Safety-aware Policy Optimisation for Autonomous Racing

Bingqing Chen[1] **Jonathan Francis**[1,2] **James Herman**[1]
Jean Oh[1] **Eric Nyberg**[1] **Sylvia L. Herbert**[3]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2]Human-Machine Collaboration, Bosch Research Pittsburgh, Pittsburgh, PA 15222
[3]Safe Autonomous Systems, University of California, San Diego, CA 92093
{bingqinc, jmf1, jamesher, jeanoh, ehn}@cs.cmu.edu, sherbert@ucsd.edu

Abstract: To be viable for safety-critical applications, such as autonomous driving and assistive robotics, autonomous agents should adhere to safety constraints, throughout interactions with their environments. Constrained reinforcement learning methods that depend on collecting samples from the environment for policy evaluation cannot guarantee safety during initial phases of learning. In comparison, methods such as Hamilton-Jacobi (HJ) reachability compute safe sets with theoretical guarantees, using models of the system dynamics. However, these guarantees hinge on the quality of the model, which may not capture the true dynamics. Furthermore, existing works on HJ reachability exclusively study problems defined on physical states, e.g., pose, instead of high-dimensional sensory inputs, such as RGB images. In this paper, we demonstrate that the HJ safety value can be learned directly on vision context, thereby expanding HJ reachability to applications where dynamics models may not be available. We evaluate our method on `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment. While not necessary for convergence, we warm-start the safety value function using values pre-computed with a nominal model. Next, the value function is updated directly on transitions of ego-agent's frontal camera view and vehicle speed. We report new state-of-the-art results on the `L2R` benchmark task, and we show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase. We release our code and dataset in the supplementary material.

Keywords: Safe Reinforcement Learning, Hamilton-Jacobi Reachability, Autonomous Driving

1 Introduction

Without safety constraints, autonomous agents are not guaranteed to adhere to safe behaviours, throughout interactions with their environment. In the context of safety-critical control applications, such as autonomous driving and human-robot interaction, recent literature has been concerned with studying how to best learn policies that are simultaneously safety-aware and performant. In the reinforcement learning (RL) literature, it is common to study constraint satisfaction under the constrained Markov decision process (CMDP) framework [1], which extends the Markov decision process (MDP) by incorporating constraints on expected cumulative costs. That is to say, aside from maximising expected cumulative rewards, the agent must ensure that the cumulative costs do not exceed pre-specified limits. The primary challenge of solving a CMDP problem is the need to evaluate whether a policy will violate the constraints [2]. Methods, such as constrained policy optimisation (CPO) [2], projection-based constrained policy optimisation (PCPO) [3] and conservative safety critic (CSC) [4], depend on collecting samples from the environment for policy evaluation. As a result, safety cannot be guaranteed, most notably during the initial learning interactions [5].

Given the practical limitations of letting agents learn safety by experiencing failures, it may be favourable to bootstrap the learning process with a model. Methods, such as Hamilton-Jacobi (HJ)

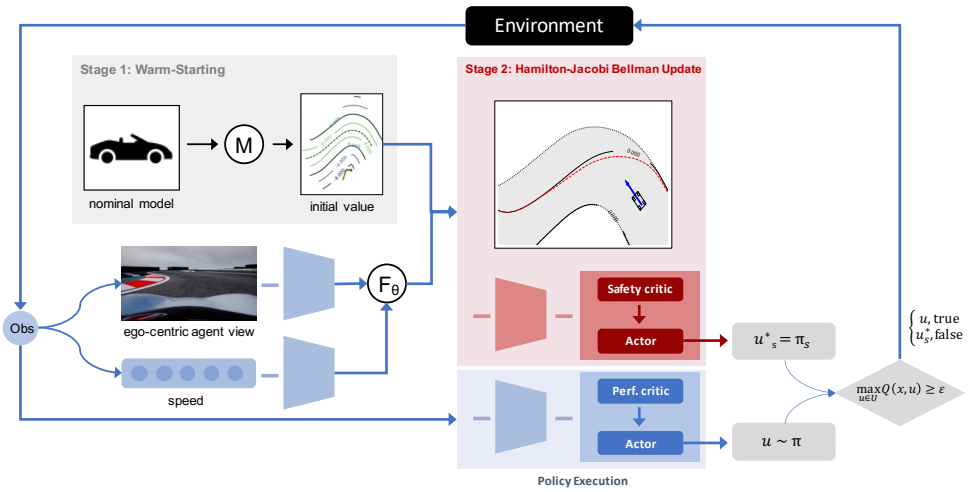


Figure 1: Summary of SPAR. We decompose the problem of learning under safety constraints into optimising for performance and updating safety value function. Thus, SPAR consists of two policies, both implemented with actor-critic architecture, which are in charge of safety and performance independently. First, we warm-start the safety critic, using values pre-computed with a nominal model. Then, we refine the safety critic based on observations from the environment, i.e., frontal camera view and speed, using HJ Bellman update. Simultaneously, we optimise the performance policy. At each time step, the safety critic verifies if the current state is safe, and only intervenes when necessary.

reachability, compute safe sets with theoretical guarantees, using models of the system dynamics. In our autonomous racing experiments, we demonstrate that a safe set that is pre-computed with a simple kinematic vehicle model [6] can empirically keep even an agent that is generating actions at random on the drivable area, given a sufficiently large safety margin. For many engineering applications, such nominal models can often be specified by domain experts, despite complexity in the true dynamics.

In this paper, we combine constrained RL with HJ Reachability theory in our problem formulation. Since safety verification under HJ Reachability theory does not depend on the performance policy, we can decompose the problem of learning under safety constraints into optimising for performance and updating safety value function. Given this intuition, we learn two independent policies (as shown in Figure 1), both implemented with actor-critic architecture, that manages safety and performance independently. While the performance policy focus exclusively on optimising performance, the safety critic verify if the current state is safe, and intervenes when necessary. The safety critic is updated via Hamilton-Jacobi Bellman update [7], grounded in control theory, and the safety controller is updated via the gradients from the safety critic. At the same time, the performance controller solves an unconstrained optimisation problem and thus can be trained with any appropriate RL algorithm.

Aside from the problem formulation and corresponding framework, our key contributions are as follows. Firstly, we build upon existing work in neural approximation of HJ Reachability and demonstrate that safety value function can be updated directly from visual context, thereby expanding HJ reachability to applications where explicit models may not be available. We evaluate our methods on `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment, which requires the vehicle to make safety-critical decision in a fast changing environment. We report new state-of-the-art results on the `L2R` benchmark task, and we show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

2 Related Work

Safe reinforcement learning. There is growing interest in enforcing some notion of safety in RL algorithms, e.g. satisfying safety constraints, avoiding worst-case outcomes, or robust to environmental stochasticity [8]. Some examples include methods that bound expected costs characterising violations of safety constraints under the CMDP formulation [2, 3], methods that combine control

theory and RL [5, 7, 9, 10], and methods that enable the agent to explore with probabilistic safety guarantees under uncertain environment [11, 12, 13].

In this work, we focus on the notion of safety as constraint satisfaction. CMDP [1] is a widely-used framework for studying RL under constraints, where the agent maximises cumulative rewards, subject to limits on cumulative costs characterising constraint violations. Solving a CMDP is challenging, because the policy needs to be optimised over the set of feasible ones (given the constraints). This requires evaluation of the constraint functions to determine whether a policy is feasible [2]. CPO [2], PCPO [3], and CSC [11, 4] upper bound the expected cost via estimation from samples collected from interacting with the environment. As a result, safety grows with experience but requires externally validated safe initial learning interactions [5]. Regardless of this limitation, we are inspired by CSC, which uses a safety critic to verify if a state is safe. In our work, we instead update the safety critic using a learning rule informed by HJ reachability theory [7], to be elaborated in Section 4.

Guaranteed safe control. Guaranteeing the safety of general continuous nonlinear systems is challenging, but there are several approaches that have been successful. These methods typically rely on knowledge of the dynamics (potentially with bounded uncertainty over some parameters), as well as the environment.

Control barrier functions (CBFs) provide a measure of safety with gradients that inform the acceptable safe actions [14]. Recent work can guarantee safety in the face of bounded disturbances on the system [15, 16]. For specific forms of dynamics and unlimited actuation bounds, this approach can be scalable to higher-dimensional systems and can be paired with an efficient online quadratic program for computing the instantaneous control [5]. Unfortunately, finding a valid control barrier function for a general system with control bounds is a nontrivial task.

Hamilton-Jacobi (HJ) reachability is a technique that uses continuous-time dynamic programming to directly compute a value function that captures the optimal safe control for a system [17, 18]. This method can provide hard safety guarantees for general nonlinear systems subject to bounded uncertainties and disturbances. There are two major drawbacks to HJ reachability. The first is that the technique suffers from the curse of dimensionality and scales exponentially with number of states in the system. Because of this, the technique can only be used directly on systems of up to 4-5 dimensions. When using specific dynamics formulations and/or restricted controllers, this upper limit can be extended [19, 20]. Second, because of this computational cost, the value function is typically computed offline based on assumed system dynamics and bounds on uncertainties. This can lead the safety analysis to be invalid (if, for example, the environment is different than expected), or overly conservative (if the bounds on disturbances/uncertainties were over-approximated).

Neural approximations of HJ Reachability. To resolve the first issue of scalability for HJ reachability, there has been recent work on neural approximations of value functions [21, 22, 23, 24] and controllers [25]. We are inspired by recent work [7, 26] that bridges the mathematical framework of HJ reachability with reinforcement learning applied to deep Q-learning. This method loses hard safety guarantees, but enables scalable safety-aware Q-learning. Building on top of [7, 26], which exclusively focus on learning the safety value function, we simultaneously learn an performance controller along with the safety critic.

Learning safety using HJ Reachability To resolve the second issue of adapting safety analysis in the face of violated assumptions (e.g. changes in environment, system, or disturbances), we are motivated by recent work on updating the safety analysis based on new information [18, 27, 28]. By merging these techniques for updating the value function with our neural approximation for Q-learning, we can push the scalability of this approach for more realistic experimental platforms.

3 Preliminaries

Constrained MDPs. The problem of reinforcement learning (RL) with safety constraints is often formulated as a CMDP. On top of the MDP $(\mathcal{X}, \mathcal{U}, R, \mathcal{F})$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\mathcal{F} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ characterises the system dynamics, and $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, CMDP includes an additional set of cost functions, $C_1, \ldots, C_m$, where each $C_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ maps state-action transitions to costs characterising constraint violations.

The objective of RL is to find a policy $\pi : \mathcal{X} \to \mathcal{U}$ that maximises the expected cumulative rewards, $V_R^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k R(x_k, u_k) | x_0 = x\right]$, where $\gamma \in [0, 1)$ is a tem-

poral discount factor. Similarly, the expected cumulative costs are defined as $V_{C_i}^{\pi}(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k C_i(x_k, u_k) | x_0 = x\right]$. CMDP requires the policy to be feasible by imposing limit for the costs, i.e. $V_{C_i}(\pi) \leq d_i, \forall i$. Putting everything together, the RL problem in a CMDP is:

$$\pi^* = \arg\max_{\pi} \; V_R^{\pi}(x) \quad \text{s.t.} \quad V_{C_i}^{\pi}(x) \leq d_i \;\; \forall i \tag{1}$$

HJ Reachability. To generate the safety constraint, we use HJ reachability applied to a simple, low-fidelity model of the vehicle kinematics, denoted as $\dot{x} = f(x, u)$. Here x is the state, u is the control contained within a compact set $\mathcal{U}$. The dynamics are assumed bounded and Lipschitz continuous. For discrete-time approximations the time step $\Delta t > 0$ is used.

We denote all allowable states as $\mathcal{K}$, for which there exists a terminal reward $l(x)$, such that $x \in \mathcal{K} \iff l(x) \geq 0$. An $l(x)$ that satisfy this condition is the signed distance to the boundary of $\mathcal{K}$. For instance, $\mathcal{K}$ is the drivable area and $l(x)$ is the distance to road boundary or obstacle in an autonomous driving scenario. This set $\mathcal{K}$ is the complement of the failure set that must be avoided. The goal of this HJ reachability problem is to compute a safety value function that maps a state to its safety value with respect to $l(x)$ over time. This is done by capturing the minimum reward achieved over time by the system applying an optimal control policy:

$$V_S(x, T) = \sup_{u(\cdot)} \min_{t \in [0, T]} l(\xi_{x,T}^{u,d}(t)), \tag{2}$$

where ξ is the state trajectory, $T < 0$ is the initial time, and 0 is the final time. To solve for this safety value function, a form of continuous dynamic programming is applied backwards in time from $t = 0$ to $t = T$ using the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI):

$$\min\left\{\frac{\partial V_S}{\partial t} + \max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle, l(x) - V(x, t)\right\} = 0, \quad V_S(x, 0) = l(x). \tag{3}$$

The super-zero level set of this function is called the reachable tube, and describes all states from which the system can remain outside of the failure set for the time horizon. For the infinite-time, if the limit exists, we define the converged value function as $V_S(x) = \lim_{T \to -\infty} V_S(x, T)$.

Once the safety value function is computed, the optimal safe control can be found online using the Hamiltonian: $u_S^* = \arg\max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle$. This safe control is typically applied in a least-restrictive way wherein the safety controller becomes active only when the system approaches the boundary of the reachable tube, i.e. $u \sim \pi$ if $V_S(x, T) \geq 0$, u_S^* otherwise.

The newly introduced discounted safety Bellman equation [7] modifies the HJI-VI in (3) in a time-discounted formulation for discrete time:

$$V_S(x) = (1 - \gamma) l(x) + \gamma \min\left\{l(x), \max_{u \in \mathcal{U}} V(x + f(x, u)\Delta t\right\}, \quad V_S(x, 0) = l(x). \tag{4}$$

This formulation induces a contraction mapping, which enables convergence of the value function when applied to approximate dynamic programming schemes commonly used in RL.

4 SPAR: Safety-aware Policy Optimisation for Autonomous Racing

In this section, we describe our framework for safety-aware policy optimisation. We are inspired by guaranteed-safe methods, such as Hamilton-Jacobi reachability, which provides a systematic way to verify safety. Thus, we formulate our problem as a combination of constrained RL and HJ reachability theory. The need for an accurate model of the system dynamics can be restrictive, particularly for applications where such a model is difficult to develop or not available. Building upon existing work on neural approximation of HJ Reachability [7], we demonstrate that it is possible to directly update the safety value function on high-dimensional multimodal sensory input—thereby expanding the scope of applications to problems previously inaccessible.

We highlight the notable aspects of our framework:

i) Leverages domain knowledge. Aside from utilising HJ reachability theory for updating the safety critic, we also warm-start the safety value function with values pre-computed via a nominal model. We demonstrate in Section 6 that despite the naivety of the assumed model, it can empirically keep a

random agent on the racetrack with a sufficiently large safety margin. This demonstrates the benefit of bootstrapping the learning process with readily-available domain knowledge.

ii) Scales to high-dimensional problems. Compared to standard HJ Reachability methods, whose computational complexity scales exponentially with the state dimension, we successfully updated the safety value directly from vision embedding. In comparison, the largest systems, to the best of our knowledge, studied via HJ reachability theory has 30 states [29].

(iii) Simultaneously learns performance and safety. We formulate our problem as a combination of CMDP and HJ Reachability theory. As the result of the formulation, the problem of learning under safety constraints can be decomposed as optimising for performance and updating safety value estimation. Thus, we simultaneously, but independently, optimise a performance controller and a safety controller, both implemented with an actor-critic architecture. The safety backup controller is applied in a least restrictive way, only intervening when the RL agent is about to enter into an unsafe state and thus allowing the RL agent maximum freedom in exploring safely.

Problem formulation. We formulate our problem as a combination of CMDP and HJ Reachability theory. As described in Section 3, the objective of CMDP is to maximise cumulative rewards, subject to limits on cumulative costs characterising constraint violations. Without loss of generality, we can interpret the negative of a cost as a reward for safety and reverse the direction of the inequality constraint. Also recall that the super-zero set of the safety value function, i.e., $\{x|V_s(x) \geq 0\}$, designates all states that can remain within the set of allowable states, $\mathcal{K}$, over infinite time horizon. Thus, the safety value function derived from HJ reachability theory can be naturally embedded in CMDP formulation (Eqn. 5):

$$\pi^* = \arg\max_{\pi} \quad V_R(x), \quad \text{s.t.} \quad V_S(x) \geq \epsilon, \tag{5}$$

where $\epsilon \geq 0$ is a safety margin. Note that the condition of $V_S(x) \geq \epsilon$ does not depend on the policy, and as long as the the policy consists of a safety controller that intervenes when $V_S(x) < \epsilon$, the policy is feasible for the CMDP. Instead of solving a constrained optimisation problem, learning under safety constraints can be decomposed as optimising for performance and updating safety value estimation, which is arguably easier.

Update of Safety Value Function. For the update of safety value function, we adopt the learning rule proposed by [7] (Eqn. 6). Note that $Q(x, u)$ is updated model-free using state action transitions, i.e., (x, u, x'). The only domain knowledge required is the function $l(x)$ that corresponds to the constraint set $\mathcal{K}$, in this case defining the drivable area.

$$Q(x, u) = (1 - \gamma)l(x) + \gamma \min\{l(x), \max_{u' \in \mathcal{U}} Q(x', u'\} \tag{6}$$

While convergence is theoretically proved in [7], we compare empirically the convergence of different implementations on two classical control benchmarks, *Double Integrator* and *Dubins' Car*. Due to the page limit, we summarise the key observations here and present the experiments in the supplementary material. It is common practice in RL algorithms to keep a copy of slow-moving target network Q' for stability [30], due to the circular dependency between value estimate and policy. The target network is commonly updated with $Q' \leftarrow \tau Q + (1 - \tau)Q'$, where τ is generally a small number in (0, 1]. In this case, since the safety controller is used least-restrictively, the replay buffer largely consists of samples collected by the performance controller. As a result, the target network for the safety controller can move faster than typically implemented for RL. We also implemented the clipped double Q-learning trick, used in both TD3 [30] and SAC [31], which did not improve upon vanilla Q-learning. We hypothesis that's because 1) $V_s(x) = \max_{u \in \mathcal{U}} Q_s(x, u)$ for HJ safety value, and 2) the target is already clip by $l(x)$. Thus, overestimation bias [30] is less of a concern.

5 Experiments: Autonomous Racing

Overview. In this paper, we evaluate our approach using the Arrival Autonomous Racing Simulator, through the newly-introduced and OpenAI-gym compliant `Learn-to-Race` (`L2R`) task and evaluation framework [32]. `L2R` provides multiple simulated racing tracks, modelled after real-world counterparts, such as Thruxton Circuit in the UK (`Track01:Thruxton`; see Figure 2) and the North Road Track at Las Vegas Motor Speedway in the US (`Track02:Vegas`). In each episode, a racing agent is spawned on a given track. At each time-step, the agent may have access to RGB images from

(a) Aerial (b) Third-person (c) Ego-view

Figure 2: We use the Arrival Autonomous Racing Simulator, within the `Learn-to-Race` (`L2R`) framework [32]. This environment provides simulated racing tracks that are modelled after real-world counterparts, such as the famed Thruxton Circuit in the UK (`Track01:Thruxton`, (a)). Here, learning-based agents can be trained and evaluated according to challenging metrics and realistic vehicle and environmental dynamics, making `L2R` a compelling target for safe reinforcement learning. Each track features challenging components for autonomous agents, such as sharp turns (visualised in (b)), where SPAR only use ego-camera views (shown in (c)) and speed.

any specified location, semantic segmentation, and vehicle state (e.g., pose, velocity) and uses its choice of information to determine normalised steering angle and acceleration. The main objective is to complete laps in as little time as possible. Additional metrics are defined to evaluate driving quality. Complementary to other benchmarks, the high-speed nature of `Learn-to-Race`, coupled with its realistic simulated dynamics, allows for a comprehensive study of Safe RL methods. In this section, we describe the evaluation of our approach, **S**afety-aware **P**olicy Optimisation for **A**utonomous **R**acing (`SPAR`), against a series of benchmarks and research goals.

Task: Learning to Race with Safety Constraints. Our objective is to train RL agents to adhere to safety constraints, i.e. successfully complete laps without safety violations, while being as performant as possible. To characterise the empirical safety performance of our approach, we report results according to the following metrics, in dedicated ablation experiments: the Average Adjusted Track Speed (*AATS*) and the Episode Completion Percentage (*ECP*) metrics [32] as proxies for agent performance and empirical safety, respectively. Other metrics on driving qualities as defined by `L2R`is presented in the supplementary materials.

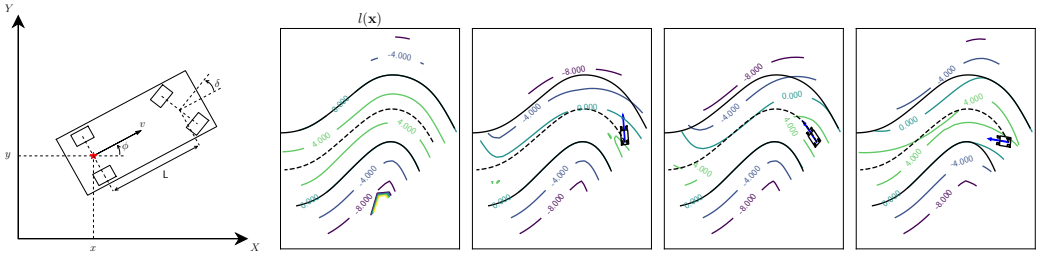
We use `Track01:Thruxton` in `L2R` (Fig. 2) for all stages of agent interaction with the environment. During training, the agent is spawned at random locations along the race track and uses a stochastic policy. During evaluation, the agent is spawned at a fixed location and uses a deterministic policy. The episode terminates when the agent successfully finishes a lap, leaves the drivable area, collides with obstacles, or does not progress for a number of steps. For each agent, we report averaged results across 5 random seeds evaluated every 5000 steps over an episode, i.e., one lap.

5.1 Warm-starting Value Estimation

While warmstart initialisation is not necessary for convergence of the safety value update, we want to demonstrate that the incorporation of readily available domain knowledge can drastically boost safety performance, especially during the initial learning phase. To do that, we first compute the safety value using a nominal model, and then pretrain the safety value function by learning the mapping from ego-view image and speed to HJ safety value.

Constructing the safe set via kinematic vehicle model. We use the kinematic vehicle model [6] to compute the safety value (see Figure 3a), which is very simplistic compared to a realistic race car model. In comparison, [33] implements develops a high-fidelity vehicle model identified through extensive testing, and additionally characterise unmodeled dynamics via Gaussian process.

The dynamics and optimal safety control from solving the Hamiltonian is given in Equation 7, where the state is $\mathbf{x} = [x, y, v, \phi]$, and the action is $\mathbf{u} = [a, \delta]$. x, y, v, ϕ are the vehicle's location, speed, and yaw angle. a is the acceleration, and δ is the steering angle. $L = 3m$ is the car length. Setting $V_s(x, 0) = l(x)$, we calculated the backward reachable tube using the code base by [34]. For efficient computation, we divided the racetrack into overlapping segments and computed the safety value segment by segment. Fig. 3b illustrates resulting safety value function at slices of state space, when the vehicle enters into a sharp turn.



(a) nominal model (b) $V_s(x, y, v, \phi)$ computed via the nominal model, where v=12m/s

Figure 3: (a) We compute the safety value function, via a kinematic vehicle model. (b) We illustrate different views of the 4D state space, given fixed velocity and three different yaw angles, indicated by the blue arrows.

$$\begin{cases} \dot{x} &= v\cos(\phi) \\ \dot{y} &= v\sin(\phi) \\ \dot{v} &= a \\ \dot{\phi} &= v\tan\delta/L \end{cases}, \qquad a^* \begin{cases} \underline{a} & \text{if } \nabla_v V \leq 0 \\ \overline{a} & \text{else} \end{cases}, \qquad \delta^* \begin{cases} \overline{\delta} & \text{if } \nabla_\theta V \geq 0 \\ \underline{\delta} & \text{else} \end{cases} \tag{7}$$

We characterise the empirical performance of this safety value function by comparing an agent that takes actions at random (`Random`) with a random agent that is coupled with our static safety backup controller (`SafeRandom`). We evaluate `SafeRandom`, through a series of safety margins, i.e., $\{3.2, 3.8, 4.2, 4.6, 5.0\}$. Empirically, $\epsilon \geq 4.2$ is sufficient to ensure the vehicle do not go out of bound at least in the speed range covered by the `SafeRandom` agent. Finally, we collected $\sim$300K state-action transitions with the `SafeRandom` agent to pretrain the safety critic in `SPAR` as a regression task.

5.2 Online Safety

We pursue multiple experimental objectives, to illustrate the effect of safety constraints on learning. We use soft-actor critic (SAC) [31] as the base policy class for all learning agents.

Learning to race with safety constraints. We examine the effect of imposing safety constraints on performance and sample efficiency, by comparing an RL agent (`SAC`) with an instance of itself that is coupled with the static safety controller (`SafeSAC`), using the pre-computed safety value function.

Continual safe value function updates. In this ablation, we study the effect of continual refinements of the safety value, using high-dimensional multimodal information, while optimising for driving performance. We want to show that the safety critic can learn about environment dynamics directly from vision context, and with a better characterisation of the safety value function, the agent no longer depend on a large safety margin. We compare the performance of `SafeSAC` (static safety backup) with our proposed `SPAR` agent, which dynamic updates its safety value function.

For the `SafeSAC` agent, we set the safety margin ϵ to be 4.2, which is selected based on empirical safety performance of the `SafeRandom` agent to account for unmodeled dynamics. The `SPAR` agent uses a less conservative safety margin of 3.0, since it is going to refine the safety value function. We include model implementation details in the supplementary.

6 Results

The performance comparison between different agents are summarised in Fig. 4 and Table 1.

Safety value function precomputed with the nominal model provides a good initial estimate. While the kinematic vehicle model is a significant simplification of realistic dynamics, it provides a reasonable initial estimate that can ensure a random agent maintain on the racetrack with a sufficiently large margin (Table 1).

Static safety backup controllers with large safety margin lead to overly conservative policies. Without the ability to perform dynamic updates, `SafeSAC` depends on a conservative safety margin to remain safe. Not surprisingly, over-conservatism compromises performance. Despite its high

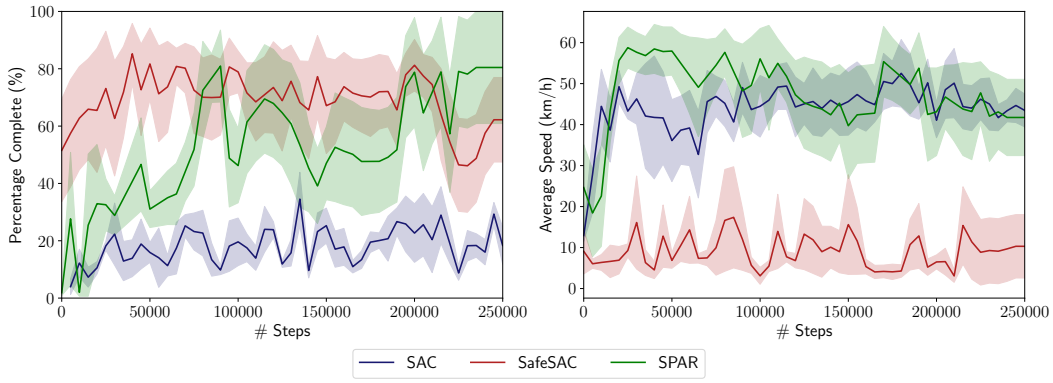


Figure 4: **Left:** Episode percent completion and **Right:** speed evaluated every 5000 steps over an episode (a single lap) and averaged over 5 random seeds. Results reported based on `Track01:Thruxton` in the `Learn-to-Race` environment [32].

Table 1: `Learn-to-Race` task [32] results on `Track01` (Thruxton Circuit), with respect to the task metrics, Episode Completion Percentage (**ECP**) and Average Adjusted Track Speed (**AATS**). Arrows (↑↓) indicate directions of better performance.

Agent	**ECP** (%) (↑)	**AATS** (km/h) (↑)
`HUMAN`	100.0 ± 0.0	171.2 ± 3.4
`Random`	0.5 ± 0.3	11.9 ± 3.8
`SafeRandom`	100 ± 0.0	10.9 ± 2.43
`SAC`	18.8 ± 13.9	41.1 ± 11.9
`SafeSAC` (ours)	62.2 ± 29.5	10.3 ± 15.4
`SPAR` (ours)	**80.4**$\pm$39.1	**41.7**$\pm$18.6

average episode completion percentage (ECP), `SafeSAC` agents exhibit extremely low speeds. This conservatism even compromises ECP in the end, as episodes terminate early due to non-progression.

SPAR agent learns to avoid unsafe scenarios, while driving reasonably fast. As the `SPAR` agent gains more experience, it learns to avoid unsafe scenarios, and thus the ECP ramps up very quickly, along with the speed. For example, at initialisation the agent tend to collides with a sidewall perhaps due lower visual contrast between the track and the inadmissible region, but it quickly learns that those states are unsafe. But, its worth-noting that the speed of the `SPAR` agent start to decrease after 50K steps, which we hypothesise is due to increased awareness of unsafe scenarios and more frequent braking as a result. While `SPAR` outperforms other baselines, there is still significant performance gap with `HUMAN`, which calls for further research.

7 Discussion, Limitations, and Conclusion

In this paper, we introduced a method that enables an agent to learn safety-aware and performance-oriented behaviour. Instead of solving a constrained optimisation problem, we effectively decompose the problem of learning under safety constraints into two more-tractable sub-tasks: optimising for performance and updating safety value. We demonstrated that an approximate HJ safety value can be learned directly on visual context, thereby expanding HJ reachability to applications where dynamics models may not be available. We evaluated our methods on `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment: we showed that, by updating the safe set, agents can maintain a higher level of safety than standard SAC methods while learning to drive quickly.

Whereas our empirical results demonstrated that it is possible to learn a safety-aware and performant policy, through approximation of the minimum payoff distance and through subsequent dynamic updates to the safety value function, one limitation of our method is that it is by no means free from failure, even for high safety margins $\epsilon \geq 4.2$. However, the method proposed in this paper represents a subtle shift away from safety constraint-satisfaction exclusively through model-free exploration,

as has become popular in the recent literature. Rather than letting agents learn safe behaviours through experiencing failures, our approach provides avenues for potential online safety analysis, as in the injection of domain knowledge (when available; e.g., using a nominal dynamics model for warm-starting safety value estimation).

Acknowledgments

We thank the reviewers for their detailed reviews, and we thank Jaime Fisac and Mo Chen for useful comments and discussions. This work was supported, in part, by a doctoral research fellowship from Bosch Research and computing resources from Honda Research and CMU's Parallel Data Lab.

References

[1] E. Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

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

[4] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[5] 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*, volume 33, pages 3387–3395, 2019.

[6] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[7] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

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

[9] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708*, 2018.

[10] K. P. Wabersich and M. N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *arXiv preprint arXiv:1812.05506*, 2018.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[12] S. Dean, S. Tu, N. Matni, and B. Recht. Safely learning to control the constrained linear quadratic regulator. In *2019 American Control Conference (ACC)*, pages 5582–5588. IEEE, 2019.

[13] S. Huh and I. Yang. Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach. *arXiv preprint arXiv:2002.10126*, 2020.

[14] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[15] Q. Nguyen and K. Sreenath. Optimal robust control for constrained nonlinear hybrid systems with application to bipedal locomotion. In *2016 American Control Conference (ACC)*, pages 4807–4813. IEEE, 2016.

[16] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

[17] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[18] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[19] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018.

[20] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020.

[21] S. Bansal and C. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. *arXiv preprint arXiv:2011.02082*, 2020.

[22] V. R. Royo and C. Tomlin. Recursive regression with neural networks: Approximating the hji pde solution. 2016.

[23] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin. Using neural networks to compute approximate and guaranteed feasible hamilton-jacobi-bellman pde solutions. *arXiv preprint arXiv:1611.03158*, 2016.

[24] B. Djeridane and J. Lygeros. Neural approximation of pde solutions: An application to reachability computations. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3034–3039. IEEE, 2006.

[25] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin. A classification-based approach for approximate reachability. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7697–7704. IEEE, 2019.

[26] A. K. Akametalu, S. Ghosh, J. F. Fisac, and C. J. Tomlin. A minimum discounted reward hamilton-jacobi formulation for computing reachable sets. *arXiv preprint arXiv:1809.00706*, 2018.

[27] S. L. Herbert, S. Bansal, S. Ghosh, and C. J. Tomlin. Reachability-based safety guarantees using efficient initializations. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4810–4816. IEEE, 2019.

[28] S. Herbert, J. J. Choi, S. Sanjeev, M. Gibson, K. Sreenath, and C. J. Tomlin. Scalable learning of safety guarantees for autonomous systems using hamilton-jacobi reachability. *arXiv preprint arXiv:2101.05916*, 2021.

[29] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.

[30] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[31] 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*, pages 1861–1870. PMLR, 2018.

[32] J. Herman, J. Francis, S. Ganju, B. Chen, A. Koul, A. Gupta, A. Skabelkin, I. Zhukov, M. Kumskoy, and E. Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. *arXiv preprint arXiv:2103.11575*, 2021.

[33] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[34] G. Giovanis, M. Lu, and M. Chen. Optimizing dynamic programming-based algorithms. https://github.com/SFU-MARS/optimized_dp, 2021.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Convergence Analysis on Classical Control Benchmarks

The objective of this section is to examine how different implementation of the learning rule proposed by [7], i.e., $Q(x,u) = (1-\gamma)l(x) + \gamma \min\{l(x), \max_{u' \in \mathcal{U}} Q(x', u'\}$, affect convergence.

We evaluate it on two classical control benchmarks, *Double Integrator* and *Dubins' Car*, as described in Section A.1. The specific questions we attempt to answer are:

- *Is a slow-moving target network necessary for convergence?* It is common practice in RL algorithms to keep a copy of slow-moving target network Q' for stability [30], to avoid the accumulation of residual error and divergent updates that result from the circular dependency between value estimate and policy. The target network is commonly updated with $Q' \leftarrow \tau Q + (1-\tau)Q'$, where τ is generally a small number in (0, 1]. We examine how different value of τ affects convergence.
- *Is the Clipped Double Q-learning technique helpful?* The Clipped Double Q-learning is a technique that was popularised by TD3 [30] and also adopted in SAC [35]. The technique addresses the overestimation of the value function by keeping two value networks and computing the Bellman backup, with the smaller estimate of the two. We examine if the same technique is conducive to learning the safety value.

A.1 Model Dynamics

Double Integrator. The double integrator models a particle moving along the x-axis at velocity v. The control input is the acceleration a. The goal in this case is keep the particle within a fixed boundary, in this case $x \in [-1, 1]$, subject to $a \in [-1, 1]$.

$$\begin{cases} \dot{x} &= v \\ \dot{v} &= a \end{cases} \tag{8}$$

Dubins' Car. The Dubins' car models a vehicle moving at constant speed, in this case $v = 1$. Similar to the kinematic vehicle model, x, y, ϕ describes the position and heading of the vehicle, and control input is the turning rate $u \in [-1, 1]$. The goal is to reach a unit circle centred at the origin.

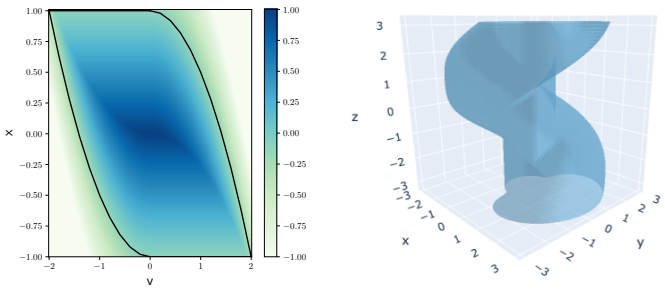
$$\begin{cases} \dot{x} &= v\cos(\phi) \\ \dot{y} &= v\sin(\phi) \\ \dot{\phi} &= u \end{cases} \tag{9}$$

Figure 5 shows the ground truth safety value function for the two benchmark tasks, as calculated using the code at [34].

Implementation & evaluation. We use a neural network with hidden layers of size [16, 16] for the double integrator and [32, 32, 32] for Dubins' car. We use a learning rate of 0.001, batch size of 64, and ADAM as the optimiser [36]. We update the safety value function over 10,000 steps and report the accuracy of the learned safety value function in classifying whether a state is safe or not every 500 steps.

A.2 Results

A slow-moving target network is not necessary for convergence. As we can see in Figure 6, the safety value converged without problem using $\tau = 1$, which is equivalent to not keeping a target network at



(a) Double Integrator (b) Dubins' Car

Figure 5: Ground truth safety value of the two control benchmarks. (a) shows the value function, where the black line indicates $V_S(x) = 0$, and (b) shows the isosurface of the value function at 0, i.e., $V_S(x) = 0$.

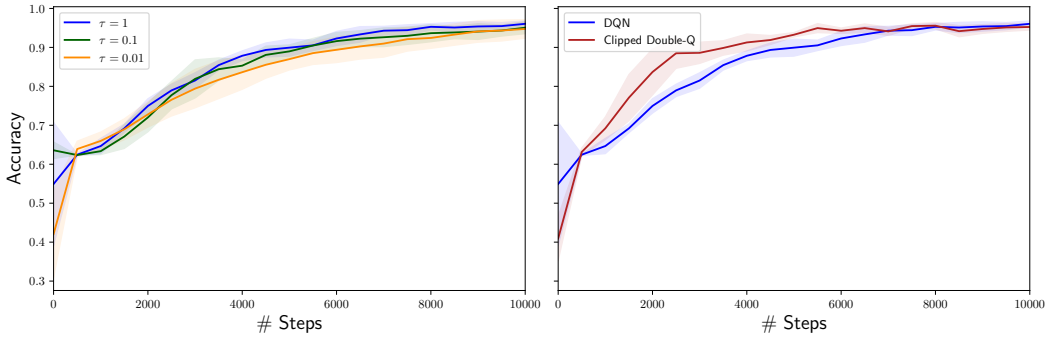


Figure 6: Performance of different implementations on the Double Integrator

all. In fact, it converged slightly faster than using $\tau = 0.1$ and 0.01. We hypothesise that is because the actions in the replay buffer are mostly taken by the performance actor and thus independent of the safety critic, removing the circular dependency between value estimation and policy. This is consistent with the observation in [30] that value estimate without target network is convergent under a fixed policy.

Clipped Double Q-learning has some, but limited benefits. In Figure 6, we also provide a comparison on vanilla DQN vs. clipped double Q-learning. While the clipped double Q-learning converged faster initially, the asymptotic performance is about the same. We hypothesise that the safety Bellman backup already clip the Q at next state with $l(x)$ and thus, the clipped double Q-learning does not have drastic improvements in comparison to vanilla DQN.

B SPAR Algorithm

SPAR relies on a dual actor-critic structure. One of the actor-critic instances functions as a performance policy, while the other functions as a safety policy. This pairing of a safety- and performance-oriented control is important, as we are able to decompose the problem of learning under safety constraints into optimising for performance and updating the safety value function, separately.

We optimise the performance policy using SAC, but it may be switched for any other comparable RL algorithms. While we use the clipped Double-Q technique for the performance critic as in standard SAC implementation, we do not use the technique for the safety critic based on our observations in Section A. We still keep a slow-moving target network for the safety critic, even though we use a τ that's a magnitude bigger compared to that of the performance critic.

The safety policy is used least-restrictively, that is only intervene when the RL agent is about to enter into an unsafe state and thus allowing the performance policy maximum freedom in exploring safely. Instead of using the optimal safe policy from solving Hamiltonian, the safe policy is updated via gradients through the safety critic, same as other actor-critic algorithms.

Algorithm 1: SPAR: Safety-aware Policy Optimisation for Autonomous Racing

Initialise: performance critic Q_{ϕ_1}, Q_{ϕ_2} and actor π_θ;
Initialise: safety critic Q_{ϕ_S}, and actor π_{θ_S};
Initialise: target networks $\phi'_1 \leftarrow \phi_1$, $\phi'_2 \leftarrow \phi_2$, $\theta' \leftarrow \theta$, $\phi'_S \leftarrow \phi_S$, $\theta'_S \leftarrow \theta_S$;
Initialise: replay buffer $\mathcal{M}$;
for *i = 0, . . . , # Episodes* **do**
  x = env.reset()
  while *not terminal* **do**
    u $\sim \pi_\theta$;
    // The safe actor intervenes when the current state-action is deemed unsafe by the safety critic. **if** $Q_{\phi_S}(x, u) \geq \epsilon$ **then**
      $u = u$
    else
      $u = \pi_{\theta_S}$
    end
    x', r = env.step(u)
    $\mathcal{M}$.store(x, a, x', r)
    Update performance critic Q_{ϕ_1}, Q_{ϕ_2} and actor π_θ with preferred RL algorithm, e.g. SAC;
    Sample N transitions (x, u, x') from $\mathcal{M}$;
    // Update the safety critic:
    Calculate the target value with the discounted Bellman safety update

$$y = (1-\gamma)l(x) + \gamma \min\{l(x), \max_{u' \in \mathcal{U}} Q_{\phi'_S}(x', u')\}$$

$$\mathcal{L}_{\phi_S} = N^{-1} \sum (Q_{\phi_S}(x, u) - y)^2$$

$$\phi_S \leftarrow \phi_S - \alpha \nabla_{\phi_S} \mathcal{L}_{\phi_S}$$

    // Update the safety actor with deterministic policy gradient:

$$\theta_S \leftarrow \theta_S + \alpha N^{-1} \sum \nabla_u Q(x, u) \nabla_{\theta_S} \pi_{\theta_S}(x)$$

    // Update the target networks:

$$\phi'_S \leftarrow \tau\phi_S + (1-\tau)\phi'_S, \quad \theta'_S \leftarrow \tau\theta_S + (1-\tau)\theta'_S$$

  end
end

C Additional Implementation Details

Training details. Interaction between a learning agent and the simulator is facilitated by instantiating two simulator environment instances: one as a training environment and another a testing environment. During the training regime, an agent executes state transitions in the training environment for 5000 number of steps, after which point the agent undergoes evaluative interactions with the test simulator instance, then returns to the training simulator instance. We set the max episode steps to be 50,000.

As a consequence of its interaction with the environment, the agent receives a raw RGB image frame and a 30-dimensional state vector as its observation, at each time-step. The agent encodes the RGB image frame and its speed to a 40-dimensional feature representation, subsequently used as input to both actor-critic networks. During its action-selection for the current step, the agent performs a logical evaluation of safety via the safety backup controller: if the agent is found to be in a safe state, given its observation, the action from the performance policy is used; otherwise, the action from the safety backup controller is taken, with transition (s, a, r, s') added to the replay buffer in either case.

We initialise the replay buffer with 2000 random transitions. After 2000 steps, we perform multiple policy updates at each time step. First, we impose an MSE loss against the Bellman backup, for the performance controller's target Q-networks.

Next, with the parameters for the target Q-networks frozen, we perform the policy-update step as an entropy-regularised loss on the actor's prediction of action distribution, given the current observation. We update target networks through polyak averaging.

Finally, we update the safety controller's Q-networks by imposing an MSE loss against the target from the safety Bellman backup. We update the target network through polyak averaging.

Hyperparameters. We summarise all network hyperparameters in Table 2. For all experiments, we implemented models using the `PyTorch` deep learning library, version 1.8.0. We directly utilised standard implementations of the Adam optimiser [36], with a learning rate of 0.003. During training, we used a batch size of 265 for all implementations, with total step sizes of 250,000 and replay buffer sizes of 250,000 elements.

Computing hardware. For rendering the simulator and performing local agent verification and analysis, we used a single GPU machine, with the following CPU specifications: Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz; 1 CPU, 4 physical cores per CPU, total of 4 logical CPU units. The machine includes a single GeForce GTX TITAN X GPU, with 12.2GB GPU memory. For generating multi-instance experimental results, we used a cluster of three multi-GPU machines with the following CPU specifications: 2x Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz; 80 total CPU cores using a Cascade Lake architecture; memory of 512 GiB DDR4 3200 MHz, 16x32 GiB DIMMs. Each machine includes 8x NVIDIA GeForce RTX 2080 Ti GPUs, each with 11GB GDDR6 of GPU memory. Experiments were orchestrated on the these machines using Kubernetes, an open-source container deployment and management system.

All experiments were conducted using version 0.3.0.137341 of the Arrival Racing Simulator.

Table 2: Network Hyperparameters

Operation	Input (dim.)	Output (dim.)	Parameters
	VISUAL ENCODER		
Conv2d	(N, chan, 42, 144), chan : 3→32	`conv1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv1`, chan : 32→64	`conv2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv2`, chan : 64→128	`conv3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv3`, chan : 128→256	`conv4`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Flatten	—	—	—
	VISUAL ENCODER BOTTLENECK REPRESENTATION		
Linear (mu)	$N \times$ h_dim	$N \times 32$	—
Linear (sigma)	$N \times$ h_dim	$N \times 32$	—
	VISUAL DECODER (only for pre-training Visual Encoder)		
Unflatten	—	—	—
ConvTranspose2d	`encoder.conv4`: encoder.conv4.chan: 256 →128	`convtranspose1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose1`, chan : 128 →64	`convtranspose2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose2`, chan : 64 →32	`convtranspose3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose3`, chan : 32 →3	`convtranspose4`	k:=(4,4), s:=2, p:=1, activation:=Sigmoid
	SAFETY ACTOR-CRITIC		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	PERFORMANCE ACTOR-CRITIC		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	ACTOR NETWORK (POLICY): SQUASHEDGAUSSIANMLPACTOR		
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear (projection: mu_layer)	$N \times 32$	$N \times 3$	—
Linear (projection: log_std_layer)	$N \times 32$	$N \times 3$	—
	Q FUNCTION		
`speed_encoder`	—	—	—
`regressor`	—	—	—
	SPEED ENCODER		
Linear	$N \times 1$	$N \times 8$	activation:=ReLU
Linear	$N \times 8$	$N \times 8$	activation:=Identity
	REGRESSOR		
Linear	$N \times 42$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 1$	activation:=Identity

Table 3: Performance of the Pre-trained Safety Critic

	F1	Precision	Recall
$V_S(\mathbf{x}) \leq 3$	0.65	0.74	0.59
$V_S(\mathbf{x}) \leq 4$	0.83	0.78	0.88
$V_S(\mathbf{x}) \leq 5$	0.96	0.96	0.96

D Additional Results

Performance of the `SafeRandom` agent. Recall that the `SafeRandom` agent takes random actions and uses the safety value function precomputed from the nominal model. The optimal safety controller intervene whenever the safety value of the current state falls belong the safety margin. The safety margin is necessary because 1) the nominal model is a significant over-simplification of vehicle dynamics, and 2) the HJ Reachability computation does not take into consideration of the physical dimension of the vehicle.

The performance of the `SafeRandom` agent at different safety margin is summarised in Figure 7. For safety margin $\epsilon \geq 4.2$, the `SafeRandom` agent can reliably complete laps, and thus we use $\epsilon = 4.2$ as the safety margin for the `SafeSAC` agent. On the other hand, the performance decrease drastically when the safety margin is reduced to 3.

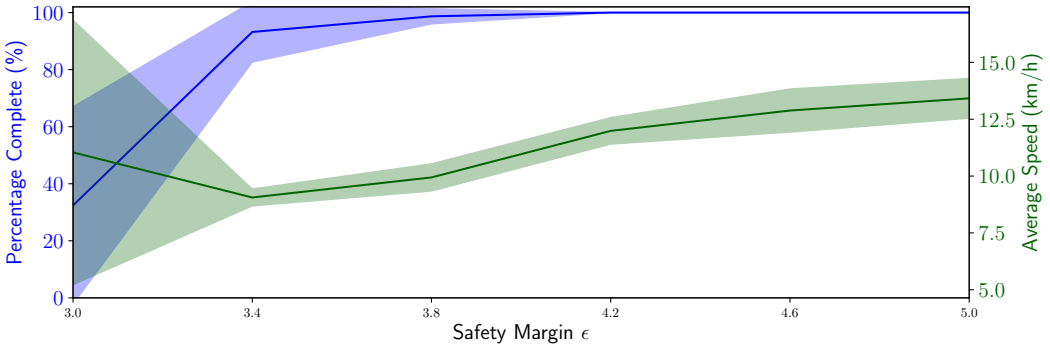


Figure 7: Performance of the `SafeRandom` agent at different safety margin (averaged over 5 random seeds)

Pre-training the safety critic. We pre-trained the safety critic on 300,000+ state-action pairs collected by the `SafeRandom` agent. Recall the state consists of the front camera view and speed, and the action consists of the steering angle and acceleration. We regress the state-action pair to the safety value computed from the nominal model. After training for 10 epochs, we evaluate how well the safety critic classifies a state belonging to the sub-level set at 3, 4, and 5. We report the F1-score, precision, and recall in Table 3.

Since the `SPAR` agent use a safety margin of 3, the performance of classifying $V_S(\mathbf{x}) \leq 3$ is of the most importance. After pre-training, the safety critic misses 41% of the unsafe scenario, leaving much for `SPAR` to learn. The later observation that `SPAR` learns to identify unsafe scenarios that it initially misses indicate that `SPAR` is indeed learning the safety value function.

Interventions and violations for safe policies. At a high level, policies that frequently traverse through unsafe states require more *interventions* from the safety backup controller. Agents that perform unrecoverable actions that lead to, e.g., leaving the drivable area or colliding with objects in the environment are said to commit *violations*. We report results on the basis of interventions per total distance travelled (in kilometers) and violations per total distance travelled, as functions of the number of agent steps, for the `SPAR` and `SafeSAC` agent classes; for both metrics, lower is better. Aggregating across 6 random seeds, `SPAR` agents receive an average of 1.37 interventions/km and commit 0.11 violations/km, while `SafeSAC` agents received 2.41 interventions/km and commit 0.22 violations/km. Figure 8 illustrates the number of interventions from safety backup controller, received by the SPAR and SafeSAC policies, throughout training, while Figure 9 illustrates the number of violations committed by the corresponding policy class.

We observe that SPAR, with its dynamic safety updates, requires overall fewer interventions over time and commits fewer violations, compared to SafeSAC. Interestingly, in both cases, SPAR experiences a slight jump in early phases, due to the warm-start initialisation based on pre-training on context provided by a nominal dynamics model.

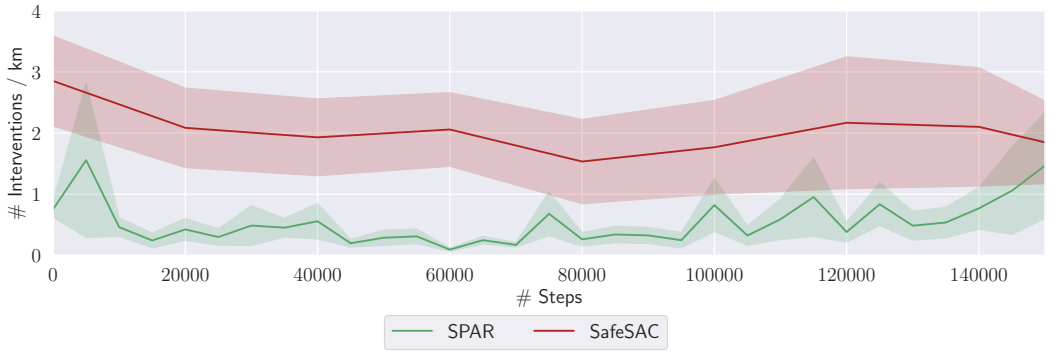


Figure 8: Plot of interventions/km versus steps, for policies that are coupled with safety backup controllers.

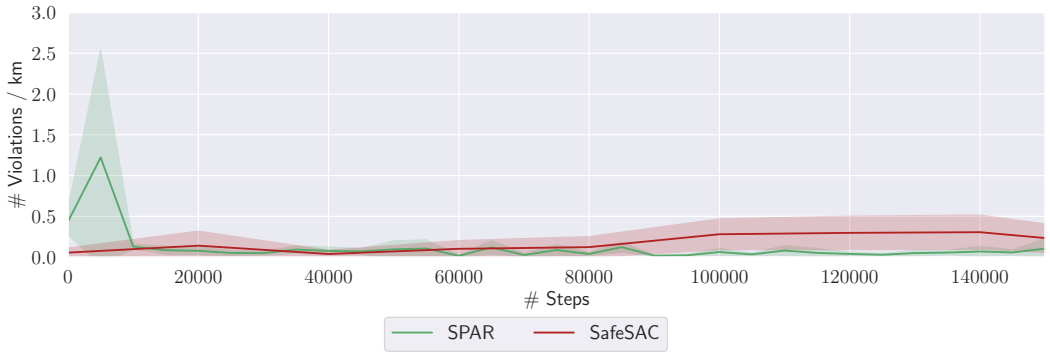


Figure 9: Plot of violations/km versus steps, for policies that are coupled with safety backup controllers.

Learn-to-Race benchmark results. In Table 4, we follow [32] in reporting on all of their driving quality metrics, for the Learn-to-Race benchmark: Episode Completion Percentage (ECP), Episode Duration (ED), Average Adjusted Track Speed (AATS), Average Displacement Error (ADE), Trajectory Admissibility (TrA), Trajectory Efficiency (TrE), and Movement Smoothness (MS).

Table 4: Learn-to-Race task [32] results on Track01 (Thruxton Circuit), with respect to the task metrics: Episode Completion Percentage (**ECP**), Episode Duration (**ED**), Average Adjusted Track Speed (**AATS**), Average Displacement Error (**ADE**), Trajectory Admissibility (**TrA**), Trajectory Efficiency (**TrE**), and Movement Smoothness (**MS**). Arrows (↑↓) indicate directions of better performance. Asterisks (*) indicate metrics which may be misleading, for incomplete racing episodes.

Agent	**ECP** (↑)	**ED*** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
HUMAN	100.0 ± 0.0	235.8 ± 1.7	171.2 ± 3.4	2.4 ± 0.1	0.93 ± 0.01	1.00 ± 0.02	11.7 ± 0.1
Random	0.5 ± 0.3	14.0 ± 5.5	11.9 ± 3.8	1.5 ± 0.6	0.81 ± 0.04	$0.33 \pm 0.38^*$	6.7 ± 1.1
MPC	100.0 ± 0.0	904.2 ± 0.7	45.1 ± 0.0	0.9 ± 0.1	0.98 ± 0.01	0.85 ± 0.03	10.4 ± 0.6
SAC	18.8 ± 13.92	8.39 ± 11.12	41.12 ± 11.87	1.79 ± 2.17	0.28 ± 0.38	0.05 ± 0.08	3.22 ± 3.59
SafeRandom (ours)	49.03 ± 44.16	484.53 ± 725.30	9.72 ± 6.65	3.73 ± 0.47	0.86 ± 0.12	0.01 ± 0.01	**11.06**$\pm$3.31
SafeSAC (ours)	62.2 ± 29.5	676.09 ± 805.83	10.31 ± 15.46	1.58 ± 1.03	0.98 ± 0.05	0.01 ± 0.01	9.15 ± 3.20
SPAR (ours)	**80.45**$\pm$39.10	**34.81**$\pm$6.14	**41.74**$\pm$18.59	**0.68**$\pm$1.58	**0.99**$\pm$0.03	**0.11**$\pm$0.10	8.06 ± 1.41

We highlight the fact that such metrics as TrA, TrE, and MS are most meaningful for agents that *also* have high ECP results. Taking TrA, for example, safe policies score higher ECP values but may spend more time in inadmissible positions (as defined by the task, i.e., with at least one wheel touching the edge of the drivable area), compared to policies without a safety backup controller that may quickly terminate episodes by driving out-of-bounds (thus spending less time in the inadmissible

positions). On the other hand, policies that have low completion percentages also have low ED scores, due to more frequent failures and subsequent environment resets.

We observe new state-of-the-art performance received by our approach, across the driving quality metrics, in the `Learn-to-Race` benchmark.