Neural Computing and Applications (2023) 35:16841–16852
https://doi.org/10.1007/s00521-022-08061-z

S.I.: HUMAN-ALIGNED REINFORCEMENT LEARNING FOR AUTONOMOUS AGENTS AND ROBOTS

Hu-bot: promoting the cooperation between humans and mobile robots

Karine Miras[1] · Decebal Mocanu[2,3] · A. E. Eiben[1]

Received: 14 December 2021 / Accepted: 16 November 2022 / Published online: 13 January 2023

Abstract
This paper investigates human–robot collaboration in a novel setup: a human helps a mobile robot that can move and navigate freely in an environment. Specifically, the human helps by remotely taking over control during the learning of a task. The task is to find and collect several items in a walled arena, and Reinforcement Learning is used to seek a suitable controller. If the human observes undesired robot behavior, they can directly issue commands for the wheels through a game joystick. Experiments in a simulator showed that human assistance improved robot behavior efficacy by 30% and efficiency by 12%. The best policies were also tested in real life, using physical robots. Hardware experiments showed no significant difference concerning the simulations, providing empirical validation of our approach in practice.

Keywords Automation · Mobile robots · Reinforcement learning · Safety · Human–robot cooperation · Human-in-the-loop

1 Introduction

Automation can be a sign of societal progress for multiple reasons. For instance, it can lower production costs of goods [1] and this way reduce their prices, making them available to a more broad niche. Furthermore, it might tackle the classical *Ds* of labor-saving [2], so that humans can avoid tasks that are "dangerous, dirty, demanding, dull...". Beyond the *Ds*, there are other types of tasks that people would prefer to be done by robots, for instance, tasks that require memorization, keen perceptual skills, and service-orientation [3].

Automation has been increasingly growing in different domains [4]. Considering robotic automation, a great deal of it is nevertheless model-based, and systems often lack the ability to adapt. Critically, although modeling a controlled environment is possible, this is only feasible for well-structured environments. If an environment is unstructured or semi-structured—if it is a changing environment—a model and a procedural system relying on it would have to be constantly updated to remain valid. Furthermore, a system capable of coping with a lack of structure could promote leeway for stakeholders of an application. For instance, imagine a warehouse in which complete standardization (boxes sizes, physical organization, etc) had been imposed to allow automation: an adaptable automation system could reduce the constraints imposed on the value chain involved with this warehouse.

This scenario motivates a need for adaptable robot capacities: learning. Encouragingly, there is a multitude of artificial learning techniques that can be applied for robot learning in a warehousing scenario, e.g., reinforcement learning, evolutionary algorithms, Bayesian optimization, etc. Notwithstanding, these techniques are not free from mistakes and uncertainty, thus raising serious concerns about safety. Though these mistakes are often less dramatic when robots work in isolation, they could have severe safety implications if happening in an environment where there can be people inserted in, i.e., a warehouse.

✉ Karine Miras
k.dasilvamirasdearaujo@vu.nl

1 Department of Computer Science, Vrije Universiteit Amsterdam, De Boelelaan, Amsterdam, The Netherlands

2 Faculty of Electrical Engineering, Mathematics and Computer Science, Universiteit Twente, Enschede, The Netherlands

3 Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Faced with such a problem, one possible solution is the cooperation between humans and robots. While some tasks are better performed by humans, others are better performed by robots, and a proper allocation of such tasks must be done [5] so that robots and humans can cooperate optimally. One way of concretize this cooperation is using human help for supervising and fine-tuning the behavior of these robots, intervening when undesired behavior is observed. This is specially important when the undesired behavior means unsafe behavior. The use of humans to influence the learning process of artificial agents has been done in different ways in the literature [6], e.g., demonstrations [7–11], imitation [12–14], supervision [15], intervention for safety [16, 17], instructions [18], corrective feedback [19], and evaluative feedback [20]. Moreover, it is frequently not necessary that the human involved be an expert on the task they are helping with [21]. However, this is more commonly done with agents that act in simulation 2D worlds [7, 12, 17, 21, 22]. Even when real robots are utilized, they are usually constrained so that the robots can not transit freely within an environment, like for example robotic arms [15, 18, 20].

This paper investigates human–robot collaboration in a novel setup: a human helps a *mobile* robot that can move and navigate freely in an environment. Specifically, the human helps by remotely taking over control during the process of learning a task. Importantly, we test our final results with physical mobile robots in the real world.

The main objective of the paper is to show how an automatically learnt non-optimal policy can be improved by interaction with a non-specialist human. The specific research questions are:

a. Can human help prevent undesired robot behavior?
b. What are the challenges involved in (a)?

2 Related work

As mentioned in the introduction, diverse methods for teaching artificial agents have been applied in the literature. While different nomenclatures have been used to refer to these methods, and despite their specificities, the main idea behind them is often very similar: humans interfere directly or indirectly, providing inputs about how a given behavior should be carried out.

One term commonly used within the teaching of artificial agents is demonstration. For example, learning by demonstration was applied to teach a simulated robot arm [11] through storing human demonstration data in a replay buffer, while accounting for cases in which demonstrations were suboptimal in comparison to the performance of the agent itself.

An actor-critic architecture was used [15] to apply human supervision to robot learning in different ways. First, the supervisor could supply additional sources of evaluative feedback to be used as rewards. Second, the supervisor could interfere by sending actions directly to the system, overwriting robot actions. Finally, the supervisor could bias the exploration of the actor by providing hints regarding promising actions. The second aspect (interference) is similar to what we have in our system. However, their supervisor is involved from the beginning and withdraws at some point, while ours interferes solely in a later stage and only if necessary. Additionally, most cases within their work use a hand-coded controller simulating a human supervisor instead of an actual human. Additionally, in all cases but one, the agents were simulated, and the only real robot case was a non-mobile robot (robot arm). In another work, feedback was provided to correct actions of agents [19], but in this case, the human teacher informed the magnitude of possible errors of continuous actions. Regarding the acting/withdrawing of the human, other possibilities have been explored in the literature: having the human not necessarily present from the beginning nor from a specific point on, but taking control when deeming necessary [14].

Supervision was also applied [17] through having a human that intervenes in the actions of agents: the human watched the episodes fully (in slow motion) and blocked actions that seemed likely to be catastrophic. The main difference between this and our method is that the human actions were used to change the agent behavior in the current timestep but were not used for training, and the (incorrect) state-action pairs intended by the robot were penalized with negative rewards.

A similar learning concept has been applied in another study [12] but is referred to as imitation learning. The authors used a simple 2D matrix as an environment where agent A tried to imitate agent B. To imitate, agent A observed and stored the transitions performed by agent B in its own replay buffer. Agent B was not a human though, but a hand-coded agent.

In another example, instead of demonstrating how and when to perform an action, unlabeled guided signals were given by a human teacher to instruct a real robot arm about correct behavior [20]. Additionally, the teacher provided a reward when the action executed corresponded to the instructed action. This way, the teacher did not have to execute the task themselves but only guided agents and provided them with evaluative feedback.

3 Background

Reinforcement Learning (RL) is a machine learning paradigm in which agents learn from their interactions with an environment, and it originated in the psychology of animal learning [23]. In RL, an agent senses *states* of an environment and takes *actions* reacting to these states. The decision of which actions to take is done with the use of what is called a *policy*: the distribution of due actions over possible states. Differently from supervised learning, in RL there are no labeled examples, but criteria of success called *rewards*. The objective is to maximize the expected return. In practice, rewards are used to adjust the policy so that in each given state, the best actions can be taken so to maximize the rewards that can be received by the agent.

In the current work, we utilize RL to carry out imitation learning [13]. The environment is modeled as a (Partial Observable) Markov Decision Process (MDP), where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. An experience is registered as a tuple (s, a, r, s'), where $s \in \mathcal{S}$ is the current state, $a \in \mathcal{A}$ is the action taken at state s, r is the reward received for performing action a, and $s' \in \mathcal{S}$ is the new state resulting from that action. Among multiple steps described in Sect. 4, a human performs a demonstration to be learned by an agent generating such a tuple. These tuples generated by human control are used during the learning of the agent, together with tuples generated by the agent itself. More details follow in the next section.

Comparing RL with semi-supervised methods, e.g., Evolutionary Algorithms, there is one fundamental difference: the latter does not allow credit assignment per state-action pair but only for a group of state-action pairs as a whole. Furthermore, for complex problems in which there is a large number of states involved, neural networks are often utilized, and this is called deep RL. Some examples of deep RL algorithms are DDPG, DQN, PPO [24], SAC [25], and TD3 [26]. TD3 is one of the most successful among the latest and suitable to our needs.

4 Methodology

4.1 Robot description

The robot utilized is called Robobo,[1] and it is depicted and described in Fig. 1. The interaction between the robot body and the robot control is explained in Fig. 2. The robot is actuated by controlling the acceleration of its wheels, and thus two parameters are needed every time that actuation commands are sent to it: speed of *left* and *right* wheels, both ranging from -3 to 3 in simulation and from -100 to 100 in hardware.

When these commands are sent, actions start to be executed simultaneously and are executed for 400 ms. A positive value makes the wheel move forward, a negative value makes the wheel move backward, and zero means the wheel does not move. Furthermore, there is a parameter to tilt the smartphone in its holder, and it was set to a fixed value of 55 degrees because this way the camera could see both the floor and the walls.

The robot uses its proximity sensors and pictures from its camera for perceiving the environment. The values of the sensors in simulation represent the distances between the sensors and something being sensed and are zero if nothing is close enough to activate them. These values were used without transformation. Conversely, in hardware, these values grow exponentially larger the closer the sensors get to something that activates them. Thus, each of these values v when not zero was transformed into $v = 1/\log 2(v)$, so as to be closer to the simulation values. Moreover, the raw pictures were not provided directly to the robot controller, but were transformed into the following 6 features: proportion of *green* pixels, proportion of *gray* pixels, average of *green* pixels coordinates in the x axis, average of *green* pixels coordinates in the y axis, average of *gray* pixels coordinates in the x axis, and average of *gray* pixels coordinates in the y axis—the averages were normalized to be between 0 and 1.

4.2 Environment and task

The environment for the task is a square arena with a flat floor surrounded by walls, and it contains 7 green boxes of different sizes. The arena can be seen in both simulation and hardware in Fig. 2. The task the robot needs to perform is finding and approaching all the boxes in the environment. It is not necessary to touch the box, but only to get a few millimeters close enough to it. Each time a package is approached, it is removed from the environment. This toy-task could be compared to a warehousing scenario, in which a robot had to approach the stored items, e.g., for counting them, for placing something on them, etc. Moreover, the robot can start an episode in one of 7 different pre-determined positions, and it should learn how to cope with any of these.

There are small differences between the simulated and hardware environments: (1) The arena has 2 x 2 meters in simulation, and 2 m 24 cm x 2 m 24 cm in hardware; (2) The walls are gray in simulation, but red in hardware[2]: this is necessary because there is too much noise that appears

[1] https://theroboboproject.com.

[2] Robots trained in simulation can be transferred to hardware by using color filters.

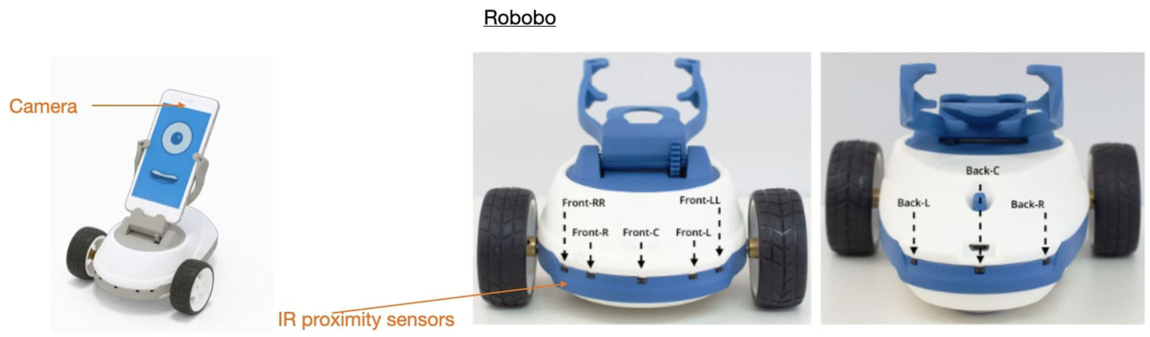

Fig. 1 The Robobo robot has a body that is like a two-wheeled vehicle, with 5 infrared proximity sensors at the front and 3 infrared proximity sensors at the back. Additionally, a smartphone with a frontal camera is attached to a holder at the top of the robot body

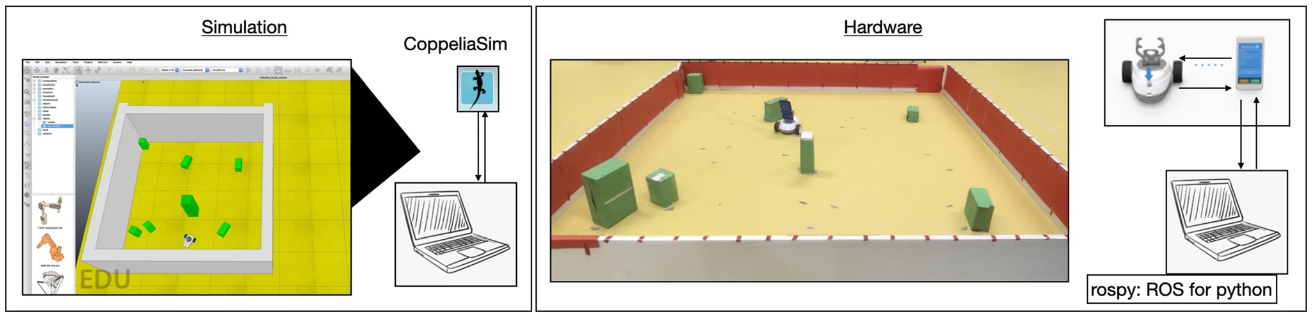

Fig. 2 In both simulation and hardware, a computer program controls the robot by sending actuation commands to it, and when the actions are completed the robot sends its sensing information back to the program, including pictures from the camera and infrared values. Simulation: the Robobo and the environment are simulated using CoppeliaSim (https://www.coppeliarobotics.com). Hardware: a computer running ROS communicates with the smartphone via Wi-Fi, and the smartphone communicates with the robot using Bluetooth

gray in the hardware environment, e.g., windows and other objects or people in the room, while the simulation environment is static; (3) In simulation, the packages are placed at the very same spots in each episode start because this is done via a script, and this script is also responsible for removing approached packages. Nevertheless, these positions present small variations in hardware because the placing and removal are carried out manually by a human; (4) The shapes of the boxes are not the same but approximated; and (5) There are differences in the physics of the simulated environment and the hardware environment, which is the classical reality gap problem.

4.3 Experimental setup

Our experiments are divided into three parts: setup I, setup II-a, and setup II-b. Setup I was conducted first, and both setups II were built upon the results from setup I. In setup I, the robot learns fully autonomously. In both setups II, the learning is continued using parameters different from setup I. In setup II-a, a human contributing to the learning is present in the loop. In setup II-b, there is no human in the loop. The purpose of setup II-b is to make sure that results obtained in setup II-a are not simply due to the parameters changes—these changes were applied to suit the human-in-the-loop dynamics.

We utilize RL as our learning method because dealing with credit assignment is fundamental, since there will be a human-helper interfering only on sub-parts of the episode. In particular, we use TD3 [26] to learn a control policy.

Both setups were used to solve the same task (Sec. 4.2), and they share the identical parameters and procedures unless otherwise mentioned. The metrics of success utilized in all experiments are: *total_success* measuring the efficacy and ranging from 0 to 7—it is the average number of packages collected during validation[3]; and *steps* measuring the efficiency and ranging from 1 to 200 episode-steps.[4]

In setup I, 21 independent learning runs were performed, using a total number of 35.000 time-steps each. In each episode, the robot was placed randomly in one of the 7 possible initial positions. After every 1.000 steps, the learning process was frozen and the current policy was validated by playing one episode from each of the possible

[3] Note that there are 7 packages in the environment, and that the robot is tested starting from 7 different positions.

[4] If the robot uses fewer steps to finish the episode, then it is more efficient.

initial positions and averaging the success metrics. This resulted in 35 stages of validation.[5]

Thereafter, the human inspected the results of each run both quantitatively (using the success metrics) and qualitatively (by observing robot behavior). The stage considered for inspection was only the very best stage of each run.[6] For the qualitative inspection, the human-helper watched one episode played by the policy from each of the possible initial positions. Subsequently, the human assessed if the behavior presented any problems, and if so, what kind of problems these were. The types of problems found through the experiments are described in Table 1 from Sect. 5. Given the simplified nature of our experimental setup, these behavioral problems are not necessarily hazardous, but they illustrate how robots could be hazardous by carrying out undesired behavior.

Later on, for setups II, only the runs considered to present problems were utilized: each problematic run was copied twice (once for II-a and once for II-b) and all stages after its best stage were removed (this means the remaining policy was the one present in the best stage). Sub sequentially, in setup II-a, these learning runs were continued while having the human-helper in the loop, using a maximum budget of 5.000 steps (5 stages). It was not mandatory for the human-helper to use all the budget of steps. The following sequence was repeated: (a) monitor episodes being played and interfere when judged necessary; (b) when a stage is finished, watch the validation and decide if the run should end or if another stage should be carried out (unless budged was used up). As for setup II-b, the runs were continued without the human in the loop, using the same budget utilized by the human-helper in setup II-a.

Note: the episodes are not fully deterministic—there can be small variations in robot behavior when playing with the same policy—because of technical issues regarding competing processes in the computer. Therefore, a final test was carried out for setups II, in which the policy of the best stage of each run was played 3 times in each of the 7 possible initial positions. The metrics from these repetitions were averaged and used in the final comparison between setup II-a and setup II-b.

4.3.1 Setup I: Fully autonomous reinforcement learning

In this setup, the robot learns the control policy only by itself. Each episode of the task has a maximum budget of $k = 200$ steps. Each step starts when an action is initialized and ends when the action is finished. Because the duration of the actions was set to have always the same length, the steps have always the same size. An episode is considered to be over when all packages are successfully approached by the robot, or when all the k steps are used.

The *states* provided for the control policy are the values of the 8 proximity sensors and the 6 features extracted from the picture. The *actions* of the policy are two continuous actions ranging from 0 to 1, and each one represents the speed of each wheel.[7]

The rewarding scheme utilized when the robot was learning by itself was: (1) when the robot succeeds in approaching a package a reward of 100 is applied; (2) when there is any green pixel in the visual field of the robot, a reward relative to the proportion of green pixels is applied; (3) when there is no green pixel in the visual field of the robot, a negative reward of -0.1 is applied. While reward 1 is a direct measure of success, reward 2 helps to attribute credit by pointing out if the robot is getting closer to the targets. Finally, reward 3 disincentivizes actions that waste time and do not lead the robot closer to the targets. The TD3 parameters were: *batch_size* = 128, *learning_starts* = 100, *learning_rate* = 3e-4, *target_policy_noise* = 0.2, *target_noise_clip* = 0.5, *gamma* = 0.99, *tau* = 0.005, *expl_noise* = 0.1. The replay buffer was sampled uniformly randomly.

4.3.2 Setup II: Human help in the reinforcement learning

In this setup, the robot also learns the control policy by itself, but additionally, a human can help the learning process (only for setup II-a). This human is referred here as the human-helper,[8] and may interfere in the robot behavior using a joystick (Fig. 3). Moving the left directional button maximally to the front will set the left wheel speed to the maximum, and moving it maximally to the back will set this wheel speed to the minimum (the same happens with the right button for the right wheel).[9] Human actions are only sent to the robot when at least one of the directional buttons is being moved by the human. Whenever the human moves any directional button, the human actions have precedence over the actions decided by the policy so that the latter are ignored, but as soon as the human releases the button(s), the policy returns to control the robot. If one directional button is moved and the other is

[5] What we here call a ‘validation stage’ is formed by multiple episodes played to assess the quality of the policy learned so far.

[6] The best stages were chosen, thus continuing from the point where the autonomous RL stagnated.

[7] Before being sent to the robot, the values of the actions had to be converted to the ranges of speed defined in the robot parameters.

[8] The human-helper in these experiments was the first author of this paper.

[9] The action values from the joystick were also converted to the ranges of speed defined in the robot parameters.

Table 1 Validation results of setup I (before human help) and II-a (after human help)

Run	Behavioral problem before help	Behavioral problem after help	Help
1	(−−) always straight	(+) useless rotations	5
2	(+) stuck at wall/slow rotations	(++) no apparent problem	1
3	(+) useless rotations	(+) slow rotations	5
4	(++) no apparent problem	–	0
5	(++) no apparent problem	–	0
6	(+) useless rotations	(+) quick and slight wall bumps	2
7	(+) useless rotations	(++) no apparent problem	2
8	(−) useless rotations/slow rotations	(++) no apparent problem	1
9	(−−) always rotating	(+) useless rotations	5
10	(−−) always rotating	(−) stuck at wall	5
11	(+) slow rotations	(++) no apparent problem	1
12	(++) no apparent problem	–	0
13	(+) useless rotations	(+) no apparent problem	2
14	(+) useless rotations/stuck at wall	(++) no apparent problem	2
15	(++) no apparent problem	–	0
16	(++) no apparent problem	–	0
17	(++) no apparent problem	–	0
18	(++) no apparent problem	–	0
19	(++) no apparent problem	–	0
20	(++) no apparent problem	–	0
21	(++) no apparent problem	–	0

(++)=very good, (+)=good, (−)=bad, and (−−)=very bad, help=number of stages utilized by human

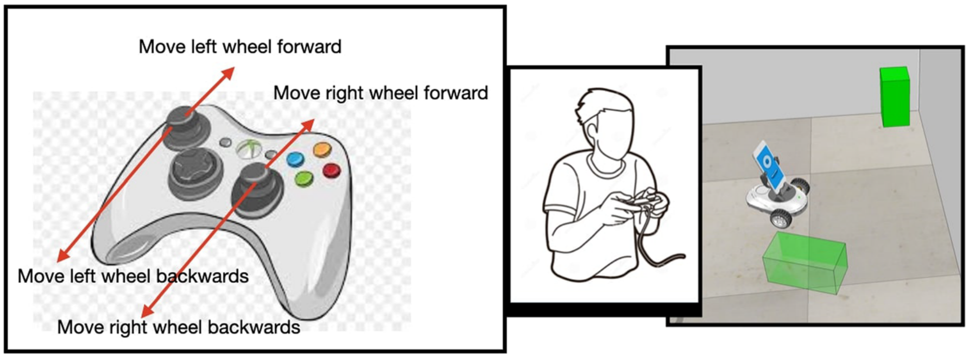

Fig. 3 Human–robot interaction: the human can interfere in the robot behavior using a joystick. Each of two directional buttons can be used to move the left and right wheel of the robot forward or backwards

not, the wheel of the unmoved button receives a speed of zero.

It is important to mention that the human-helper does not need to be experienced with joystick robot control, and so the human-helper was allowed to practice for an unlimited number of episodes.

Beyond using human actions to interfere in the robot behavior, the human help has to be considered in the learning process in four other ways. First, the parameter expl_noise is set to 0, because policy exploration can make the helping process very confusing and tiresome for the human-helper, since it may be difficult to clearly see if the robot is learning what is being taught and since the robot may too often act in a "stubborn" way. Second, in every step in which there is human interference, it is the human state-action pair that is added to the replay buffer,[10] while the state-action pair of the policy is ignored.

Third, the rewarding scheme needs the following alterations: (1) a human help reward is defined as $h = 0.1$; (2) the negative rewards are replaced by h only when there is human interference; (3) when rewards are positive, they are summed up with h, regardless the presence of human interference. Alteration 2 is necessary because when the human-helper is making a demonstration of what to do, there may be intermediate actions that do not lead to a positive reward immediately (sparse rewards), but that will

[10] Values returned by the joystick were normalized to correspond to the policy action ranges.

do so after a sequence of steps. For example, when demonstrating how to escape a corner, the human-helper may drive backward for a few steps, and then rotate for a few other steps, to only then have green pixels in the visual field. In this case, by using the standard rewarding scheme, it would be very difficult for the agent to learn that those actions are valuable, because rewards associated with them are too many steps away. Therefore, it is necessary to impose a reward that assumes these actions are good because they are decided by the human-helper. When there is no human interference, the negative reward for mistakes made by the policy continues to apply, so that the policy can still autonomously learn what it should not do. Additionally, 3 is applied to make sure that state-action pairs of alteration 2 do not seem more profitable than the actual successful ones since the proportion of green can often be lower than h (in case a visualized package is very far). Additionally, alteration 3 is applicable during the whole episode, even in the steps when the human-helper does not interfere, because otherwise, the states when human-help was necessary may seem more profitable than other states, but this is not necessarily true.

Finally, the batch size is changed to have the size of k (same as maximum size of an episode), and each item in the buffer can stay in it for only (also) k steps. Note that this way the buffer is never larger than the maximum size of an episode. The reason for these changes is related to the idea that the human-helper is not necessarily an expert on the task, and can make mistakes. While this is not a problem for the cases when an actual reward from the environment is obtained, it could become a problem for the cases when the human reward is applied. Therefore, assuming that the human-helper will make mistakes, we do not maintain their actions in the buffer for too long. As the stages progress, the human-helper is also learning and getting better, and the latest state-action pairs will influence the learning, while the old ones will have no direct effect in the learning anymore. The increase in the batch and buffer size is applied to make sure every pair has a fair chance to participate in the learning, since the pairs do not stay for too long anymore, and since the human help lasts much fewer stages.

Let us now describe the human-helper guidelines utilized during the human-help experiments. The human-helper could interfere in the robot behavior anytime throughout the episodes of setup II, however, the following principles were taken into consideration: the focus of help in the first stage should be on the problems observed on the qualitative inspection of the run; after watching the validation of each human-helped stage, a new qualitative assessment should be made, and the next stage should focus on the current problems; if no problems are presented, the learning run can end before the budget is used up; during the episodes, interference should be carried out as minimally as possible, that is, for fixing actions that are clearly problematic; if slight imperfect behavior is observed, but the human-helper has the impression they can not help much with it, help should not be carried out; observe not only the third-person view of the simulation but also the first-person view window, so that the decision making of the human-helper is based on aspects perceivable by the robot.

5 Results

5.1 Setup I: Fully autonomous RL

The results of the experiments[11] in the setup before human help show that RL performed overall very well on the task (Fig. 4), with an average of *6.7* for total_success (efficacy) and 180 for *steps* (efficiency). This means that almost always all the packages were approached most of the time, while using 90% of the time budget for completing the task. This good result was reached around stage 20, remaining mostly stagnated for the remaining 15 stages. Table 1 shows the results of the validations separated per run before and after human help. More than half the runs achieved their best result before stage 27.

Analyzing the independent runs, we can see in more detail the imperfections of each run. After the human inspections of setup I, each run was classified into the levels of behavioral quality *very good*, *good*, *bad*, and *very bad*, according to their quantitative and qualitative success. All runs with efficacy 7 and with non-apparent behavioral problems were classified as *very good*. Runs with efficacy between 6 and 7 that had apparent problems were classified as *good*.[12] Runs with efficacy below 6 were classified as *bad*. Runs with efficacy below 3 were classified as *very bad*. Furthermore, every run not considered *very good* was classified into one or more categories of behavioral problems described in the caption of Table 1.

A run can be judged to present behavioral quality problems if an inadequate behavior happens in multiple or even only one from the starting positions. Naturally, runs with imperfect effectiveness presented some sort of problematic behavior. Nevertheless, some runs that were maximally effective presented problematic behavior that was reflected only on the efficiency. This means that the robot

[11] The markers above the boxplots in the Results section represent significance levels from Wilcoxon tests. ns: $0.05 < p <= 1$, *$0.01 < p <= 0.05$, **$0.001 < p <= 0.01$, ***$0.0001 < p <= 0.001$, ****$p <= 0.0001$.

[12] There was one exception (run 13) that presented no apparent problem but had efficacy 6.9 instead of 7, so to be strict, it was considered only *good*.

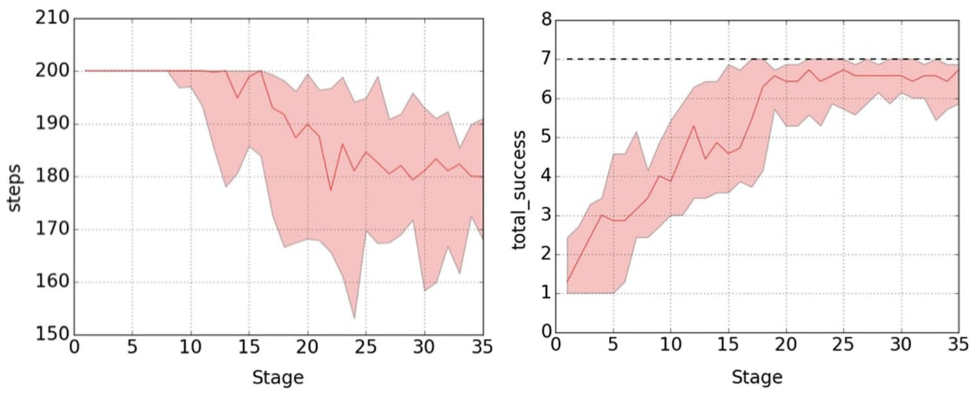

Fig. 4 Results of setup I: Averages of the metrics of success throughout the learning process among the independent runs before human help. Lines represent the medians and clouds represent the first and third quartiles

was doing something that, from the point of view of the human-helper, clearly could be done better.

Among the 21 runs, 10 (48%) presented no apparent problem.[13] By "no apparent" we mean that the human-helper could not identify any problems, or was not sure if there was a significant problem with which they could help. For instance, the human-helper may have the impression that a route is not optimum, but they are aware that sometimes there is too much subjectivity involved, and it may not be worth it risking to worsen the policy. As for the 11 imperfect runs, 3 of them were *very bad*, approaching less than two packages. Additionally, 1 of them was *bad*, approaching 5.7 packages. Finally, 7 of them were *good*, with 3 of these approaching almost all packages, and the other 4 approaching all packages but with inefficient behavior. Note that because the metrics are averages of results in each starting position, a *total_success* of almost 7 often means that all packages were approached most times but sometimes some were missed.

5.2 Setup II: Effects of human help on RL

The human help was applied to the 11 imperfect runs and it brought improvements to 7 (64%) of them (Fig. 5). In 5 (71%) of the improved runs, there was an increase in efficacy, and in 2 (29%) of them there was increase only in efficiency. Note that efficiency and efficacy are correlated, because when not all packages all approached, the whole budget of steps is used. All runs that improved in efficacy improved also in efficiency.

Considering the runs from the perspective of behavioral quality, in most cases the human helped in shifting from a less desired behavior to a more desired behavior: among the *very bad* runs, 2 of them became *good* runs, and one of them became a *bad* run; the *bad* run became a *very good* run; and among the *good* runs, 4 became *very good* runs, and 3 remained *good* runs. This is illustrated in Table 1 in the columns 'behavioral problem before human help' and 'behavioral problem after human help'.

The plots of the runs that did not improve with human help are available in Fig. 3 of Appendix 1. In summary, the improvement obtained through human help was an increase of 30% in efficacy and 12% in efficiency (Fig. 6).

To finalize our study, we tested our best policies in hardware, with the purpose of demonstrating that these robots can also function in the real-world, beyond simply in simulation. For this test, we utilized two different policies: (a) the best policy among the ones that did not need human help, and (b) the best policy among the ones improved by human help. These final hardware tests were carried out just like the final simulation tests described in Sect. 4.3: the policy was played 3 times repeatedly in each of the possible initial positions. In Fig. 7 we see that the efficacy[14] in hardware is only slightly lower than in simulation: 3% worse in run 7 and 1% worse in run 12, while the difference is significant only for the former. In practice, the robot could approach all packages in almost every episode, having in rare cases picked up only 6 packages. A video demonstrating the robot behavior in both simulation and hardware is available on YouTube.[15]

6 Discussion

While the fully autonomous RL had a favorable performance with almost half of the runs presenting no apparent behavioral problem, the other half contained imperfect runs with room for improvement. The imperfection of these runs can be considered undesired behavior, which in determined environments may represent unsafe behavior. From the machine learning point of view, the imperfection could be tackled in different ways, for instance, parameter

[13] The plots of validation for these runs are available in Fig. 1 of Appendix 1, in the folder supplementarymaterial of our GitHub (see Supplementary information).

[14] Efficiency was not compared because the time steps in simulation and hardware are not comparable.

[15] https://youtu.be/n-Z-QDVCJCM.

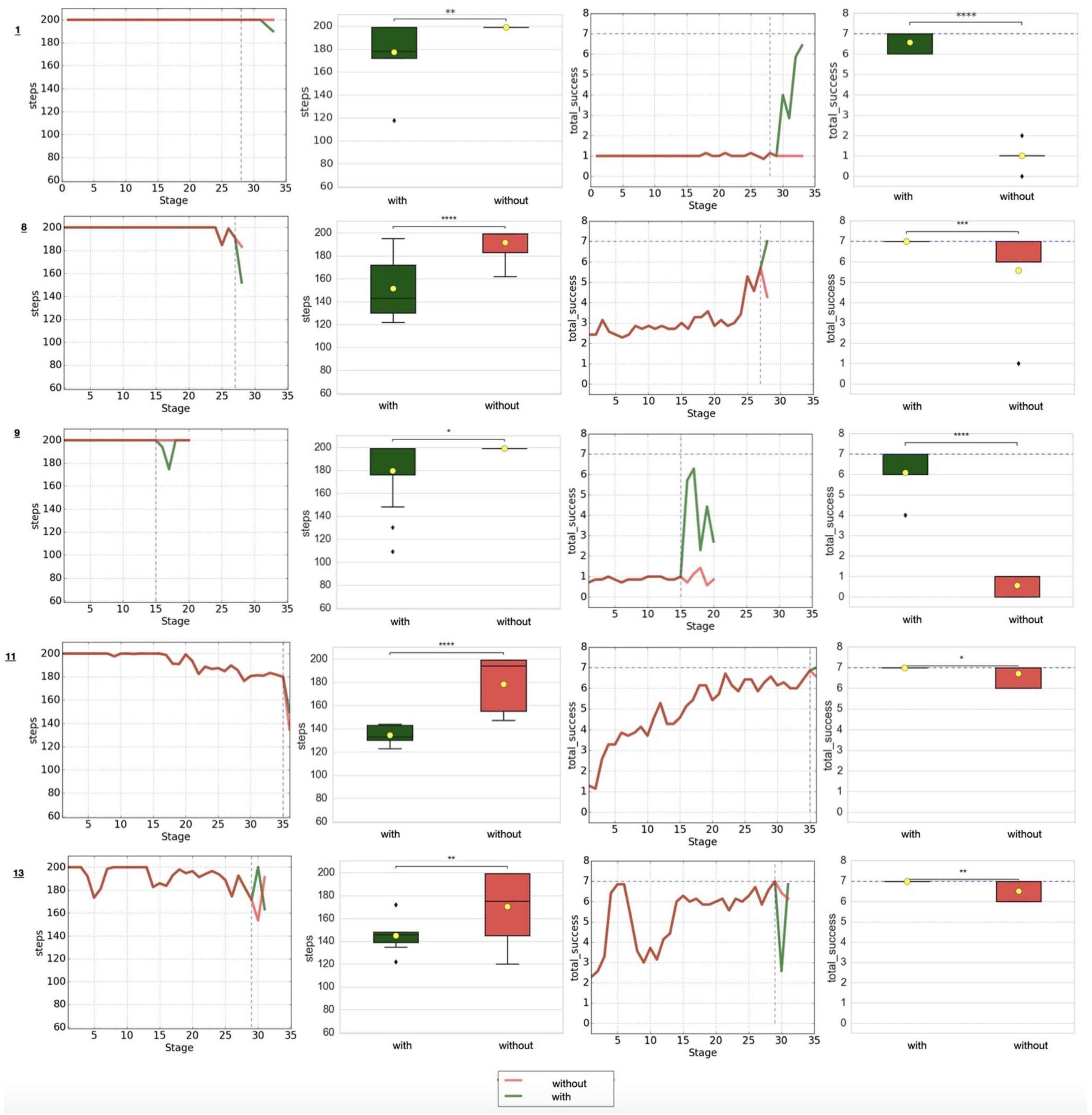

Fig. 5 Results of setup II: with=setup II-a and without=setup II-b. Each row (made of four plots) represents one run that improved both in *efficacy* and *efficiency* and with human help. The line plots show the metrics of success in the validation stages throughout the learning processes: naturally, the lines overlap in the stages that were simply copied from setup I. The boxplots compare the best controllers found by each experiment. The number of the run is displayed at the left of the plots. The horizontal dotted lines mark the maximum number of packages achievable, while the vertical ones mark the best validation stage of the run before human help. The plots of the runs that improved only in *efficiency* are available in Fig. 2 of Appendix 1

tuning [27], reward shaping refinement [28], etc. However, we did not intend to propose algorithmic improvements for the fully autonomous RL but to demonstrate and discuss the promotion of safety by keeping the human in the loop preventing the takeover of undesired behavior. Note that safety promotion is not intended to occur in the fully autonomous setup stages but in the hybrid setup stages: RL in cooperation with humans. Importantly, the robot loses control whenever the human-helper presses a button but regains control immediately after the button release. In this way, they are both responsible for the final robot behavior, and in such a way that the robot is never idle. We

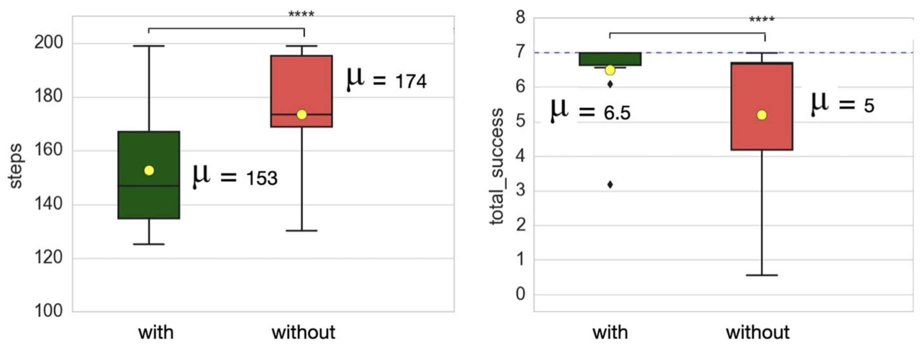

Fig. 6 Summary over all experiments of Setup II: assessment of the average change in *efficiency*: $\frac{174-153}{174}=$ 12% of decrease in steps; and *efficacy*: $\frac{6.5-5}{5}=30\%$ of increase in total_success. Each point in the bloxplot is the average within a run. *with* = Setup II-a and *without* = Setup II-b

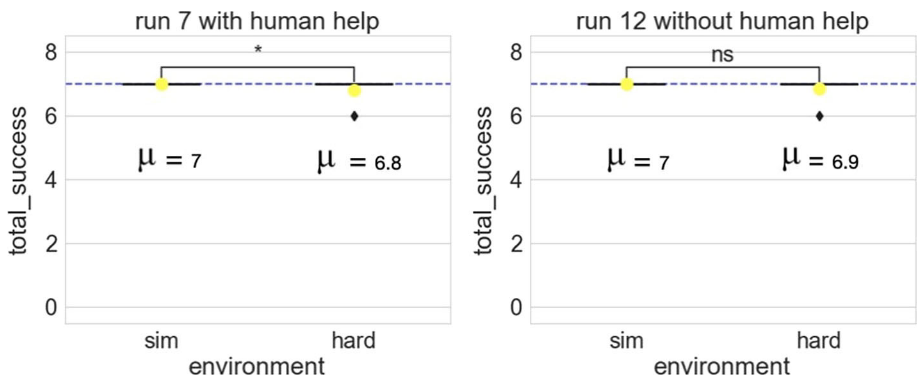

Fig. 7 Comparing the success of controllers tested in simulation and in hardware. sim = simulation and hard = hardware

emphasize that the human-helper has always precedence over the robot policy, providing a quick alternative for situations in which the robot is behaving dangerously.

Ultimately, the inclusion of a non-specialist human in the loop benefited significantly the efficacy and/or efficiency of the learnt robot behavior in the majority of the imperfect runs.[16] Whereas our research goal was achieved, there were multiple challenges involved. These are described below and are related to the design choices for human help presented in Sect. 4.3.2.

1. Translating the human choice of actions into the "language" of the policy: we did that by adding the human-generated state-action pairs and their rewards to the replay buffer when there was human interference.
2. Understanding perceptual differences between robot and human-helper (Theory of Mind): after several failed attempts, we realized that the human-helper was taking actions based on environmental states that were not available for the robot. We tackled that by watching the robot vision field in the first person, reflecting on what environmental information was available, and then transforming these insights into new states.
3. Sparse rewards from human demonstration: we solved it by adding a "human reward" for state-action pairs that would normally be punished.
4. Balancing real rewards and human rewards: when the human behavior achieved real rewards, they were used summed up to the human reward, so that (real) low partial rewards did not seem less attractive than the human rewards.
5. Calibrating the magnitude of human interference effects on the learning: because the human is not an expert or necessarily experienced with the task, there can be mistakes. Therefore the learning should not happen too greedily, so to avoid that mistakes be too deeply incorporated into the policy. We dealt with it by redesigning the replay buffer dynamics.
6. Dealing with policy exploration during human help: the policy exploration proved very confusing for the human-helper since it made it difficult to observe if the human demonstrations had been learned or not. We decided to completely turn off the exploration, considering that this should not be a problem: that is the case because the policies had already gone through a stage of learning, and were just being fine-tuned while guided by the human-helper.

Beyond improving robot behavior and promoting safety, this solution could impact society in another way: employment. Automation can drastically and abruptly lead to high levels of unemployment [29], in opposition to the

[16] To facilitate clear results, human interference was applied only after the convergence of each run. In a real-world scenario nevertheless, human interference could be applied as soon as undesired behavior was observed.

promotion of societal progress. Picturing a scenario in which this system would be deployed in a warehouse, an imaginable outcome is: the robots act as a labor-saving technology, and thus the demand for non-specialist human labor is reduced, but this reduction is mitigated by the need for labor to fine-tune the robots. Moreover, there may continuously be a need for behavior fine-tuning because such an environment is never static. Therefore, non-specialist human labor demand could be preserved to some extent, thus tackling unemployment.

To conclude, we demonstrated how to promote safety using human help to prevent undesired robot behavior. One limitation of this study was using a single human-helper, and who was involved with the development of the system. Additionally, because in some cases the human-helper could not improve the robot behavior, further work is necessary to tackle the current challenges. For future work, it would be interesting (a) to repeat this experimental setup using a large number of human-helpers with different profiles (age, sex, education, etc); (b) to design a framework to implement such a system in a real-world warehouse subject to changing environmental conditions, e.g., through what means would humans monitor and assess robot behavior, where would the joysticks be localized, etc., and (c) to provide the full images of the camera to the neural network, as opposed to extracting features beforehand.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-022-08061-z. All the code produced is available on GitHub (https://github.com/karinemiras/learning_robobos/tree/d6fb1e3a4826575ca385d6930a4660578bd3166b). The data generated is in supplementarymaterial/data_neural2022.zip of this same repository.

Acknowledgements We would like to thank Kim Baraka for his insightful advice.

Funding This research was funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organization for Scientific Research (https://www.hybrid-intelligence-centre.nl), Grant No. 024.004.022.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Park J, Nakamura KA (2015) Labor-saving and employment in a plant factory. Environ Control Biol 53:89–92
2. Connell JK (1993) The rise of labour migration to Japan. University of Sydney, Sydney
3. Takayama L, Ju W, Nass CBd (2008) Dangerous and dull: what everyday people think robots should do. In: ACM/IEEE international conference on human–robot interaction (HRI), pp 25–32
4. Jämsä-Jounela Sirkka-Liisa (2007) Future trends in process automation. Annu Rev Control 31:211–220
5. Repperger D, Phillips C (2009) The human role in automation. In: Handbook of automation, pp 295–304
6. Zhang R, Torabi F, Guan L, Ballard D, Stone P (2019) Leveraging human guidance for deep reinforcement learning tasks. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19, p 7
7. Ho M, Littman M, MacGlashan J, Cushman F, Austerweil J (2016) Showing versus doing: teaching by demonstration. Adv Neural Inf Process Syst 29:3027–3035
8. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483
9. Brys T, Harutyunyan A, Suay HB, Chernova S, Taylor ME, Nowé A (2015) Reinforcement learning from demonstration through shaping. In: Twenty-fourth international joint conference on artificial intelligence
10. Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, Horgan D, Quan J, Sendonaris A, Osband I et al (2018) Deep q-learning from demonstrations. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
11. Nair A, McGrew B, Andrychowicz M, Zaremba W, Abbeel P (2018) Overcoming exploration in reinforcement learning with demonstrations. In: 2018 IEEE international conference on robotics and automation (ICRA), pp 6292–6299. IEEE
12. Fournier P, Sigaud O, Colas C, Chetouani MC (2021) Curriculum learning and imitation for object control in nonrewarding environments. IEEE Trans Cogn Dev Syst 13:239–248
13. Hussein A, Gaber M, Elyan E, Jayne C (2017) Imitation learning: a survey of learning methods. ACM Comput Surv (CSUR) 50:1–35
14. Kelly M, Sidrane C, Driggs-Campbell K, Kochenderfer MJ (2019) Hg-dagger: interactive imitation learning with human experts. In: International conference on robotics and automation (ICRA), pp 8077–8083. IEEE
15. Rosenstein M, Barto A, Si J, Barto A, Powell W, Wunsch D (2004) Supervised actor-critic reinforcement learning. In: Learning and approximate dynamic programming: scaling up to the real world, pp 359–380
16. Goecks VG, Gremillion GM, Lawhern VJ, Valasek J, Waytowich NR (2019) Efficiently combining human demonstrations and interventions for safe training of autonomous systems in real-time. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 2462–2470
17. Saunders W, Sastry G, Stuhlm"uller A, Evans OTwE (2018) Towards safe reinforcement learning via human intervention. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, pp 2067–2069

18. Najar A, Sigaud O, Chetouani M (2020) Interactively shaping robot behaviour with unlabeled human instructions. Auton Agents Multi-Agent Syst 34:1–35
19. Pérez-Dattari R, Celemin C, Ruiz-del-Solar J, Kober J (2018) Interactive learning with corrective feedback for policies based on deep neural networks. In: International symposium on experimental robotics, pp 353–363. Springer
20. Najar A, Sigaud O, Chetouani M (2016) Training a robot with evaluative feedback and unlabeled guidance signals. In: 2016 25th IEEE international symposium on robot and human interactive communication (RO-MAN), pp 261–266
21. Rutard F, Sigaud O (2020) Chetouani: Tirl: enriching actor-critic rl with non-expert human teachers and a trust model. In: 2020 29th IEEE international conference on robot and human interactive communication (RO-MAN), pp 604–611
22. Knox WB, Stone P (2009) Interactively shaping agents via human reinforcement: the tamer framework. In: Proceedings of the fifth international conference on knowledge capture, pp 9–16
23. Sutton R, Barto A (2018) Reinforcement learning: an introduction
24. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: a brief survey. IEEE Signal Process Mag 34(6):26–38
25. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning, pp 1861–1870. PMLR
26. Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of the 35th international conference on machine learning, vol 80, pp 1587–1596
27. Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, Thomas J, Ullmann T, Becker M, Boulesteix A-L et al (2021) Hyperparameter optimization: foundations, algorithms, best practices and open challenges. arXiv preprint arXiv:2107.05847
28. Laud AD (2004) Theory and application of reward shaping in reinforcement learning
29. Acemoglu D, Restrepo P (2020) Robots and jobs: evidence from us labor markets. J Polit Econ 128(6):2188–2244