PS	AcAd	AC(λ) [15, 91]
	GridWorld [56]	VC	Passive	PS	Critique, AcAd	BQL [26]
	Pac-Man, Frogger [37]	GUI	Passive	PS	Critique	BQL
Exploration Process	Pac-Man, Cart-Pole simulation [114]	GUI	Passive	GEP	AcAd	QL
	Simulated cleaning Robot [24, 23]	VC	Passive	GEP	AcAd	SARSA
	Pac-Man [7]	GUI	Active	GEP	AcAd	SARSA(λ)
	Pac-Man [32]	GUI	Active	Myopic Agent	AcAd	QL, QRL
	Sophie's Kitchen game [103]	GUI	Active	ACTG	Guidence	QL
	Street Fighter game [13]	Not apply	Passive	EB using Safe RL	Demonstration	HRL
	Nao Robot [93]	GUI	Passive	ACTG	Guidence	QL
	Nexi robot [54]	AT + CT	Passive	Myopic Agent	AcAd	SARSA(λ)
Value Function	Mountain Car simulation [52]	GUI	Passive	Weighted VF	Demonstration	SARSA(λ)
	Keepaway simulation [101]	GUI	Passive	Weighted VF	Demonstration	SARSA
	Mario, Cart Pole [18]	Not apply	Passive	Initialization of VF	Demonstration	QL(λ)

Table 1. Selected works for each design dimension. HKI=human knowledge integration, RS=reward shaping, PS=policy shaping, AcAd=action advice, VF=value function, GUI=grafical user interface, FE=facial expression, VC=voice command, GC=game controller, AT=artifact, CT=controller, HF=human feedback, ER=environmental reward, GEP=guided exploration process, PBRS=potential based reward shaping, ACTG=actions containing the target goal, QL=q-learning, λ= with elegibility traces, DQL= deep q-learning, QRL= R-learning, QSL=QS-learning, AC(λ)=actor-critic with elegibility traces, DAC= deep actor-critic, DRL= deep reinforcement learning, BQL=bayesian q-learning, HRL=hierarchical reinforcement learning, SARSA=state-action-reward-state-action, IRD= inverse reward design, Mario=Super Mario Bros.

Policy

In the policy design dimension, we include interactive RL methods that augment the policy of an agent using human knowledge. This process is called *policy shaping* (PS) [36].

Some authors also classify approaches that use a binary critique as feedback [36]. However, when an action is labeled as "bad" in this type of algorithm, it is generally pruned from the action set at the corresponding action-state, which creates a bias in the action selection of the agent, not its policy. Consequently, we categorize this method as a guided exploration process.

The PS approach consists of formulating human feedback as action advice that directly updates the agent's behavior. The user can interact with the RL algorithm using an action advice type of human feedback. For this design dimension, we therefore need access to the (partial) policy of an (expert) user in the task the agent is learning to perform.

One advantage of the PS approach is that it does not rely on the representation of the problem using a reward function. Therefore, in real-life scenarios with many conflicting objectives, the PS approach might make it easier for the agent to indicate if its policy is correct, rather than trying to explain it through a reward-shaping method. Nevertheless, the user should know a near-optimal policy to improve the agent's learning process. Even though the effect of the feedback's quality has been investigated in interactive RL [32], more research on this topic is needed to better understand which design dimension is least sensitive to the quality of feedback.

Exploration Process

RL is an ML paradigm based on learning by interaction [98]. To learn a given task, the agent aims to maximize the accumulated reward signal from the environment. This is achieved by performing the actions that have been effective in the past — a learning step called *exploitation*. However, to discover the best actions to achieve its goal, the agent needs to perform actions that have not been previously tried — a learning step called *exploration*. Therefore, to learn an effective policy, the agent has to compromise between exploration and exploitation.

The sampling inefficiency of basic RL methods hinders their use in more practical applications, as these algorithms could require millions of samples to find an optimal policy [20, 78]. An approach called the *guided exploration process* can mitigate this problem.

The guided exploration process aims to minimize the learning procedure by injecting human knowledge that guides the agent's exploration process to states with a high reward [114, 24, 7, 13]. This method biases the exploration process such that the agent avoids trying actions that do not correspond to an optimal policy. This design dimension also assumes that the user understands the task well enough to identify at least one near-optimal policy. Despite this assumption, there is empirical evidence indicating that using human feedback (i.e., guidance) to direct the exploration process is the most natural and sample-efficient interactive RL approach [93].

For example, in [93, 103], users direct the agent's attention by pointing out an object on the screen. Exploration is driven by performing actions that lead the agent to the suggested object. This interaction procedure thus requires access to a model of the environment, so the agent can determine what actions to perform to get to the suggested object.

Another popular approach to guiding the exploration process consists of creating myopic agents [51, 54, 7, 32]. This kind of shortsighted agent constructs its policy by choosing at every time-step to perform the action that maximizes the immediate

reward, according to the action suggested by the user; the action's long-term effects are not considered. Although this guiding approach creates agents that tend to overfit, it has been successfully used in different applications [51, 54, 7, 32].

Value Function

A *reward function* defines the objectives of the agent as immediate rewards that map state or state-action pairs to a scalar value, while a *value function* is an estimate of the expected future reward from each state when following the current policy. Computing a value function is a necessary intermediate step to find a *policy* [94]. The main approach of the value function design dimension is creating an *augmented value function*.

The procedure to augment a value function consists of combining a value function created by human feedback with the value function of the agent. This procedure has been demonstrated to bias the behavior of the agent and accelerate the learning rate [18, 51, 101]. There are too few studies on this design dimension to conclusively compare its performance to other design dimensions [52]. Nonetheless, the reuse of value functions might be a good way to minimize the human feedback required; an expert's value function could be used in multiple scenarios that share similar state-action spaces.

Design Dimension Alternatives

So far, we have explained how RL experts can inject human knowledge through the main components of a basic RL algorithm. There are other ways, however, to interact with low-level features of particular RL approaches. Next, we will explain the two main design dimension alternatives for an interactive RL setting.

Function Approximation (FA) allows us to estimate continuous state spaces rather than tabular approaches such as tabular Q-learning in a GridWorld domain. For any complex domain with a continuous state space (especially for robotics), we need to represent the value function continuously rather than in discrete tables; this is a scalability issue. The idea behind FA is that RL engineers can identify patterns in the state-space; that is, the RL expert is capable of designing a function that can measure the similarity between different states.

FA presents an alternative design dimension that can work together with any other type of interaction channel, which means the implementation of an FA in an interactive RL algorithm does not obstruct interaction with other design dimensions. For example, Warnell et al. proposed a reward-shaping algorithm that also uses an FA to enable their method to use raw pixels as input from the videogame they used as a testbed [108]. However, their FA is created automatically.

As far as we know, the paper in [85] is the only work that has performed user-experience experiments for an interactive RL that uses hand-engineered FAs to accelerate the base RL algorithm. Rosenfeld et al. asked the participants of the experiment to program an FA for a soccer simulation environment. The FAs proposed by the participants were successful: they accelerated the learning process of the base RL algorithm. The same experiment was performed again, this time using the game Super Mario Bros. [48] as a testbed. In the second experiment, the RL's performance worsened when using the FAs. This evidence suggests that designing an effective FA is more challenging than simply using an RS technique in complex environments [85].

Hierarchical Decomposition (HRL) is an effective approach to tackle high-dimensional state-action spaces by decomposing them into smaller sub-problems or temporarily extended actions [99, 27, 59, 106]. In an HRL setting, an expert can define a hierarchy of sub-problems that can be reused as skills in different applications. Although HRL has been successfully tested in different research areas [13, 14, 59], there are no studies from the user-experience perspective in an interactive RL application.

Types of Feedback

In this section, we present the most common types of human feedback found in the literature. In particular, we will describe how each type of feedback delivers the user's expertise to the different human knowledge integration methods that we explained in the previous section.

Binary critique

The use of *binary critique* to evaluate an RL model's policy refers to binary feedback (positive or negative) that indicates if the last chosen action by the agent was satisfactory. This signal of human feedback was initially the only source of reward used [43]. This type of feedback was shown to be less than optimal because people provide an unpredictable signal and stop providing critiques once the agent learns the task [42].

One task for the RL designer is to determine whether a type of reward will be effective in a given application. For example, it was shown that using binary critique as policy information is more efficient than using a reward signal [52, 104]. Similarly, Griffith et al. [36] propose an algorithm that incorporates the binary critique signal as policy information. From a user experience perspective, it has been shown that using critique to shape policy is unfavorable [56].

It is also worth noting that learning from binary critique is popular because it is an easy and versatile method of using non-expert feedback; the user is required to click only "+" and "−" buttons.

Heuristic Function — The *heuristic function* approach [16, 72] is another example of the critique feedback category. Instead of receiving binary feedback directly from the user, in this approach, the critique signal comes from a hand-engineered function. This function encodes heuristics that map state-action pairs to positive or negative critiques. The aim of the heuristic function method is to reduce the amount of human feedback. Empirical evidence suggests that this type of feedback can accelerate RL algorithms; however, more research is needed to test its viability in complex environments such as videogames [85].

Query — The authors of the *active inverse reward design* approach [76] present a query-based procedure for inverse reward design [38]. A *query* is presented to the user with a set of sub-rewards, and the user then has to choose the best among the set. The sub-rewards are constructed to include

as much information as possible about unknown rewards in the environment, and the set of sub-rewards is selected to maximize the understanding of different suboptimal rewards.

Scalar-valued critique

In a manner similar to binary critique, with the *scalar-valued* critique type of feedback, users evaluate the performance of a policy. In this case, the magnitude of the scalar-valued feedback determines how good or bad a policy is. This method can be used on its own to learn a reward function of the environment purely through human feedback. However, it has been shown that humans usually provide nonoptimal numerical reward signals with this approach [60].

Action Advice

In the *action advice* type of feedback, the human user provides the agent with the action they believe is optimal at a given state, the agent executes the advised action, and the base RL algorithm continues as usual. From the standpoint of user experience, the immediate effect of the user's advice on the agent's policy makes this feedback procedure less frustrating [56, 57].

There are other ways to provide action advice to the agent, such as learning from demonstration [101], inverse RL [81, 116, 117], apprentice learning [1], and imitation learning [20]. All these techniques share the characteristic that the base RL algorithm receives as input an expert demonstration of the task the agent is supposed to learn, which is ideal, as people enjoy demonstrating how agents should behave [5, 104, 47].

Guidance

The last type of feedback, *guidance*, is based on the premise that humans find it more natural to describe goals in an environment by specifying the object(s) of interest at a given time-step [103, 93]. This human knowledge leverages the base RL algorithm because the focus is on executing actions that might lead it to a goal specified by the user, which means that the RL algorithm also has access to the transition function of the dynamics in the environment.

RECENT RESEARCH RESULTS

In this section, we survey the interactive RL research area for studies using each of the human knowledge integration methods. These methods define how user feedback is mapped onto the design dimensions of RL-based agents and thus play a critical role in interactive RL. Given the vast amount of work in this research area, we focus on the methods we believe to be the most influential.

Reward Shaping

The reward shaping (RS) method aims to mold the behavior of a learning agent by modifying its reward function to encourage the behavior the RL designer wants.

Thomaz and Breazeal analyzed the teaching style of non-expert users depending on the reward channel they had at their disposal in [103]. Users were able to use two types of RS: positive numerical reward and negative numerical reward. These types of feedback directly modify the value function of the RL model. However, when the user gives negative feedback, the agent tries to undo the last action it performed; this reveals the learning progress to the user and motivates them to use more negative feedback, which achieves good performance with less feedback [103].

Another finding in [103] is that some users give anticipatory feedback to the bot; that is, users assume that their feedback is meant to direct the bot in future states. This analysis displays the importance of studying users' teaching strategies in interactive RL. We need to better understand the user's preferences to teach agents, as well as how agents should provide better feedback about their learning mechanism to foster trust and improve the quality and quantity of users' feedback.

Another RS strategy is to manually create *heuristic functions* [85, 16] that incentivize the agent to perform particular actions in certain states of the environment. This way, the agent automatically receives feedback from the hand-engineered heuristic function. The type of feedback is defined by the RL designer, and it can be given using any of the feedback types reviewed in this paper (i.e., critique or scalar-value). The experiments conducted in [85] demonstrate that using heuristic functions as input for an interactive RL algorithm can be a natural approach to injecting human knowledge in an RL method.

The main shortcoming of heuristic functions is that they are difficult to build and require programming skills. Although it has been investigated how non-experts build ML models in real life [113], there are not many studies on the use of more natural modes of communication to empower non-experts in ML to build effective heuristic functions that generalize well.

The Evaluative Reinforcement (TAMER) algorithm uses traces of demonstrations as input to build a model of the user that is later used to automatically guide the RL algorithm [51].

Warnell et al. later introduced a version of the TAMER algorithm [51] modified to work with a deep RL model [108]. In addition to the changes needed to make the TAMER algorithm work with a function approximation that uses a deep convolutional neural network, the authors of [108] propose a different approach for handling user feedback. Instead of using a loss function that considers a window of samples, they minimize a weighted difference of user rewards for each individual state-action pair.

Similarly, the authors of [10] propose an algorithm called DQN-TAMER that combines the TAMER and Deep TAMER algorithms. This novel combination of techniques aims to improve the performance of the learning agent using both environments and human binary feedback to shape the reward function of the model. Furthermore, Arakawa et al. experimented in a maze-like environment with a robot that receives implicit feedback; in this scenario, the RS method was driven by the facial expression of the user. Since human feedback can be imprecise and intermittent, mechanisms were developed to handle these problems.

Deep versions of interactive RL methods benefit mostly from function approximation, as use of this technique minimizes the feedback needed to get good results. This advantage is

due to the generalization of user feedback among all similar states – human knowledge is injected into multiple similar states instead of only one.

Policy Shaping

The policy shaping (PS) approach consists of directly molding the policy of a learning agent to fit its behavior to what the RL designer envisions.

A PS approach directly infers the user's policy from critique feedback [37, 70]. Griffith et al. introduced a Bayesian approach that computes the optimal policy from human feedback, taking as input the critiques for each state-action pair [37]. The results of this approach are promising, as it outperforms other methods, such as RS. However, PS experiments were carried out using a simulated oracle instead of human users. Further experiments with human users should be conducted to validate the performance of this interactive RL method from a user-experience perspective.

Krening and Feigh conducted experiments to determine which type of feedback — critique or action advice — creates a better user experience in an interactive RL setting [56]. Specifically, they compared the critique approach in [37] to the proposed Newtonian action advice in [58]. Compared to the critique approach, the action advice type of feedback got better overall results: it required less training time, it performed objectively better, and it produced a better user experience with it.

MacGlashan et al. introduced the Convergent Actor-Critic by Humans (COACH) interactive RL algorithm [71]. There is also an extension to this work named deep COACH [12] that uses a deep neural network coupled with a replay memory buffer and an autoencoder. Unlike the COACH implementation in [71], deep COACH [12] uses raw pixels from the testbed as input. The authors argue that using this high-level representation as input means their implementation is better suited for real scenarios. However, the testbed consists of simplistic toy problems, and a recent effort demonstrated that deep neural networks using raw pixels as input spend most of their learning capacity extracting useful information from the scene and just a tiny part on the actual behaviors [25].

Guided Exploration Process

Guided exploration process methods aim to minimize the learning procedure by injecting human knowledge to guide the agent's exploration to states with a high reward.

Thomaz and Breazeal conducted human-subject experiments in the Sophie's Kitchen platform to evaluate the performance of diverse human knowledge integration methods [103]. Their results suggest that using guidance feedback is more effective than using scalar reward feedback. Based upon the algorithm in [103], Suay et al. proposed a variation in which the user can guide exploration by pointing out goal states in the environment [93]; the bot then biases its policy to choose actions that lead to the indicated goal. These experiments were generally successful, but their highlight is their finding that using only exploration guides from the user produces the best results and reduces the amount of user feedback needed [93].

There have been efforts to create adaptive shaping algorithms that learn to choose the best feedback channels based on the user's preferred strategies for a given problem [114]. For instance, Yu et al. defined an adaptive algorithm with four different feedback methods at its disposal that are based on exploration bias and reward shaping techniques [114]. To measure the effectiveness of the feedback methods at every time-step, the adaptive algorithm asks for human feedback; with this information, a similarity measure between the policy of the shaping algorithms and the user's policy is computed. Then, according to the similarity metric, the best method is selected using a softmax function, and the value function for the selected method is updated using q-learning. Once the adaptive algorithm has enough samples, it considers the cumulative rewards to determine which feedback methodology is the best. Overall, one of the exploration bias-based algorithms produced better results and was chosen most often by the adaptive algorithm, as well as demonstrating good performance on its own. The authors of [114] call this algorithm action biasing, which uses user feedback to guide the exploration process of the agent. The human feedback is incorporated into the RL model using the sum of the agent and the user value functions as value functions.

In general, using human feedback as guidance for interactive RL algorithms appears to be the most effective in terms of performance and user experience. However, to make human feedback work, it is necessary to have a model of the environment, so the interactive RL algorithm knows which actions to perform to reach the state proposed by the user. This can be a strong limitation in complex environments where precise models are difficult to create.

Augmented Value Function

The procedure to augment a value function consists of combining the value function of the agent with one created from human feedback.

Studies have proposed combining the human and agent value functions to accelerate learning [52, 101]. In [101], the authors introduce the Human-Agent Transfer (HAT) algorithm, which is based on the rule transfer method [100]. The HAT algorithm generates a policy from the recorded human traces. This policy is then used to shape the q-value function of the agent. This shaping procedure gives a constant bonus to state-action pairs of the agent q-learning function that aligns with the action proposed by the previously computed human policy.

Brys et al. present an interactive RL algorithm named RLfD^2 that uses demonstrations by an expert as input [18]. With these demonstrations, they create a potential-based piecewise Gaussian function. This function has high values in state-action pairs that have been demonstrated by the expert and 0 values where no demonstrations were given. This function is used to bias the exploration process of a Q(λ)-learning algorithm in two different ways. First, the Q-function of the RL algorithm is initialized with the potential-based function values. Second, the potential-based function is treated as a shaping function that complements the reward function from the environment. The combination of these two bias mechanisms is meant to

leverage human knowledge from the expert throughout the entire learning process.

From the user-experience standpoint, the augmented value function design dimension has the advantage of *transfer learning*. For instance, a value function constructed by one user can be used as a baseline to bias the model of another user trying to solve the same task — the learned knowledge from one user is transferred to another. Multiple sources of feedback (coded as value functions) can be combined to obtain more complete feedback in a wider range of states. It is also convenient that a model of the environment is not essential for this approach.

Inverse Reward Design

Inverse reward design (IRD) is the process of inferring a true reward function from a proxy reward function.

IRD [38, 76] is used to reduce reward hacking failures. According to the proposed terminology in [38], the hand-engineered reward function named the *proxy reward function* is just an approximation of the true reward function, which is one that perfectly models real-world scenarios. The process of inferring a true reward function from a proxy reward function is IRD.

To infer the true reward function, the IRD method takes as input a proxy reward function, the model of the test environment, and the behavior of the RL designer who created it. Then, using Bayesian approaches, a distribution function that maps the proxy reward function to the true reward function is inferred. This distribution of the true reward function makes the agent aware of uncertainty when approaching previously unseen states, so it behaves in a risk-averse manner in new scenarios. The results of the experiments in [38] reveal that reward hacking problems lessen with an IRD approach.

The main interaction procedure of IRD starts as regular RS, and the system then queries the user to provide more information about their preference for states with high uncertainty. This kind of procedure can benefit from interaction techniques to better explain uncertainty to humans and from compelling techniques to debug and fix problems in the model.

DESIGN PRINCIPLES FOR INTERACTIVE RL

The design of RL techniques with a human-in-the-loop requires consideration of how human factors impact the design of and interaction with these machine learning algorithms. To produce behaviors that align with the user's intention, clear communication between the human and the RL algorithm is key. In this section, we will introduce general design principles for interactive RL meant to guide HCI researchers to create capable and economical interactive RL applications that users can understand and trust.

Feedback

Delay of human feedback has a great impact on the performance of interactive RL applications. Delay refers to the time that a human user needs to evaluate and deliver their feedback to the interactive RL algorithm. Many examples propose different strategies to deal with feedback delay [10, 12, 43, 105, 103, 108, 82].

The most common approach consists of a delay parameter that expresses to how many past time-steps the current feedback will be applied; this parameter is usually found in practice [12]. Warnell et al. conducted an experiment to quantify the effect of different reward distribution delays on the effectiveness of their algorithm Deep TAMER [108]. This experiment revealed that a small change in the parameters of the expected user feedback timing distribution can have a great impact on performance; expert knowledge on this timing distribution is key to achieving good results.

A different approach to deal with feedback delay is proposed in [93]. In this case, the interactive RL algorithm pauses the agent exploration process every time-step to give the user time to give feedback. More elaborated approaches assume that the feedback delay follows a probability distribution [10]. Likewise, it has been found that less avid users need more time to think and decide on their feedback [82]. It is therefore important to adapt the feedback delay depending on users' level of knowledge.

Fatigue of users and its effects on the quantity and quality of feedback should be considered. It has been observed that humans tend to reduce the quantity of feedback they give over time [43, 39, 50]. The quality also diminishes, as humans tend to give less positive feedback over time. According to [71], this degradation of the quantity and quality of human feedback also depends on the behavior exhibited by the agent. The authors of [71] found that humans tend to give more positive feedback when they notice that the agent is improving its policy over time: feedback is therefore policy-dependent [75]. On the other hand, the experiments of [20] offer evidence to support that human users gradually diminish their positive feedback when the agent shows that it is adopting the proposed strategies. Fachantidis et al. performed experiments to determine the impact of the quality and distribution of feedback on the performance of their interactive RL algorithm [32].

Motivating users to give feedback — elements of gamification have been adopted with good results; gamification strategies have been shown to improve the quantity and quality of human feedback [62, 65].

Some studies focus on improving the quality and quantity of human feedback by incorporating an active question procedure in the learning agent [37, 63, 7, 32]; that is, the agent can ask the user to give feedback in particular states. Amir et al. present an active interactive RL algorithm in which both the agent and the demonstrator (another agent) work together to decide when to make use of feedback in a setting with a limited advice budget. First, the agent determines if it needs help in a particular state and asks the demonstrator for attention. Depending on the situation of the agent, the demonstrator determines if it will help or not. The results of these experiments are promising because they achieve a good level of performance while requiring less attention from the demonstrator.

Maximizing the use of feedback is necessary because interactive RL in complex environments might require up to millions of interaction cycles to get good results [20, 78]. It has been

demonstrated that the inclusion of an approximation function that propagates human feedback to all similar states is effective to tackle the sample inefficiency of interactive RL in complex environments such as videogames [108].

Typification of the End-user

We found that the most important features to typify an end-user in interactive RL involves:

Knowledge level in the task at hand — this has an impact on the quality, quantity, and delay of feedback, and could limit the use of some feedback types that require more precise information, such as demonstrations or action advice.

Preferred teaching strategy — this is essential to select the feedback type appropriate for a given application (see Figure 1).

Safety concerns — this refers to the severity of the effects of the agent's actions if an error in its behavior occurs. If the severity of the errors is high (e.g. property damage, personal harm, politically incorrect behavior, etc.), end-users' perception and willingness to interact with the agent will diminish [79]. The quality and quantity of feedback are therefore also affected.

An end-user typification using these features would help researchers to select the best combination of feedback type and design dimension for a particular interactive RL and for the expected type of end-user. For instance, although all design dimensions assume that the user knows a policy that is at least good enough to solve the task, some design dimensions can better handle non-optimal policies from human feedback [37]. Empirical evidence suggests that the exploration process is the most economical design dimension for interactive RL applications [103, 37, 56]. Nonetheless, myopic agents — which use the policy design dimension — have demonstrated great success in some tasks [32, 60].

The hierarchical decomposition and function approximation design dimensions require human users with a deep understanding of the problem to make effective use of them. Another possibility is combining different design dimensions in the same interactive RL method. It would also be useful to design an active interactive RL that learns which design dimension best suits the application and type of use; in this way, the interactive RL algorithm can minimize the feedback needed from the user [37]. For example, a combination of function approximation and policy design dimensions would enable the use in a videogame environment of an interactive RL that needs only 15 minutes of human feedback to get positive results [108].

Fast Interaction Cycles

In an iterative procedure, such as an interactive RL algorithm, a fast evaluation of the agent's behavior can substantially reduce the time needed for each teaching loop. Some of the approaches meant to lessen the evaluation process consist of evaluation, visualization, or explanatory techniques.

Evaluation techniques such as queries, in which the user selects the best reward function among a set [38, 92], the presentation of multiple trajectories by the agent that then summarizes its behavior [111], or crowd evaluations that distribute the evaluation task among various users [61].

Visualization techniques, such as the HIGHLIGHTS algorithm [6], focus on creating summaries of agents so people can better understand an agent's behavior in crucial situations. These summaries are sets of trajectories that provide an overview of the agent's behavior in key states. It is therefore fundamental to define a metric that measures the importance of states. This is calculated as the difference between the highest and lowest expected returns for a given state. That is, a state in which taking a particular action leads to a significant decrease in the expected reward is considered important.

Explanatory techniques are a different way to address the evaluation problem by explaining the agent's decision process to the human user. This approach enhances the user's understanding of how the agent learns, interprets, and determines the best action to take in a given state.

One example of uncertainty explanation is introduced in [103]. The authors use the gaze of the agent in their experiments as an indicator of its level of uncertainty in any given state.

The main idea of the gaze mechanism is to make the agent explicitly express at every time-step the options it considers most promising by staring at the object of interest. If there is more than one promising action in a given state, the agent will have to focus its gaze on different objects in the environment, communicating indecision to the user and giving the user time to give feedback to the agent. The authors also found that users tend to give more feedback when the agent displays uncertainty, which optimizes the exploration process of the robot.

Design Implications

Our analysis has found some design factors that need more exploration to better understand their impact on interactive RL. These include adaptive systems that can choose between different design dimensions, enabling the use demonstration as feedback in complex robot environments where it is difficult for the user to show the agent how to perform the task it has to learn. This makes interactive RL based applications accessible to more types of users (e.g. non-experts in RL, non-experts in the task at hand, those with physical limitations, etc.), so this design dimension is better for non-experts in the task at hand.

OPEN CHALLENGES

Below, we list what we consider the most promising research directions in interactive RL.

High-dimensional Environments

Making interactive RL feasible for real-world applications with high-dimensional environments is still a challenge. Broadening the domains and applications in which interactive RL is useful is key to extending the impact of ML in people's life.

Recent interactive RL applications that use an autoencoder to spread human feedback to similar states perform well in high-dimensional environments [12]. One underexploited approach

is using crowd-sourced methods to provide human feedback [43] or build a reward function.

Lack of Evaluation Techniques

The evaluation of interactive RL applications from the HCI perspective is limited; most studies focus on improving the performance of algorithms rather than the interaction experience, but proper evaluation is essential to create practical applications for different types of end-users. For instance, the studies by Krening et al. [56, 57] present an adequate evaluation of the user experience in an interactive RL setting.

In the same context, the *problem of generalization from simple testbeds* (e.g., GridWorld) exists, as the results are not consistent when using more complex testbeds. For example, even though RS has been demonstrated to be effective and easy to use by all types of users in a GridWorld environment [10, 12], using the same human knowledge integration method to train agents for the Infinite Mario platform is challenging and ineffective, even for expert users [85].

Lack of Human-like Oracles

The use of simulated oracles in the early stages of implementation of interactive RL applications is useful to save time and get an idea of the results with human users [24, 114, 37]. However, it has been found that results with simulated oracles can differ from those with human users [56, 114] — simulated oracles do not behave like humans. More studies are needed to determine what features make human feedback different from simulated oracles.

Modeling Users

As far as we know, a formal model of users for interactive RL (or IML in general) has not been proposed. Such a model could be used to create interactive RL applications with an active initiative that avoids user fatigue by detecting and predicting user behavior. That is, the interactive RL system would ask for help from the user with the correct frequency to avoid user fatigue. Another possibility is implementing RL-based applications that adapt the interaction channel and feedback type according to the user's preferences. This would require empirical studies to find a way to map between user types and their preferred interaction channels (see next subsection).

Combining Different Design Dimensions

A better understanding of the strengths and weaknesses of each design dimension and feedback type in interactive RL would lead the community to develop effective combinations of interactive RL approaches. Achieving this would require extensive user experience studies with all different combinations of design dimensions and feedback types, as well as in testbeds with small and big state spaces. Furthermore, it would be favorable to find a mapping that considers the type of end-user. This, in combination with an end-user model, would enable the design of interactive RL applications that adapt to the current end-user.

Safe interactive RL

With a human-in-the-loop that can teach an agent new skills, it is important to have a mechanism to restrict agents from going to negative states or performing actions that lead to dangerous situations for humans, politically incorrect situations, or unwanted bias. The addition of Safe RL techniques [34] to an interactive RL setting can be an excellent starting point. Safe RL techniques ensure good performance and respect safety constraints during learning and deployment. Similarly, exploring new interaction methods to impose limits on interactive RL models is a promising research direction.

Fast Evaluation of Behaviors

Fast evaluation methods through visualization techniques have not been adequately studied. These approaches include applications to summarize behavior [6, 38, 92, 111] and explain uncertainty [88, 64]. The main goal of these methods is to reduce the time needed to evaluate the behavior of agents, and assist users to give higher quality feedback.

Explainable interactive RL

RL-based agents have a particular way of representing their environment, depending on the sensors at their disposal and how they interpret the incoming signals. It can therefore be complicated for humans to elicit the state representation of agents, which can lead to giving incorrect feedback to agents. Explainable interactive RL methods aim to help people understand how agents see the environment and their reasoning procedure for computing policies [68, 57, 69, 3]. With more transparency in an agent model, people could find better ways to teach agents.

Another way to take advantage of a transparent agent is by debugging its model. That is, users would be able to interpret why the agent made a certain decision in a particular state. This feature would help people find errors or unwanted bias in the model [80], which is essential in applications that could have a substantial impact on people's lives [22].

CONCLUSIONS

We have presented a survey of interactive RL to empower HCI researchers with the technical background in RL needed to design new interaction techniques and applications. We strongly believe that our paper is a step toward the wider use of interactive RL in the HCI community.

ACKNOWLEDGEMENTS

This work was supported by JST CREST Grant Number JPMJCR17A1, Japan.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 1.

[2] David Abel, John Salvatier, Andreas Stuhlmüller, and Owain Evans. 2017. Agent-agnostic human-in-the-loop reinforcement learning. *arXiv preprint arXiv:1701.04079* (2017).

[3] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[4] Adrian K Agogino and Kagan Tumer. 2004. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society, 980–987.

[5] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.

[6] 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*. International Foundation for Autonomous Agents and Multiagent Systems, 1168–1176.

[7] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara Grosz. 2016. Interactive Teaching Strategies for Agent Training. In *In Proceedings of IJCAI 2016* (in proceedings of ijcai 2016 ed.). https://www.microsoft.com/en-us/research/publication/interactive-teaching-strategies-agent-training/

[8] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[9] Sule Anjomshoae, Kary Främling, and Amro Najjar. 2019. Explanations of Black-Box Model Predictions by Contextual Importance and Utility.

[10] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda. 2018. DQN-TAMER: Human-in-the-Loop Reinforcement Learning with Intractable Feedback. *arXiv preprint arXiv:1810.11748* (2018).

[11] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

[12] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. 2019. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257* (2019).

[13] Christian Arzate Cruz and Jorge Ramirez Uresti. 2018. HRLB $\wedge$ 2: A Reinforcement Learning Based Framework for Believable Bots. *Applied Sciences* 8, 12 (2018), 2453.

[14] Aijun Bai, Feng Wu, and Xiaoping Chen. 2015. Online planning for large markov decision processes with hierarchical decomposition. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 4 (2015), 45.

[15] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. 2009. Natural actor–critic algorithms. *Automatica* 45, 11 (2009), 2471–2482.

[16] Reinaldo AC Bianchi, Murilo F Martins, Carlos HC Ribeiro, and Anna HR Costa. 2013. Heuristically-accelerated multiagent reinforcement learning. *IEEE transactions on cybernetics* 44, 2 (2013), 252–265.

[17] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016).

[18] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. 2015. Reinforcement Learning from Demonstration Through Shaping. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 3352–3358. http://dl.acm.org/citation.cfm?id=2832581.2832716

[19] Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2334–2346.

[20] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*. 4299–4307.

[21] Jack Clark and Dario Amodei. 2016. Faulty Reward Functions in the Wild. (2016). https://openai.com/blog/faulty-reward-functions/ Accessed: 2019-08-21.

[22] European Commission. 2018. 2018 reform of EU data protection rules. (2018). https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf Accessed: 2019-06-17.

[23] Francisco Cruz, Sven Magg, Cornelius Weber, and Stefan Wermter. 2016. Training agents with interactive reinforcement learning and contextual affordances. *IEEE Transactions on Cognitive and Developmental Systems* 8, 4 (2016), 271–284.

[24] Francisco Cruz, Johannes Twiefel, Sven Magg, Cornelius Weber, and Stefan Wermter. 2015. Interactive reinforcement learning through speech guidance in a domestic scenario. In *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[25] Giuseppe Cuccu, Julian Togelius, and Philippe Cudré-Mauroux. 2019. Playing atari with six neurons. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 998–1006.

[26] Richard Dearden, Nir Friedman, and Stuart Russell. 1998. Bayesian Q-learning. In *Aaai/iaai*. 761–768.

[27] Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13 (2000), 227–303.

[28] Thomas Dodson, Nicholas Mattei, and Judy Goldsmith. 2011. A natural language argumentation interface for explanation generation in Markov decision processes. In *International Conference on Algorithmic DecisionTheory*. Springer, 42–55.

[29] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. 2018. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217* (2018).

[30] Francisco Elizalde and Luis Enrique Sucar. 2009. Expert Evaluation of Probabilistic Explanations.. In *ExaCt*. 1–12.

[31] Francisco Elizalde, L Enrique Sucar, Manuel Luque, J Diez, and Alberto Reyes. 2008. Policy explanation in factored Markov decision processes. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM 2008)*. 97–104.

[32] Anestis Fachantidis, Matthew E. Taylor, and Ioannis Vlahavas. 2018. Learning to Teach Reinforcement Learning Agents. *Machine Learning and Knowledge Extraction* 1, 1 (2018), 21–42. DOI: http://dx.doi.org/10.3390/make1010002

[33] Jerry Alan Fails and Dan R Olsen Jr. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 39–45.

[34] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *J. Mach. Learn. Res.* 16, 1 (Jan. 2015), 1437–1480. http://dl.acm.org/citation.cfm?id=2789272.2886795

[35] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, and others. 2015. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning* 8, 5-6 (2015), 359–483.

[36] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea Thomaz. 2013a. Policy shaping: Integrating human feedback with reinforcement learning. In *In Proceedings of the International Conference on Neural Information Processing Systems (NIPS*.

[37] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013b. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*. 2625–2633.

[38] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse Reward Design. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 6765–6774. http://papers.nips.cc/paper/7253-inverse-reward-design.pdf

[39] Mark K Ho, Michael L Littman, Fiery Cushman, and Joseph L Austerweil. 2015. Teaching with rewards and punishments: Reinforcement or communication?. In *CogSci*.

[40] Andreas Holzinger. 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics* 3, 2 (2016), 119–131.

[41] Andreas Holzinger, Markus Plass, Katharina Holzinger, Gloria Cerasela Crişan, Camelia-M Pintea, and Vasile Palade. 2016. Towards interactive Machine Learning (iML): applying ant colony algorithms to solve the traveling salesman problem with the human-in-the-loop approach. In *International Conference on Availability, Reliability, and Security*. Springer, 81–95.

[42] Charles Lee Isbell, Michael Kearns, Satinder Singh, Christian R Shelton, Peter Stone, and Dave Kormann. 2006. Cobot in LambdaMOO: An adaptive social statistics agent. *Autonomous Agents and Multi-Agent Systems* 13, 3 (2006), 327–354.

[43] Charles Lee Isbell Jr and Christian R Shelton. 2002. Cobot: A social reinforcement learning agent. In *Advances in neural information processing systems*. 1393–1400.

[44] Natasha Jaques, Shixiang Gu, Richard E Turner, and Douglas Eck. 2016. Generating music by fine-tuning recurrent neural networks with reinforcement learning. (2016).

[45] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. 2018. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *arXiv preprint arXiv:1810.08647* (2018).

[46] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[47] Tasneem Kaochar, Raquel Torres Peralta, Clayton T Morrison, Ian R Fasel, Thomas J Walsh, and Paul R Cohen. 2011. Towards understanding how humans teach robots. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 347–352.

[48] Sergey Karakovskiy and Julian Togelius. 2012. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 55–67.

[49] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. Pcgrl: Procedural content generation via reinforcement learning. *arXiv preprint arXiv:2001.09212* (2020).

[50] W Bradley Knox, Brian D Glass, Bradley C Love, W Todd Maddox, and Peter Stone. 2012. How humans teach agents. *International Journal of Social Robotics* 4, 4 (2012), 409–421.

[51] W. Bradley Knox and Peter Stone. 2009. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. In *The Fifth International Conference on Knowledge Capture*. http://www.cs.utexas.edu/users/ai-lab/?KCAP09-knox

[52] W Bradley Knox and Peter Stone. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 5–12.

[53] W Bradley Knox and Peter Stone. 2012. Reinforcement learning from simultaneous human and MDP reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 475–482.

[54] W Bradley Knox, Peter Stone, and Cynthia Breazeal. 2013. Training a robot via human feedback: A case study. In *International Conference on Social Robotics*. Springer, 460–470.

[55] Raj Korpan, Susan L Epstein, Anoop Aroor, and Gil Dekel. 2017. Why: Natural explanations from a robot navigator. *arXiv preprint arXiv:1709.09741* (2017).

[56] Samantha Krening and Karen M Feigh. 2018. Interaction algorithm effect on human experience with reinforcement learning. *ACM Transactions on Human-Robot Interaction (THRI)* 7, 2 (2018), 16.

[57] Samantha Krening and Karen M Feigh. 2019a. Effect of Interaction Design on the Human Experience with Interactive Reinforcement Learning. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. ACM, 1089–1100.

[58] Samantha Krening and Karen M Feigh. 2019b. Newtonian Action Advice: Integrating Human Verbal Instruction with Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 720–727.

[59] Young-Seol Lee and Sung-Bae Cho. 2011. Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer. In *International conference on hybrid artificial intelligence systems*. Springer, 460–467.

[60] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871* (2018).

[61] Levi HS Lelis, Willian MP Reis, and Ya'akov Gal. 2017. Procedural Generation of Game Maps With Human-in-the-Loop Algorithms. *IEEE Transactions on Games* 10, 3 (2017), 271–280.

[62] Pascal Lessel, Maximilian Altmeyer, Lea Verena Schmeer, and Antonio Krüger. 2019. "Enable or Disable Gamification?": Analyzing the Impact of Choice in a Gamified Image Tagging Task. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 150, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300380

[63] Guangliang Li, Hayley Hung, Shimon Whiteson, and W. Bradley Knox. 2013. Using Informative Behavior to Increase Engagement in the Tamer Framework. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '13)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 909–916. http://dl.acm.org/citation.cfm?id=2484920.2485064

[64] Guangliang Li, Shimon Whiteson, W. Bradley Knox, and Hayley Hung. 2016b. Using informative behavior to increase engagement while learning from human reward. *Autonomous Agents and Multi-Agent Systems* 30, 5 (01 Sep 2016), 826–848. DOI: http://dx.doi.org/10.1007/s10458-015-9308-2

[65] Guangliang Li, Shimon Whiteson, W. Bradley Knox, and Hayley Hung. 2018. Social interaction for efficient agent learning from human reward. *Autonomous Agents and Multi-Agent Systems* 32, 1 (01 Jan 2018), 1–25. DOI:http://dx.doi.org/10.1007/s10458-017-9374-8

[66] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016a. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* (2016).

[67] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3-4 (1992), 293–321.

[68] Yang Liu, Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2019a. Experience-based Causality Learning for Intelligent Agents. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 18, 4 (2019), 45.

[69] Yang Liu, Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2019b. Experience-based Causality Learning for Intelligent Agents. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 18, 4, Article 45 (May 2019), 22 pages. DOI: http://dx.doi.org/10.1145/3314943

[70] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. 2016. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems* 30, 1 (2016), 30–59.

[71] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2285–2294.

[72] Murilo Fernandes Martins and Reinaldo AC Bianchi. 2013. Heuristically-accelerated reinforcement learning: A comparative analysis of performance. In *Conference Towards Autonomous Robotic Systems*. Springer, 15–27.

[73] Sean McGregor, Hailey Buckingham, Thomas G Dietterich, Rachel Houtman, Claire Montgomery, and Ronald Metoyer. 2017. Interactive visualization for testing Markov Decision Processes: MDPVIS. *Journal of Visual Languages & Computing* 39 (2017), 93–106.

[74] Qinggang Meng, Ibrahim Tholley, and Paul WH Chung. 2014. Robots learn to dance through interaction with humans. *Neural Computing and Applications* 24, 1 (2014), 117–124.

[75] Raymond G Miltenberger. 2011. *Behavior modification: Principles and procedures*. Cengage Learning.

[76] Sören Mindermann, Rohin Shah, Adam Gleave, and Dylan Hadfield-Menell. 2018. Active Inverse Reward Design. *arXiv preprint arXiv:1809.03060* (2018).

[77] 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).

[78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[79] Cecilia G Morales, Elizabeth J Carter, Xiang Zhi Tan, and Aaron Steinfeld. 2019. Interaction Needs and Opportunities for Failing Robots. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. 659–670.

[80] Chelsea M Myers, Evan Freed, Luis Fernando Laris Pardo, Anushay Furqan, Sebastian Risi, and Jichen Zhu. 2020. Revealing Neural Network Bias to Non-Experts Through Interactive Counterfactual Examples. *arXiv preprint arXiv:2001.02271* (2020).

[81] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[82] Bei Peng, James MacGlashan, Robert Loftin, Michael L Littman, David L Roberts, and Matthew E Taylor. 2016. A need for speed: Adapting agent action speed to improve task learning from non-expert humans. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 957–965.

[83] Athanasios S Polydoros and Lazaros Nalpantidis. 2017. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems* 86, 2 (2017), 153–173.

[84] Martin L Puterman. 2014. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[85] Ariel Rosenfeld, Moshe Cohen, Matthew E Taylor, and Sarit Kraus. 2018. Leveraging human knowledge in tabular reinforcement learning: A study of human subjects. *The Knowledge Engineering Review* 33 (2018).

[86] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, England.

[87] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[88] Hanna Schneider, Julia Wayrauther, Mariam Hassib, and Andreas Butz. 2019. Communicating Uncertainty in Fertility Prognosis. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 161, 11 pages. DOI: http://dx.doi.org/10.1145/3290605.3300391

[89] Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. (2016).

[90] Patrice Y Simard, Saleema Amershi, David M Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and others. 2017. Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742* (2017).

[91] Satinder P Singh and Richard S Sutton. 1996. Reinforcement learning with replacing eligibility traces. *Machine learning* 22, 1-3 (1996), 123–158.

[92] Patrikk D Sørensen, Jeppeh M Olsen, and Sebastian Risi. 2016. Breeding a diversity of super mario behaviors through interactive evolution. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–7.

[93] Halit Bener Suay and Sonia Chernova. 2011. Effect of human guidance and state space size on interactive reinforcement learning. In *2011 Ro-Man*. IEEE, 1–6.

[94] Richard Sutton and JAMH. 2019. The value-function hypothesis. (2019). http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/valuefunctionhypothesis.html Accessed: 2019-08-21.

[95] Richard Sutton, Michael Littman, and Al Paris. 2019. The reward hypothesis. (2019). http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html Accessed: 2019-08-21.

[96] Richard S Sutton. 1985. Temporal Credit Assignment in Reinforcement Learning. (1985).

[97] Richard S Sutton. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*. 1038–1044.

[98] Richard S Sutton and Andrew G Barto. 2011. Reinforcement learning: An introduction. (2011).

[99] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[100] Matthew E Taylor and Peter Stone. 2007. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*. ACM, 879–886.

[101] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 617–624.

[102] Ana Tenorio-González, Eduardo Morales, and Luis Villaseñor-Pineda. 2010. Dynamic Reward Shaping: Training a Robot by Voice. 483–492. DOI: http://dx.doi.org/10.1007/978-3-642-16952-6_49

[103] AL Thomaz and C Breazeal. 2006. Adding guidance to interactive reinforcement learning. In *Proceedings of the Twentieth Conference on Artificial Intelligence (AAAI)*.

[104] Andrea L Thomaz and Cynthia Breazeal. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172, 6-7 (2008), 716–737.

[105] Andrea Lockerd Thomaz, Guy Hoffman, and Cynthia Breazeal. 2005. Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*.

[106] Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. 2016. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993* (2016).

[107] Ning Wang, David V Pynadath, Ericka Rovira, Michael J Barnes, and Susan G Hill. 2018. Is It My Looks? Or Something I Said? The Impact of Explanations, Embodiment, and Expectations on Trust and Performance in Human-Robot Teams. In *International Conference on Persuasive Technology*. Springer, 56–69.

[108] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. 2018. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[109] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[110] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. 2003. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 792–799.

[111] Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*. 1133–1141.

[112] Qian Yang. 2017. The role of design in creating machine-learning-enhanced user experience. In *2017 AAAI Spring Symposium Series*.

[113] Qian Yang, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. Grounding interactive machine learning tool design in how non-experts actually build models. In *Proceedings of the 2018 Designing Interactive Systems Conference*. 573–584.

[114] Chao Yu, Tianpei Yang, Wenxuan Zhu, Guangliang Li, and others. 2018. Learning Shaping Strategies in Human-in-the-loop Interactive Reinforcement Learning. *arXiv preprint arXiv:1811.04272* (2018).

[115] Shao Zhifei and Er Meng Joo. 2012. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics* 5, 3 (2012), 293–311.

[116] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. (2008).

[117] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2009. Human Behavior Modeling with Maximum Entropy Inverse Optimal Control.. In *AAAI Spring Symposium: Human Behavior Modeling*. 92.