IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Safe Reinforcement Learning with Dead-Ends Avoidance and Recovery

Xiao Zhang[1], Hai Zhang[1], Hongtu Zhou[1], Chang Huang[1], Di Zhang[1], Chen Ye[*,1,2], Junqiao Zhao[*,1,2,3], *Member, IEEE*

Abstract*—Safety is one of the main challenges in applying reinforcement learning to realistic environmental tasks. To ensure safety during and after the training process, existing methods tend to adopt overly conservative policies to avoid unsafe situations. However, overly conservative policy severely hinders the exploration and makes the algorithms substantially less rewarding. In this paper, we propose a method to construct a boundary that discriminates between safe and unsafe states. The boundary we construct is equivalent to distinguishing dead-end states, indicating the maximum extent to which safe exploration is guaranteed, and thus has a minimum limitation on exploration. Similar to Recovery Reinforcement Learning, we utilize a decoupled RL framework to learn two policies, (1) a task policy that only considers improving the task performance, and (2) a recovery policy that maximizes safety. The recovery policy and a corresponding safety critic are pre-trained on an offline dataset, in which the safety critic evaluates the upper bound of safety in each state as awareness of environmental safety for the agent. During online training, a behavior correction mechanism is adopted, ensuring the agent interacts with the environment using safe actions only. Finally, experiments of continuous control tasks demonstrate that our approach has better task performance with fewer safety violations than state-of-the-art algorithms. The code is available at https://github.com/tiev-tongji/dea-rrl.

***Index Terms*—Reinforcement Learning, Safety.**

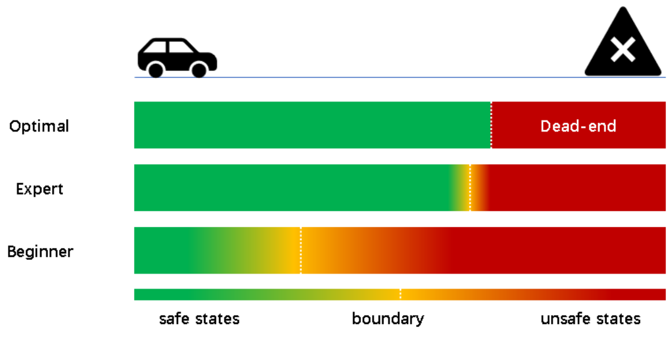

Fig. 1. This schematic shows the agent needs to control the car to avoid a collision. A safety critic evaluates the safety of the policy in all states and combines it with a threshold value to obtain the boundary that divides the safe (green) and unsafe states (red). When the car is close enough to an obstacle, at a given speed, no policy can avoid a collision, and these states are termed Dead-ends. The boundary that only identifies all dead-ends as unsafe states is called the optimal boundary (Optimal). Since the optimal safe policy is safe in all states except dead-ends, by evaluating the safety of the optimal safe policy, it is possible to distinguish whether a state is a dead-end or not. In contrast, suboptimal safe policies can lead the safety critic to conservatively consider more states as unsafe (Expert and Beginner).

I. INTRODUCTION

REINFORCEMENT learning (RL) has made impressive achievements in long-term control tasks, including Atria games [1], car driving [2] and robot controlling [3]. While RL performs well in games and simulation environments, safety becomes one of the greatest challenges when applying RL to real-world tasks. In real environments such as autonomous driving tasks, unsafe actions can lead to damage to the agent itself and the environment, resulting in significant maintenance costs and even human casualties.

To learn a safe policy that satisfies state-wise safety constraints in the "safe critical" task, the agent needs to evaluate the safety of each state and avoid entering unsafe states [4]. In some Safe RL algorithms, the agent's awareness of the safety of a state is achieved by the safety critic who evaluates the safety of the task policy in states. The safety critic and a safety threshold together construct a boundary that divides the state space into safe and unsafe subspaces, as shown in Figure 1. The division of the state space depends on the policy, and sub-optimal policies will lead to more states being considered unsafe, thus limiting agent exploration.

In RRL [5], a recovery policy and a behavioral correction mechanism are introduced. The task policy and the recovery policy are trained simultaneously to improve task performance and to satisfy safety constraints respectively. The behavioral correction mechanism determines whether the action selected by the task policy is a safe one by querying the value of safety critic corresponding to the action, and replacing the unsafe actions with those selected by the recovery policy. The safety critic and recovery policy are first trained offline, and then they are trained online alongside the task policy. This allows them to leverage prior information about the environment to enhance safety during online training. The agent can be considered as evaluating the safety of states and interacting with the environment through a composite policy composed of a task policy and a recovery policy.

Manuscript received: June 27, 2023; Revised: September 23, 2023; Accepted: October 22, 2023.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

This work is supported by the National Key Research and Development Program of China (No. 2021YFB2501104, No. 2020YFA0711402) *(Corresponding Author: Junqiao Zhao.)*

[1]All the authors are with the Department of Computer Science and Technology, Tongji University, China.

[2]Junqiao Zhao and Chen Ye are with the MOE Key Lab of Embedded System and Service Computing, Tongji University, Shanghai, China (`e-mail: zhaojunqiao@tongji.edu.cn; yechen@tongji.edu.cn`).

[3]Institute of Intelligent Vehicles, Tongji University, Shanghai, China.

Digital Object Identifier (DOI): see top of this page.

However, there are certain issues that arise when implementing this method. Specifically, the recovery policy used for selecting actions in states approaching danger inhibits exploration. This restriction can introduce a substantial bias in the distribution of data observed by the agent in these states. Training the safety critic and recovery policy on this biased data may not lead to optimal convergence, subsequently impacting the agent's assessment of state safety and resulting in an overly conservative algorithm. Additionally, the training of the safety critic necessitates composite policy sampling, and the behavior of the composite policy, in turn, is influenced by the safety critic. This interaction can lead to an unstable training process.

We believe that the safety of states should be solely determined by the recovery policy, which should be trained alongside a safety critic with the objective of maximizing safety. This approach enables the safety critic and recovery policy to be acquired through fully offline training. Additionally, it allows us to identify the optimal boundaries of the safe state space, specifically the boundaries of dead-ends.

Motivated by this, we propose our safe RL framework *Safe Reinforcement Learning with Dead Ends Avoidance and Recovery* (DEARRL), following the Recovery RL framework [5] and decoupled RL framework [6]. In contrast to RRL, our approach features a safety critic that assesses the safety of the recovery policy rather than the composite policy. This reduction in conservatism relaxes the constraints on exploration without increasing the risk of violations.

The advantages of our approach are as follows:

- Our method allows task policy to fully explore the environment and significantly improve task performance in complex environments;
- We show theoretically that our approach can learn optimal boundary, which is equivalent to the discovery of dead-ends.
- Our method completely decouples the task policy and safe policy, so that the recovery policy can be plug-and-play within other task policies without fine-tuning.

II. Related Work

A. Safe RL

Safe RL addresses two main safety-related problems: the asymptotic safety and the in-training safety of policies. Asymptotic safety means the safety of policies after convergence, which is commonly achieved by reward penalties and cost constraints. Lagrange Relaxation [7] is the most widely used method due to its simplicity, and other methods such as Trust Regions [8], Lyapunov-based [9], Guide Policy [10] are proposed for their stricter safety guarantees, fewer violations during the training and having faster convergence.

Although these approaches do find safe policies after training, they learn safety by trial-and-error the same as traditional RL algorithms, which means violations are inevitable before convergence. To ensure in-training safety, state-wise safety constraints and prior knowledge of the environment are utilized [4]. [11]–[13] assume a white-box or a black-box environment dynamics model is available, limiting policy optimizing in a safe policy set which is constructed based on the known model. However, these assumptions are hard to meet in real-world problems.

RRL [5] trains a recovery policy using offline data to guarantee the safety of the online training process. [14] combines RRL and Meta-RL [15], improving the generalizability of RRL by enabling pre-trained safety critic and recovery policy to be quickly adapted to different tasks during the fine-tuning phase. Although these methods can achieve in-training safety, they are prone to obtain over-conservative policies due to the reason explained in Section I.

B. Decoupling Performance and Safety

Decoupling performance and safety offers new ideas for solving the exploration-exploitation dilemma [6], [16] because separating processes of maximizing reward and guaranteeing safety prevents policies from under-performing due to over-conservatism. [17] and [5] introduce an implicit and an explicit safe policy respectively. The task policy no longer needs to consider safety explicitly when being updated. However, the identification of unsafe actions in these methods correlated with the task policy, thus the agent is still prevented from obtaining optimal policies by the exploration-exploitation dilemma. Although [18] achieves full decoupling of exploration and exploitation, it focuses only on the rewards and ignores safety in training.

C. Dead-Ends Discovery and Avoidance

The concept of dead-ends discovery (DeD) is introduced in [19] and later applied to the medical field [20]. [21] combines risk sensitivity with DeD, which allows dead-ends to be identified earlier. [22] also proposes the "irrecoverable state" with a similar meaning to the dead-end state, preventing dangerous situations from occurring through reward shaping and model-based rollout [23].

We contend that distinguishing dead-end states from normal states is essential to enhance the performance of safe RL. This distinction allows us to identify the broadest range of policies that can be explored safely. The safety critic in RRL [5] could not distinguish dead-end states, as elaborated later in Section IV-A.

III. Preliminary

A. Constraint Markov Decision Processes

We consider safe RL under Constraint Markov Decision Process (CMDPs) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma, \rho, \mathcal{C}, \gamma_{safe}, \epsilon_{safe})$ [24], where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ the transition function, $\gamma \in [0, 1)$ the discount factor for reward, $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the cost function, $\gamma_{safe} \in [0, 1)$ the discount factor for cost and $\epsilon_{safe} \in \mathcal{R}$ the safe threshold. Let Π be the set of Markovian stationary policies. Given policy $\pi \in \Pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maps states to action distributions and $\pi(a|s)$ denotes the probability of choosing action a in state

s. The task performance of π is defined as the discounted cumulative reward $\mathcal{J}(\pi)$:

$$\mathcal{J}(\pi)=\mathbb{E}_{\tau\sim\pi,P}[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t,a_t,s_{t+1})] \tag{1}$$

Similar to [25], state-cost function V_c and action-cost function Q_c are introduced to indicate the expected cumulative cost of π in s:

$$V_c^\pi(s)=\mathbb{E}_{\tau\sim\pi,P}[\sum_{t=0}^{\infty}\gamma_{safe}^t\mathcal{C}(s_t,a_t,s_{t+1})|s_0=s] \tag{2}$$

$$Q_c^\pi(s,a)=\mathbb{E}_{s'\sim P(\cdot|s,a)}V_c^\pi(s') \tag{3}$$

Under the state-wise safety constraint formulation [4], we define the set of safe policies

$$\Pi_c=\{\pi\in\Pi|\forall s\in\mathcal{S},V_c^\pi(s)<\epsilon_{safe}\} \tag{4}$$

The objective of CMDPs is to find a policy that maximizes Equation (1) in the set of safe policies Π_c:

$$\pi^*_{task}=\max_\pi\mathcal{J}(\pi),\text{s.t.}\pi\in\Pi_\text{c} \tag{5}$$

B. Safe Markov Decision Processes

In safety-critical tasks, any unsafe action is fatal. Therefore, we define SMDPs, a special case of CMDPs to describe this kind of task.

Definition 1: Safe Markov Decision Processes (SMDPs) $\mathcal{M}=(\mathcal{S},\mathcal{A},\mathcal{R},P,\gamma,\rho,\mathcal{C},\gamma_{safe},\epsilon_{safe})$, a special case of CMDPs where episodes terminate after any danger occurred. In SMDPs, the cost function is a binary indicator of the safety of the state:

$$\mathcal{C}(s_t,a_t,s_{t+1})=\begin{cases}1, & safety\ violation\\ 0, & otherwise\end{cases} \tag{6}$$

Observe that if $\gamma_{safe}=1$, Q_c^π indicates the probability of ending up with a failure state in the future with π. $Q_{\phi,c}^\pi$, parameterized by ϕ, can be optimized by minimizing the MSE loss:

$$\begin{aligned}\mathcal{L}_{Q_c}(\phi;\pi)=\frac{1}{2}(Q_{\phi,c}^\pi(s_t,a_t)-(c_t+(1-c_t)\\ \gamma_{safe}\mathbb{E}_{a_{t+1}\sim\pi(\cdot|s_{t+1})}Q_{\hat{\phi},c}^\pi(s_{t+1},a_{t+1})))^2\end{aligned} \tag{7}$$

where (s_t,a_t,s_{t+1},c_t) is the transition sampled from offline data or replay buffer and c_t is short for $\mathcal{C}(s_t,a_t,s_{t+1})$. Equation (7) can be considered to be the policy evaluation of π in terms of safety.

Definition 2: The state space is divided into three subspaces:

- $\mathcal{S}_{fail}$: Failure state. Indicates the end of an episode due to a danger.
- $\mathcal{S}_{dead}$: Dead-ends state. Any dead-end state will transform into a failure state, regardless of the policy π the agent takes ($\pi\in\Pi$).
- $\mathcal{S}_{safe}$: Safe state. $\mathcal{S}_{safe}$=$\mathcal{S}\setminus(\mathcal{S}_{fail}\cup\mathcal{S}_{dead})$.

Therefore,

$$\forall s\in\mathcal{S}_{safe},\exists(a,s')\in(\mathcal{A},\mathcal{S}_{safe}),P(s,a,s')>0 \tag{8}$$

The cost function in SMDPs can also be expressed as

$$\mathcal{C}(s_t,a_t,s_{t+1})=\mathbb{I}(s_{t+1}\in\mathcal{S}_{fail}) \tag{9}$$

Similar to [22], we assume that a safety violation must come fairly soon after entering any dead-end state region:

Assumption 1: There exists a horizon $H\in\mathbb{N}$, that any trajectory starting from $s_0\in\mathcal{S}_{dead}$ will end up in H steps.

We additionally introduce a sampling policy $\overline{\pi}$ for training the safety critic and modify the definition of the set of safe policies as follows:

$$\begin{aligned}\Pi_c^{\overline{\pi}}=\{\pi\in\Pi|\forall(s,a)\in(\mathcal{S}_{safe},\mathcal{A})\text{ and }\pi(\text{a}|\text{s})>0,\\ Q_c^{\overline{\pi}}(s,\pi(s))<\epsilon_{safe}\}\end{aligned} \tag{10}$$

Similar to CMDPs, the objective of SMDPs is to find a policy with Equation (1) in the set of safe policies $\Pi_c^{\overline{\pi}}$:

$$\pi^*_{task}=\max_\pi\mathcal{J}(\pi),\text{s.t.}\pi\in\Pi_\text{c}^{\overline{\pi}} \tag{11}$$

IV. METHODOLOGY

A. Recovery RL

In most approaches of Safe RL without decoupling, $\overline{\pi}$ is the same as π_{com} (the composite policy), including RRL [5]. In pre-train phase, RRL trains safety critic $Q_{\phi,c}^{\pi_{com}}$ and recovery policy $\pi_{\theta,rec}$ (parameterized by ϕ and θ respectively) by minimizing $\mathcal{L}_{Q_c}(\phi;\pi_{com})$ according to Equation (7) and maximizing $\mathcal{J}_{\pi_{rec}}(\theta;\pi_{com})$ according to Equation (12).

$$\mathcal{J}_{\pi_{rec}}(\theta;\overline{\pi})=-\mathbb{E}_{s\sim\mathcal{D}}[Q_c^{\overline{\pi}}(s,\pi_{\theta,rec}(\cdot|s))] \tag{12}$$

In fine-tuning phase, unsafe actions will be corrected by π_{rec}:

$$a_t=\begin{cases}a^{\pi_{task}}, Q_{\phi,c}^{\overline{\pi}}(s_t,a^{\pi_{task}})<\epsilon_{safe}\\ a^{\pi_{rec}}, otherwise\end{cases} \tag{13}$$

Similar to Equation (11) The objective of RRL can be expressed as:

$$\pi^*_{task}=\max_\pi\mathcal{J}(\pi),\text{s.t.}\pi\in\Pi_\text{c}^{\pi_\text{com}} \tag{14}$$

As described in Section I, the restriction of exploration results in a bias in the observed data. Training the safety critic and recovery policy on such biased data may not lead to optimal convergence. Consequently, RRL tends to be overly conservative in its assessment of state safety.

B. Dead-ends Discovery and Avoidance

Different from RRL, our proposed DEARRL ensures safety by distinguishing between dead-end states and safe states and only prevents the agent from entering dead-ends. In the pre-train phase, DEARRL trains safety critic $Q_{\phi,c}^{\pi_{rec}}$ and recovery policy $\pi_{\theta,rec}$ by minimizing $\mathcal{L}_{Q_c}(\phi;\pi_{rec})$ and $\mathcal{J}_{\pi_{rec}}(\theta;\pi_{rec})$ as in Equation (7) and Equation (12), which is completely decoupled from π_{task}. This is equivalent to solving the optimal Bellman equation [25] for safety. Thus, the optimal recovery policy and the corresponding policy evaluation function can be obtained.

$$\begin{aligned}\pi^*_{rec}(\theta)&=\min_\theta\mathbb{E}_{\tau\sim P,\pi_{rec}(\theta)}[Q_c^{\pi_{rec}}(s,\pi_{rec}(s;\theta))]\\ &=\min_\theta\mathbb{E}_{\tau\sim P,\pi_{rec}(\theta)}[\sum_{t=0}^{\infty}\gamma_{safe}^t c_t]\end{aligned} \tag{15}$$

By using the same behavior correction mechanism as Equation (13) where $\bar{\pi}$ is π^*_{rec}, the object of DEARRL can be expressed as:

$$\pi^*_{task} = \max_{\pi} \mathcal{J}(\pi), \text{s.t.} \pi \in \Pi_c^{\pi^*_{\text{rec}}} \tag{16}$$

C. Theoretical Proof

We will illustrate the advantages of DEARRL over RRL theoretically.

Theorem 1: $\Pi_c^{\pi_{com}}$ and $\Pi_c^{\pi^*_{rec}}$ are the accessible space for π to explore safely in RRL and DEARRL respectively, we have $\Pi_c^{\pi_{com}} \subseteq \Pi_c^{\pi^*_{rec}}$.

Proof: Since Q_c^* (the shorthand of $Q_c^{\pi^*_{rec}}$) is the optimal cost value function, we have $Q_c^*(s,a) \leq Q_c^{\pi_{com}}(s,a)$ for every (s,a). As in Equation (10), $\forall(\pi, s, a) \in (\Pi_c^{\pi_{com}}, \mathcal{S}_{safe}, \mathcal{A})$ and $\pi(a|s) > 0$,

$$Q_c^*(s, \pi(s)) \leq Q_c^{\pi_{com}}(s, \pi(s)) < \epsilon_{safe}$$

therefore $\pi \in \Pi_c^{\pi^*_{rec}}$, which means $\Pi_c^{\pi_{com}} \subseteq \Pi_c^{\pi^*_{rec}}$.

We will show that with appropriate ϵ_{safe}, Q_c^* can be used to identify dead-end states.

Lemma 1: Suppose that Assumption 1 holds and uncertainties are ignored in the environment i.e. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \{0, 1\}$,

- $\forall(s,\pi) \in (\mathcal{S}_{fail}, \Pi),\ V_c^\pi(s) = V_c^*(s) = 1$,
- $\forall(s,\pi) \in (\mathcal{S}_{dead}, \Pi),\ V_c^\pi(s) \geq V_c^*(s) \geq \gamma_{safe}^{H-1}$
- $\forall(s,\pi) \in (\mathcal{S}_{safe}, \Pi),\ V_c^\pi(s) \geq V_c^*(s) = 0$

Proof: Since V_c^* is the optimal value function, we have $V_c^\pi(s) \geq V_c^*(s)$ for all $s \in \mathcal{S}$. By the definition of cost function Equation (9) and state-cost function Equation (2), $\forall s \in \mathcal{S}_{fail}, V_c^\pi(s) = 1$.

Assumption 1 shows that the episode would terminate after at most H steps since the agent reaches a dead-end state, we can derive from Equation (2) that

$$\begin{aligned} &V_c^\pi(s) = \mathbb{E}_{\tau\sim\pi,P}[\sum_{t=0}^{\infty} \gamma_{safe}^t \mathcal{C}(s_t, a_t, s_{t+1})|s_0 = s \in \mathcal{S}_{dead}] \\ &\geq \sum_{t=0}^{H-2} \gamma_{safe}^t * 0 + \gamma_{safe}^{H-1} * 1 = \gamma_{safe}^{H-1}. \end{aligned} \tag{17}$$

By Equation (8) in Definition 2, for every $s \in \mathcal{S}_{safe}$, there always exists at least one action a such that $s' \sim P(\cdot|s,a), s' \in \mathcal{S}_{safe}$. Start at $s \in \mathcal{S}_{safe}$, the agent would never reach a dead-end state by choosing the safe action, which means that $V_c^*(s) = 0$.

Theorem 2: Suppose that Assumption 1 holds and uncertainties are ignored in the environment, and let

$$\epsilon_{safe} = \gamma_{safe}^H \tag{18}$$

Then, with the behavior correction mechanism shown by Equation (13), the agent will be prevented from reaching dead-end states.

Proof: According to Equation (13), the actions allowed to be performed satisfy:

$$Q_c^\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)} \gamma_{safe} V_c^\pi(s') < \gamma_{safe}^H \tag{19}$$

therefore

$$V_c^\pi(s') < \gamma_{safe}^{H-1}, \forall s \in \mathcal{S}_{safe} \tag{20}$$

which ensuring $s' \in \mathcal{S}_{safe}$. Because of the initial state $s_0 \in \mathcal{S}_{safe}$, it is ensured that the agent is always explored in safe states.

Theorem 1 shows that DEARRL may provide a broader accessible space for exploration compared to RRL, thereby reducing conservatism without compromising safety. Lemma 1 and theorem 2 collectively suggest that when starting from safe states, both RRL and DEARRL can prevent agents from entering dead-end states by utilizing an appropriate ϵ_{safe}. It is noteworthy that in RRL, there is no guarantee that $V_c^{\pi_{com}}(s) < \gamma_{safe}^{H-1}, \forall s \in \mathcal{S}_{safe}$ since π_{com} may not necessarily be the optimal safe policy. Consequently, RRL may mistakenly identify certain safe states as dead-end states due to its inherent conservatism.

D. Offline Pre-train

In offline pre-training, managing Out-Of-Distribution (OOD) actions is crucial to mitigate any estimation bias in the safety critic. An effective approach to this challenge is avoiding the querying of Q_c values for OOD actions. However, during online training, it's essential not to completely disregard OOD actions when determining recovery policy actions. In situations where all known actions within the current state are unsafe, it is preferable to opt for an unknown yet potentially safe action rather than deploying a known but certainly dangerous one.

Inspired by the implicit Q-learning (IQL) algorithm [26], we use Expectile Regression to train $Q_c^{\pi_{rec}}$, effectively avoiding the impact of OOD actions. Different from IQL, we employ Advantage Policy Gradient instead of Advantage Weighted Regression in [26] to train π_{rec}. This modification permits an OOD action to be attempted in a state where all known actions are deemed unsafe.

The state-cost function V_c^π and action-cost function Q_c^π is updated by minimizing following loss functions:

$$\mathcal{L}_{V_c^\pi}(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\phi,c}^\pi(s,a) - V_{\psi,c}^\pi(s))] \tag{21}$$

$$\begin{aligned} &\mathcal{L}_{Q_c^\pi}(\phi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(c(s,a,s')+ \\ &(1 - c(s,a,s'))\gamma_{safe} V_{\psi,c}^\pi(s') - Q_{\phi,c}^\pi(s,a))^2] \end{aligned} \tag{22}$$

where L_2^τ is the expectile regression loss and $\mathcal{D}$ the offline dataset.

E. Online Training

Any of the RL algorithms can be used to train π_{task}. In our work, we utilize the Soft Actor Critic algorithm (SAC) [27]. The process of online fine-tuning is illustrated in Algorithm 1, and we give some remarks on online training.

- We relabel all actions with the action proposed by π_{task} as the same as [5], which is important to achieve decoupled safe reinforcement learning, as it prompts the agent to view behavior correction as part of the environment.

- Exploration is not permitted for the recovery policy as it would increase the frequency of safety constraint violations during the training process, rendering the training procedure unsafe.
- We choose not to fine-tune $Q_c^{\pi_{rec}}$ and π_{rec} in online training since it is risky to train on data collected by policies with constrained exploration. Additionally, a fixed behavior correction mechanism provides the agent with a stable MDP dynamic model, thus improving the stability of training.
- Finally, we use a small ϵ_{safe} to enhance the safety of the algorithm due to aleatoric and epistemic uncertainty.

Algorithm 1 DEARRL Training Online

```
1:  Input: safety critic $Q_c^{\pi_{rec}}$, recovery policy $\pi_{rec}$, task horizon $H$, training steps $N$
2:  Initialize replay buffer $\mathcal{D}_{task} \leftarrow \varnothing$
3:  $s_0 \leftarrow env.reset()$
4:  **for** $steps, \leftarrow 1, N$ **do**
5:      **for** $t \in \{1, ..., H\}$ **do**
6:          **if** $c_{t-1} = 1 \; or \; t = H$ **then**
7:              $s_t \leftarrow env.reset()$
8:          **end if**
9:          Sample $a^{\pi_{task}}$, $a^{\pi_{rec}}$ from $\pi_{task}$, $\pi_{rec}$
10:         **if** $Q_c^{\pi_{rec}}(s_t, a^{\pi_{task}}) < \epsilon_{safe}$ **then**
11:             $a_t = a^{\pi_{task}}$
12:         **else**
13:             $a_t = a^{\pi_{rec}}$ (behavior correction)
14:         **end if**
15:         Execute $a_t$, Observe $s_{t+1}, r_t, c_t$
16:         $\mathcal{D}_{task} \leftarrow \mathcal{D}_{task} \cup (s_t, a^{\pi_{task}}, s_{t+1}, r_t)$
17:         Train $\pi_{task}$ on $\mathcal{D}_{task}$ by maximizing $\mathcal{J}(\pi)$
18:     **end for**
19: **end for**
```

V. EXPERIMENTS

In the experiments, we investigate whether our approach can:

- exceeds state-of-the-art algorithms in terms of task performance, post-training safety and in-training safety;
- enhance the safety of the training process with minimal impact on task performance;
- obtain a task-independent behavior correction strategy that ensures the safety of trained policies in testing.

A. Domains

Experiments are conducted under the standard safe reinforcement learning test environment Safety Gym [28]. As shown in Figure 2, we selected two simple environments (StaticEnv, DynamicEnv) set up in [29] and three more complex environments (PointGoal1, CarGoal1, DoggoGoal1) predefined in Safety Gym. Unlike the general Safety Gym setup, we mandate that any violation of safety constraints leads to the immediate termination of the episode, mirroring real-world tasks. This setup significantly increases the complexity of the task and poses an extreme challenge to the agent's capacity to balance exploration and exploitation.

B. Offline Data Collection

Unlike general offline RL where the training results are influenced by the performance of the behavior policy that is used for data sampling, the Recovery RL framework requires offline data to contain a wide coverage of unsafe trajectories, allowing Q_c to learn to identify unsafe actions efficiently. The offline dataset we used contains 2M transitions including: (1) 1M transitions sampled from the replay buffer of SAC training, and (2) 1M transitions obtained by interacting with the environment using random actions. The latter is included because in our experiments we find that the failed transitions in the SAC replay buffer tend to share similar characteristics. The random sampling data can diversify the failed transitions in the offline data. We filter out the parts of the data that are less relevant to safety by keeping only 100 transitions before violations.

C. Evaluation Metric

The average cumulative reward (ACR) over episodes is used as a measure of task performance, and a higher return indicates a better task performance. The default reward functions provided by Safety Gym are used in our experiments, which define algorithm-independent tasks. We use the average rate of constraint violation (AVR) in testing as a measure of asymptotic safety and the number of constraint violations (TV) in training as a measure that indicates in-training safety. To highlight the differences between our method and RRL, we use the ratio of steps that the behavior correction mechanism is used (ARR) as a measure of the extent to which the behavior correction mechanism intervenes in the training and testing phase.

D. Comparisons with Baselines

- Unconstrained Baseline [27]: We use SAC as a baseline for unconstrained methods, optimizing task performance and ignoring safety constraints.
- Worst-Case Soft Actor Critic (WCSAC) [29]: A combined Lagrangian relaxation and risk-sensitive approach that maximizing
$$\mathcal{J}(\pi) - \lambda(\mathbb{E}_{\tau\sim P,\pi}[CVaR_\alpha(Q_c^\pi(s,a))] - \epsilon_{safe})$$
where $CVaR_\alpha \doteq \mathbb{E}_{p^\pi}[Q_c^\pi | Q_c^\pi \geq F_C^{-1}(1-\alpha)]$ and F_C is the CDF of $p^\pi(Q_c^\pi | s, a)$, updating policy parameters and λ via dual gradient descent.
- RRL [5]: A *semi-decoupled* approach that preventing agent using unsafe action for which $Q_c^{\pi_{com}}(s,a) \geq \epsilon_{safe}$ by behavior correction mechanism. To ensure safety during training, we do not allow the recovery policy to explore the environment.
- Implicit Q-Learning (IQL) [26]: An offline RL algorithm that optimizes critic and actor by expectile regression and advantage-weighted regression.

To better compare the ability of each method to balance safety and task performance, we let each method have similar asymptotic safety achieved by using different ϵ_{safe}.

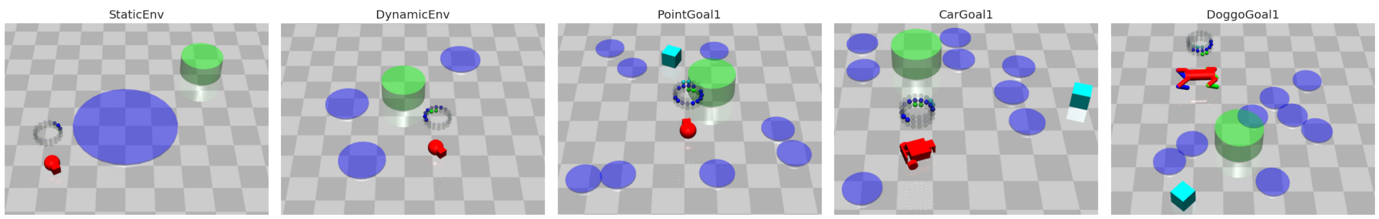

Fig. 2. The experiments are conducted in the safety gym standard test environment. Five task environments are shown from left to right, including StaticEnv, DynamicEnv, PointGoal1, CarGoal1 and DoggoGoal1. These environments are based on the Safety Gym, where the task is to navigate a robot from its initial position to the target area (green area) and avoid entering unsafe areas (blue areas) during the process. PointGoal1, CarGoal1 and DoggoGoal1 can be seen as upgraded versions of DynamicEnv, which include more unsafe areas, larger action space and more complex robots.

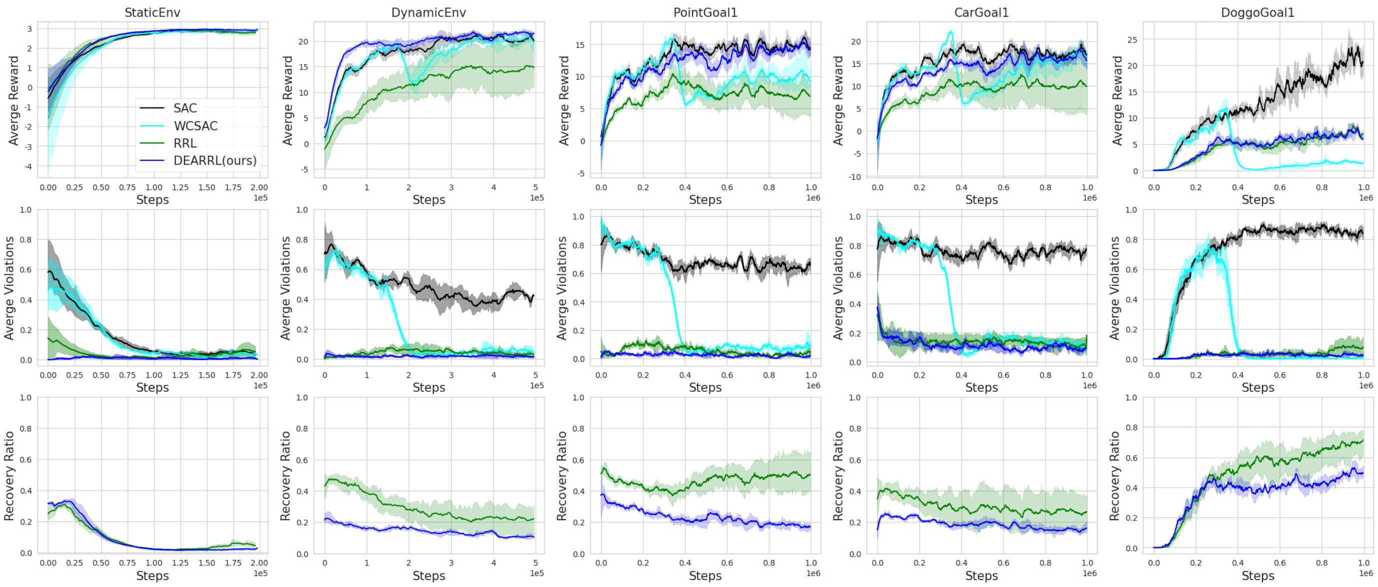

Fig. 3. Training curves. Means (solid lines) and variances (shaded) of the training curves for the four algorithms under different tasks, with the first row showing the average cumulative reward, the second row showing the average proportion of constraint violations over update steps and the third row showing the average use of behavior correction. We adapted ϵ_{safe} for RRL and DEARRL to make the two algorithms have similar safety in training. For each method, we used random seeds for training.

TABLE I
RESULTS OF TRAINING AND TESTING EACH METHOD INDIVIDUALLY

Environments	SAC			WCSAC			RRL			DEARRL(ours)			IQL	
	ACR	AVR	TV	ACR	AVR	TV	ACR	AVR	TV	ACR	AVR	TV	ACR	AVR
StaticEnv	2.754	0.043	1642	**2.936**	**0.008**	1453	2.813	0.032	392	2.882	0.023	**119**	2.630	0.095
DynamicEnv	17.434	0.519	3330	19.004	0.069	1451	14.230	**0.006**	112	**20.784**	0.018	**86**	18.044	0.565
PointGoal1	12.609	0.753	12375	7.811	0.055	5739	7.203	0.094	476	**14.079**	**0.035**	**236**	12.740	0.76
CarGoal1	16.727	0.73	14850	**18.672**	**0.035**	6199	9.647	0.057	1109	15.430	0.091	**786**	11.357	0.94
DoggoGoal1	16.956	0.908	31450	1.055	**0.013**	10825	5.924	0.192	**623**	7.002	0.048	706	**18.889**	0.838

* ACR, AVR, TV respectively stand for Average Cumulative Return, Average Violation Rate, and Total Violations in training.
* The best and second-best results have been marked in bold and underlined, respectively.

E. Results

1) Main Results: The performance and safety of all methods are shown in Figure 3 and Table I. The results show that our method exhibits high asymptotic safety (low AVR) and in-training safety (low TV) across all tasks, along with a cumulative reward (ACR) similar to the unconstrained method in most tasks. This suggests that our method substantially enhances the safety of the algorithm with minimal impact on performance.

WCSAC achieves a policy with higher safety (low AVR) compared to SAC after training in all tasks. However, due to its learning of information related to the safety of the environment during training, WCSAC inevitably experiences numerous constraint violations in the early stages of training (high TV).

IQL avoids interaction with the environment by training completely offline so that safety constraints are not violated during training. Nevertheless, it fails to attain a safe policy

TABLE II
RESULTS OF TESTING THE COMBINATION OF TRAINED SAC AND PRE-TRAINED BEHAVIORAL CORRECTION MECHANISMS

Environments	SAC(with RRL)			IQL(with RRL)			SAC(with DEARRL)			IQL(with DEARRL)		
	ACR	AVR	ARR	ACR	AVR	ARR	ACR	AVR	ARR	ACR	AVR	ARR
StaticEnv	2.451	**0**	0.351	2.730	0.006	0.314	2.667	**0**	0.009	**2.947**	**0**	0.031
DynamicEnv	10.012	**0.014**	0.492	9.822	0.046	0.523	18.769	0.032	0.242	**19.425**	0.078	0.235
PointGoal1	6.059	0.120	0.430	4.190	0.132	0.572	**10.596**	**0**	0.275	9.926	0.04	0.322
CarGoal1	8.332	0.060	0.527	7.545	0.094	0.559	8.165	**0.028**	0.334	**10.783**	0.116	0.297
DoggoGoal1	3.270	0.002	0.962	3.103	0.024	0.951	**13.285**	0.002	0.780	9.407	**0**	0.846

* ARR stands for the Average Ratio of using the Recovery policy.

after training (high AVR).

RRL, on the other hand, learns information about the safety of the environment through pre-training and therefore results in a low number of constraint violations throughout the training process (low TV). As shown in Figure 3, the ratio of steps RRL using behavior correction during the training process is higher than that of DEARRL, indicating a larger number of interventions in the training process of π_{task}, which is an important reason why RRL can only learn suboptimal policies in complex environments. It is worth noting that the training curve of RRL exhibits large variance, which is caused by the fact that π_{task} is trained in an unstable environment.

In terms of online training time, DEARRL requires action correction, thus it takes about 1.5 times longer than SAC to train with the same number of steps, slightly less than RRL. Conversely, offline training takes significantly less time (accounting for only 20% of the total training time), as the major time cost lies in interacting with the environment and collecting feedback.

Finally, we remark that DEARRL has a lower return than SAC in DoggoGoal1 because the environment is more difficult to explore with the introduction of the behavior correction mechanism, and it takes longer for agents to learn the optimal policy.

2) Ablations: We design ablation experiments to investigate the effect of settings in pre-training and online training on safety and task performance.

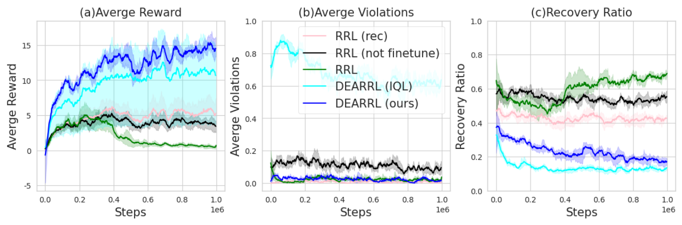

Fig. 4. Ablations of offline pre-training. Average reward, average proportion of constraint violations and average proportion of use of behavioral correction for different methods. For each method, we used the same $\epsilon_{safe} = 0.7$ and random seeds for training.

As shown in Figure 4, RRL (rec) represents the use of recovery policy as the sampling strategy for training the safety critic. This makes the training process smoother and improves safety critic's ability to recognize dead-end states, as compared to RRL. Not fine-tuning the safety critic and recovery policy during the online phase, represented by RRL (not finetune), can also yield better results than training on the observed data (RRL). This is due to the significant difference between the actual data distribution and the observed data distribution in online training. DEARRL (IQL) utilizes IQL as the offline training algorithm, which helps reduce the impact of OOD actions during the training of the safety critic, thus improving the performance of the safety critic. DEARRL employs the Advantage Policy Gradient algorithm within IQL, as described in Section IV-D, allows the recovery policy to select actions outside the dataset, thereby enhancing the capability of the behavior correction mechanism to ensure safety.

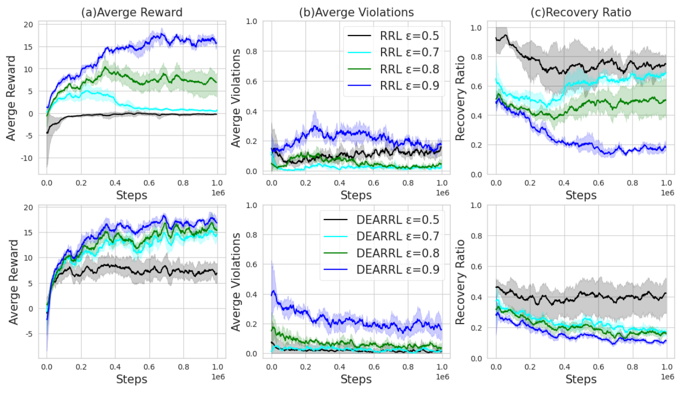

Fig. 5. Ablations of online training. Average reward, the average proportion of constraint violations and the average proportion of use of behavioral correction for different ϵ_{safe} choices for RRL and DEARRL in PointGoal1. For each method, we used random seeds for training.

We then compared the performance and safety of RRL and DEARRL under different ϵ_{safe}. As shown in Figure 5, as ϵ_{safe} decreases, both RRL and DEARRL exhibit similar performance, i.e. a decrease in both the average reward and the proportion of constraint violations. Notice that when ϵ_{safe} is equal, DEARRL demonstrates comparable safety levels and far superior task performance than RRL. DEARRL substantially reduces the influence of the behavior correction mechanism on π_{task}, eventually enabling the algorithm to find a better π_{task}.

3) Decoupled Framework: In DEARRL, π_{rec} and $Q_c^{\pi_{rec}}$ are completely decoupled from π_{task}, so that the pre-trained π_{rec} and $Q_c^{\pi_{rec}}$ can be directly combined with other trained π_{task} to enhance the safety of these algorithms during testing. We combined RRL and DEARRL's behavior correction mechanism with policies acquired from SAC and IQL, and tested them across 500 random episodes (see Table II). The findings indicate that DEARRL can be seamlessly integrated into any RL algorithm, significantly elevating their safety during testing, albeit with a minor decrease in returns. Notice that in some environments, π_{task} and π_{rec} obtained by IQL (with DEARRL) even rival those achieved through online training, despite being trained entirely offline.

Finally, we observe that when combined with the behavior correction mechanism, SAC and IQL show a significant increase in episodic length but a slight decrease in return on testing. This can be attributed to the fact that the behavior correction mechanism modifies the original MDP model for which policies trained using SAC and IQL were initially optimized.

VI. CONCLUSION

In this paper, we propose DEARRL, a fully decoupled safe reinforcement learning framework. We achieve the decoupling of task performance and safety through pre-training recovery policies that maximize safety. The task policies are free to explore without considering safety, and safety in exploration is achieved by combining recovery policies with behavior correction. We show our approach's superiority in identifying the dead-end states in determined MDPs, which defines the maximum range that a policy can explore safely. Through a series of experiments, we demonstrate that our approach strikes a balance between task performance and safety, suggesting its potential for practical applications in diverse environmental tasks.

Future work includes: (1) introducing model-based reinforcement learning to estimate uncertainty, allowing ϵ_{safe} to be adaptive, and (2) obtaining an offline safe reinforcement learning algorithm that involves the behavior correction mechanism in the offline training process of π_{task} to improve task performance.

REFERENCES

[1] 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, pp. 529–533, 2015.

[2] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8248–8254.

[3] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile qt-opt for risk-aware vision-based robotic grasping," *arXiv preprint arXiv:1910.02787*, 2019.

[4] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu, "State-wise safe reinforcement learning: A survey," *arXiv preprint arXiv:2302.03122*, 2023.

[5] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[6] L. Schäfer, F. Christianos, J. Hanna, and S. V. Albrecht, "Decoupling exploration and exploitation in reinforcement learning," in *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

[7] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.

[8] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[9] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," *arXiv preprint arXiv:2010.03152*, 2020.

[10] D. Kim, Y. Kim, K. Lee, and S. Oh, "Safety guided policy optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2462–2467.

[11] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[12] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.

[13] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9797–9806.

[14] M. Luo, A. Balakrishna, B. Thananjeyan, S. Nair, J. Ibarz, J. Tan, C. Finn, I. Stoica, and K. Goldberg, "Mesa: Offline meta-rl for safe adaptation and fault tolerance," *arXiv preprint arXiv:2112.03575*, 2021.

[15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[16] W. F. Whitney, M. Bloesch, J. T. Springenberg, A. Abdolmaleki, K. Cho, and M. Riedmiller, "Decoupled exploration and exploitation policies for sample-efficient reinforcement learning," *arXiv preprint arXiv:2101.09458*, 2021.

[17] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[18] L. Zhang, Z. Yan, L. Shen, S. Li, X. Wang, and D. Tao, "Safety correction from baseline: Towards the risk-aware policy in robotics via dual-agent reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9027–9033.

[19] M. Fatemi, S. Sharma, H. Van Seijen, and S. E. Kahou, "Dead-ends and secure exploration in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1873–1881.

[20] M. Fatemi, T. W. Killian, J. Subramanian, and M. Ghassemi, "Medical dead-ends and learning to identify high-risk states and treatments," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4856–4870, 2021.

[21] T. W. Killian, S. Parbhoo, and M. Ghassemi, "Risk sensitive dead-end identification in safety-critical offline reinforcement learning," *arXiv preprint arXiv:2301.05664*, 2023.

[22] G. Thomas, Y. Luo, and T. Ma, "Safe reinforcement learning by imagining the near future," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 859–13 869, 2021.

[23] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[24] E. Altman, *Constrained Markov decision processes*. CRC press, 1999, vol. 7.

[25] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[26] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[28] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," *arXiv preprint arXiv:1910.01708*, vol. 7, no. 1, p. 2, 2019.

[29] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, "Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 639–10 646.