SAFETY-AWARE POLICY OPTIMISATION FOR AUTONOMOUS RACING

Anonymous authors
Paper under double-blind review

ABSTRACT

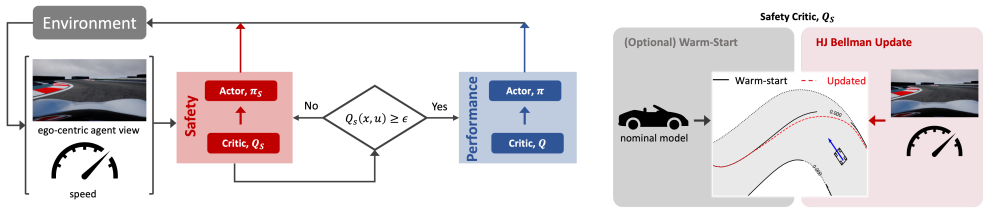
To be viable for safety-critical applications, such as autonomous driving and assistive robotics, autonomous agents should adhere to safety constraints throughout the interactions with their environments. Instead of learning about safety by collecting samples, including unsafe ones, methods such as Hamilton-Jacobi (HJ) reachability compute safe sets with theoretical guarantees using models of the system dynamics. However, HJ reachability is not scalable to high-dimensional systems, and the guarantees hinge on the quality of the model. In this work, we inject HJ reachability theory into the constrained Markov decision process (CMDP) framework, as a control-theoretical approach for safety analysis via model-free updates on state-action pairs. Furthermore, we demonstrate that the HJ safety value can be learned directly on vision context, the highest-dimensional problem studied via the method to-date. We evaluate our method on several benchmark tasks, including Safety Gym and `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment. Our approach has significantly fewer constraint violations in comparison to other constrained RL baselines, and achieve the new state-of-the-art results on the `L2R` benchmark task. We release our code in the supplementary material.

1 INTRODUCTION

Autonomous agents need to adhere to safe behaviours when interacting with their environment. In the context of safety-critical applications, such as autonomous driving and human-robot interaction, there is growing interest in learning policies that are simultaneously safe and performant. In the reinforcement learning (RL) literature, it is common to define safety as satisfying safety specifications (Ray et al., 2019a) under the constrained Markov decision process (CMDP) framework (Altman, 1999), which extends the Markov decision process (MDP) by incorporating constraints on expected cumulative costs. One challenge for solving a CMDP problem is the need to evaluate whether a policy will violate constraints (Achiam et al., 2017). Model-free methods depend on collecting diverse state-action pairs from the environment, including unsafe ones. As a result, safety is not guaranteed, most notably during the initial learning interactions (Cheng et al., 2019).

Given the practical limitations of learning about safety by collecting experiences of constraint violations or even failures, it may be favourable to leverage control theory and/or domain knowledge to bootstrap the learning process. 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. Due to scalability issues with HJ reachability, these models tend to be low-dimensional representations of the true system (up to 5D for offline computation, and 2D for online computation) Furthermore, existing works on HJ reachability exclusively study problems defined on physical states, e.g. poses, instead of high-dimensional sensory inputs, such as RGB images.

Building upon prior work on learning HJ safety value via model-free updates on state-action pairs (Fisac et al., 2019), we inject HJ reachability into the CMDP framework (Section 4). Since safety verification under HJ Reachability theory does not depend on the performance policy, we can bypass the challenges from solving a constrained optimisation problem with a neural policy, and naturally decompose the problem of learning under safety constraints into (a) optimising for performance, and (b) updating the safety value function. Given this intuition, we learn two policies that independently



(a) SPAR Architecture (b) Safety Critic

Figure 1: SPAR Overview. (a) By incorporating HJ reachability theory into the CMDP framework, we can decompose learning under safety constraints into optimising for performance and updating safety value function. Thus, SPAR consists of two policies, which are in charge of safety and performance independently. The safety controller only intervenes when the current state-action pair is deemed unsafe by the safety critic. (b) The safety critic is updated via HJ Bellman update and may optionally be warm-started via a nominal model.

manage safety and performance (Figure 1): the performance policy focuses exclusively on optimising performance, while the safety critic verifies if the current state is safe and intervenes when necessary. Primarily focused on the application of autonomous racing, we refer to our approach as **S**afety-aware **P**olicy Optimisation for **A**utonomous **R**acing (SPAR),

Aside from the problem formulation and corresponding framework, our key contributions are as follows. Firstly, we compare the HJ Bellman update rule (Fisac et al., 2019) to alternatives for learning safety critic (Srinivasan et al., 2020; Bharadhwaj et al., 2020) on two classical control benchmarks, where safe vs. unsafe states are known analytically. Given the same off-policy samples, the HJ Bellman update rule learns safety more accurately and sample efficiently. We also compare empirically how different implementations of the HJ Bellman update affect convergence.

Secondly, we demonstrate that HJ safety value function can be learned directly on visual context, the highest-dimensional problem studied by HJ safety analysis to-date, thereby expanding the applications of HJ reachability to high-dimensional systems where explicit model may not be available.

Finally, we evaluate our methods on `Safety Gym` (Ray et al., 2019a) and `Learn-to-Race` (`L2R`) (Herman et al., 2021), a recently-released, high-fidelity autonomous racing environment, which challenges the agent to make safety-critical decisions in a complex and fast-changing environment. While `SPAR` is by no means free from failure, it has significantly fewer constraint violations compared to other constrained RL baselines in `Safety Gym`. We also report new state-of-the-art results on the `L2R` benchmark task, and show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

2 Related Work

Constrained 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 (Garcıa & Fernández, 2015). We focus on the notion of safety as satisfying constraints. CMDP (Altman, 1999) 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 problem is challenging, because the policy needs to be optimised over the set of feasible ones. This requires off-policy evaluation of the constraint functions to determine whether a policy is feasible (Achiam et al., 2017). As a result, safety grows with experience, but requires diverse state-action pairs, including unsafe ones (Srinivasan et al., 2020). Furthermore, one needs to solve a constrained optimisation problem with a non-convex neural policy. This may be implemented with techniques from convex optimisation, such as primal-dual updates (Bharadhwaj et al., 2020) and projection (Yang et al., 2020), or by upper bounding the expected cost at each policy iteration (Achiam et al., 2017). Most relevant to our work is Bharadhwaj et al. (2020); Srinivasan et al. (2020); Thananjeyan et al. (2021), which also uses a safety critic to verify if a state is safe. We compare our control-theoretical learning rule with theirs in Section 5.1.

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 environment dynamics. Control barrier functions (CBFs) provide a measure of safety with gradients that inform the acceptable safe actions (Ames et al., 2019). For specific forms of dynamics, e.g. control-affine (Cheng et al., 2019), 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 (Cheng et al., 2019). Unfortunately, finding a valid control barrier function for a general system is a nontrivial task. Lyapunov-based methods (Chow et al., 2018; 2019) suffer from the same limitation of requiring hand-crafted functions.

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 general nonlinear system (Bansal et al., 2017; Fisac et al., 2018). This method can provide hard safety guarantees for 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 (Chen et al., 2018; Kousik et al., 2020). 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 or overly conservative.

There are many attempts in injecting some form of control theory into RL algorithms. In comparison to works that assume specific problem structure (Cheng et al., 2019; Dean et al., 2019) or existence of a nominal model (Cheng et al., 2019; Bastani, 2021), our proposed approach is applicable to general nonlinear system and does not require a model. But, we do assume access to a distance metric defined on the state space. Our primary inspiration is recent work by Fisac et al. (2019) that connects HJ reachability with RL and introduced a HJ Bellman update, which can be applied to deep Q-learning for safety analysis. This method loses hard safety guarantees due to the neural approximation, but enables scalable learning of safety value function. However, an agent trained using the method in Fisac et al. (2019) will focus exclusively on safety. Thus, We extend the method by formulating it within the CMDP framework, thereby enabling performance-driven learning.

Applications to autonomous racing. There is a large body of research on autonomous driving, predominately focused on urban driving. However, racing presents its unique set of challenges, e.g., making sub-second decisions under complex dynamics (Rhinehart et al., 2018). Existing open-source racing simulators, e.g., CarRacing-v0 (Brockman et al., 2016) and TORCS (TOR), lack in realism both in terms of the graphics and vehicular dynamics, which limits the researchers' ability to effectively evaluate their algorithms. Florian et al. (2020) developed an interface to and trained their agents in the Gran Turismo video game, but did not make their environment publicly available. Furthermore, their agents assume access to an unrealistic amount of privileged information. In this work, we set the first safe learning results in the recently-introduced, high-fidelity, open-source `Learn-to-Race` autonomous racing environment (Herman et al., 2021). Apart from RL-based approaches, optimisation-based approaches have been used in works such as Liniger et al. (2015); Kabzan et al. (2019). We refer interested readers to Herman et al. (2021) for a recent and comprehensive review on autonomous racing.

3 Preliminaries

Constrained MDPs. The problem of 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 temporal 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:

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

HJ Reachability. To generate the safety constraint, one can apply HJ reachability to a general nonlinear system model, 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 $\mathcal{K}$. Taking autonomous driving as an example, $\mathcal{K}$ is the drivable area and $l(x)$ is the shortest distance to road boundary or obstacle. 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:

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

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):

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

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 by solving the Hamiltonian: $\pi^*_S(x) = \arg\max_{u \in \mathcal{U}} \langle f(x,u), \nabla V_S(x) \rangle$. 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 $V_S(x,T) \geq 0$ and π^*_S otherwise. By switching maximisation to minimisation, and vice versa, the method is also applicable to finding a reachable set (tube) to a desired goal (Bansal et al., 2017).

The newly introduced discounted safety Bellman equation (Fisac et al., 2019) modifies the HJI-VI in equation 3 in a time-discounted formulation for discrete time:

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

This formulation induces a contraction mapping, which enables convergence of the value function when applied to 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 HJ reachability, which provides a systematic way to verify safety. Thus, we formulate our problem as a combination of constrained RL and HJ reachability theory, adopting a control-theoretical approach to learn safety. The need for an accurate model of the system dynamics can be restrictive. Building upon prior work on neural approximation of HJ Reachability (Fisac et al., 2019), 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) Injects control theory into RL. We incorporate HJ Reachability theory into the CMDP framework, thereby updating the safety critic in a control-theoretical manner. An unintended, but welcome outcome is that the original constrained optimisation problem is naturally decomposed into two unconstrained optimisation problems, making the problem more amenable to gradient-based learning.

ii) Scales to high-dimensional problems. Compared to standard HJ Reachability methods, whose computational complexity scales exponentially with the state dimension, we updated the safety value directly on vision embedding using the neural approximation. This is the highest-dimensional problem studied studied via HJ reachability to-date.

Problem formulation. We inject HJ Reachability theory into the CMDP framework. Starting with Eqn. 1, we can interpret the negative of a cost as a reward for safety and, without loss of generality, reverse the direction of the inequality constraint. Recall that the super-zero set of the safety value function, i.e., $\{x|V_s(x) \geq 0\}$, designates all states from which the system can remain within the set of allowable states, $\mathcal{K}$, over infinite time horizon. Thus, the safety value function derived from HJ reachability can be naturally embedded into CMDP (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. A key difference from the original CMDP formulation (Eqn. 1) is that constraint satisfaction, $V_S(x) \geq \epsilon$, no longer depends on the policy, π. Thus, we can bypassing the challenges of solving CMDPs (Section 2) and decompose learning under safety constraints into optimising for performance and updating safety value estimation. While a number of works have similar dual-policy architecture (Cheng et al., 2019; Bastani, 2021; Thananjeyan et al., 2021), ours design is informed by HJ reachability theory. A downside of the formulation is that HJ reachability considers safety as absolute, and there isn't a mechanism to allow for some level of safety infractions.

Update of Safety Value Function. For the update of safety value function, we adopt the learning rule proposed by Fisac et al. (2019) (Eqn. 6). Note that $Q_S(x, u)$ 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 allowable states $\mathcal{K}$.

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

On top of the theoretical analysis in Fisac et al. (2019), we compare how common RL implementation techniques, including delayed target network, clipped double Q-learning (Fujimoto et al., 2018), and baseline reduction (Schulman et al., 2015) affect convergence on two classical control benchmarks, *Double Integrator* and *Dubins' Car* and summarise the observations in Appendix A.2.

SPAR. We propose SPAR, which consists of a performance policy and a safety policy. 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, i.e. $u \sim \pi$, if $Q_S(x, u) \geq \epsilon$ and $u \sim \pi_S$ otherwise. Thus, the agent enjoys the most freedom in safe exploration. The performance policy may be implemented with any RL algorithm. Since we expect the majority of samples to be from the performance policy, it is more appropriate to update the safety actor critic with an off-policy algorithm. The safety critic is updated with Eqn. 6, where $u' \sim \pi_S$. The safety actor is updated via deterministic policy gradient through the safety critic, i.e. $\nabla_u Q_S(x, u)$. The algorithm for SPAR is detailed in Appendix B.

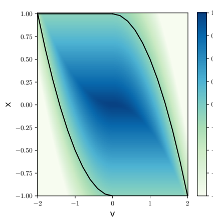
5 EXPERIMENTS

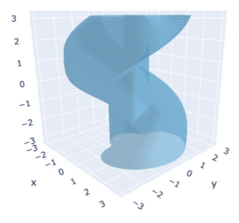
We evaluate SPAR on three set of benchmarks of increasing difficulty. While the our intended application is autonomous racing, the first two set of benchmarks can be considered as some abstraction of vehicles with the objective of avoiding obstacles and/or moving towards goals. Firstly, we evaluate on two classical control tasks where the safe vs. unsafe states are known analytically, and compare the HJ Bellman update used in SPAR to alternatives for learning safety critic in the literature. Secondly, we compare SPAR to constrained RL baselines in Safety Gym. Finally, we challenge SPAR in Learn-to-Race and conduct ablation to better understand how different components of SPAR contribute to its performance.

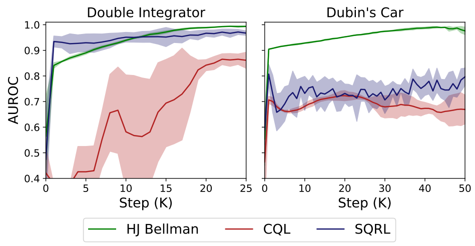
5.1 EXPERIMENT: CLASSICAL CONTROL BENCHMARKS

As mentioned earlier, safety critics have been trained in other works (Bharadhwaj et al., 2020; Srinivasan et al., 2020) with different learning rules. The objective here is to compare the HJ Bellman update with alternatives. Thus, we focus on safety analysis with off-policy samples, and evaluate on two classical control benchmarks *double integrator* and *Dubins' Car*, where the safe / lively[1] states are known analytically (Figure 2a and 2b). The double integrator (Fisac et al., 2019) characterise a particle moving on x-axis, with velocity v. By controlling the acceleration, the objective is to keep the particle on a bounded range on x-axis. Dubins' car (Bansal et al., 2017) is a simplified car model, where the car moves at a constant speed. By controlling the turning rate, the goal is to reach a unit circle regardless of the heading. More information on the two tasks are provided in Appendix A.1.

[1]Liveliness refers the ability to reach the specified goal (Hsu et al., 2021).


(a) Double Integrator


(b) Dubins' Car



(c) Performance comparison of learning rules (averaged over 5 random seeds)

Figure 2: We uses two classical control benchmarks, *double integrator* and *Dubins' car*, to evaluate the performance of different learning rules for safety analysis. (a) shows the safety value function of the double integrator and the black line delineates $V_S(x) = 0$, within which the particle can remain within the allowable range of $x \in [-1, 1]$. (b) shows the iso-surface of the safety value function at 0, i.e., $V_S(x) = 0$, for Dubins' Car, within which the car can reach a unit circle at the origin. The performance comparison is summarized in (c).

In this experiment, we generate state-action pairs with a random policy, and evaluate the safety value function with respect to the optimal safety controller, π_S^*, which is also known. In both Bharadhwaj et al. (2020); Srinivasan et al. (2020), the safety critic is defined as the expected cumulative cost, i.e. $Q_C^\pi(x,u) := \mathbb{E}_{x_k,u_k\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^k C(x_k)|x_0 = x, u_0 = u\right]$, where $C(x_k) = 1$ if a failure occurs at x_k and 0 otherwise. $Q_C^\pi(x,u)$ is additionally set to 1 for terminal states in Srinivasan et al. (2020). Note that both the environment and optimal safety policy are deterministic. Following the definition, $Q_C^{\pi_S^*}(x, \pi_S^*(x))$ should be 0 if x is a safe state. Safety Q-functions for RL (SQRL) (Srinivasan et al., 2020) uses the standard Bellman backup to propagate the failure signal. On top of that, Conservative Safety Critic (CSC) (Bharadhwaj et al., 2020) uses conservative Q-learning (CQL) (Kumar et al., 2020) to correct for the difference between the behaviour policy, i.e. the random policy, and the evaluation policy, i.e. the optimal safety policy, and overestimate Q_C to err on the side of caution.

Since the safe vs. unsafe states are known for these benchmark tasks, we can directly compare the performance of these safety critics learned with different learning rules (Figure 2c). While the theoretical cut-off for safe vs. unsafe states is 0, the performance of SQRL is very sensitive to the choice of the cut-off. Thus, we report AUROC instead. For both CQL and SQRL, we do a grid search around the hyperparameters used in the original paper and report the best results. The implementation details and additional results, e.g. visualisation of learned critics and other metrics, are included in Appendix A.3. Directly applying Bellman update for safety analysis as in SQRL, performs well on *double integrator*, but does not on the more challenging *Dubins' Car*. In our experiment, CQL is highly sensitive to the choice of hyperparameters and has large variance in performance. In comparison, HJ Bellman update has AUROC close to 1 and has very little variance over different runs. Note that we are only comparing the efficacy of the different learning rules for safety critic given the same off-policy samples, and learning safety critic is only part of SQRL and CSC.

One caveat is that SQRL and CQL uses binary signal for failures, while HJ Bellman update has access to the distance, $l(x)$. On the one hand, HJ Bellman update does assume more information. On the other hand, it may be more practical to learn safety from distance measurements then experiencing failures. Applied to autonomous driving, this translates to learning to avoid obstacle from distance measurements that are readily available on cars with assisted driving capabilities (BMW, 2021), in comparison to experiencing collisions.

5.2 EXPERIMENT: `Safety Gym`

We additionally evaluate our proposed approach, SPAR, in Safety Gym (Ray et al., 2019b), the OpenAI gym environment for safe RL. Specifically, we evaluate on the CarGoal1-v0 and PointGoal1-v0 benchmarks, wherein we compare SPAR against Constrained Policy Optimisation (CPO) (Achiam et al., 2017), Proximal Policy Optimisation (PPO) (Schulman et al., 2017), and PPO-Lagrangian (Ray et al., 2019b). Episodic Performance and Cost curves are shown in Figure 3, and additional SPAR implementation details are included in Appendix D.

PPO-SPAR has significantly fewer constraint violations compared to other baselines and the violations decreases over time. This is because HJ reachability theory considers safety as absolute, and there isn't a mechanism for allowing a certain number of violations, which unfortunately compromises

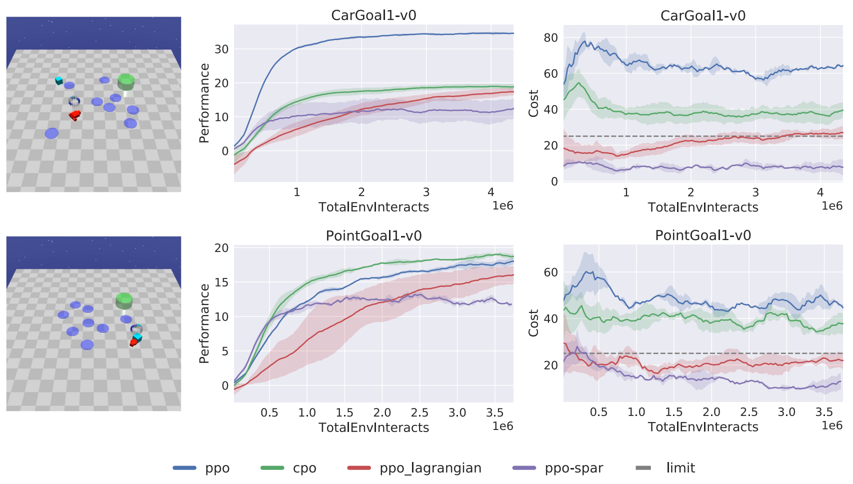
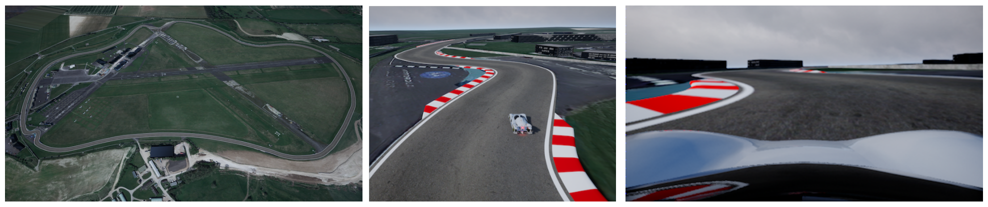


Figure 3: Results of SPAR in the CarGoal1-v0 (top row) and PointGoal1-v0 (bottom row) benchmarks, under the OpenAI Safety Gym framework. In Goal tasks, agents must navigate to observed goal locations (indicated by the green regions), while avoiding static obstacles (e.g., vases, in cyan) and hazards (blue regions).



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

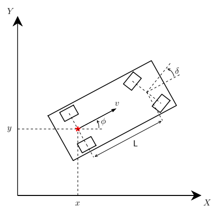
Figure 4: We use the `Learn-to-Race` (`L2R`) framework (Herman et al., 2021) for evaluation; this environment provides simulated racing tracks that are modelled after real-world counterparts, such as the famed Thruxton Circuit in the UK (`Track01:Thruxton`, (a)). Here, learning-based agents can be trained 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 (shown in (b)), where SPAR only uses ego-camera views (shown in (c)) and speed.

performance. The violations that do occur are results of neural approximation error, and the number of violations decrease over time as the safety actor-critic gain experiences. While constraint violation are non-terminal in the Safety Gym environment, the task in the next subsection *does* terminate episodes upon safety infraction, which is better suited for HJ safety analysis.

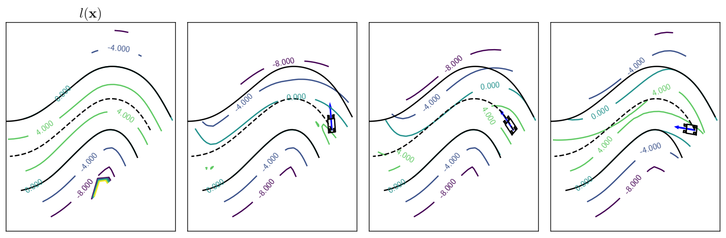
5.3 EXPERIMENT: `Learn-to-Race`

Task 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 (Herman et al., 2021). `L2R` provides multiple simulated racing tracks, modelled after real-world counterparts, such as Thruxton Circuit in the UK (`Track01:Thruxton`; see Figure 4). `L2R` provides access to RGB images from any specified location, semantic segmentation, and vehicle states (e.g., pose, velocity). In each episode, an agent is spawned on the selected track. At each time-step, it uses its observations to determine normalised steering angle and acceleration. All learning-based agents receive the reward specified by `L2R`, which is formulated as a weighted sum of reward for driving fast and penalty for leaving the drivable area; the main objective is to complete laps in as little time as possible. Additional metrics are defined to evaluate driving quality.

Implementation Details. To characterise the performance of our approach, we report results on the Average Adjusted Track Speed (*AATS*) and the Episode Completion Percentage (*ECP*) metrics (Herman et al., 2021) as proxies for agent performance and safety, respectively. For reference, one lap in `Track01:Thruxton` is 3.8km, whereas CARLA, the de facto environment for autonomous driving research, has in total 4.3km drivable roads in the original benchmark (Codevilla et al., 2019). Thus,



(a) nominal model



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

Figure 5: (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.

successfully completing an episode, i.e. a lap, is very challenging. Agents' results on other metrics of driving quality, as defined by the L2R environment, are presented in Appendix F.

We use Track01:Thruxton in L2R (Fig. 4) 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. We use soft-actor critic (SAC) (Haarnoja et al., 2018b) as the performance policy, and all agents only have access to ego-camera view (Figure 4c) and speed, unless specified otherwise. The implementation, including network architecture and hyperparameters, are detailed in Appendix E.

Static Safety Actor-Critic from Nominal Model. To demonstrate the benefit of utilising domain knowledge in the form of a nominal model and to compare with the learnable safety actor-critic in SPAR, we use the kinematic vehicle model (Kong et al., 2015) (see Figure 5a), which is a significant simplification of a realistic race car model (Kabzan et al., 2019), to compute the safety value and corresponding 'optimal'[2] safety controller. The dynamics and 'optimal' safety control is given in Eqn. 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. $L = 3$m is the car length. Intuitively, the 'optimal' safety policy brakes and steers towards the centre of the track as much as possible. The derivation of the safety policy is provided in Appendix C. Setting $V_S(\mathbf{x}, 0) = l(\mathbf{x})$, we calculated the backward reachable tube using the code from Giovanis et al. (2021). Fig. 5b illustrates resulting safety value function at slices of state space, as the agent enters into a sharp turn.

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{v} = a \\ \dot{\phi} = v\tan\delta/L \end{cases} \quad \text{and} \quad a^* = \begin{cases} \underline{a} & \text{if } \partial V_S/\partial v \le 0 \\ \overline{a} & \text{else} \end{cases} \quad \delta^* = \begin{cases} \overline{\delta} & \text{if } \partial V_S/\partial \phi \ge 0 \\ \underline{\delta} & \text{else} \end{cases} \tag{7}$$

We assume the static actor-critic have access to vehicle poses in order to evaluate safety value and determine the safety action. We evaluate the performance of this static actor-critic by coupling a random agent with it (SafeRandom). We test SafeRandom on a series of safety margins to account for unmodelled dynamics; the performance averaged over 10 random seeds is summarised in Figure F.1. For instance, $\epsilon \geq 4.2$ achieves 80+% ECP. This high safety performance, in comparison to 0.5% ECP by Random agent showcase the benefit of utilising domain knowledge.

Ablation Study. We conduct ablation to better understand how different components of SPAR contribute to its performance. We examine the effect of imposing safety constraints on performance and sample efficiency, by comparing SAC agent with an instance of itself that is coupled with the static safety actor-critic (SafeSAC). We set the safety margin ϵ to be 4.2, based on empirical results from SafeRandom. We also compare the performance of using the static safety actor-critic (SafeSAC) and a learnable one (SPAR). Since SPAR is expected to have a better characterisation of the safety value, the agent no longer depend on a large safety margin to remain safe and thus the SPAR agent uses a safety margin of 3.0m, which accounts for the dimension of the vehicle[3].

[2]only with respect to the nominal model

[3]The HJ reachable tube is computed with respect to the back axle of the vehicle and does not account for the physical dimension of the vehicle.

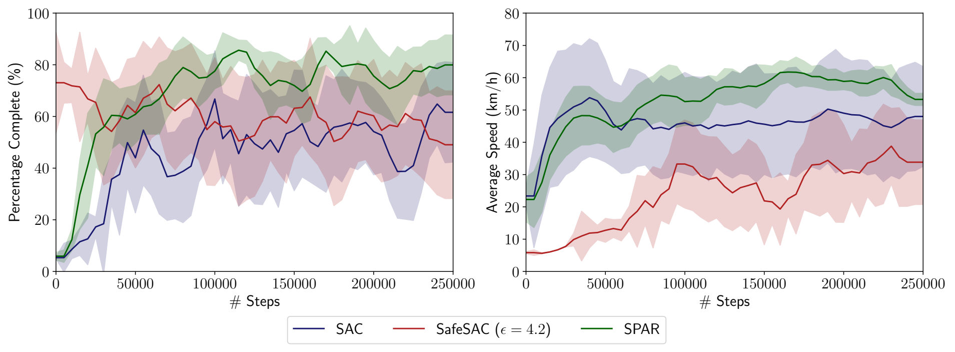


Figure 6: **Left:** Episode percent completion and **Right:** speed evaluated every 5000 steps over an episode (a single lap) and averaged over 5 random seeds. Results reported based on `Track01:Thruxton` in `L2R`.

Results. The performance comparison between different agents is summarised in Figure 6. Reporting the average of ECP obscure the fact that the failures concentrate in a small number of locations, e.g. with sharp turns, after the agents acquire basic driving skills.

The static safety actor-critic significantly boost initial safety performance. With the help of the static safety actor-critic, the `SafeSAC` can complete close to 80% of a lap, in comparison to slightly more than 5% with `SAC`. This, again, showcase that injecting domain knowledge in the form of a nominal model is extremely beneficial to safety performance, especially in the initial learning phase. However, there are two notable limitation with the static safety controller. Firstly, it is extremely conservative, braking whenever the vehicle less safe. As a result the `SafeSAC` agent has an initial speed of less than 10km/h. Secondly, as the `SAC` learns to avoid activating the safety controller and drive faster, the static safety controller is no longer able to recover the vehicle from marginally safe states. In fact, by applying the 'optimal' safety action from Eqn. 7, i.e., maximum brake and steer, the vehicle will lose traction and spin out of control. As a result, the ECP actually decreases over time for `SafeSAC`.

SPAR learns safety directly from vision context and can recover from marginally safe states more smoothly. Having a safety actor-critic that is dedicated to learning about safety significantly boosted the initial safety performance of `SPAR` in comparison to the `SAC` agent, even though the safety actor-critic is randomly initialised to show the safety value function can be learned from vision embedding from scratch. In practice, we envision the safety actor-critic to be warm-started with the nominal model, and fine-tuned by observations from the environment. Furthermore, the learnable safety actor-critic can recover from marginally safe states more smoothly. A qualitative comparison of such behaviours is available at video link. While `SPAR` outperforms other baselines, there is still significant performance gap with human, as the speed record at Thruxton Circuit is 237 km/h (average speed).

6 Conclusion

In this paper, we incorporate HJ reachability theory into the CMDP framework as a principled approach to learn about safety. As a result of the problem formulation, we effectively decompose the problem of learning under safety constraints into two more-tractable sub-tasks: optimising for performance and updating safety value. We show on two classical control benchmarks that the HJ Bellman update is more effective than alternatives for learning the safety critic. Comparing to constrained RL baselines in the Safety Gym, we show that `SPAR` has significantly few constraint violations, while maintaining similar performance. Finally, we report the new state-of-the-art result on `Learn-to-Race`. We demonstrate that the HJ safety value can be learned directly on visual context, thereby expanding HJ reachability to broader applications.

Whereas our empirical results demonstrated that it is possible to learn a safety-aware and performant policy, `SPAR` is by no means free from failure. However, the method proposed in this paper represents a subtle shift away from constraint-satisfaction exclusively through model-free learning, as has become popular in recent literature. Rather than letting agents learn safe behaviours through experiencing failures, our approach provides potential avenues for online safety analysis, through the injection of domain knowledge (e.g. a nominal model), and by informing the learning rule with control theory.

References

Torcs, the open racing car simulator. `http://torcs.sourceforge.net/index.php?name=Sections&op=viewarticle&artid=19`. Last accessed: 2021-01-30.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pp. 3420–3431. IEEE, 2019.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253. IEEE, 2017.

Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American Control Conference (ACC)*, pp. 3488–3494. IEEE, 2021.

Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

BMW. Automotive sensors – the sense organs of driver assistance systems, Sep 2021. URL `https://www.bmw.com/en/innovation/automotive-sensors.html`.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

Mo Chen, Sylvia L Herbert, Mahesh S Vashishtha, Somil Bansal, and Claire J Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel 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, pp. 3387–3395, 2019.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708*, 2018.

Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9329–9338, 2019.

Sarah Dean, Stephen Tu, Nikolai Matni, and Benjamin Recht. Safely learning to control the constrained linear quadratic regulator. In *2019 American Control Conference (ACC)*, pp. 5582–5588. IEEE, 2019.

Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8550–8556. IEEE, 2019.

F. Florian, S. Yunlong, E. Kaufmann, D. Scaramuzza, and P. Duerr. Super-human performance in gran turismo sport using deep reinforcement learning, 2020.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

George Giovanis, Michael Lu, and Mo Chen. Optimizing dynamic programming-based algorithms. `https://github.com/SFU-MARS/optimized_dp`, 2021.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018b.

James Herman, Jonathan Francis, Siddha Ganju, Bingqing Chen, Anirudh Koul, Abhinav Gupta, Alexey Skabelkin, Ivan Zhukov, Max Kumskoy, and Eric Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. *arXiv preprint arXiv:2103.11575*, 2021.

Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. 2021.

Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099. IEEE, 2015.

Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram 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.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019a.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019b.

Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3): 4915–4922, 2021.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

A Classical Control Benchmarks

The objective of this section is 1) to examine how different implementation of the learning rule proposed by Fisac et al. (2019), i.e., $Q(x, u) = (1-\gamma)l(x) + \gamma \min\{l(x), \max_{u' \in \mathcal{U}} Q(x', u'\}$, affect convergence, and 2) to compare the learning rule with alternatives for learning safety value function. We evaluate it on two classical control benchmarks, *Double Integrator* and *Dubins' Car*, as described in Section A.1, where the analytical solution to safe and unsafe states are known.

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 $a \in [-1, 1]$.

$$\begin{cases} \dot{x} = v \\ \dot{v} = a \end{cases} \tag{A.1}$$

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.

$$\begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{\phi} = u \end{cases} \tag{A.2}$$

Implementation & evaluation. We use a neural network with hidden layers of size [16, 16] for the double integrator and [64, 64, 32] for Dubins' car. We use ADAM Kingma & Ba (2014) as the optimiser with a learning rate of 0.001, batch size of 64. We update the safety value function over 25K steps for Double Integrator and 50K steps for Dubin's Car, and report classification performance every 1000 steps averaged over 5 random seeds.

While the safety value is defined over continuous state space, we evaluate the performance over a mesh on the state space. Qualitative comparison between the ground truth value and that learned via HJ Bellman update is shown in Figure A.1 and A.2.

A.2 Comparison of Implementation Details

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 Fujimoto et al. (2018), as the circular dependency between value estimate and policy results in accumulation of residual error and divergent updates. The target network is commonly updated with $Q' \leftarrow \tau Q + (1-\tau)Q'$, where τ is 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 Fujimoto et al. (2018) and also adopted in SAC Haarnoja et al. (2018a). 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.

- *Is the baseline technique helpful?* Reducing an action-independent baseline from the value function is commonly used variance reduction technique in policy gradient algorithms. We also examine if the technique is conducive to learning the safety value.

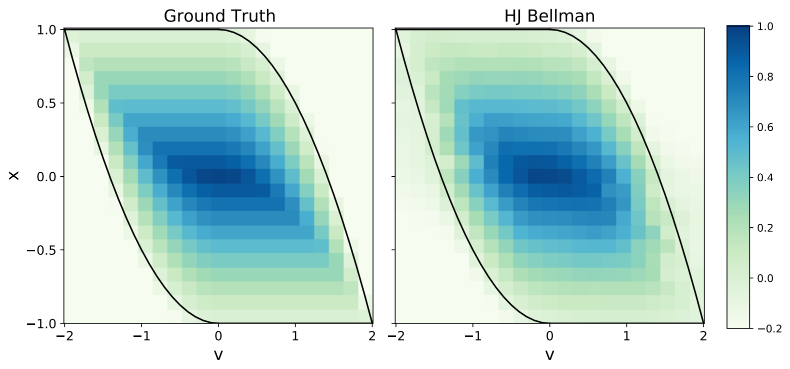
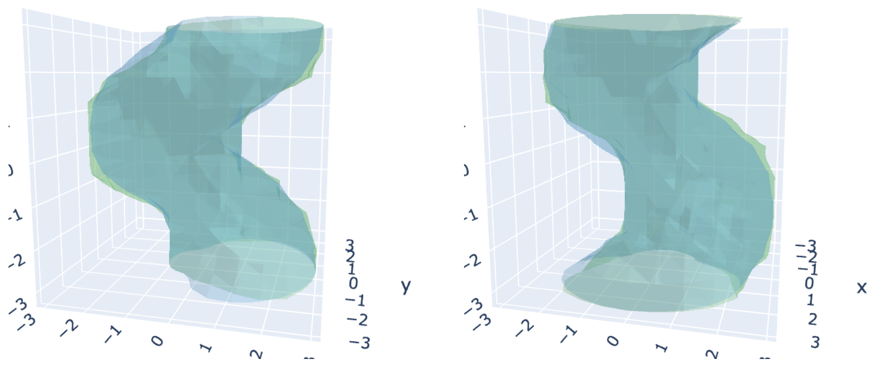


Figure A.1: A comparison between the ground truth safety value (and that learned via HJ Bellman update for double integrator



(a) (b)

Figure A.2: A comparison between the ground truth safety value (blue) and that learned via HJ Bellman update (green) for Dubins' car

A slow-moving target network is not necessary for convergence. As we can see in Figure A.3, the safety value converged without problem using $\tau = 1$, which is equivalent to not keeping a target network at 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 Fujimoto et al. (2018) that value estimate without target network is convergent under a fixed policy.

Clipped Double Q-learning is unnecessary. In Figure A.4, we also provide a comparison on vanilla DQN vs. clipped double Q-learning, including both minimum and maximum of the double Q functions. We hypothesise that the safety Bellman backup already clips Q at next state with $l(x)$ and thus the overestimation error is not a concern.

Baseline technique does not significantly improve the results of learning safety critic, but is conducive learning the safety actor. We consider updating the safety critic with the baselined version of safety Q-value, i.e. $\Psi_S(x, u) = Q_S(x, u) - l(x)$. The corresponding update rule is given by Eqn. A.3.

$$\Psi_S(x, u) = \gamma \min\{0, \max_{u' \in \mathcal{U}} \Psi_S(x', u') + l(x') - l(x)\} \tag{A.3}$$

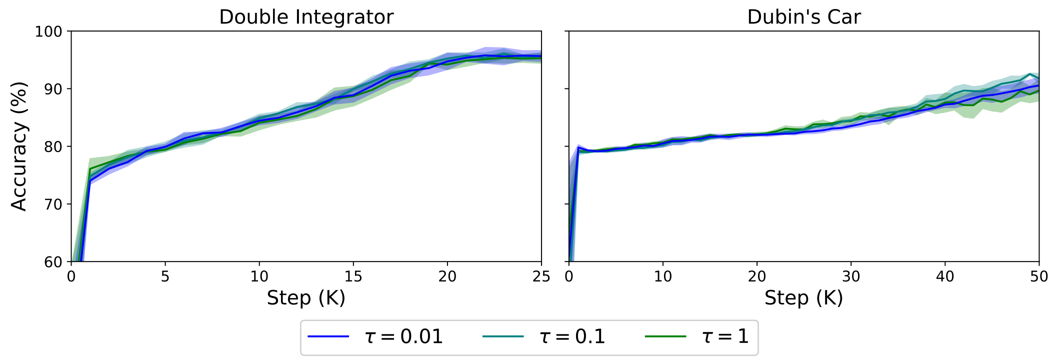


Figure A.3: Performance comparison under τ=0.01, 0.1, 1

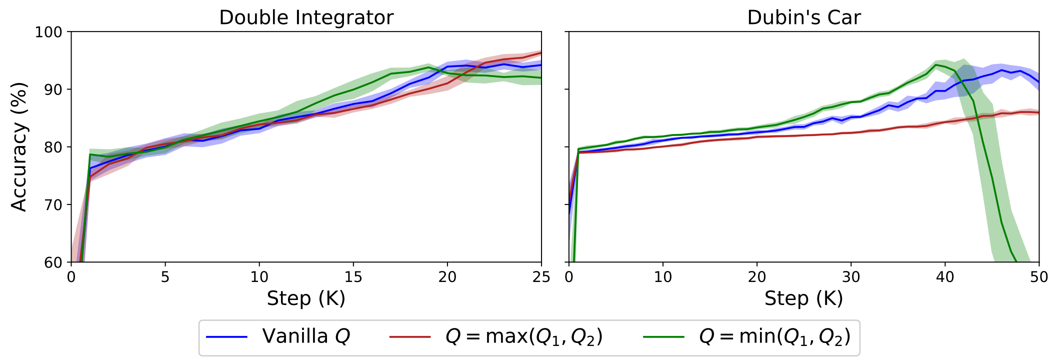


Figure A.4: Performance comparison with clipped double Q-learning

As shown in Figure A.5, using the baseline technique does not improve the accuracy of the safety critic, but qualitatively, we observe that the safety actor predicts safe action better. Thus, we use the baselined-Q for the rest of the paper.

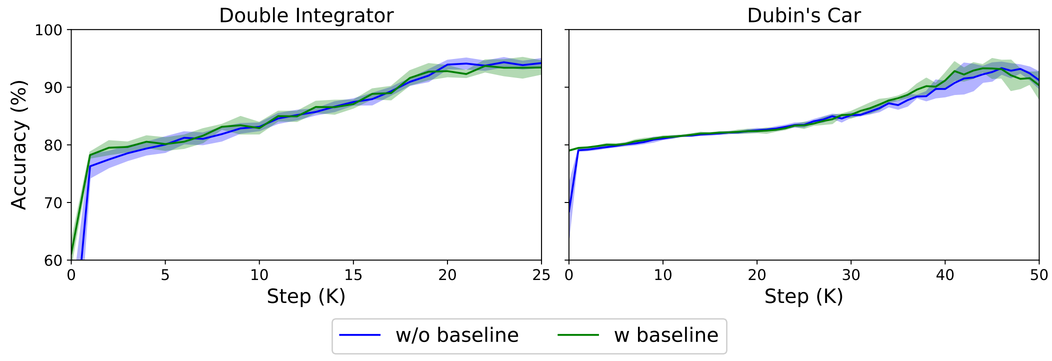


Figure A.5: Performance comparison with clipped double Q-learning

A.3 Comparison of Learning Rules for Safety Critic

Implementation Details. For the CQL implementation, we used $\gamma = 0.99$ and learning rate = 0.0002, following Bharadhwaj et al. (2020). We used $\alpha = 0.05$ for smaller variance. For the SQRL implementation, we $\gamma = 0.7$ for Dubins' car and learning rate = 0.0003, following Srinivasan et al. (2020). We used $\gamma = 0.9$ for double integrator, which improved performance.

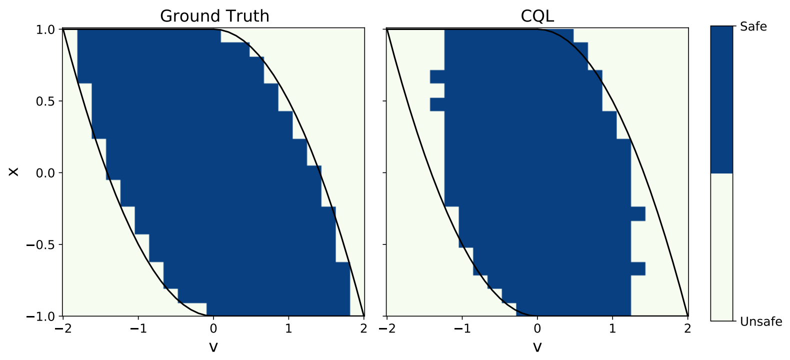


Figure A.6: A comparison between the ground truth safety value and that learned via Conservative Q-learning (CQL) for double integrator

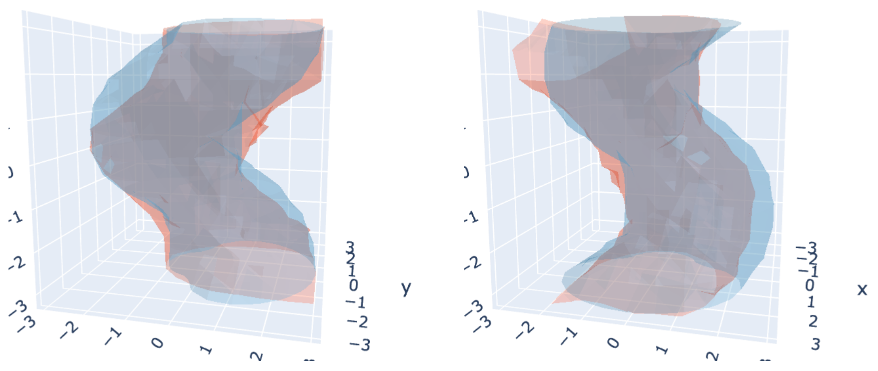


Figure A.7: A comparison between the ground truth safety value (blue) and that learned via CQL (red) for Dubins' car

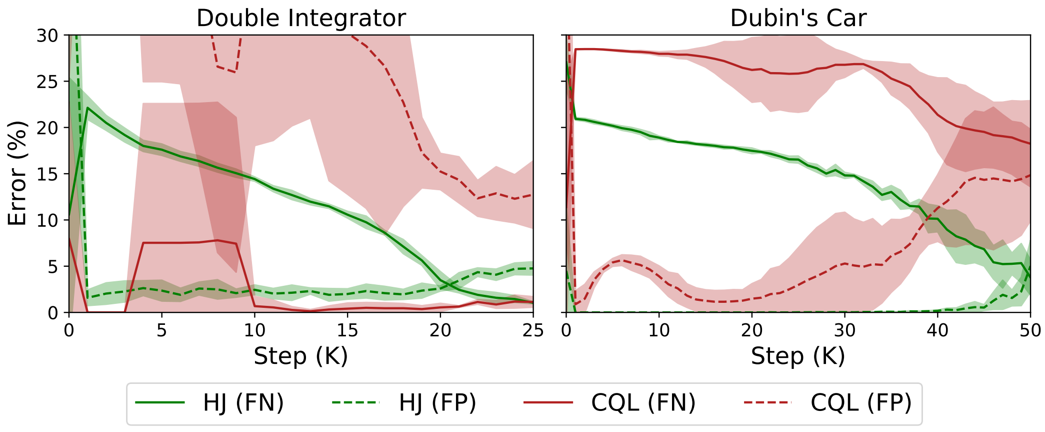


Figure A.8: A breakdown of false negative (FN) and false negative (FP) rate

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.2. 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 $Q_\phi$ and actor $\pi_\theta$;
Initialise: safety critic $Q_{\phi_S}$, and actor $\pi_{\theta_S}$; target networks $\phi'_S \leftarrow \phi_S$ , $\theta'_S \leftarrow \theta_S$;
Initialise: replay buffer $\mathcal{M}$;
for i = 0, . . . , # Episodes do
    $x$ = env.reset()
    while not terminal do
        u $\sim \pi_\theta(x)$;
        // 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
            $u \sim \pi_{\theta_S}(x)$
        end
        $x', r$ = env.step($u$)
        $\mathcal{M}$.store($x, a, x', r$)
        $x = x'$
        Update performance critic $Q_\phi$ and actor $\pi_\theta$ with preferred RL algorithm;
        Sample N transitions $(x, u, x')$ from $\mathcal{M}$;
        // Update the safety critic:
        Calculate the target value with the discounted Bellman safety update
            $$y = (1 - \gamma)l(x) + \gamma \min\{l(x), Q_{\phi'_S}(x', u')\}$$
         where $u' \sim \pi_\theta$.
            $$\mathcal{L}_{\phi_S} = N^{-1} \sum (Q_{\phi_S}(x, u) - y)^2$$
            $$\phi_S \leftarrow \phi_S - \alpha \nabla_{\phi_S} \mathcal{L}_{\phi_S}$$
         // 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 Derivation of the Optimal Safety Controller

Recall that the nominal model is given by Eqn. C.1, where the state is $\mathbf{x} = [x, y, v, \phi]$, and the action is $\mathbf{u} = [a, \delta]$. For the action, $a \in [\underline{a}, \overline{a}]$ and $\delta \in [\underline{\delta}, \overline{\delta}]$ x, y, v, ϕ are the vehicle's location, speed, and yaw angle. a is the acceleration, and δ is the steering angle. $L = 3$m is the car length.

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{v} = a \\ \dot{\phi} = v\tan\delta/L \end{cases} \tag{C.1}$$

The optimal safety control is derived by solving the Hamiltonian as given in Eqn. C.2a. By definition,

$$\nabla V_S(\mathbf{x}) = [\partial V_S/\partial x, \partial V_S/\partial y, \partial V_S/\partial v, \partial V_S/\partial \phi]$$

.

$$\pi_S^*(x) = \arg\max_{\mathbf{u}\in\mathcal{U}} \langle f(x,u), \nabla V_S(x)\rangle \tag{C.2a}$$

$$= \arg\max_{[a,\delta]\in\mathcal{U}} [v\cos(\phi)\frac{\partial V_S}{\partial x} + v\sin(\phi)\frac{\partial V_S}{\partial y} + a\frac{\partial V_S}{\partial v} + v\tan\delta/L\frac{\partial V_S}{\partial \phi}] \tag{C.2b}$$

$$= \arg\max_{[a,\delta]\in\mathcal{U}} [a\frac{\partial V_S}{\partial v} + v\tan\delta/L\frac{\partial V_S}{\partial \phi}] \tag{C.2c}$$

From Eqn C.2c, it is clear that the optimal safety controller maximising the Hamiltonian is given by Eq. C.3

$$a^* = \begin{cases} \underline{a} & \text{if } \partial V_S/\partial v \le 0 \\ \overline{a} & \text{else} \end{cases}, \qquad \delta^* = \begin{cases} \overline{\delta} & \text{if } \partial V_S/\partial \phi \ge 0 \\ \underline{\delta} & \text{else} \end{cases} \tag{C.3}$$

D Additional Implementation Details for Safety Gym

Following the default CarGoal1-v0 and PointGoal1-v0 benchmarks in Safety Gym, all agents were given observation of hazard, goal, and vase LiDARs, with hazards being constrained. Both environments were initialised with a total of 8 hazards and 1 vase. Agent's are endowed with accelerometer, velocimeter, gyro, and magnetometer sensors; their LiDAR configurations included 16 bins, with max distance of 3.

To illustrate the generality of our approach, relative to an arbitrary learning-based policy class, our SPAR implementation wraps its safety actor critic around a standard PPO base agent, utilising default configuration from the *PPO-Spinningup* repository from OpenAI (Ray et al., 2019b). By default, distance measurements from LiDAR are available in these benchmarks, and thus SPAR has direct access to $l(x)$. Despite PPO being an on-policy algorithm, the SPAR safety critic was implemented with off-policy updates, using prioritised memory replay based on the TD-error of predicting safety value. Since $l(x)$ is small in this environment, we scaled cost by a factor of 100. We used a gamma of 0.99, $\tau = 1 -$ `polyak` of 0.005, critic learning rate of 0.001, actor learning rate of 0.003, target KL (for early stopping criteria) of 0.05, an alpha of 0.2, and a safety margin of 0.25. We used the `SquashedGaussianMLPActor` and `MLPFunction` (Ray et al., 2019b) for the safety actor critic architecture.

All experiments in Safety Gym were run on an Intel(R) Core(TM) i9-9920X CPU @ 3.50GHz – with 1 CPU, 12 physical cores per CPU, and a total of 24 logical CPU units.

E Additional Implementation Details for `Learn-to-Race`

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 E.1. 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 Kingma & Ba (2014), with a learning rate of 0.003. During training, we used a batch size of 256 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.

F 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 F.1. For safety margin $\epsilon \geq 4.2$, the SafeRandom agent can finish 80+% of the lap, 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.

Learn-to-Race benchmark results. In tables F.1 and F.2, we follow (Herman et al., 2021) 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).

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

Table E.1: 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

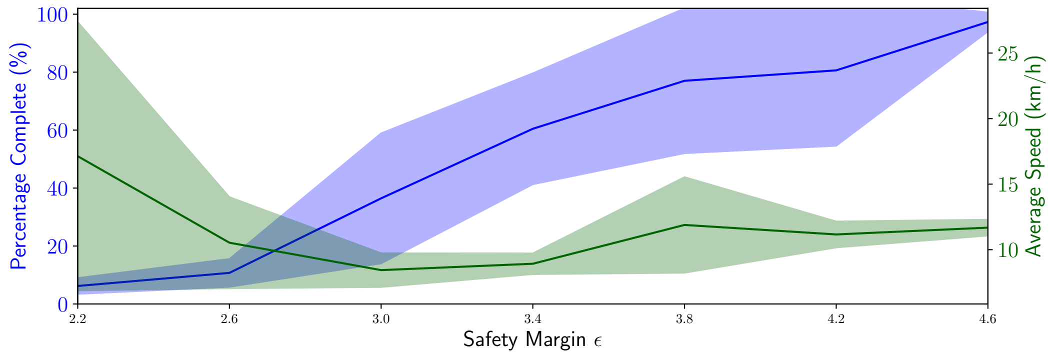


Figure F.1: Performance of the `SafeRandom` agent at different safety margin (averaged over 10 random seeds)

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.

Table F.1: `Learn-to-Race` task (Herman et al., 2021) results on `Track01` (Thruxton Circuit), for learning-*free* agents, 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, across agents. **Bold** results in tables F.1 and F.2 are generally best, however, asterisks (*) indicate metrics which may be misleading, for incomplete racing episodes.

Agent	**ECP** (↑)	**ED*** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
`HUMAN`	**100.0** $\pm$ 0.0	78.6 $\pm$ 5.2	**79.29** $\pm$ 4.7	2.4 $\pm$ 0.1	0.93 $\pm$ 0.01	**1.00** $\pm$ 0.02	**11.7** $\pm$ 0.1
`Random`	0.50 $\pm$ 0.30	**4.67** $\pm$ 3.2	11.90 $\pm$ 3.80	1.5 $\pm$ 0.60	0.81 $\pm$ 0.04	0.33 $\pm$ 0.38*	6.7 $\pm$ 1.1
`MPC`	100.0 $\pm$ 0.0	301.40 $\pm$ 10.10	45.10 $\pm$ 0.0	**0.90** $\pm$ 0.10	**0.98** $\pm$ 0.01	0.85 $\pm$ 0.03	10.4 $\pm$ 0.60

Table F.2: `Learn-to-Race` task (Herman et al., 2021) results on `Track01` (Thruxton Circuit), for learning-*based* agents.

Agent	**ECP** (↑)	**ED*** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
`HUMAN`	**100.0** $\pm$ 0.0	78.6 $\pm$ 1.7	79.29 $\pm$ 4.7	2.4 $\pm$ 0.1	0.93 $\pm$ 0.01	1.00 $\pm$ 0.02	**11.7** $\pm$ 0.1
`SAC`	61.6 $\pm$ 38.6	8.39 $\pm$ 11.12	48.0 $\pm$ 30.9	1.79 $\pm$ 2.17	0.28 $\pm$ 0.38	0.05 $\pm$ 0.08	3.22 $\pm$ 3.59
`SafeRandom` (ours)	83.1 $\pm$ 24.5	484.53 $\pm$ 725.30	11.2 $\pm$ 0.9	3.73 $\pm$ 0.47	0.86 $\pm$ 0.12	0.01 $\pm$ 0.01	11.06$\pm$3.31
`SafeSAC` (ours)	49.1 $\pm$ 41.7	676.09 $\pm$ 805.83	33.8 $\pm$ 26.2	1.58 $\pm$ 1.03	0.98 $\pm$ 0.05	0.01 $\pm$ 0.01	9.15 $\pm$ 3.20
`SPAR` (ours)	**79.9** $\pm$ 23.2	**34.81**$\pm$6.14	**53.3**$\pm$3.8	**0.68**$\pm$1.58	**0.99**$\pm$0.03	**0.11**$\pm$0.10	8.06 $\pm$ 1.41