Safe Resetless Reinforcement Learning: Enhancing Training Autonomy with Risk-Averse Agents

Tristan Gottwald, Maximilian Schier, and Bodo Rosenhahn

Institute for Information Processing, L3S, Leibniz University Hannover, Germany
{gottwald,schier,rosenhahn}@tnt.uni-hannover.de

Abstract. Training Reinforcement Learning agents directly in any real-world environment remains difficult, as such scenarios entail the risk of damaging the training setup or violating other safety constraints. The training process itself further requires extensive human supervision and intervention to reset the environment after each episode. Thus, we propose an innovative Safe Reinforcement Learning framework that combines Safe and Resetless RL to autonomously reset environments, while also reducing the number of safety constraint violations. In this context, we develop a novel risk-averse RL agent suitable for stringent safety constraints by combining Safe RL, Distributional RL, and Randomized Ensembled Double Q-Learning. Experiments conducted in a novel mobile robotics scenario indicate that our Safe Resetless RL framework reduces the number of human interactions required during training compared to state-of-the-art methods, filling a gap in current problem formulations and enhancing the autonomy of RL training processes in real-world settings.

Keywords: Safe Reinforcement Learning · Distributional Reinforcement Learning · Resetless Reinforcement Learning · Autonomous Navigation · Mobile Service Robotics

1 Introduction

As the capabilities of Reinforcement Learning agents have increased in recent years [8, 11, 19, 37], interest in deploying RL agents to the real world has also seen a sharp rise [17, 18, 21, 44]. However, most RL agents are trained in simulated environments [4, 6, 12, 20, 28, 34], which are unable to capture the dynamics and the full complexity of the real world scenario that they are trying to simulate [32]. Therefore, deploying the agent to the real world often requires further changes in order to adapt to the differences between the domains. Sim2Real approaches [27, 43] try to bridge this gap between the simulation and the real world. However, these methods do not address the root of the problem: the simulated training process. While training the agent directly in its real world domain solves many of the problems associated with the domain gap, it also leads to other previously irrelevant problems [13]. The two main problems of moving

the training process to the real world include the questions of how to reset the training environment and how to ensure the safety of the training process. The former problem arises from the fact that in simulated environments, there is typically a meta-action reset available. However, in the real world this meta-action is equivalent to extrinsic interventions, *e.g.* manually driving the robot back to its starting point. This makes the training process less autonomous, as constant human supervision of the training process is needed to ensure the agent can start its next episode. The second problem has to do with the fact that simulated environments do not reflect the permanent and fatal consequences an agent's actions might have. For example, in the real world a robot might need to be repaired after executing an undesired action, which again amounts to human intervention being required. Thus, the agent needs to be additionally incentivized to minimize the number of times an action leading to an unsafe state is selected. Most existing agents avoid entering these unsafe states only because they do not maximize the agent's expected return, not because of the state's fatal consequences. Training a RL agent in the real world thus requires the definition of some safety constraints which the agent should respect to avoid damages to the experimental setup [17]. Our main contributions are:

- We present a novel training framework for Safe Resetless Reinforcement Learning (SRRL) in real-world environments that reduces the number of human interactions necessary.
- We introduce our new risk-averse RL agent called DREDQ, which converges faster to higher returns than state-of-the-art approaches.
- Our proposed framework reduces the number of human interactions compared to existing Resetless RL approaches.
- In addition, we present a novel mobile robotics environment with realistic sensor and kinematic simulation.
- The code is available at `https://github.com/tgottwald/srrl`.

2 Related Works

Our method combines the two research fields of Safe Reinforcement Learning and Resetless Reinforcement Learning. We build upon previous works published in the respective fields and adapt them to our holistic approach.

Safe Reinforcement Learning Most state-of-the-art Safe RL agents [3, 28, 38, 39, 41] are based on on-policy algorithms, which do not provide the sample efficiency needed for real-world experiments. Yang *et al.* utilize a risk-averse distributional safety critic in their off-policy agent [42]. Their approach however adheres to the classical separation of reward and cost signal in Safe RL and thus uses separate critics for both. Consequently, only their cost critic uses distributional RL, while the distributional critic in our approach considers both cost and reward with respect to the distortion risk-measure. Recently, Sootla *et al.* proposed Sautè RL [36], which makes Safe RL environments solvable with any standard RL agents by wrapping the environment. The wrapper introduces a safety cost

budget, which encodes the cost threshold and is concatenated to the agent's observation space. Every time the agent executes an action that incurs cost, the remaining safety budget is reduced accordingly. If the safety budget is used up, the reward returned to the agent is replaced with -1. Thus, the agent can learn a causality between cost incurring actions and negative rewards and therefore is able to avoid safety constraint violations. A different approach to Safe RL rooted in control theory, is to use control barrier functions. However, these methods often need to be pretrained with offline data [9, 29] or require the training of a large auxiliary generative model [40].

Resetless Reinforcement Learning In [15] Eysenbach *et al*. propose their "Leave No Trace" (LNT) approach which formalizes a Resetless RL structure many other researchers have built upon [22, 24, 44]. LNT introduces a second agent called the *reset* agent, which is responsible for resetting the environment to the initial state distribution after the regular agent (now referred to as the *forward* agent) has concluded its episode. Both agents use the same action and observation space. However, the reset agent's policy is independent of the one of the forward agent. Thus, the structure of the framework alternates between the forward agent trying to solve the task and the reset agent trying to reset the environment. Additionally, Eysenbach *et al*. added an option for the reset agent to trigger an *early abort* during the forward episode if the reset agent's Q-value is below a certain threshold, to prevent the forward agent from entering states the reset agent cannot (yet) reset from. The main drawback of LNT is that it requires the user to define an additional reset reward function for the reset agent to work. Expressing the initial state distribution in terms of a reward function might prove to be difficult and removes some autonomy from the training process. Therefore, Kim *et al*. replace the reset agent with an example-based agent in [22].

3 Preliminaries

The Reinforcement Learning problem is formalized using a five-element tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, which is known as a Markov Decision Process (MDP) [5]. A MDP consists of a continuous space of states $\mathcal{S}$ in which the agent can influence the currently active state by applying an action from action space $\mathcal{A}$ according to the dynamics model $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. The dynamics model specifies the transition probability $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ to the next state $\boldsymbol{s}_{t+1} \in \mathcal{S}$ given a current state $\boldsymbol{s}_t \in \mathcal{S}$ and action $\boldsymbol{a}_t \in \mathcal{A}$. Typically, there also exists an initial state distribution $\rho_0 : \mathcal{P}(\mathcal{S})$ in which the agent starts. A consecutive sequence of states and actions is commonly referred to as a trajectory τ. Furthermore, the agent receives a reward after each transition according to the reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ based on the current state, the selected action and the successor state. The rewards collected during an N-transition long trajectory are expressed as the discounted future return $G(\tau) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^N r_{t+N}$. Here, the discount factor γ is typically chosen to be less than 1 so that high rewards in the distant future are less relevant to the agent's decision in the present. The

agent has to find an optimal policy π^* according to the following equation:

$$\pi^* = \arg\max_{\pi} J_r(\pi), \text{ with } J_r(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi}[G(\tau)]\,. \tag{1}$$

Safe Reinforcement Learning extends the classical MDP definition to a Constrained Markov Decision Process (CMDP) [2,28] by additionally including one or more cost functions $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The discounted future cost $C(\tau)$ can be defined analogous to $G(\tau)$. However, a policy is only valid if the accumulated cost of the corresponding trajectory are below a cost threshold d. This leads to Eq. (1) now being constrained to a set of safe policies $\mathit{\Pi}_C$, thus resulting in

$$\pi^* = \arg\max_{\pi\in\mathit{\Pi}_C} J_r(\pi), \text{ where } \mathit{\Pi}_C = \{\pi : J_c(\pi) \leq d\}, \text{ with } J_c(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi}[C(\tau)]\,. \tag{2}$$

4 Proposed Method

Our approach to Safe Resetless Reinforcement Learning (SRRL) is based on the fact that both Safe and Resetless RL have the common motivation in enabling RL agents to be trained in the real world. In our framework, we will only consider *fatal* safety constraints. These type of constraint violations leads to a catastrophic, non-reversible loss of the system controlled by the agent. Therefore, any actions selected and observations made by the agent after violating the safety constraints are useless or will result in the same observation, respectively. *Non-fatal* safety constraints which have been typically used in Safe RL [2,28,42] leave the system intact and controllable after safety violations. The agent is able to keep observing the environment and selecting actions.

4.1 Safe Reinforcement Learning Agent

Since the safety constraints we assume in this setting are immediately violated if the agent enters a set of unsafe states and thus the trade-off between reward and cost is simple (see Sec. 5), we will only consider Safe RL environments that have been Sautèd. We combine the Sautéd environment with our novel distributional RL agent called DREDQ, which combines REDQ [8] and IQN [10] into a single algorithm. Assuming a Sautéd environment furthermore has the advantage that DREDQ is not restricted to Safe RL problems only, but might also be applied to environments where undesired states are indicated by the reward function. IQN uses Quantile Regression [23], where the inverse of the Cumulative Density Function F_G of the generic return distribution G is learned by mapping a H-element vector of quantile fractions $\boldsymbol{\iota} \in [0,1]$ to their corresponding quantile values $G^{\boldsymbol{\iota}}(\boldsymbol{s},\boldsymbol{a}) = F_G^{-1}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\iota})$. The latter is also known as the quantile function. In the case of $G(\boldsymbol{s},\boldsymbol{a})$, the quantile values corresponds to the discounted return if the trajectory starts in state $\boldsymbol{s}$ with action $\boldsymbol{a}$ and follows the policy after that. Please note that in contrast to prior publications on the topic of Distributional RL [10,11,42], we will use $\boldsymbol{\iota}$ to denote the quantiles instead of τ, as the latter

is already reserved for trajectories. IQN and most agents based on it [10, 25] do not use ensembles like the N critics found in the randomly sampled ensemble of REDQ. Therefore, DREDQ presents a new approach to utilize ensembles of any size with a distributional agent. Our TD error term is based on Ma *et al.*'s formulation for Distributional SAC (DSAC) [25], but we generalized it to be compatible with REDQ. For the critic, we calculate the conservative target in the element-wise TD error $\delta_t^{\iota_i,\iota'_j}$ in Eq. (3). Here, we use the smallest predicted quantile value of all ensemble networks as the target.

$$\delta_t^{\iota_i,\iota'_j} = r_t + \gamma \min_{m \in \mathcal{M}} \left[G_{\theta_m}^{\iota'_j}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) - \alpha \log \pi_\phi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1}) \right] - G_{\theta_m}^{\iota_i}(\boldsymbol{s}_t, \boldsymbol{a}_t) \quad (3)$$

The IQN-loss itself remains unchanged to [10] and makes use of the Huber quantile loss $\varrho_{\iota_i}^{\kappa}$ [11], where κ is the boundary at which the L1 loss transitions into the L2 loss:

$$\mathcal{L}_{G^{\boldsymbol{\iota},\boldsymbol{\iota}'}}(\theta_l) = \mathop{\mathbb{E}}_{\substack{(\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+1})\sim\mathcal{D} \\ \boldsymbol{a}_{t+1}\sim\pi(\cdot|\boldsymbol{s}_{t+1})}} \left[\frac{1}{H'} \sum_{i=1}^{H} \sum_{j=1}^{H'} \varrho_{\iota_i}^{\kappa}(\delta_t^{\iota_i,\iota'_j}) \right] . \quad (4)$$

The hyperparameters H and H' denote the number of quantile fractions to sample for the current and the successor state respectively. For DREDQ's policy loss, the mean of state-action values used in REDQ is replaced by a distorted expectation Υ, resulting in the updated policy loss:

$$\mathcal{L}_\pi(\phi) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}_t\sim\mathcal{D} \\ \boldsymbol{a}_t\sim\pi_\phi(\cdot|\boldsymbol{s}_t)}} \left[\alpha \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t) - \Upsilon(\boldsymbol{s}_t, \boldsymbol{a}_t, \eta) \right] . \quad (5)$$

In DREDQ, Υ corresponds to the Conditional Value at Risk [30] of the lower tail (see Eq. (6)), but any risk distortion measure can theoretically be used here. Choosing CVaR has the advantage of the policy becoming risk-averse, as only the worst $\eta\%$ of the distribution is considered.

$$\Upsilon(\boldsymbol{s}_t, \boldsymbol{a}_t, \eta) = \mathbb{E}\left[G'|G' \leq F_G^{-1}(\eta)\right], \text{ with } G' = \frac{1}{M} \sum_{m=1}^{M} G_{\theta_m}^{\tilde{\boldsymbol{\iota}}} . \quad (6)$$

To make the distorted expectation compatible with the REDQ's M critics, we calculate Υ using the mean quantile value over all available critics G'. Here, K different quantile fractions $\tilde{\boldsymbol{\iota}}$ sampled from the region of interest are used. A pseudocode version of our algorithm can be found in Algorithm 1. The UTD-ratio L specifies how often to update the networks for each collected transition. In practice, the actual start of the training is preceded by a warm-up phase in which transitions are collected using a random policy to ensure that enough data is available.

Algorithm 1 Distributional Randomized Ensembled Double Q-Learning

Network Parameters $\theta_0, \dots, \theta_N, \phi$
Hyperparameters: $\lambda_Q, \lambda_\pi, \lambda_\alpha, \vartheta, \mathcal{H}, L, M, N, H, H', K, \eta, \kappa$
$\overline{\theta_0} \leftarrow \theta_0, \dots, \overline{\theta_N} \leftarrow \theta_N$ ▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize replay buffer
for each iteration **do**
  $\boldsymbol{a}_t \sim \pi_\phi(\cdot|\boldsymbol{s}_t)$ ▷ Sample action from policy
  $\boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$ ▷ Sample successor state from environment
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}), \boldsymbol{s}_{t+1})\}$ ▷ Store transition in replay buffer
  for L update steps **do**
    $\boldsymbol{\iota} \sim U([0,1])$, $\boldsymbol{\iota}' \sim U([0,1])$, $\tilde{\boldsymbol{\iota}} \sim U([0,\eta])$ ▷ Sample quantile fractions
    $\mathcal{D}\prime \sim \mathcal{D}$ ▷ Sample mini-batch from replay buffer
    $\mathcal{M} \sim \{1, \dots, N\}$ ▷ Sample M indices without replacement
    $\theta_n \leftarrow \theta_n - \lambda_Q \hat{\nabla}_{\theta_i} \mathcal{L}_G^{\boldsymbol{\iota},\boldsymbol{\iota}'}(\theta_n)$ for $n \in \{1, \dots, N\}$ ▷ Update critics (Eq. (4))
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi \mathcal{L}_\pi(\phi)$ ▷ Update actor (Eq. (5))
    $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha \mathcal{L}(\alpha)$ ▷ Adjust entropy temperature
    $\overline{\theta_n} \leftarrow \vartheta\theta_n + (1-\vartheta)\overline{\theta_n}$ for $n \in \{1, \dots, N\}$ ▷ Update target networks
  end for
end for

4.2 Example-based Reinforcement Learning Agent

Previous example-based agents like [14,16] cannot be used in the context of Safe RL, as they are unable to learn about the possibly negative consequences of their actions because of the agent only having access to the observations. For our example-based reset agent, we assume that the agent has access to a signal indicating whether the agent has caused a safety constraint violation, making it the first safe example-based agent. Our method can be seen as attempting to embed a function proportional to a hypothetical reward function directly into the Q-function. The agent receives a set of success examples $\mathcal{S}^*$, which consists of states where the task has been completed. The first part of Eq. (7) represents the critic being trained to predict an arbitrary positive value $\mathcal{K} \in \mathbb{R}_{>0}$ for states randomly sampled from $\mathcal{S}^*$. The action needed to query the current Q-functions approximation is taken from the current policy estimate.

$$\mathcal{L}_Q(\theta) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}^* \sim \mathcal{S}^*, \\ \boldsymbol{a}^* \sim \pi_\phi(\cdot|\boldsymbol{s}^*)}} \left[\frac{1}{2} \left(Q_\theta(\boldsymbol{s}^*, \boldsymbol{a}^*) - \mathcal{K} \right)^2 \right] + \mathop{\mathbb{E}}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - \left(\gamma \min_{j \in \{0,1\}} \left[\mathop{\mathbb{E}}_{\boldsymbol{s}_{t+1} \sim p} \left[V_{\overline{\theta_j}}(\boldsymbol{s}_{t+1}) \right] \right] \right) \right)^2 \right] \tag{7}$$

The second part of the equation is identical to the regular critic loss term found in SAC but without the reward. It is responsible for propagating the discounted state-action value of the success states back through all other states, generating a "value trace" the reset agent can follow to its target state. This kind of approach

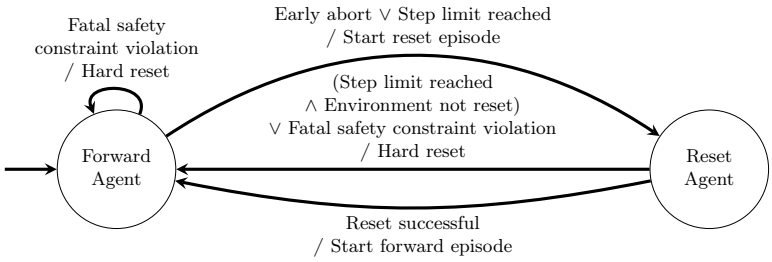

Fig. 1: Dynamics between the forward and reset agent in our Safe Resetless Reinforcement Learning training framework. The reset agent takes over if the forward agent's episode reaches the step limit or if the forward agent would execute an action which causes an early abort. Control is transferred back to the forward agent if the reset agent successfully resets the environment. In this case, the forward agent will start its episode from the state in which the reset agent ended theirs. If either the forward or the reset agent violates a fatal safety constraint, the environment must be reset using a hard reset. Additionally, the reset agent may cause a hard reset if it is unable to reset the environment within its step limit.

requires some success examples states to be visited by the initial trajectories of the agent, as otherwise the second part of Eq. (7) cannot create a connection between the success examples and all other state. However, since we only deploy this type of agent as part of the Resetless Framework, this is guaranteed to be the case as the framework generates an implicit curriculum [15, 22].

4.3 Resetless Reinforcement Learning Structure

Our training framework is based on the Resetless RL framework proposed by Kim *et al.* [22]. We chose this framework over LNT [15] because the use of an example-based reset agent grants greater autonomy and reusability compared to LNT. We have modified the proposed transitions by Kim *et al.* to include additional transitions when the agent causes a fatal safety constraint violation (see Fig. 1). After a violation, the agent requests a hard reset as an additional safety measure to prevent any further potential damage to the experimental setup, since we do not have access to information about the severity of the constraint violation. Like in LNT, the reset agent may trigger an early abort of the forward agents episode if the reset agent's Q-function for the transition returns a value less than the threshold $Q_{\min}$:

$$\mathcal{E} = \{(\boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}) | Q(\boldsymbol{s}, \boldsymbol{a}) < Q_{\min}\}\,. \tag{8}$$

This allows the reset agent to prevent the forward agent from entering states with uncertain state-action values for the reset agent, which the reset agent might not be able to reset from yet. This creates an implicit curriculum for the reset agent, which ensures that the reset agent's starting states will become gradually more

difficult instead of always placing the agent at one of the most difficult states like in the standalone version of the algorithm. Letting the reset agent start in or near its goal distribution ρ_0 in the starting phase of the training, further guarantees that it's Q-function is not sparse, since the reset agent does not have to start from states whose Q-values are not connected to the goal state distribution. To reduce the proportion of environments being used up by the agent, the reset agent's episode also terminate early if the agent has reset the environment by reaching a state that satisfies

$$\mathcal{T} = \{\boldsymbol{s} \in \mathcal{S} | \rho_0(\boldsymbol{s}) > 0\}\,. \tag{9}$$

In contrast to [22] and [15], we do not mark the transition in which the reset agent successfully resets the environment with a `done` flag in the replay buffer. This has to do with the fact that the `done` flag signals the agent not to consider the state-action value for the next state when calculating the TD target (see Eq. (7)). For regular RL algorithms, this makes the TD target identical to the reward if the `done` flag is set. Since our example-based reset agent does not use a reward, the TD target would be set to zero. This encourages the reset agent's policy to stay in states close to the goal states but not to enter them, as the agent cannot accumulate any further rewards once they entered the goal states and the episode terminates.

5 Environment

For our experiments we implemented a new mobile robotics scenario based on Schier *et al.*'s *CarEnv* [35] in which the agent has to drive a mobile platform back to its charging area. Colliding with one of the walls leads to the episode being terminated. We simulate the mobile platform using a bicycle model with rear wheel steering. The robot is controllable using the action space $(\omega_x, \beta_{\text{target}}) \in [-1, 1]^2$, where ω_x is the longitudinal acceleration or deceleration and β_{target} the target steering angle of the robot, which the steering controller tries to reach as fast as possible. The observation space $(m_1, m_2, ..., m_N, v_x, \beta_{\text{curr}})$ consists of N lidar distance measurements, which are concatenated by the robot's odometry in the form of the longitudinal velocity v_x and its current steering angle β_{curr}. All dimensions are again normalized to $[-1, 1]$ with a -1 in a lidar dimension indicating that the ray did not return an echo. The resolution and maximum range of the lidar is fully customizable. We choose $N = 300$ with a maximum range of 10 m, which results in a challenging high-dimensional observation space unmatched by other existing environments and a higher degree of realism.

$$\begin{aligned} r(\boldsymbol{s}, \boldsymbol{a}) &= -\mathbb{1}_{\text{collision}} + h \cdot r_{\text{pose}}(\boldsymbol{s}, \boldsymbol{a}) \\ r_{\text{pose}}(\boldsymbol{s}, \boldsymbol{a}) &= \max\left(0, 1 - \frac{|x - l|}{x_{\max}}\right) \cdot \max\left(0, 1 - \frac{|y - u|}{y_{\max}}\right) \cdot \max\left(0, \cos\left(\theta - \alpha\right)\right) \end{aligned} \tag{10}$$

The reward function depends on whether the robot has collided, its current pose (x, y, θ) and target pose (l, u, α). For h, $x_{\max}$ and $y_{\max}$ the same values as in [35]

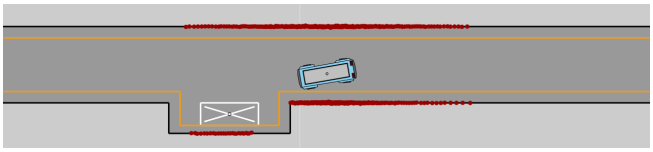

Fig. 2: Screenshot showcasing our environment. The walls limiting the drivable area are represented by *black lines*. The area indicated by the *white floor markings* represents the charging area in which the robot should stop. Furthermore, we visualized the lidar scan available to the agent by marking the lidar echos with *red dots*.

are used. We further want to highlight that the target pose is not accessible to the agent through its observation space like in other environments [27] and the agent instead has to learn to interpret its raw sensor readings. Figure 2 shows the setting of our environment. The environment does not offer a cost function and thus does not formally qualify as a Safe RL environment as defined by Ray *et al*. [28]. However, we argue that this environment (and probably many others) can be seen as a Safe RL environment if we view the environment as a Sautéd [36] version of an underlying Safe RL problem. Assuming a binary cost function returning a cost of k for state-action tuples violating a safety constraint and a Sautèd Safe RL environment with a cost budget of k, a single unsafe action would immediately use up the cost budget and thus make the environment return a negative reward. This behavior is identical to our environment's original reward function. The only difference remaining is that Safe RL environments typically do not terminate episodes after the safety constraint has been violated, as they only assume non-fatal safety constraints. With fatal safety constraints, there is no benefit in continuing the episode after a constraint violation, since the same transitions will be added to the replay buffer over and over again after the violation. This can even have a negative impact on the agent's performance, since the replay buffer is filled with fewer transitions relevant to solving the task.

6 Experiments

We carried out our evaluation using best-practice methods [1]. Thus, all results reported use the interquartile mean and 95% confidence interval based on 10 randomly seeded training runs for each algorithm.

In our distributional critics, we reuse IQN's network architecture [10] with a 512 dimensional embedding and $H = H' = K = 32$. All other critics use a $256 - 256 - 1$ MLP with ReLU activation. For the Huber loss parameter, we determined a value of $\kappa = 0.01$. The safety level η is set to 0.25. $L = 10$, $M = 10$ and $N = 2$ is used for all REDQ and DREDQ runs, while all other agents use the default values $L = 1$, $M = 2$ and $N = 2$ [19]. All the parameters used were determined by grid search unless noted otherwise. The actor is build using a $256-256$ MLP with ReLU as the shared network and separate dense layers for the mean and log-std transforming the output to the action space's dimensionality. All agents use the same default learning rate of 0.0003 for λ_Q, λ_π and λ_α.

Table 1: Interquartile means of the performance metrics for all standalone agents evaluated on our environment after 10^6 training steps. Best results are highlighted in bold and second-best results are underlined. Our DREDQ agent achieves the overall best results, while the classical Safe RL agent causes the most violations.

Agent	Return ↑		Safety Violations ↓		Hard Resets ↓	
	IQM	CI95	IQM	CI95	IQM	CI95
SAC	12.68	[12.24, 13.06]	160.33	[144.66, 182.00]	3431.33	[3421.83, 3445.16]
REDQ	12.95	[12.60, 13.21]	118.50	[101.66, 137.33]	<u>3418.50</u>	[3402.50, 3435.00]
WCSAC	12.95	[12.76, 13.13]	372.00	[337.00, 412.33]	3565.00	[3542.16, 3596.33]
DSAC	<u>12.96</u>	[12.70, 13.23]	**110.66**	[95.00, 121.50]	**3406.16**	[3393.00, 3415.66]
DREDQ (**ours**)	**13.44**	[13.40, 13.47]	<u>112.50</u>	[100.00, 124.50]	3418.66	[3407.50, 3428.66]

6.1 DREDQ Evaluation

In order to make the influence of all of our proposed methods more comprehensible, we will first evaluate our DREDQ agent (Sec. 4.1) as a standalone algorithm before incorporating it in our SRRL framework. Therefore, we will be looking at the performance of DREDQ in a classical RL scenario, where the environment is reset using a reset meta-action. Table 1 shows that all best and second-best results can be either associated with either DSAC or DREDQ. When only considering the final results in Tab. 1, both algorithms reduce the number of safety violations compared to SAC by around 30% and achieve overall similar results with the most significant difference being visible in the episodic return where DREDQ achieves a higher mean episodic return and higher lower 95% confidence interval bound than DSAC's upper confidence bound. Interestingly, the Safe RL agent WCSAC [42] caused by far the most violations, showing that classical Safe RL approaches are unsuited for problems with fatal safety constraints. The benefit of combining the overall safer performance of a distributional agent with the higher convergence speed of REDQ becomes more noticeable when looking at the course of our metrics during the training in Fig. 3. DREDQ already converges after about 5.5×10^5 environment steps in terms of the episodic return achieved, at which point around 69 violations have occurred, reducing the number of violations compared to SAC by 66%. This makes the DREDQ agent best suited for training in a real-world environment as it requires the overall fewest training steps and thus time to learn to solve the task and also causes the fewest number of violations in the meantime. Furthermore, DREDQ remains real-time capable at a sampling rate of 10 Hz.

6.2 SRRL Evaluation

For the evaluation of our Safe Resetless Reinforcement Learning framework (Sec. 4.3), we will compare the performance of our approach to a standalone

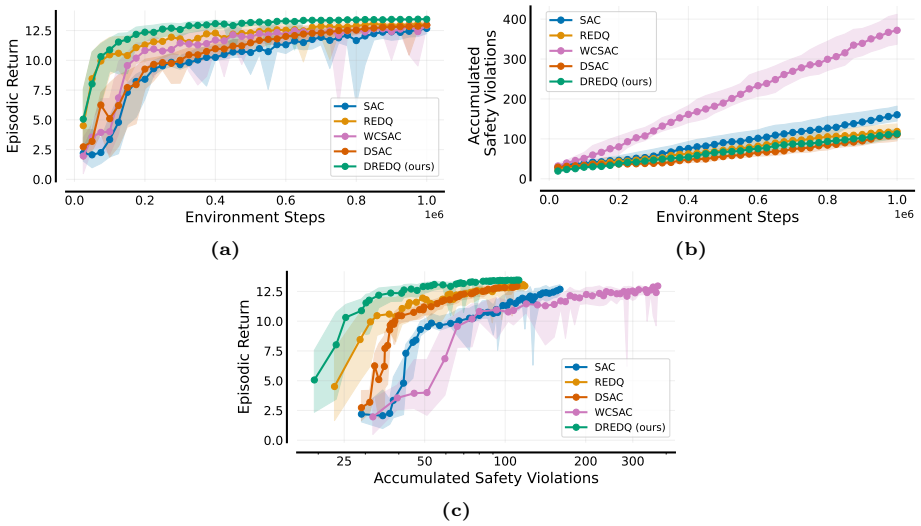

Fig. 3: Course of the episodic return (Fig. 3a) and number of safety violations (Fig. 3b) during a training with 10^6 environment steps for multiple standalone agents. The solid line represents the interquartile mean, while the shaded area indicates the respective 95% confidence interval. Additionally, Figure 3c depicts the episodic return over the interquartile mean of the number of violations during training. Please note that the violation axis is scaled logarithmically for this figure. The higher the return and the lower the number of violations, the better the agent. DREDQ is the fastest to converge to a high episodic return, with one of the lowest total number of safety violations.

SAC agent with external hard resets and Leave No Trace with SAC for both the forward and the reset agent. In contrast to our approach, which only uses examples like [22], LNT requires an additional reset reward function to be defined. For this, we change the target pose in Eq. (10) from the center of the charging bay to the center of the spawn area. For all other pairings, we used our example-based variant of SAC (EBSAC) from Sec. 4.2. All methods were trained using a total of 1.5×10^6 environment steps. Looking at the results in Tab. 2, the pairing of a vanilla SAC agent with our EBSAC agent appears to perform the best from a safety violation and hard reset perspective. However, these values need to be considered with respect to the forward return, where the pairing achieves significantly less return than with DREDQ as a forward agent. The reason behind the lower number of safety violations and hard resets of the other pairings can be found when looking at the final deviation from the target pose in Tab. 3. DREDQ is the only forward agent that consistently reaches the target pose in our SRRL framework. All other pairing achieve higher residuals in terms of their x and y deviation. The latter indicates that the forward agent has not learned to drive the robot into the charging bay yet and ends its episodes with the robot still inside the main corridor. This provides the reset agent with a less complex and risky initial state compared to the agent taking over with the robot inside

Table 2: Interquartile mean and 95% confidence interval for the resetless performance metrics for different forward and reset agent combinations after 1.5×10^6 environment steps. The best result in each category is emphasized in bold and the second-best underlined. DREDQ is the only forward to achieve a high enough return to complete the task at this point in training.

Forward Agent	Reset Agent	Forward Return ↑		Forward Violations ↓		Reset Violations ↓		Hard Resets ↓	
		IQM	CI95	IQM	CI95	IQM	CI95	IQM	CI95
SAC (External Resets)	-	<u>12.92</u>	[10.63, 13.18]	234.33	[192.00, 279.16]	-	-	5159.50	[5129.16, 5196.16]
SAC (LNT)	SAC	11.75	[9.44, 12.43]	102.16	[81.00, 151.16]	<u>208.83</u>	[162.33, 254.16]	1251.00	[1001.33, 1584.50]
SAC	EBSAC	11.46	[11.04, 11.84]	**76.20**	[61.20, 97.00]	**157.50**	[136.00, 188.10]	**830.10**	[757.30, 929.30]
DREDQ (**ours**)	EBSAC	**12.98**	[12.75, 13.15]	<u>94.00</u>	[47.33, 118.66]	244.66	[162.00, 323.66]	<u>1162.33</u>	[821.33, 1527.66]

Table 3: IQM and 95% CI for the forward agent's deviation from the target pose after 1.5×10^6 training steps. DREDQ is the only agent to achieve $< 10\,\text{cm}$ for Δx and Δy.

Forward Agent	Reset Agent	Forward $\Delta x\,[m]$ ↓		Forward $\Delta y\,[m]$ ↓		Forward $\Delta\theta\,[°]$ ↓		Forward $\Delta v\,[m/s]$ ↓	
		IQM	CI95	IQM	CI95	IQM	CI95	IQM	CI95
SAC (External Resets)	-	<u>0.26</u>	[0.15, 0.84]	<u>0.12</u>	[0.05, 0.78]	4.86	[2.90, 17.57]	0.07	[0.00, 0.96]
SAC (LNT)	SAC	0.46	[0.23, 0.86]	0.51	[0.27, 0.81]	**2.63**	[0.91, 6.68]	0.00	[0.00, 0.05]
SAC	EBSAC	0.36	[0.22, 0.60]	0.68	[0.44, 0.83]	<u>4.33</u>	[2.28, 7.40]	**0.00**	[0.00, 0.00]
DREDQ (**ours**)	EBSAC	**0.05**	[0.02, 0.10]	**0.06**	[0.00, 0.19]	8.21	[2.27, 12.66]	<u>0.00</u>	[0.00, 0.04]

the charging bay, since in this case the reset agent only has to drive backwards in a straight line to reset the environment. Thus, the number of hard resets and safety violations is lower for these pairings. The generally high $\Delta\theta$ values of the 95% CI upper bound can be explained by the non-linearity of the cosine function used in Eq. (10). The non-linearity leads to the angular error becoming less relevant for maximizing the return the closer θ becomes to zero, thus incentivizing the agent to optimize for Δx and Δy instead. Overall, DREDQ is the only non-standalone forward agent capable of learning to solve the environment's task within the step limit. Further experiments have shown that a SAC forward agent requires at least 3×10^6 environment steps to solve the task, while still reaching lower final returns, causing more forward violations and a comparable number of reset violations and requiring a similar number of hard resets. Figure 4c and Tab. 2 show the absolute number of hard resets performed during the training. Our SRRL framework reduces the number of required hard resets to approximately one fifth of the hard resets of the standalone SAC trained for the same number of steps. Even when we account for the fact that the standalone SAC would be sufficiently trained after 10^6 environment steps like in Tab. 1, the number of hard resets is still reduced by 66%. Figure 4c visualizes the development in the share of successful soft resets by the reset agent in the total number of resets requested. All pairings using our EBSAC reset agent converge to the point where almost all resets are performed by the reset agent faster than the LNT

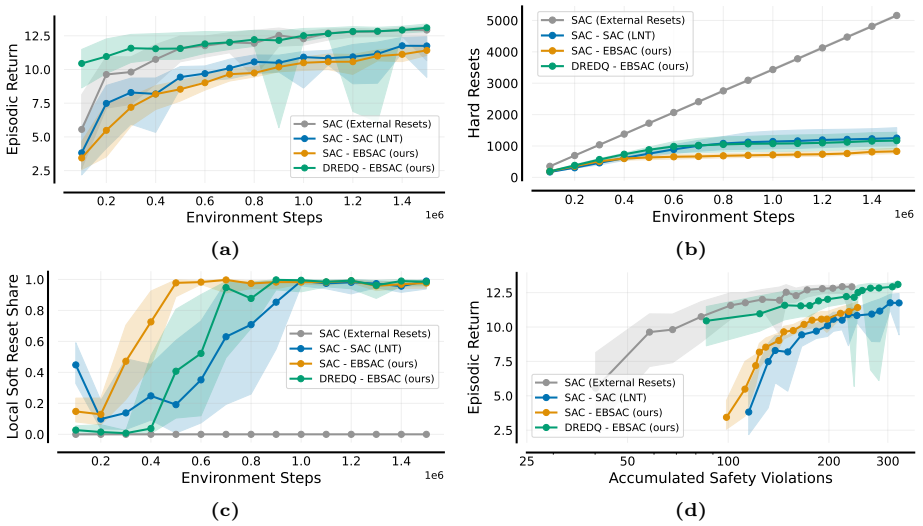

Fig. 4: Interquartile mean of the episodic return (Fig. 4a), the number of hard resets performed during training (Fig. 4b) and proportion of the soft resets in the last 100 requested resets (Fig. 4c) plotted over 1.5×10^6 environment steps. The 95% confidence interval of all metrics are represented by the shaded areas. The pairing with a DREDQ forward agent converges faster to high episodic returns than all other pairings and even standalone SAC. All pairings significantly reduce the number of required hard resets, with DREDQ and EBSAC also requiring fewer resets than LNT. LNT's reset agent takes the longest to learn to consistently reset the environment. For the relationship between episodic return and accumulated safety violations in Fig. 4d, the safety violation axis is scaled logarithmically and based on the interquartile mean. The higher the return and the lower the number of violation, the better the agent. DREDQ is the only forward agent that learns to complete the task within the step limit.

baseline. This implies that the additional reset reward function of LNT restricts the reset agent's learning progress and the example-based approach leaves more freedom for the reset agent to exploit. However, the reset agent appears to be the main cause of violations regardless of the framework used, as can be seen in Tab. 2. To further assess the quality of the policy learned by our EBSAC reset agent, we illustrate the last 100 trajectories of the forward and reset agent of the DREDQ - EBSAC pairing in Fig. 5. The trajectories of both agents overlap considerably, thus indicating that the EBSAC reset agent learned a policy which is almost exactly inverse to the forward agent's policy. This confirms that our simple approach to example-based RL works in the context of our framework.

7 Current Limitations and Future Work

Currently, our approach relies on an extrinsic signal to indicate whether our condition for a successful reset is met. Although this requirement can be chal-

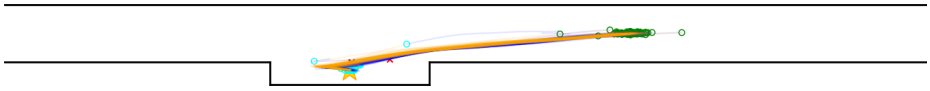

Fig. 5: Exemplary illustration of the last 100 trajectories driven by the robot controlled by a DREDQ forward agent (*blue lines*) and an EBSAC reset agent (*orange lines*). The forward agent's starting positions are marked with *green circles* and positions where it collided with an obstacle are highlighted by a *red cross*. For the reset agent, the starting positions are colored *turquoise* and points of collision in *purple*. The reset agent's trajectories closely follow those of the forward agent. Moreover, the converged reset policy results in the forward agent's initial state distribution forming a smaller cluster than the environment's original distribution.

lenging in practical scenarios due to the need for an external tracking setup, it allows us to simplify the overall framework and concentrate on demonstrating the core concept. In future work, we aim to eliminate this reliance on external signals by incorporating anomaly detection methods [26,31,33]. This will enable the agent to autonomously determine if the current state is part of the initial state distribution, further enhancing the framework's practicality. Furthermore, Fig. 5 shows the forward agent's starting positions are closely clustered due to the reset agent converging to a policy. This clustering can impact the forward agent's ability to generalize well. To address this, we plan to introduce randomness and thus variance to the final steps of the reset agent's trajectory using methods like RND [7]. Moreover, we are optimistic about reducing reset safety violations and minimizing the steps required by the reset agent even further by developing an example-based variant of DREDQ.

8 Conclusion

In this paper, we demonstrated the benefits of combining Safe and Resetless Reinforcement Learning in our novel Safe Resetless Reinforcement Learning framework to further increase the autonomy of the training process. We also explored the possibility of using distributional agents on Sautéd Safe RL environments and introduced DREDQ a new high-performing algorithm which leverages the advantages of both distributional RL and REDQ to achieve an unseen convergence speed while also reaching the highest episodic returns in our evaluation.

Acknowledgements This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the AI service center KISSKI (grant no. 01IS22093C), the Lower Saxony Ministry of Science and Culture (MWK) through the zukunft.niedersachsen program of the Volkswagen Foundation and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122).

References

1. Agarwal, R., Schwarzer, M., Castro, P.S., Courville, A., Bellemare, M.G.: Deep reinforcement learning at the edge of the statistical precipice. Advances in Neural Information Processing Systems (2021)
2. Altman, E.: Constrained Markov decision processes. Routledge (2021)
3. Bai, Q., Bedi, A.S., Agarwal, M., Koppel, A., Aggarwal, V.: Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 3682–3689 (2022)
4. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)
5. Bellman, R.: A markovian decision process. Journal of mathematics and mechanics pp. 679–684 (1957)
6. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
7. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint arXiv:1810.12894 (2018)
8. Chen, X., Wang, C., Zhou, Z., Ross, K.: Randomized ensembled double q-learning: Learning fast without a model. arXiv preprint arXiv:2101.05982 (2021)
9. Cohen, M.H., Belta, C.: Safe exploration in model-based reinforcement learning using control barrier functions. Automatica **147**, 110684 (2023)
10. Dabney, W., Ostrovski, G., Silver, D., Munos, R.: Implicit quantile networks for distributional reinforcement learning. In: International conference on machine learning. pp. 1096–1105. PMLR (2018)
11. Dabney, W., Rowland, M., Bellemare, M., Munos, R.: Distributional reinforcement learning with quantile regression. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
12. Dockhorn, A., Kruse, R.: Forward model learning for motion control tasks. In: 2020 IEEE 10th International Conference on Intelligent Systems (IS). pp. 1–5. IEEE (2020)
13. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning **110**(9), 2419–2468 (2021)
14. Eysenbach, B., Levine, S., Salakhutdinov, R.R.: Replacing rewards with examples: Example-based policy search via recursive classification. Advances in Neural Information Processing Systems **34**, 11541–11552 (2021)
15. Eysenbach, B., Gu, S., Ibarz, J., Levine, S.: Leave no trace: Learning to reset for safe and autonomous reinforcement learning. arXiv preprint arXiv:1711.06782 (2017)
16. Fu, J., Singh, A., Ghosh, D., Yang, L., Levine, S.: Variational inverse control with events: A general framework for data-driven reward definition. Advances in neural information processing systems **31** (2018)
17. Ha, S., Xu, P., Tan, Z., Levine, S., Tan, J.: Learning to walk in the real world with minimal human effort. arXiv preprint arXiv:2002.08550 (2020)
18. Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., Levine, S.: Learning to walk via deep reinforcement learning. arXiv preprint arXiv:1812.11103 (2018)
19. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al.: Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)

20. Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., Yang, Y.: Safety gymnasium: A unified safe reinforcement learning benchmark. Advances in Neural Information Processing Systems **36** (2023)
21. Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. Nature **620**(7976), 982–987 (2023)
22. Kim, J., hyeon Park, J., Cho, D., Kim, H.J.: Automating reinforcement learning with example-based resets. IEEE Robotics and Automation Letters **7**(3), 6606–6613 (2022)
23. Koenker, R., Hallock, K.F.: Quantile regression. Journal of economic perspectives **15**(4), 143–156 (2001)
24. Lee, S.H., Seo, S.W.: Self-supervised curriculum generation for autonomous reinforcement learning without task-specific knowledge. IEEE Robotics and Automation Letters (2024)
25. Ma, X., Xia, L., Zhou, Z., Yang, J., Zhao, Q.: Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning. arXiv preprint arXiv:2004.14547 (2020)
26. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM computing surveys (CSUR) **54**(2), 1–38 (2021)
27. Rao, K., Harris, C., Irpan, A., Levine, S., Ibarz, J., Khansari, M.: Rl-cyclegan: Reinforcement learning aware simulation-to-real. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11157–11166 (2020)
28. Ray, A., Achiam, J., Amodei, D.: Benchmarking safe exploration in deep reinforcement learning. arXiv preprint arXiv:1910.01708 **7**(1), 2 (2019)
29. Robey, A., Hu, H., Lindemann, L., Zhang, H., Dimarogonas, D.V., Tu, S., Matni, N.: Learning control barrier functions from expert demonstrations. In: 2020 59th IEEE Conference on Decision and Control (CDC). pp. 3717–3724. IEEE (2020)
30. Rockafellar, R.T., Uryasev, S., et al.: Optimization of conditional value-at-risk. Journal of risk **2**, 21–42 (2000)
31. Rudolph, M., Wandt, B., Rosenhahn, B.: Same same but differnet: Semi-supervised defect detection with normalizing flows. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 1907–1916 (2021)
32. Salvato, E., Fenu, G., Medvet, E., Pellegrino, F.A.: Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. IEEE Access **9**, 153171–153187 (2021)
33. Schaller, M., Schlör, D., Hotho, A.: Modeconv: A novel convolution for distinguishing anomalous and normal structural behavior. arXiv preprint arXiv:2407.00140 (2024)
34. Schier, M., Reinders, C., Rosenhahn, B.: Deep reinforcement learning for autonomous driving using high-level heterogeneous graph representations. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 7147–7153. IEEE (2023)
35. Schier, M., Reinders, C., Rosenhahn, B.: Learned fourier bases for deep set feature extractors in automotive reinforcement learning. In: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). pp. 931–938. IEEE (2023)
36. Sootla, A., Cowen-Rivers, A.I., Jafferjee, T., Wang, Z., Mguni, D.H., Wang, J., Ammar, H.: Sauté rl: Almost surely safe reinforcement learning using state augmentation. In: International Conference on Machine Learning. pp. 20423–20443. PMLR (2022)

37. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 30 (2016)
38. Wachi, A., Hashimoto, W., Shen, X., Hashimoto, K.: Safe exploration in reinforcement learning: A generalized formulation and algorithms. Advances in Neural Information Processing Systems **36** (2024)
39. Wachi, A., Sui, Y.: Safe reinforcement learning in constrained markov decision processes. In: International Conference on Machine Learning. pp. 9797–9806. PMLR (2020)
40. Wang, Y., Zhan, S.S., Jiao, R., Wang, Z., Jin, W., Yang, Z., Wang, Z., Huang, C., Zhu, Q.: Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In: International Conference on Machine Learning. pp. 36593–36604. PMLR (2023)
41. Xu, T., Liang, Y., Lan, G.: Crpo: A new approach for safe reinforcement learning with convergence guarantee. In: International Conference on Machine Learning. pp. 11480–11491. PMLR (2021)
42. Yang, Q., Simão, T.D., Tindemans, S.H., Spaan, M.T.: Safety-constrained reinforcement learning with a distributional safety critic. Machine Learning **112**(3), 859–887 (2023)
43. Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: 2020 IEEE symposium series on computational intelligence (SSCI). pp. 737–744. IEEE (2020)
44. Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., Levine, S.: The ingredients of real-world robotic reinforcement learning. arXiv preprint arXiv:2004.12570 (2020)