Model-Free Reinforcement Learning with Safe-Reachability Objectives

Ruohan Zhan, Maxime Bouton, and Mykel J. Kochenderfer

Stanford University, Stanford CA 94305, USA
{rhzhan,boutonm,mykel}@stanford.edu

Abstract. Model-free reinforcement learning (RL) algorithms can be effective in finding policies maximizing accumulated reward in uncertain environments, but it is often challenging to find safe policies. We propose a model-free algorithm to learn a safety mask in an unknown environment modeled as a Markov decision process (MDP). The safety mask provides a quantitative metric of safety for each state-action pair of the MDP, and is learned without requiring external knowledge. This mask is then used to promote safe exploration while training a policy to maximize expected accumulated reward. We demonstrate that our method has mathematical safety guarantees in tabular cases, and we empirically show that it can scale to large and continuous domains and enhance safety.

Keywords: Deep Reinforcement Learning · Safety Constraints · Robotics

1 Introduction

Reinforcement learning (RL) can be used to find approximately optimal decision strategies in environments with unknown dynamics. In safety critical domains, such as autonomous driving or health care, RL methods often do not offer sufficient safety guarantees to be deployed. The agent must learn to avoid dangerous states before trying to optimize for a given reward function. However, encoding dangerous states with very large penalties in the reward function of standard RL algorithms might lead to overly conservative behavior.

In the context of reinforcement learning, safety can be defined in different ways [13], such as variance of rewards [18], temporal difference errors [8], and unsafe states [14]. This paper focuses on unsafe states. We focus on multi-objective scenarios where the goal of the agent is not only to avoid the unsafe states, but also to achieve the best possible performance. Those objectives are referred to as *safe-reachability objectives.* For instance, in an autonomous driving scenario, optimizing solely for collision avoidance may result in a trivial driving policy that never moves. In a safe-reachability problem, success is when the agent reaches a goal state without visiting any unsafe states.

Designing a safe and effective strategy is challenging. External knowledge is often needed to facilitate decision making, and is commonly incorporated in three forms [13, 35]: prior knowledge [23, 24], generalization from a finite set of demonstrations [1], and active expert advising [13, 26, 28, 31, 33]. However, the performance of the algorithm can largely depend on the quality of prior knowledge and demonstration. In addition, it is often not practical to query an expert during training.

In this work, we propose a generic, scalable, safety-masked reinforcement learning methodology that can be easily adapted to a variety of RL algorithms. Our algorithm aims to enhance safety of the learned policy. This paper addresses several challenges associated with solving safe-reachability problems using model-free reinforcement learning. Such challenges include handling sparse rewards and balancing safety and efficiency. We first present an algorithm to approximate the probability of success of the agent in a given state-action pair in continuous states environments. The algorithm is shown to converge in the tabular case. We then show how to use this success probability as a safety mask to filter unsafe actions in a standard reinforcement learning loop maximizing a reward function. The main result of this paper is two deep reinforcement learning algorithms for solving problems with safe-reachability objectives. We refer two those algorithms as two-stage masked RL and joint learning. The former learns a safety mask (based on the probability of success) and a reward maximizing policy in two sequential training procedures, while the latter learns the mask and the policy jointly. The learning algorithms are built upon deep Q-learning [25], but can be easily extended to other value-based RL methods. We empirically demonstrate that this approach can improve safety both during learning and deployment in a variety of domains.

2 Related Work

Recent approaches to safe RL have focused on constraining the action space of the agent to only allow safe exploration using a masking mechanism (also referred to as shield) to enhance safety [3, 6, 10, 11, 16, 19, 21, 22]. Bouton et al. guided learning with a masking mechanism relying on discrete value iteration [6]. This technique requires discretizing the environment and modeling its transition and reward functions. The discretization prevents the method from scaling to domains with large state spaces and makes it difficult to balance computational time, storage of the value iteration mask, and the accuracy of the local approximation. Alshiekh et al. used a deterministic abstraction of the environment to form a safety shield [3]. Safety is only guaranteed if the underlying abstraction of the environment is correct, which can be difficult to define for complex problems. The constructed shield specifies safety on the discrete abstract states, and the extension to continuous state spaces is not straightforward.

Formal methods with linear temporal logic (LTL) objectives are often used to derive such a monitoring system for safe learning [3, 6, 11, 12, 15, 16, 17]. Hasanbeig et al. proposed a synchronised product MDP to incorporate LTL properties by modifying a reward function, which by construction ensures that the search space satisfies the properties [16]. However the learning procedure requires a partition of the state space beforehand based on the LTL property, which might be complicated to achieve. Fulton and Platzer proposed verificationpreserving model updates for heterogeneous environments [12]. The authors constructed a set of candidate models and then selected the one meeting safety guarantees at run time. However, in order to update the models, differential equations need to be synthesized according to collected data, which requires certain assumptions about how the world behaves.

The need for domain specific knowledge and strong modeling assumptions limits the applicability and scalability of some existing methods [3, 6, 12], while other model-free approaches mainly focus on finite MDPs with finite state spaces [15, 16, 21, 22, 34]. The method proposed by Dalal et al. can be applied to continuous space, but is limited to linear approximations to estimate the level of safety of a given state [10].

Contrary to previous work, our algorithm allows us to easily benefit from all the advances in deep reinforcement learning to solve safe-reachability problems. Although we focused on safe-reachability objectives, it has been shown that any LTL satisfaction problem in MDPs can be reduced to safe-reachability problems [4]. One could use the approach proposed by Hasanbeig et al. to build the synchronised MDP [17].

In this work, motivated by previous masking techniques, we propose a model-free reinforcement learning methodology that can learn a safety mask in continuous state spaces and uses it to enhance safety when optimizing for reaching a goal. In contrast with previous work, our approach does not rely on external knowledge or models of the environment. Our proposed deep RL approach scales to complex and continuous domains, and we propose empirical techniques to improve the convergence of deep RL algorithms with safe-reachability objectives.

3 Preliminaries

A Markov Decision Process (MDP) is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the probability the system transitions to s' from state s by taking action a, $R(s, a)$ is the immediate expected reward of taking action a at state s, and $\gamma \in [0, 1]$ is a scalar that discounts future rewards. A policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ specifies the action to take at a current state, and a utility function $U^\pi(s)$ corresponds to the expected accumulated reward obtained by following policy π from state s. In an MDP, a trajectory corresponds to a sequence of state action pairs.

The state-action value function $Q(s, a)$ is defined as the accumulated reward for taking action a at state s and then following an optimal policy. Given $Q(s, a)$, we can obtain the optimal utility $U(s) = \max_a Q(s, a)$ and an optimal policy $\pi(s) = \arg\max_a Q(s, a)$. The Bellman equation for $Q(s, a)$ can be formulated as follows

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{1}$$

Reinforcement learning can be used to solve for an optimal policy when P and R are not known exactly. Training data is sampled from a set of transitions (s, a, r, s') obtained by interacting with the environment. Value function approximation methods such as deep Q-learning (DQN) have been used to solve for policies in large and complex environments with a neural network representation of the value function [25].

4 Problem Formulation

In this work, we are interested in a multi-objective problem. Our algorithm aims at improving both the agent's safety and its accumulated reward. The problem is decomposed into two parts: learning a safety mask and maximizing accumulated reward with masked safe actions. An alternative method to incorporate safety objectives into policy learning is to assign large negative rewards to unsafe states, but choosing a value for those penalties is difficult. It has been shown that using large penalties does not always permit a desirable range of policies and might lead to switching behaviors (either too risky or too conservative) [32]. In contrast, our method formulates a mask to explicitly account for safety and integrates it into policy learning by constraining the agent to only explore actions with sufficient safety.

4.1 Safety of a State Action Pair

Terminal states are divided into *unsafe states* and *goal states.* Safety in this paper is defined as a safe-reachability problem (or constrained reachability) [4, 7] where the objective is to reach goal states without visiting unsafe states. We define unsafe trajectories as trajectories that end in unsafe states. We introduce $P^{\pi}_{\text{safe}}(s, a)$ to represent the probability of reaching a goal state safely under policy π starting from (s, a). Thus, a quantitative metric for safety of a state-action pair can be defined as the maximum probability of reaching the goal safely over all policies:

$$P_{\text{safe}}(s, a) = \max_{\pi} P^{\pi}_{\text{safe}}(s, a). \tag{2}$$

Geibel and Wysotzki proposed a similar approach to estimate the probability of entering unsafe states to facilitate safety [14]. However, avoiding unsafe states is not the only objective of our agent; it has to also reach a goal state. If we define safety to only avoid unsafe states, the agent could wander around and not reach goal states. By defining the safety to be the probability of avoiding unsafe states *and* reaching goal states, we force the agent to reach the goal with the constraints of avoiding unsafe states.

Other works focus on a different formulation of safety, where the constraint is on the additive safety-related costs over the entire trajectory [2, 9]. Our method, on the contrary, focuses on quantifying each state-action pair's probability of reaching a goal state safely to enhance safety.

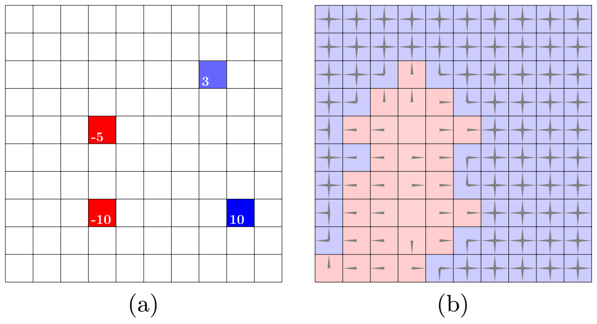

Fig. 1. Illustration of the masking mechanism. The reward function of the environment is shown in Fig. 1(a). To achieve a perfect score, the agent must go to the darker blue cell as fast as possible. We illustrate the safe action set in Fig. 1(b). The arrows represent the available actions in that set to satisfy a certain probability of safety. The red cells are the states where the agent must default to $\arg\max_a P_{\text{safe}}(s, a)$.

4.2 Computing Probability of Safety

To be able to compute P_{safe}, we introduce the following equation based on the Markov property [4]:

$$P_{\text{safe}}(s, a) = \sum_{s'} P(s' \mid s, a) \max_{a'} P_{\text{safe}}(s', a'). \tag{3}$$

Comparing Eq. (3) and Eq. (1), it can be seen that $P_{\text{safe}}(s, a)$ is actually a state-action value function for a specific MDP $\mathcal{M}_{\text{safe}} = (\mathcal{S}, \mathcal{A}, P, R_{\mathcal{S}}, \gamma_{\text{safe}})$, which has the same state space $\mathcal{S}$, action space $\mathcal{A}$ and transition probability P as the original MDP $\mathcal{M}$, but the following two distinctions:

Reward R_{safe} All rewards are 0 except upon reaching a goal state, which gives reward 1. The reward for reaching an unsafe state is also 0, corresponding to zero probability of entering goal states.

Discount factor γ_{safe} Since there is no discount in the probability calculation, we set $\gamma_{\text{safe}} = 1$.

In other words, the value function of $\mathcal{M}_{\text{safe}}$ corresponds to the maximum probability of being safe, which aligns with our definition of P_{safe} in Eq. (2). Note that reaching a goal state sends the agent to an absorbing state, and the value of goal states in $\mathcal{M}_{\text{safe}}$ is 1. Solving $\mathcal{M}_{\text{safe}}$ gives us its state-action value function $P_{\text{safe}}(s, a)$, which quantifies the safety of a state-action pair in our original MDP $\mathcal{M}$. We can apply Q-learning to $\mathcal{M}_{\text{safe}}$ without needing to know the transition model of the original MDP $\mathcal{M}$. Furthermore, if the environment is discrete, our estimate of P_{safe} will converge.

Theorem 1. *Assume that all policies lead to terminal states with probability* 1. *By initializing P_{safe} to zeros everywhere except ones in goal states, Q-learning [35]* *will converge to the true probability of reaching the goal with appropriate learning rate decay and exploration method which is greedy in the limit with infinite visits for each state-action pair (GLIE [29]).*

Details of the proof can be found in Appendix A. Our initialization and no intermediate rewards guarantee that $P_{\text{safe}}^{(k)}$ at the kth update step of Q-learning in $[0, 1]$ for all k. The assumption that all policies lead to terminal states with probability 1 holds naturally for finite horizon problems. If the underlying Markov chain is ergodic, the termination time is bounded universally in discrete space. Under these assumptions, the dynamic programming value iteration operator is a contraction mapping [5], which leads to the convergence of Q-learning [20]. Theorem 1 also applies to other value-based RL algorithms, such as Sarsa(0) [27]. A proof is provided in Appendix A.

4.3 Maximizing Reward under Safety Constraints

Once we compute our quantitative safety metric P_{safe}, we can learn a policy Q to maximize reward for our original MDP $\mathcal{M}$ using standard RL methods. The only modification is *exploration*, which is constrained by a safety mask. The actions available to the agent are filtered using a threshold on P_{safe}. In a given state s, the agent is only allowed to take actions that are safe enough, that is with $P_{\text{safe}}(s, a)$ above some threshold. By construction, this will guarantee the safety of our agent so long as we have an accurate safety metric P_{safe}.

We define a *safety mask* $M = (P_{\text{safe}}, \theta)$ that constrains the exploration, where θ is a safety threshold. For each state s encountered in the trajectory, we define the set of safe actions as follows:

$$\mathcal{A}_{\text{safe}}(s) = \{a \in \mathcal{A} \mid P_{\text{safe}}(s, a) \geq \theta\}. \tag{4}$$

The safety threshold θ controls how conservative the exploration will be. When $\theta = 0$, this reduces to standard RL. When $\theta = 1$, the agent requires fully guaranteed safety. It is possible that $\mathcal{A}_{\text{safe}}(s)$ is empty if the mask is too strict with θ close to 1. In such cases, we can choose the action with the largest probability of safety, that is, $\mathcal{A}_{\text{safe}}(s) = \{\arg\max_{a \in \mathcal{A}} P_{\text{safe}}(s, a)\}$. We summarize our masked exploration in Algorithm 1. Figure 1 illustrates our problem formulation. The agent must learn to maximize reward while operating within the safety constraints imposed by the mask. Experiments with varying θ are provided in Appendix D.

5 Learning Algorithms

As shown in Sections 4.2 and 4.3, we have safety guarantees for the agent in tabular cases. However, in more complex and continuous domains, value function approximation techniques are needed to compute the probability of safety and solve the RL problem. In this section, we propose two scalable learning algorithms based on deep Q-learning (DQN) : *two-staged masked RL* and *joint learning*. Safety is not guaranteed in such scenarios due to function approximation. More details on algorithms and their specific realization can be found in Appendix B, and comparison of these two approaches will be discussed in Section 6.

Algorithm 1 Masked Exploration

```
**Input:** safety mask $M = (P_{\text{safe}}, \theta)$, current state $s$, current value function $Q(s,a)$.
Calculate masked actions $\mathcal{A}_{\text{safe}}(s_j)$.
    $\mathcal{A}_{\text{safe}}(s_j) = \begin{cases} \tilde{A}(s_j) & \text{for } \tilde{A}(s_j), \neq \emptyset \\ \arg\max_{a \in \mathcal{A}} P_{\text{safe}}(s_j, a), & \text{otherwise} \end{cases},$   (5)
where $\tilde{A}(s_j) = \{a \in \mathcal{A} \mid P_{\text{safe}}(s,a) \geq \theta\}$.
Choose action $a \in \mathcal{A}_{\text{safe}}(s_j)$ based on some exploration strategy with $Q(s,a)$.
**Return:** action $a$.
```

5.1 Two-staged Masked RL

Stage one applies DQN to $\mathcal{M}_{\text{safe}}$ to learn the safety metric P_{safe}, and stage two applies DQN to $\mathcal{M}$ under masked exploration Algorithm 1 to learn a policy to maximize a reward function. However, the *sparse reward* and *undiscountalibity* of $\mathcal{M}_{\text{safe}}$ make learning P_{safe} challenging. Extensive exploration is required before reaching non-zero reward states, and the lack of discounting could lead to divergence.

In addition, the learned P_{safe} should have as little effect as possible in learning the reward policy, which means that the mask should only filter unsafe actions to ensure enough exploration of the agent. To ensure that the mask is not too conservative, the learned representation of P_{safe} must be sufficiently accurate, especially at states with large utilities. In the example of autonomous driving, if P_{safe} is large in only conservative actions, such as decelerating or staying idle, on stage two it would take the agent longer to reach the goal, resulting in smaller accumulated reward due to discounting.

To overcome these difficulties, we propose three techniques to augment deep RL methods when applied to $\mathcal{M}_{\text{safe}}$. Experimental parameters can be found in Appendix B.

Gradually increase γ_{safe} to 1 which leads to faster convergence and more exploration in states with large utility.

Less conservative exploration space (compared to solving $\mathcal{M}$), which means larger ϵ for ϵ-greedy exploration, higher temperature T for Boltzmann (softmax) exploration, and larger exploration bonus for upper confidence bound (UCB) exploration.

Sigmoid in the last layer where $S(x) = \frac{1}{1+e^{-x}}$ is applied to the last layer to scale outputs within $[0,1]$, the range of probability values.

After P_{safe} is learned in stage one, it can be used to mask actions in stage two to learn a policy to maximize the accumulated reward.

5.2 Joint Learning

In order to have a minimal impact on the reward policy learning, the learned P_{safe} should be accurate at states with large utilities. This motivates us to propose an approach to jointly learn the probability of safety P_{safe} and the state-action value function Q. We introduce the temporal difference loss function for joint learning as follows:

$$\begin{aligned} L(\omega_{P_{\text{safe}}}, \omega_Q) = \mathbb{E}_{s,a,r,s' \sim \rho(\cdot)} \Bigg[& \lambda \underbrace{\left(\max_{a'} P_{\text{safe}}(s', a') - P_{\text{safe}}(s, a) \right)^2}_{P_{\text{safe}} \text{ loss}} \\ & + \underbrace{\left(r + \max_{a'} Q(s', a') - Q(s, a) \right)^2}_{Q\text{-value loss}} \Bigg] \end{aligned} \tag{6}$$

where $\omega_{P_{\text{safe}}}$ and ω_Q are the parameters of the P_{safe} and Q networks, λ balances safety and rewards, tuples (s, a, r, s') are randomly sampled from a replay buffer $\mathcal{B}$ with distribution ρ.

We use the same replay buffer $\mathcal{B}$ for P_{safe} and Q. During training, the action is chosen based on Q from candidate actions given by P_{safe}. P_{safe} and Q are updated, leading to mutual improvements for both. Masked by P_{safe}, Q is able to avoid unsafe actions that may lead to unsafe states. Sharing the data buffer with Q, P_{safe} is more accurate in states with larger utilities.

Initially, both P_{safe} and Q could be unreliable. Thus, we use a decaying parameter β starting from 1.0 to balance the masking effect. The action candidates are the entire action space $\mathcal{A}$ with probability β, and the masked action set $\mathcal{A}_{\text{safe}}(s)$ with probability $1 - \beta$. At the very beginning, the exploration is similar to that in standard deep Q-learning. As P_{safe} and Q becomes more accurate, β decreases and exploration is more constrained. Eventually, with $\beta = 0$, the algorithms becomes similar to the last stage of two-staged DQN.

6 Experiments

We apply standard deep Q-learning (DQN), two-staged masked RL (Two-DQN) and joint learning (Joint-DQN) to four tasks: Sub Hunt, Continuum World, Navigation, and Urban Driving (Appendix C). The Sub Hunt domain has a discrete state space small enough to be solved with a tabular representation. However, other domains are larger with continuous state spaces and require function approximation. For consistency, we do not change the representation of the policy throughout the experiments. All experiments use a safety threshold of 0.99. Appendix D discusses how to choose an appropriate safety threshold.

6.1 Sub Hunt

The Sub Hunt problem involves a submarine attempting to track and attack an enemy submarine [30]. The game ends when the agent attacks the enemy (resulting in reward 100) or the enemy reaches its goal.

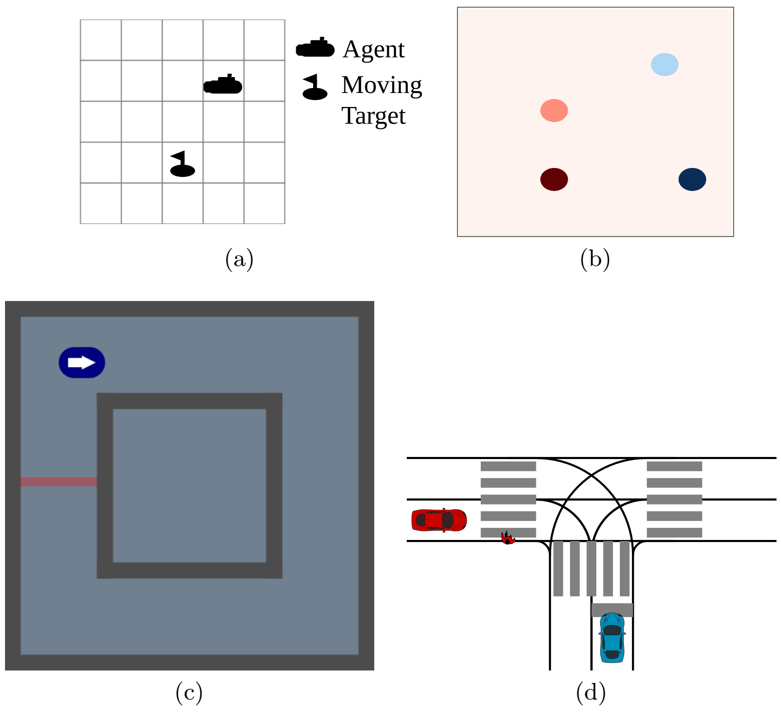

Fig. 2. Illustration of the different domains used for experiments: (a) Sub Hunt, (b) Continuum World, (c) Navigation, and (d) Urban Driving. For Sub Hunt, the world is a 20×20 grid.

The unsafe states are defined to be those where the enemy reaches its goal, and the goal states are those when the submarine successfully attacks the enemy. Table 1 shows that, at test time, Two-DQN and Joint-DQN never encounter an unsafe state, and Two-DQN even obtains larger rewards than standard DQN. During training, Figure 3 (left) shows that with masking no trajectories end in unsafe states, and Figure 3 (right) shows that masking also leads to higher accumulated rewards than standard DQN.

6.2 Continuum World

Continuum World is a continuous state environment illustrated in Figure 2(b). The terminal locations are four circular regions. The orange and dark red circles are unsafe state with medium and large penalties respectively. The light blue and dark blue circles are goal states with medium and large rewards respectively.

Table 1 and Figure 4 show that masked methods perform more safely during both deployment and learning. The masked methods cannot achieve perfect safety (probability of success of one) in this domain due to the stochastic motion of the

Table 1. Average accumulated reward and percentage of trajectories ending in unsafe states in 10 000 simulations.

Task	Accumulated Rewards			unsafe trajectories (%)		
	DQN	Two-DQN	Joint-DQN	DQN	Two-DQN	Joint-DQN
Sub Hunt	83.308	**83.782**	82.500	0.350	**0.000**	**0.000**
continuum World	**6.218**	5.674	6.052	0.870	**0.070**	0.220
Navigation	32.507	39.090	**40.030**	9.770	**0.540**	0.640
Urban Driving	0.498	0.408	**0.501**	0.510	**0.070**	0.310

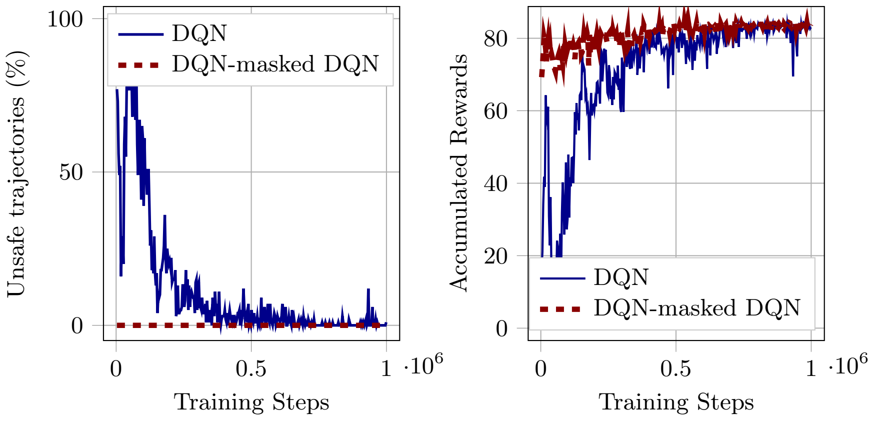

Fig. 3. Percentage of trajectories ending in unsafe states during training (left) and evolution of the accumulated rewards (right) on the Sub Hunt domain. Each point is obtained by freezing the weights and averaging the results over 100 simulations.

agent. In addition, the agent is randomly initialized and might start within an unsafe region which explains most of the unsafe trajectories.

To further investigate how close our learned safety mask is to the ground truth, we visualize the safety probability for each state $U_{\text{safe}}(s) = \max_a P_{\text{safe}}(s, a)$ in Figure 5. Figure 5 (left) shows the probability of success calculated using value iteration on a discretization of the continuum world with a 30×30 grid. We use multi-linear interpolation over the continuum. Figure 5 (right) is obtained using DQN augmented with the techniques described in Section 4.2. This comparison shows how our method, without prior knowledge, approximates the probability of reaching unsafe states in a continuous domain.

6.3 Navigation

The Navigation environment [3] is a continuous state problem where a vehicle must navigate clockwise in a corridor to reach a finish line without hitting the

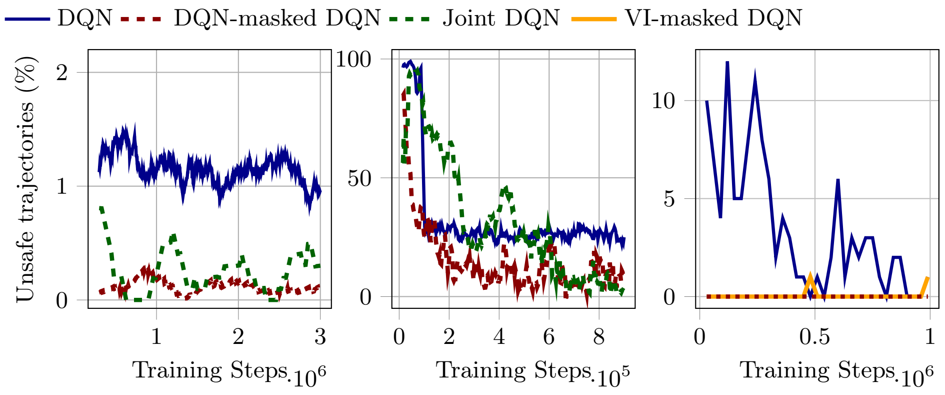

Fig. 4. Percentage of trajectories ending in unsafe states during training on the Continuum World domain (left) and Navigation domain (middle) and Urban driving (right). Each point is obtained by freezing the weights and averaging the results over 100 simulations. VI-masked DQN uses a safety mask computed using value iteration on a discretized environment (urban driving only).

walls illustrated in Figure 2(c). The vehicle is penalized for counterclockwise motions and rewarded for clockwise motions. The unsafe states are those where the vehicle collides with the wall, and goal states are those where the vehicles reaches the finish line.

Table 1 and Figure 4 show that our learned mask can significantly reduce collisions and improve rewards. We notice that standard DQN performs poorly on this task, while masking improves the performance significantly, especially safety. One distinction of the navigation problem from other domains in this paper is that the original MDP has very dense rewards. At each step, the agent receives nonzero rewards by either moving clockwise or counter-clockwise. In the new MDP $\mathcal{M}_{\text{safe}}$, deep exploration is required to expose the agent to the large bonus of reaching the finish line. Standard DQN could be disctracted by greedily accumulating immediate rewards and never realize the importance of reaching the goal. In fact, this behavior was observed in our experiments, and the standard DQN agent only learned to move along a circle to avoid collision with walls. In contrast, learning the mask in the modified MDP, accompanied with our techniques to handle the sparse credit assignment (described in Section 5), the agent learns a more goal-oriented behavior and successfully reaches the goal.

6.4 Urban Driving

The Urban Driving environment [6] is an autonomous driving scenario where an autonomous vehicle must navigate in an intersection with a crosswalk, as illustrated in Figure 2(d). The controlled agent (in blue) must achieve a left-turn while avoiding another vehicle (in red) and a pedestrian. The unsafe states

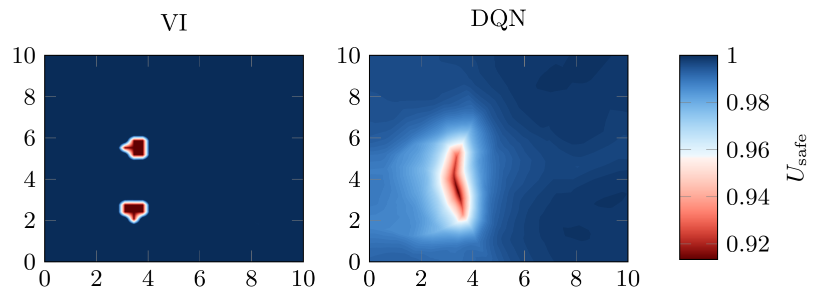

Fig. 5. Visualization of the probability of being safe in the Continuum World environment computed using discrete value iteration and using DQN.

correspond to collisions with other agents, and the goal states have positions at the exit of the intersection.

Table 1 shows that the mask can enhance safety in the deployment phase. To further demonstrate that our learned safety mask P_{safe} is a good approximation to the ground truth, we compare our learned mask with the one computed with discrete value iteration. We used multi-linear interpolation to generalize the discrete values over the whole state space [6]. The value iteration mask requires knowledge of transition probabilities and reward function, while our method does not require any prior information about the environment. Moreover, the discretization necessary to perform value iteration does not scale well in this environment. By choosing resolutions of $2\,\text{m}$ for the position and $2\,\text{m}\,\text{s}^{-1}$ for the velocity of each agent, the number of states grows to approximately 23×10^6. As shown in Figure 4, both masked methods enhance safety in the learning phase, while our method prevails with zero unsafe states. It is likely that the few collisions observed with the discrete mask are due to modeling inaccuracies introduced by discretization.

6.5 Discussion

The experiments above demonstrate the ability of our approach to enhance safety in both learning and execution for safety critical problems. Our methods are sometimes outperformed by standard DQN in terms of rewards, which is expected when conservative behavior could compromise efficiency.

We also notice differences between the two proposed algorithms. During learning, as the estimation of the safety mask becomes more accurate, joint-DQN reaches a similar safety level as Two-DQN, while at the beginning of learning, it has similar performance as standard RL. At test time, Joint-DQN achieves higher rewards but has slightly more unsafe trajectories than Two-DQN. One reason for this is that Two-DQN trains the policy with a pre-trained mask, which constrains the exposed actions within safe candidates from the start. Joint-DQN sacrifices safety at the beginning of the training but achieves higher rewards due to more flexible exploration.

Our proposed algorithms rely on function approximation, and do not provide mathematical safety guarantees. Moreover, learning the safety mask for safety critical problems requires the availability of a simulator. However, we empirically demonstrated that our methods improve safety, without requiring external knowledge, in domains that would be intractable for algorithms requiring discretization or strong modeling assumptions.

There are two main restrictions of our approach. First, we only consider the unsafe states as terminal states. A more general extension would be controlling the entire cost related to safety of the whole path under some acceptable threshold. Second, it is difficult to verify how close our learned mask from function approximation method is to the real mask, which is indeed the price for the scalability of DQN.

We also want to note the difference between almost-sure safety and maximal safety, while the latter is considered in this work. Almost-sure safety means that the state is safe for all available actions, and the maximal safety means that the state is safe if there exists one safe action. In practice, almost-sure safety is hard to achieve since there could always exist hazardous actions leading to unsafe states. In contrast, maximal safety is more practical since safe actions could always be eligible to lead the agent away from unsafe states. With the given formulation of our safety mask, the safety guarantee holds at one step only, and the errors can accumulate. The probability of safety given by the safety-mask assumes that the agent will follow the safest policy at the next time step, which is not the case in practice. However, we showed empirically that this approach can filter actions to improve safety significantly over standard RL.

7 Conclusion

In this paper, we incorporate safety into deep reinforcement learning algorithm by defining a safety mask to filter unsafe actions. Safety is defined as the capability of reaching goal terminal states while avoiding unsafe terminal states. Without requiring external knowledge about the environment, we formulate a new MDP that when solved allows us to quantify the safety of each state-action pair. We show that, in tabular cases, our formulation guarantees safety. To deal with complex and continuous domains, we suggest two approaches based on value function approximation for solving safe-reachability objectives. We demonstrate empirically that our method can approximate the ground truth safety mask and improve safety in both the training and the deployment processes.

Future work involves extending the method to more advanced deep RL algorithms to improve the performance of our methods. In this work, we only focused on using neural networks for value approximation, but other classes of functions could be used to represent the safety mask. Although we used a model-free approach, model-based approaches could help relax the need for a fixed safety threshold. The safety mask could be relaxed in states where the domain is known.

Bibliography

[1] Abbeel, P., Coates, A., Ng, A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. International Journal of Robotics Research 29(13), 1608–1639 (2010)

[2] Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International Conference on Machine Learning (ICML). pp. 22–31 (2017)

[3] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: AAAI Conference on Artificial Intelligence (AAAI) (2018)

[4] Baier, C., Katoen, J.: Principles of Model Checking. MIT Press (2008)

[5] Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation: numerical methods, vol. 23. Prentice hall Englewood Cliffs, NJ (1989)

[6] Bouton, M., Karlsson, J., Nakhaei, A., Fujimura, K., Kochenderfer, M.J., Tumova, J.: Reinforcement learning with probabilistic guarantees for autonomous driving. In: UAI Workshop on Safety, Risk and Uncertainty in Reinforcement Learning (2018)

[7] Brázdil, T., Chatterjee, K., Chmelík, M., Fellner, A., Křetínský, J.: Counterexample explanation by learning small strategies in Markov decision processes. In: International Conference on Computer-Aided Verification. pp. 158–177 (2015)

[8] Campos, P., Langlois, T.: Abalearn: Efficient self-play learning of the game abalone. INESC-ID, Neural Networks and Signal Processing Group (2003)

[9] Chow, Y., Nachum, O., Duenez-Guzman, E., Ghavamzadeh, M.: A Lyapunov-based approach to safe reinforcement learning. In: Advances in Neural Information Processing Systems (NIPS). pp. 8092–8101 (2018)

[10] Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., Tassa, Y.: Safe exploration in continuous action spaces. arXiv preprint arXiv:1801.08757 (2018)

[11] Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In: AAAI Conference on Artificial Intelligence (AAAI) (2018)

[12] Fulton, N., Platzer, A.: Verifiably safe off-model reinforcement learning. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 413–430 (2019)

[13] Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research 16(1), 1437–1480 (2015)

[14] Geibel, P., Wysotzki, F.: Risk-sensitive reinforcement learning applied to control under constraints. Journal of Artificial Intelligence Research 24, 81–108 (2005)

[15] Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 395–412 (2019)

[16] Hasanbeig, M., Abate, A., Kroening, D.: Logically-constrained reinforcement learning. arXiv preprint arXiv:1801.08099 (2018)

[17] Hasanbeig, M., Abate, A., Kroening, D.: Logically-constrained neural fitted q-iteration. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 2012–2014 (2019)

[18] Heger, M.: Consideration of risk in reinforcement learning. In: Machine Learning Proceedings, pp. 105–111. Elsevier (1994)

[19] Isele, D., Nakhaei, A., Fujimura, K.: Safe reinforcement learning on autonomous vehicles. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)

[20] Jaakkola, T., Jordan, M.I., Singh, S.P.: Convergence of stochastic iterative dynamic programming algorithms. In: Advances in Neural Information Processing Systems (NIPS). pp. 703–710 (1994)

[21] Jansen, N., Könighofer, B., Junges, S., Bloem, R.: Shielded decision-making in mdps. arXiv preprint arXiv:1807.06096 (2018)

[22] Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.P.: Safety-constrained reinforcement learning for mdps. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 130–146 (2016)

[23] Koppejan, R., Whiteson, S.: Neuroevolutionary reinforcement learning for generalized control of simulated helicopters. Evolutionary Intelligence 4(4), 219–241 (2011)

[24] Martìn, J.A., de Lope Asiaìn, J.: Learning autonomous helicopter flight with evolutionary reinforcement learning. In: International Conference on Computer Aided Systems Theory. pp. 75–82 (2009)

[25] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)

[26] Polo, F.J.G., Rebollo, F.F.: Safe reinforcement learning in high-risk tasks through policy improvement. In: IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL). pp. 76–83 (2011)

[27] Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. University of Cambridge (1994)

[28] Saunders, W., Sastry, G., Stuhlmueller, A., Evans, O.: Trial without error: Towards safe reinforcement learning via human intervention. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 2067–2069 (2018)

[29] Singh, S., Jaakkola, T., Littman, M.L., Szepesvári, C.: Convergence results for single-step on-policy reinforcement-learning algorithms. Machine Learning 38(3), 287–308 (2000)

[30] Sunberg, Z.N., Kochenderfer, M.J.: Online algorithms for POMDPs with continuous state, action, and observation spaces. In: International Conference on Automated Planning and Scheduling (ICAPS) (2018)

[31] Thomaz, A.L., Breazeal, C., et al.: Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In: AAAI Conference on Artificial Intelligence (AAAI) (2006)
[32] Undurti, A., How, J.P.: An online algorithm for constrained pomdps. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 3966–3973 (2010)
[33] Walsh, T.J., Hewlett, D.K., Morrison, C.T.: Blending autonomous exploration and apprenticeship learning. In: Advances in Neural Information Processing Systems (NIPS). pp. 2258–2266 (2011)
[34] Wang, Y., Chaudhuri, S., Kavraki, L.E.: Bounded policy synthesis for pomdps with safe-reachability objectives. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 238–246 (2018)
[35] Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge (1989)

Appendix A: Convergence in Tabular Cases

The formulation of $\mathcal{M}_{\text{safe}}$ is equivalent to setting all intermediate rewards to zero while assigning utilities for goal states and error states to ones and zeros, respectively. In tabular cases, the state and action spaces are finite. We initialize $P^{(0)}_{\text{safe}}$ to zeros everywhere except ones in goal states.

Q-learning

Given transition (s, a, s'), consider the Q-learning update rule [35]:

$$P^{(k+1)}_{\text{safe}}(s,a) = (1 - \alpha_{(k)}(s,a))P^{(k)}_{\text{safe}}(s,a) + \alpha_{(k)}(s,a)\max_{a'} P^{(k)}_{\text{safe}}(s',a') \tag{7}$$

with $\sum_k \alpha_{(k)}(s,a) = \infty$ and $\sum_k \alpha^2_{(k)}(s,a) < \infty$ uniformly w.p.1. Since there are no intermediate rewards, the variance of reward function is zero, and $P^{(k)}(s,a)$ is bounded uniformly within $[0,1]$ by induction. For finite horizon problems, all policies lead to terminal states w.p.1. Also, if the underlying Markov chain is ergodic, the termination time is bounded uniformly in discrete space. Therefore, the value iteration operator is a contraction mapping under a weighted maximum norm [5]. With all these, theorem 2 in Jaakkola et al. [20] gives us that $P^{(k)}_{\text{safe}}(s,a)$ converges to the optimal $P_{\text{safe}}(s,a)$ values under the Q-learning update (Eq. (7)).

Appendix B: Pseudo-code of the Learning Algorithms

Two-staged masked RL and joint learning are proposed to learn mask P_{safe} and action-value function Q. We illustrate these two approaches in Fig. 6.

Two-staged Masked RL

The first stage is to learn safety metric P_{safe}. We propose three ad hoc techniques to deal with training difficulties. Specifically, in our experiments, we linearly increase γ_{safe} from γ (discount factor of the original MDP $\mathcal{M}$) to 1. In terms of exploration, we use ϵ-greedy method. The parameter ϵ is linearly decreased from 1 to a small number ϵ_{end}, and then maintained for the rest of exploration. We use a larger ϵ_{end} when solving $\mathcal{M}_{\text{safe}}$ compared to $\mathcal{M}$. The algorithm is summarized in Algorithm 2. After we learn safety metric using Algorithm 2, we can apply masked exploration to learn action-value function Q. The algorithm is summarized in Algorithm 3.

Joint Learning

The joint learning approach is to train P_{safe} and Q simultaneously. Masked by P_{safe}, Q can explore more safely. By leveraging Q's exploration, P_{safe} can be more representative in states with larger utilities, and thus least interfere in Q's exploration. We summarize the algorithm in Algorithm 4.

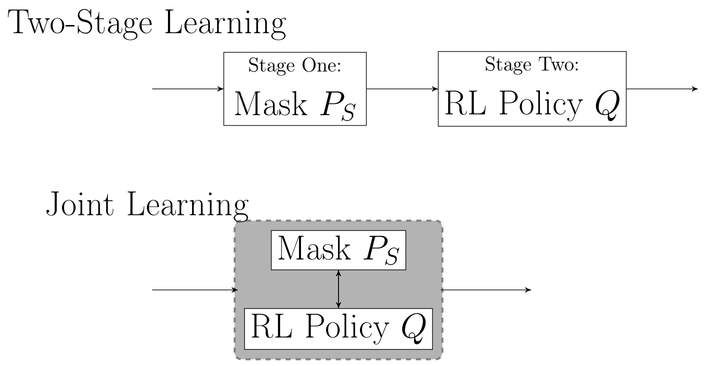

Fig. 6. Diagram of two masked RL algorithms. The upper one shows the two-stage masked RL learning method, the lower one shows the joint learning method.

Algorithm 2 First Stage of Two-staged Masked RL: Computing P_{safe}

```
Initialize replay buffer $\mathcal{B}$
Initialize state-action safety measure $P_{\text{safe}}$ with random weights, and apply sigmoid function $S(x)$ to the last layer
Set $\epsilon^{begin} = 1, \epsilon^{end} = 0.05, \gamma_S^{begin} = \gamma, \gamma_S^{end} = 1$
for $i = 1$ to $M$ do
    $\epsilon \leftarrow \epsilon^{begin} + (\epsilon^{end} - \epsilon^{begin}) \min(\frac{2i}{M}, 1)$
    $\gamma_{safe} \leftarrow \gamma_{safe}^{begin} + (\gamma_S^{end} - \gamma_S^{begin}) \frac{i}{M}$
    Choose action $a_i$ with $\epsilon$-greedy
    Add new transition tuple $(s_i, a_i, r_i, s_{i+1})$ to replay buffer $\mathcal{B}$
    Update $P_{\text{safe}}$ for $\mathcal{M}_{\text{safe}} = \{\mathcal{S}, \mathcal{A}, P, R_S, \gamma_{safe}\}$ with replay buffer data $\mathcal{B}$
end for
Choose a proper safety threshold $\theta$.
Return: safety mask $\mathbb{M} = (P_{\text{safe}}, \theta)$
```

Algorithm 3 Second Stage of Two-staged Masked RL: Computing Q with P_{safe}

```
Compute safety mask $\mathbb{M} = (P_{\text{safe}}, \theta)$ using algorithm 2
Initialize replay buffer $\mathcal{B}$
Initialize action-value function $Q$
for $j = 1$ to $N$ do
    For state $s_j$, calculate masked actions
        $$\tilde{A}(s_j) = \{a | a \in \mathcal{A}, P_{\text{safe}}(s, a) \geq \theta\}$$
        $$\mathcal{A}_{\text{safe}}(s_j) = \begin{cases} \tilde{A}(s_j) & \text{for } \tilde{A}(s_j) \neq \emptyset \\ \left\{\arg\max_{a \in \mathcal{A}} P_{\text{safe}}(s_i, a)\right\} & \text{otherwise} \end{cases} \quad (8)$$
    Choose action $a_j \in \mathcal{A}_{\text{safe}}(s_j)$ based on certain exploration strategy
    Add transition $(s_j, a_j, r_j, s_{j+1})$ to replay buffer $\mathcal{B}$
    Update $Q$ with replay buffer data $\mathcal{B}$
end for
Return: action-value function $Q$
```

Algorithm 4 Joint Learning of a Safety Mask and a Policy

```
Initialize replay buffer $\mathcal{B}$
Initialize safety measure $P_{\text{safe}}$ and action-value function $Q$
$\beta = 1$
for $j = 1$ to $N$ do
  For state $s_j$, calculate masked actions
      $\tilde{A}(s_j) = \{a | a \in \mathcal{A}, P_{\text{safe}}(s, a) \geq \theta\}$
      $\mathcal{A}_{\text{safe}}(s_j) = \begin{cases} \tilde{A}(s_j) & \text{for } \tilde{A}(s_j) \neq \emptyset \\ \left\{\arg\max_{a \in \mathcal{A}} P_{\text{safe}}(s_i, a)\right\} & \text{otherwise} \end{cases}$   (9)
  With probability $\beta$, choose action $a_j \in \mathcal{A}$ randomly; with probability $1 - \beta$, choose
  action $a_j \in \mathcal{A}_{\text{safe}}(s_j)$ based on certain exploration strategy
  Add transition $(s_j, a_j, r_j, s_{j+1})$ to replay buffer $\mathcal{B}$
  Update $P_{\text{safe}}, Q$ with replay buffer data $\mathcal{B}$ based on joint loss function
end for
Return: action-value function $Q$
```

Appendix C: Implementation of the Experimental Domains

We apply our approaches to four tasks: Sub Hunt, Continnum World, Navigation, and Urban Driving. The environments are deterministic for navigation, and stochastic for the other three.

Sub Hunt The agent in the sub hunt problem has six actions: move three steps north, south, east or west, attack the target, or use active sonar. The enemy randomly moves either two steps toward the goal or one step forward and one step to the side. The agent can only attack the enemy within a range of 2. The discount is 0.95.

Continuum World The domain of continuum world is a 2D square of $[0, 10] \times [0, 10]$. Four terminal locations are circular regions of radius 0.5 centered at $(3.5, 2.5), (3.5, 5.5), (8.5, 2.5), (7.5, 7.5)$ with rewards -10.0, -5.0, 10.0, and 3.0 respectively. The agent can go up, down, left and right with step length of 1.0. The motion is perturbed by a Gaussian distribution with standard deviation 0.5. The discount is 0.95.

Navigation The state is the position and orientation of the agent, and is in a three dimensional continuous space. The reward function consists of a penalty -1 for counterclockwise movement, a reward $+1$ for clockwise movement, a bonus $+100$ for reaching a finish line, and a cost -100 for hitting the wall similarly as Alshiekh et al. [3]. The agent moves at constant speed and can control its heading by choosing between four different steering commands from $-45°$ to $45°$. The discount is 0.95.

Urban Driving In this domain, the agents include ego car, another car and one pedestrian. Each entity is represented by a four dimensional variable corresponding to position (2D), heading, and longitudinal velocity. The state space is continuous with twelve dimensions. The discount is 0.95.

Appendix D: Safety Threshold

It is a natural question to ask how the safety threshold will influence the performances, since we choose a high safety threshold of 0.99 on all tasks. Since our algorithm approximates the maximum of all P_{safe}^{π} over all the policy π, the learned P_{safe} functions on many cases (e.g. urban driving, continuum world) are very close to 1.0 as we observed in the experiments.

We did an empirical comparison of different safety thresholds θ on urban driving, and chose θ to be 0.95, 0.96, 0.97, 0.98, 0.99. Average collisions, rewards and steps of Two-DQN and VI-DQN based on 10 000 testing simulations are illustrated in figure 7. When θ is not so strict(about 0.95), the masked RL has a similar performance as standard RL. When θ increases, the mask becomes stricter, hence both collision rate and accumulated rewards decrease, while the number of steps to reach the goal increases. We also notice that DQN-mask is more sensitive to θ, and has the comparable safety masking effects with VI-mask with large safety threshold $\theta = 0.98, 0.99$. This could be due to the training difficulties of DQN-mask we mentioned before. Actions with larger P_{safe} values could converge faster. With larger θ, safe actions $\mathcal{A}_{\text{safe}}(s)$ from VI-mask and DQN-mask can have more overlap.

However, empirically, increasing θ larger than 0.99 will not change the collision rate and the accumulated rewards noticeably. Thus, to fully demonstrate the effectiveness of the safety mask and least interfere in reward policy learning, we choose $\theta = 0.99$. In the case where the set of allowed actions is empty, the algorithm would then choose the safest one, which is also based on the learned safety metric P_{safe}. Our experiments demonstrate that such a backup strategy, if used, can still effectively enhance safety. The choice of safety threshold is domain specific.

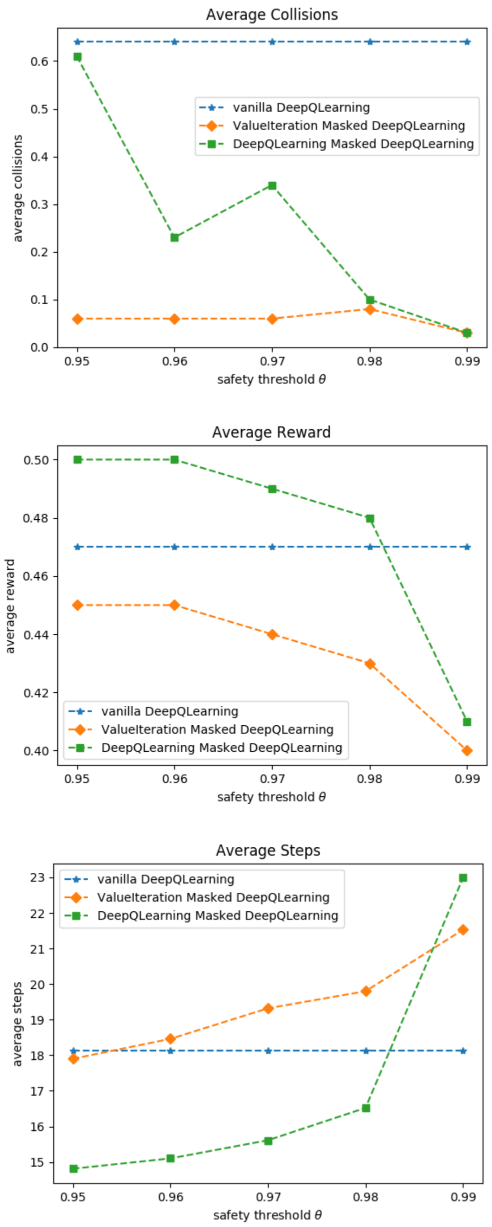

Fig. 7. Influence of safety threshold θ on the performance in Urban Driving. We compare standard deep Q-learning, value iteration masked deep Q-learning and two-staged deep Q-learning. Each point is obtained over 10 000 simulations.