2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Towards Safe Human-Robot Collaboration Using Deep Reinforcement Learning

Mohamed El-Shamouty*[1], Xinyang Wu*[2], Shanqi Yang[1], Marcel Albus[1], Marco F. Huber[2,3]

***Abstract*— Safety in Human-Robot Collaboration (HRC) is a bottleneck to HRC-productivity in industry. With robots being the main source of hazards, safety engineers use over-emphasized safety measures, and carry out lengthy and expensive risk assessment processes on each HRC-layout reconfiguration. Recent advances in deep Reinforcement Learning (RL) offer solutions to add intelligence and comprehensibility of the environment to robots. In this paper, we propose a framework that uses deep RL as an enabling technology to enhance intelligence and safety of the robots in HRC scenarios and, thus, reduce hazards incurred by the robots. The framework offers a systematic methodology to encode the task and safety requirements and context of applicability into RL settings. The framework also considers core components, such as behavior explainer and verifier, which aim for transferring learned behaviors from research labs to industry. In the evaluations, the proposed framework shows the capability of deep RL agents learning collision-free point-to-point motion on different robots inside simulation, as shown in the supplementary video.**

I. Introduction

Manufacturing needs to adapt to fluctuating market demands with flexible production and achieve mass-personalization to produce highly customized products. This motivates the need for lean manufacturing with flexible reconfigurable Human-Robot Collaboration (HRC) cells. Yet, robots are the main source of hazards. For every cell reconfiguration, the production stops until a safety engineer carries out a risk-assessment process. In such process, the safety engineer evaluates the hazards, quantifies the risks for these hazards and sets up the necessary safety measures (e.g., slowing down the robot) to reduce the assessed risks. The risk-assessment process incurs the following challenges: (1) hazards cannot be easily foreseen, (2) a considerably long time is required, which makes safety a bottleneck for layout changes (the vision is 1-hour time-limit for cell reconfiguration [1]–[3]), and (3) safety constraints are over-emphasized to tighten the possibility of hazards' occurrence, which reduces the productivity of robots.

Awad et al. address challenges (1) and (2) by introducing Computer-Aided Risk-Assessment (CARA) [4]. Given a cell layout, and product, process and resource descriptors (PPR-triple) [4], [5], CARA assesses the cell configuration using an expert rule-based system, calculates the possible hazards and risks and recommends safety measures with the reduced risk amount, such as placing laser scanners to detect human proximity and reducing the robot's speed. However, CARA is used only in planning (pre-deployment) phase for known cell-configurations and considers simplified and deterministic robot and human behavior described by the PPR-triple. Therefore, rules developed by CARA are still rather strict, falling under challenge (3). More advanced safety functionalities addressing challenge (3) come from the robot control perspective. Particularly, for motion planning, recent work exists using Model-Predictive Control (MPC) including human-motion prediction to plan collision-free paths of robots [6], [7]. Such approaches, however, are mostly still in research labs, due to the high dependency on models or sensory accuracy, or the high computational complexity [8]. A more detailed discussion of safety methods in HRC could be referred in [9].

Machine Learning (ML) provides an alternative to handle these challenges, especially the computational complexity and the curse of dimensionality. More specifically, Reinforcement Learning (RL) is thought to be one of the key components towards general artificial intelligence [10], and recent breakthroughs in deep RL suggests that artificial Neural Network (NN) is a natural platform for RL agents [11]–[13]. Despite the efficiency and versatility of NNs, combining NNs with RL agents under deep RL can easily incur the following challenges: (1) deep RL agents need reward functions and accurate instrumented environments encoding the task and context of applicability for training, (2) training deficiency - even if the data used in training is representative, it may under-represent the hazardous cases because they are often rare in the real-world [14], [15], (3) deep RL agents can easily learn unintended functionalities that can lead to unsafe behaviours [16], and (4) lack of explainability and transparency of the behaviour exerted by deep RL agents [17].

The work presented in this paper is twofold. First, we define a framework to address the aforementioned challenges using deep RL as an enabling technology combined with control methods to learn tasks safely. The framework addresses preparing the training environment, learning behaviors, verifying and eventually deploying the learned behaviors in industry. As such, the core components of the framework are introduced (conceptually). Second, we evaluate the mapped

*Authors contributed equally and names are in alphabetical order.

[1] M. El-Shamouty, S. Yang and M. Albus are with the Department of Robot and Assistive Systems, Fraunhofer IPA. [mohamed.el-shamouty, shanqi.yang, marcel.albus] @ipa.fraunhofer.de.

[2] X. Wu and M. F. Huber are with the Center for Cyber Cognitive Intelligence, Fraunhofer IPA. xinyang.wu@ipa.fraunhofer.de, marco.huber@ieee.org.

[3] M. F. Huber is also with the Institute of Industrial Manufacturing and Management IFF, University of Stuttgart.

This work was partially supported by the Ministry of Economic Affairs of the state Baden-Württemberg in Germany (Center for Cyber Cognitive Intelligence – Grant No. 017-192996 and Cyberprotect). The authors would like to also acknowledge Ramez Awad, Dennis Lamaj, Jiacheng Yang and Elias Merzhaeuser for discussions and support with implementations.

RL-environment with different deep RL baseline algorithms in learning collision-free point-to-point motion on different robots and in the presence of static and dynamic obstacles, as shown in the supplementary video. This provides a first instantiation (implementation) of the framework excluding the components for verification and explanation.

The rest of the paper is structured as follows. Section II states the used notation. Section III formulates the proposed framework. Section IV evaluates the framework on learning collision-free point-to-point motion.

II. Notation

A. Markov Decision Process

Markov Decision Process (MDP) is a classical formulation of decision making processes, namely choosing different actions in different situations, where the actions influence not only the immediate rewards, but also the future rewards, or states [18]. A MDP contains four basic elements: $(s_t, a_t, P(s_{t+1}|s_t, a_t), R(s_{t+1}|s_t, a_t))$, where s_t and s_{t+1} represent respectively the current and next states w.r.t time-step t, a_t stands for the action, $P(s_{t+1}|s_t, a_t)$ represents the transition probability of getting into state s_{t+1} when taking action a_t in state s_t, and $R(s_{t+1}|s_t, a_t)$ is the immediate reward received from the environment after transitioning from s_t to s_{t+1}. In MDP, the transition probability $P(s_{t+1}|s_t, a_t)$ is only dependant on the current state s_t and chosen action a_t. Thus, given s_t and a_t, the next state s_{t+1} is conditionally independent of previous states and actions, which satisfies the Markov Property [19].

B. Reinforcement Learning

MDPs are an ideal mathematical formulation for RL problems, for which a straightforward methodology of learning from interaction to achieve a goal is framed. In the context of RL, the learnt decision maker is called *agent*, which chooses actions and collect rewards from the environment. The agent makes decisions to gain not only the current reward, but also the cumulative rewards in subsequent states, namely, *return*. A discount factor $\gamma \in (0, 1]$ is introduced to trade-off immediate and delayed rewards. The agent and MDP together give rise to a sequence called *trajectory*, which contains a set of state-action pairs: $\tau = ((s_0, a_0), (s_1, a_1), (s_2, a_2)...)$. The discounted return for trajectory τ at time-step t is

$$R_t(\tau) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \ , \tag{1}$$

where $r_{t'}$ is the reward at time-step t'. The core optimization problem of RL is to find the optimum policy $\pi^\star$, which maximizes the expected return

$$\pi^\star = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi}[R_t(\tau)] \ . \tag{2}$$

III. Deep RL Framework for Safe HRC

We propose a framework (shown in Fig. 1) to teach robots safe and productive behaviors. In the framework, three main phases exist: *pre-learning*, *learning* and *verification* phases.

A. Pre-Learning Phase

The main goal of the pre-learning phase is to encode the task and safety requirements and create an instrumented training environment that reflects such requirements and the context of applicability. The main components in the pre-learning phase are mentioned as follows.

Intended Functionality: Compared to conventional software systems, the complete behavior of a ML-skill cannot be defined or verified as precisely [20]. ISO/PAS 21448:2019, a recent standard for road vehicles, defines the requirements specification using the term *intended functionality*. The intended functionality reflects the task definition and the context of applicability that is expected to be covered by the ML-skill.

Expert Knowledge: Currently, expert knowledge is required to provide an instrumented training environment that encodes the task and the context of applicability into a RL setting. The expert knowledge is reflected in: (1) encoding the context of applicability by providing instrumented training environments, (2) encoding the task and safety requirements in *reward functions* [18] and *constraints* [21], [22], (3) defining the *action space* by providing existing robot skills for the RL agents to learn on [11], (4) defining the *state space* by providing the necessary perception skills (e.g., object recognition), (5) and optionally providing demonstrations to increase the learning efficiency of RL agents [23].

To partially automate encoding the context of applicability in the RL setting, we leverage features from CARA. CARA requires a setup of the environment layout including resources (e.g., robot). Environment layout can be defined using existing robot description models (such as URDF or MJCF). Accordingly, CARA uses an expert-system to identify the hazards regions, e.g., by calculating the intersection between the full reachability of the robot and the human accessibility. This in turn is used in defining the training and verification context.

Reward Function: The reward function can either be sparse or dense. Sparse reward functions encode RL tasks in the form of providing a non-negative reward signal $+r$ or 0, only when a task or sub-task is achieved correctly, and negative reward signal $-r$ otherwise. Sparse reward functions are one of the most trivial and convenient ways to define tasks in the RL setting. However, sparse reward functions can easily lead the RL agent to learn unintended behavior [16], due to the delayed reward signals. On the other hand, a dense reward function provides a state-dependent reward signal that better describes the task, e.g., Euclidean distance in case of goal-based tasks. Defining a well-shaped dense reward function requires reward engineering and limits the power of RL by incurring a possible sub-optimal expected behavior that the RL agent learns.

In Hybrid Reward Architecture (HRA), Van Seijen et al. propose to decompose the reward function into multiple reward components and learn a corresponding value function for each reward component [24]. This reduces the high-complexity of the value function and hence can represent the value function using low-dimensional representation, e.g.,

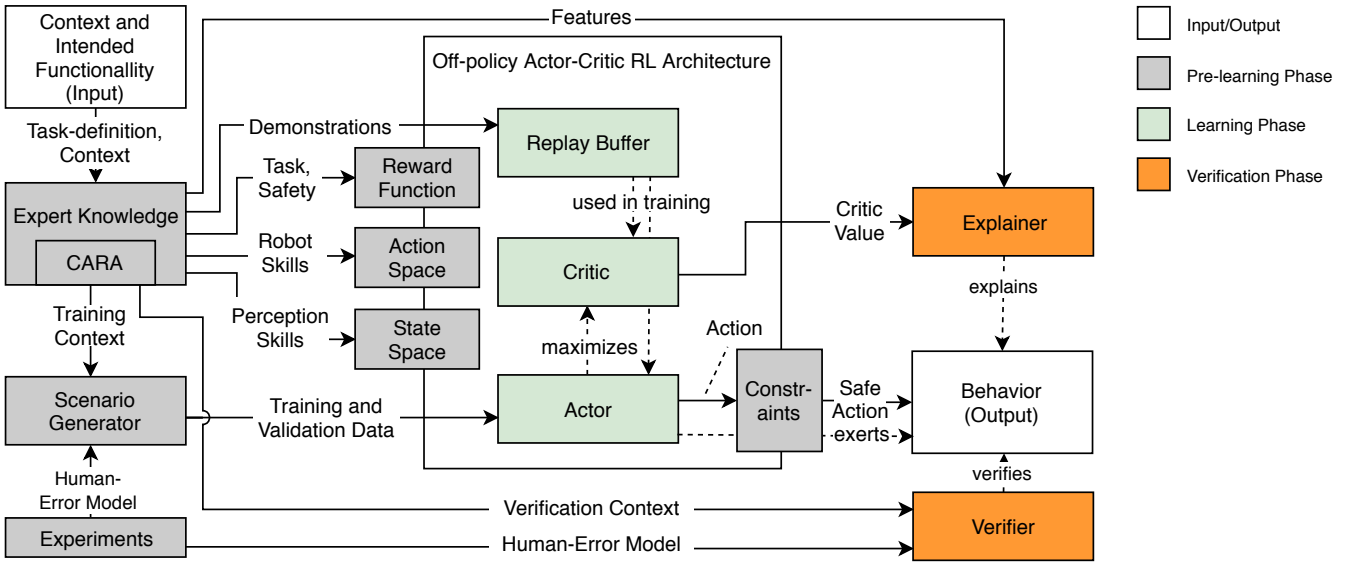

Fig. 1. Core components of the proposed deep RL framework for safe and productive HRC

tabular representation instead of deep NNs. However, the positive outcomes of HRA are proven only with a discrete action space [24]. In this paper, we evaluate HRA on continuous action space. Moreover, the manual decomposition of the reward function is based on the respective task and requires an environment with rich features. In Section IV, we evaluate both sparse rewards and structured sparse rewards for encoding task and safety requirements.

Action Space: HRC scenarios include complex interactions between robots and humans in 6D (robot's end-effector pose). As such, the action space is continuous and makes the RL agents take long time to learn (millions of time-steps if at all feasible) even when using a robust exploration-exploitation algorithm [25]. To limit the action space, recent work learns on existing Inverse Kinematics (IK) solver instead of learning directly on the joint angles of the robot [11]. This reduces the dimensions of actions from 6 or 7 dimensions in case of 6 and 7 degrees of freedom (DoF) robot arms, to 3 dimensions considering only actions in Cartesian space (Δx, Δy, and Δz) with fixed orientations. In addition, learning on existing skills produces more robust output behaviors and reduces the probability of unintended behaviors. Meanwhile, learning on existing general robot skills brings the possibility of transferrability of learnt behavior between different types of robots (discussed in Section IV).

State Space: Similar to the continuous action-space in robotics, sensors provide raw data such as forces on robot end-effector (e.g., force sensor), pixels (e.g., camera sensor), distances (e.g., sensor skin), etc., making the state-space continuous as well. For continuous state-spaces, deep NNs prove to be robust in learning a low-dimensional representation (from training data, e.g., for detecting objects) and even mapping states to output control actions [26]. In the proposed framework, we differentiate between perception (e.g., object detection and localization) and control (e.g., point-to-point motion) in order to introduce more structure in the learning architecture instead of complete end-to-end learning [27]. Thus, in Section IV, we use abstract features collected from simulation environments, such as obstacle relative positions instead of camera pixels, as input to deep NNs to reduce the input-dimensions. In future work, raw sensory data and perception skills will be considered.

Constraints: Negative reward functions are soft penalties that cannot guarantee the safety of the output behavior. Safe RL considers maximizing the expected return while satisfying safety constraints [28]. Safety constraints ensure reaching the goal safely by constraining the exploration space and exploiting actions in the constrained space. Pham et al. present *OptLayer* [29], which augments the NN architecture (of the policy) with an additional optimization layer. OptLayer takes the NN predictions as input and outputs the closest safe actions if the predictions violate constraints, e.g., ensures that predictions do not send the robots end-effector outside of a defined space [30]. Dalal et al. propose another NN architecture with a *Safety Layer* [21], which offers an analytical solution in a state-wise action correction formulation. In Safety Layer, the physical quantities are bound by corresponding linearly approximated state-wise constraints and are exploited for avoiding unexpected and unsafe actions. Compared to OptLayer, which requires expertise knowledge and manual definition, Safety Layer offers an analytical optimization solution and a linear model that can learn the constraints itself and predict the change in the safety-related physical quantities over a single time-step. Both of these approaches require one-time initial training in pre-learning phase.

The advantage of the aforementioned methods is their independence on the RL algorithms since the introduced constraints are applied directly on the policy predictions,

which makes them easily compatible with various algorithms and still exploiting the computational advantage of NNs. Another approach considers using Bayesian optimization layer [31], which is however mostly applicable to low-dimensional problems due to the computational complexity. Although constraining the predictions can improve training efficiency, limiting the exploration of the action space can lead to sub-optimal behaviours. Another limitation is the complicity of the environment dynamics model and the bias coming from the optimization formulation and domain expertise. In addition, formalizing safety constraints in a computationally efficient way (solvable in polynomial time with a closed-form solution) is complex if at all feasible, or computationally expensive and inefficient.

Scenario Generator: The Scenario Generator uses the regions of hazards generated by CARA, and the tasks that the robot and human have to execute, to generate the training and validation data including error-free and erroneous human motion. The tasks are modelled as such they include the possible generalization requirements, e.g., in point-to-point motion tasks the robot has to reach all the points in the work-place with specific orientations.

Human-Error Experiments: Currently, the generated human motion is completely random, which under-represents possible hazardous scenarios. Indeed, the humans' uncertain behaviour, e.g., due to fatigue or temporal emotional disturbance [32], is one main hurdle in deploying safe HRC systems and correctly quantifying risks. In future work, we will investigate the influencing factors of the human worker on the safe state of the HRC system. Accordingly, we will create a human model that can be used to train and verify the learnt policy, and quantify the risks of using the policy.

B. Learning Phase

The learning phase consists of general RL algorithms and architectures, which require the reward function, state-space, action-space and the training context defined in the pre-learning phase. Fig. 1 shows the deep RL framework for HRC with off-policy actor-critic architecture [33] as an example. Deep Deterministic Policy-Gradient (DDPG) [12] has an actor-critic structure, in which the actor's goal is to learn an optimum policy

$$\pi^{\star}(s_t) = \arg\max_{a_t} Q^{\star}(s_t, a_t) \ , \tag{3}$$

by selecting actions which maximize the expected return (in Eq. 1) estimated by the critic

$$Q^{\star}(s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1}\sim P} \left[r + \gamma \max_{a_{t+1}} Q^{\star}(s_{t+1}, a_{t+1}) \right] \ , \tag{4}$$

following bellman equations [18]. In DDPG, the policy and the action-value functions are represented by parameterized NNs, i.e. $\pi(s_t;\theta_a)$ and $Q(s_t, a_t;\theta_c)$, where $\theta(\cdot)$ represents the weights of the NNs, to generalize for continuous state-action spaces [12]. The Replay Buffer stores past trajectories and provides samples of them to perform gradient updates of $\theta(\cdot)$, which increases the learning efficiency. Although we use off-policy and actor-critic structure here as an example, the framework can accept varieties of RL algorithms. Indeed, we compare the aforementioned off-policy algorithms with on-policy ones in Section IV.

C. Verification Phase

Before deployment in industry, the learnt policy has to be verified and safety-assured. The safety-assurance includes realistically quantifying the risk of the policy and in the best case also explaining the behavior exerted by the policy.

Verifier: The main aim of the Verifier is to verify the intended functionality and the generalization capability of the learned behavior, and quantify the risk arising from using the learnt behavior. ISO-12100:2010 - Safety of machinery, provides the process steps for quantifying the risks, which requires the probability of occurrence of the hazard, frequency of human exposure to the hazard, possibility of human avoiding the hazard and severity of the hazard on the human. The probability of occurrence of the hazard relates to the probability of the scenario that leads to the hazard, which includes human-error. Therefore, future work will consider fitting a human-error model from experiments in order to verify the learnt behavior and quantify its risks.

Explainer: In order to provide safety guarantees, the exerted behaviors need to be explained. In Fig. 1, the explainer's main function is to represent the actor's decisions in a comprehensible representation rather than the blackbox representation of NNs [34]. The explainer is connected to the actor, since the actor implicitly controls the behaviors output by the policy. Therefore, it is easier to explain the actor (source of behavior) rather than explaining the behavior itself exerted by the policy. In this work, we consider HRA due to the simplified representation HRA promises [24] and future work will further provide explanation methods of the HRA's representations.

IV. Evaluations

In this section, we evaluate the learning-phase of the framework on a selected task using different deep RL algorithms on different robots. MuJoCo physics simulator [35] and extensions of the Gym environment [36], [37] are used.

A. Experiment Setup

1) Task: The task is to reach a goal position while avoiding collision with obstacles. The RL agent is allowed to reach each goal position within a maximum trajectory length T, with T set to 100 time-steps.

2) Algorithms under Evaluation: We compare different state-of-the-art deep RL algorithms, namely, Hindsight experience Replay (HER) [11] combined with DDPG [12], HRA [24] combined with HER (HRA-HER) and Proximal Policy Optimization (PPO) [38]. HER and HRA are using DDPG which is off-policy, while PPO is on-policy. HER and PPO use the unstructured reward function

$$r = -\alpha_1 \cdot [d(x_{\text{target}}, x_{\text{robot}}) > \delta] - \alpha_2 \cdot [\max(f_1, ..., f_n) > 0] \ , \tag{5}$$

where $d(\cdot)$ represents the Euclidean distance in cm, x_{target} and x_{robot} are the positions of target and the robot's end-effector respectively, δ is a threshold distance, f_i is the contact force measured by the robot due to hitting obstacle_i and α_i is a trade-off co-efficient between the risks and benefits from reaching the goal and avoiding collisions. The conditions, such as $(f_i > 0)$, are evaluated either to 0 or 1 when false or true, respectively. As such, the reward function is sparse, e.g., outputting $-\alpha_1$ in case of no collisions and not reaching the goal.

On the other hand, HRA-HER has two variants: HRA-2 and HRA-n. HRA-2 uses a structured reward function with one reward component for achieving the task, and one reward component for avoiding collisions with obstacles, according to

$$\vec{r} = \begin{pmatrix} -\alpha_1 \cdot [d(x_{\text{target}}, x_{\text{robot}}) > \delta] \\ -\alpha_2 \cdot [\max(f_1, ..., f_n) > 0] \end{pmatrix} , \tag{6}$$

while HRA-n considers a separate reward component for each obstacle, according to

$$\vec{r} = \begin{pmatrix} -\alpha_1 \cdot [d(x_{\text{target}}, x_{\text{robot}}) > \delta] \\ -\alpha_2 \cdot (f_1 > 0) \\ ... \\ -\alpha_{n+1} \cdot (f_n > 0) \end{pmatrix} . \tag{7}$$

With HRA-HER, we decompose the estimated $Q^\star(s_t, a_t; \theta_c)$, which we hypothesize as simplifying the complexity of the used NN. The proof of this hypothesis is beyond the scope of this paper. However, we expect a better learning efficiency exerted by HRA-HER compared to HER and PPO.

3) Scenarios: The environment setup consists of the robot, workplace, table and box-shaped obstacles or a human obstacle. The robots under test are 7 DoF Fetch robot-arm and 6 DoF UR5 robot-arm. The obstacles are either static (fixed) or dynamic (moving). Up to three obstacles can exist simultaneously in the workplace. The goal position is either static or dynamic, and the orientation of reaching the goal is fixed.

Training: The goal starting position is randomly sampled within the robot's reachability region. For the obstacles, four cases exist: static or dynamic and single or three obstacles. If the obstacle is located randomly in the workplace, it is highly unlikely that the obstacle is in-between the robot and the goal. As such, the training efficiency is rather low. Therefore, the Scenario Generator includes instrumented randomness of the obstacle's position and velocity in the training strategy based on the hazardous region identified by CARA and the sampled goal position. The obstacles starting positions are sampled as follows: 20% of the obstacles are far away from the goal, 40% are within a middle-range and 40% are within a small-range from the goal.

On the other hand, dynamic obstacles are sampled within a $20cm$ border outside the table. The initial velocity of the obstacle is sampled in the direction towards the goal and later can change randomly to go away from or towards the goal position. The velocity of the obstacle can go up to $0.6m/s$. A new obstacle will be sampled as soon as one obstacle has run out of the border of the workplace.

Validation: In validation, the robot's reachable area is divided into a grid mesh, with each point in the mesh corresponding to one goal position. Neighboring goal positions are separated by $5cm$. In case of static obstacles, the starting positions of the obstacles are uniformly distributed in the goal positions grid. In case of dynamic obstacles, the experiment with each goal position is repeated 10 times to allow different obstacles positions and velocities per goal.

4) Metrics: The metric should show not only the task performance but also the safety performance of agents, which reflects the influence of encoding safety requirement in the reward function on RL agents' behaviour. We use *safe success rate* to reflect target positions reached successfully without collisions. Additionally, during the training phase, the average safe success rate from all trajectories per epoch is calculated to reflect the learning efficiency.

B. Results

We evaluate different aspects of the RL agents in the defined validation scenarios. Firstly, we compare the performance changes and model sensitivity of different RL agents w.r.t different hyper-parameter configurations of the reward function, namely, punish weight α_i (defined in Eq. 5, 6 and 7), as shown in Fig. 2-a and 2-c. Accordingly, we state the best performance achieved per RL agent in Table I. Secondly, we use the configurations resulting in the best performance to compare the learning efficiency of the RL agents, as shown in Fig. 2-b and 2-d. Thirdly, we evaluate the transferability of a behavior learnt on Fetch robot (7 DoF) and transferred to UR5 (6 DoF) and vise versa. Finally, we evaluate the best approach on a UR5 with a human model exerting random motion.

1) Sensitivity to Hyper-parameters: In this perspective we test the performance changes of agents against different punish weights, since we use punish weight to encode safety in the reward functions (see Eq. 5, 6 and 7). Fig. 2-a and 2-c show the safe success rate vs. punish weight in fetch and UR5 environments with three static or dynamic obstacles. Results show that all agents are sensitive to the punish weight and robot type. Especially HER success rate starts to collapse with punish weights exceeding 10, compared to more consistent performance of HRA-2 and HRA-n.

Table I summarizes the best performance achieved by each agent additionally with one static or dynamic obstacle. HER, HRA-2 and HRA-n can reach target goal with at least 60% success rate, no matter with static or dynamic and one or three obstacles. Not a single agent outperforms the others in all scenarios. This shows that HRA combined with HER does not improve the safe success rate over HER. Meanwhile, PPO fails in both one and three static obstacle scenarios due to sample inefficiency. Since the dynamic obstacle scenarios are more challenging, we refrained from testing PPO further.

2) Learning Efficiency: Fig. 2-b and 2-d show the learning curve of HER, HRA-2 and HRA-n agents on the same setup. All the three agents reach satisfying safe success

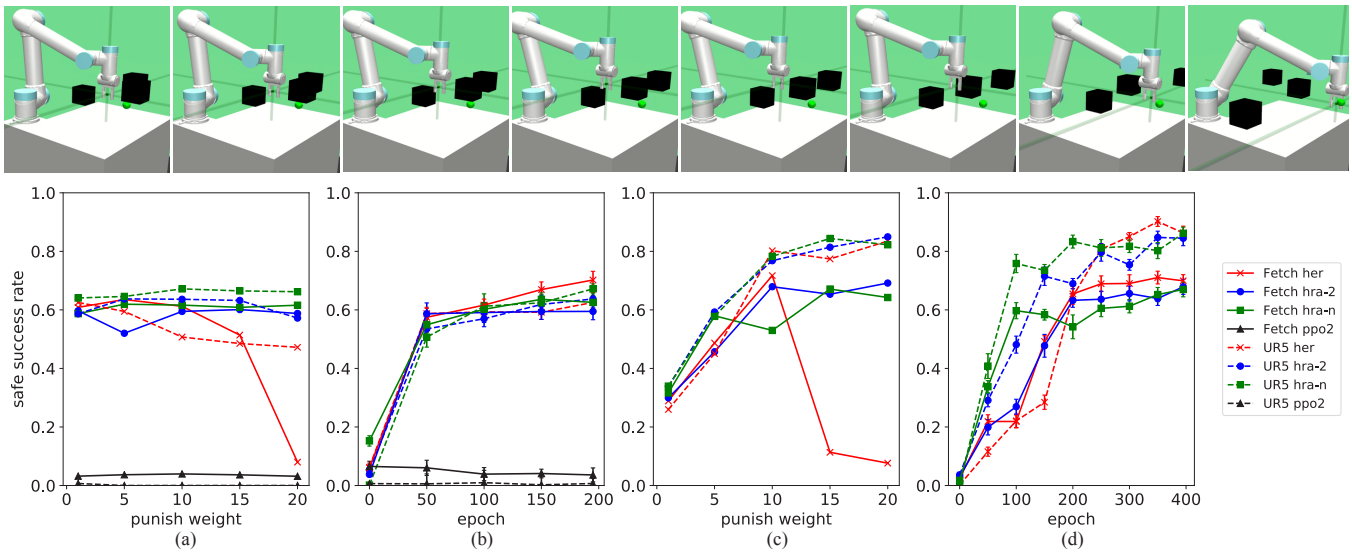

Fig. 2. (Top) UR5 robot safely keeps reaching a moving goal in presence of three dynamic obstacles. (Bottom) (a) Safe success rate of different algorithms against different punish weights, on both Fetch and UR5 robots with three static obstacles. (b) Learning efficiency of different algorithms in the same environments as (a). (c) Safe success rate of different algorithms against different punish weights on both robots with three dynamic obstacles. PPO is not tested since it fails to learn in the static scenario. (d) Learning efficiency of different algorithms in the same environment as (c). Each epoch corresponds to $5,000$ time-steps.

TABLE I

BEST SAFE SUCCESS RATES FOR REACHING STATIC GOALS

Obst.	Robot	# Obst.	HER	HRA-2	HRA-n	PPO
Static	Fetch	1	**81.6%**	75.5%	79.4%	6.7%
		3	**63.5%**	60.1%	61.9%	4.0%
	UR5	1	85.5%	84.9%	**86.3%**	2.0%
		3	62.5%	63.7%	**67.2%**	0.7%
Dynamic	Fetch	1	81.1%	78.3%	**82.3%**	-
		3	**71.7%**	69.2%	67.1%	-
	UR5	1	**73.4%**	71.0%	67.1%	-
		3	83.5%	**85.0%**	84.4%	-

rate after 300 epochs. Fig. 2-d shows that HRA-2 and HRA-n converge faster than HER in environments with dynamic obstacles, which supports the hypothesis mentioned in Section IV-A.2.

3) Transferability: Since we train the RL agents to output only the Cartesian positions to an existing IK solver, the trained agent should gain similar performance on different types of robots. However, results show that an agent trained on UR5 and executed on Fetch achieves a performance degradation from 85.0% safe success rate on UR5 to 52.0% safe success rate on Fetch. An agent trained on Fetch and executed on UR5 achieves performance degradation from 71.7% on Fetch to 64.1% on UR5. The performance degradation comes from the different shapes and DoF of the robots encoded in the agents' latent space. It can be noticed that training on a redundant robot and executing the trained policy on a non-redundant robot results in a better transferrability.

4) Human Model: We test the HER and HRA-2 agents in a similar setup, however, with a human model instead of box-shaped obstacles. The punish weight is fixed to $\alpha_2 = 1$ and static goals are sampled near to the human model. HER achieves 53.1% safe success rate, while HRA-2 achieves 91.4% safe success rate. Currently, the human motion is completely random. Thus, this experiment is only to show the capability of the agents to generalize to more complex obstacles and is not aimed at a performance comparison.

V. CONCLUSION

Safety in HRC is a bottleneck to productivity due to over-emphasized safety constraints, and expensive risk assessment process. With robots being the main source of hazards, we proposed a novel framework that uses deep RL to teach robots safe and intelligent behavior. The framework maps HRC task and safety requirements into a RL setting, and uses preliminary work in the HRC field, namely, CARA to encode the context of applicability in the respective RL setting. In the evaluations, we compared different RL agents on learning and solving the task of safe point-to-point robot arm motion. Results showed that off-policy DDPG combined with HER and HRA are more sample-efficient compared to on-policy PPO. Moreover, HRA combined with HER did not largely improve the safety performance over using HER alone. However, HRA is hypothesised to reduce the complexity of the representation of the learnt behavior, that can simplify the Verifier and Explainer. Overall, off-policy agents achieved up to $\approx 85\%$ safe motion in challenging scenarios. The results are encouraging and expected to be further improved when enhancing individual components of the framework. As future work, we are planning to introduce hard constraints instead of soft penalties to further increase the safe success rate. Another foreseen improvement is to include realistic human models for training and evaluating the RL agents, and addressing the verification and explanation for transferring the learnt behavior from simulation to a real-world scenario.

REFERENCES

[1] B. Vogel-Heuser, T. Bauernhansl, and M. Ten Hompel, "Handbuch industrie 4.0 bd. 4," *Allgemeine Grundlagen*, vol. 2, 2017.

[2] F. M. Remiel, A. Cauvin, and A. Ferrarini, "A production system reconfiguration model based on repair approach," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2014, pp. 344–351.

[3] P. Liggesmeyer and M. Trapp, "Safety: Herausforderungen und lösungsansätze," in *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Springer, 2014, pp. 433–450.

[4] R. Awad, M. Fechter, and J. van Heerden, "Integrated risk assessment and safety consideration during design of HRC workplaces," in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2017.

[5] S. Stanev, R. Awad, M. Prieur, W. Walla, S. Pölz, and J. Ovtcharova, "Production-oriented product validation method as support for the reuse of production lines in the automotive industry," in *3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV)*. Herbert Utz Verlag, 2009.

[6] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1–4, 2017.

[7] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," *arXiv preprint arXiv:1806.00109*, 2018.

[8] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE, 2013, pp. 100–107.

[9] P. A. Lasota, T. Fong, J. A. Shah *et al.*, "A survey of methods for safe human-robot interaction," *Foundations and Trends® in Robotics*, vol. 5, no. 4, pp. 261–349, 2017.

[10] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[11] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] D. A. Cieslak and N. V. Chawla, "A framework for monitoring classifiers performance: when and why failure occurs?" *Knowledge and Information Systems*, vol. 18, no. 1, pp. 83–108, 2009.

[15] K. R. Varshney, "Engineering safety in machine learning," in *2016 Information Theory and Applications Workshop (ITA)*. IEEE, 2016, pp. 1–5.

[16] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

[17] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *arXiv preprint arXiv:1905.04223*, 2019.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] A. A. Markov and N. M. Nagorny, *The Theory of Algorithms*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[20] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of iso 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435*, 2017.

[21] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[22] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. JMLR.org, 2017, pp. 22–31.

[23] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 143, 2018.

[24] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 5392–5402.

[25] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[27] P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Perez, "Differentiable algorithm networks for composable robot learning," in *Robotics: Science and Systems (RSS)*, 2019. [Online]. Available: http://lis.csail.mit.edu/pubs/karkus-rss19.pdf

[28] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[29] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[30] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[31] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *ArXiv*, vol. abs/1602.04450, 2016.

[32] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "Modeling operator behavior in the safety analysis of collaborative robotic applications," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2017, pp. 89–104.

[33] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[34] N. Schaaf and M. F. Huber, "Enhancing decision tree based interpretation of deep neural networks through L1-Orthogonal Regularization," *arXiv preprint arXiv:1904.05394*, 2017.

[35] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[36] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[37] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark," in *Conference on Robot Learning*, 2018.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.