# Exploring Safer Behaviors for Deep Reinforcement Learning

**Enrico Marchesini**[*], **Davide Corsi**[*], **Alessandro Farinelli**

Department of Computer Science, University of Verona
enrico.marchesini@univr.it, davide.corsi@univr.it

**Abstract**

We consider Reinforcement Learning (RL) problems where an agent attempts to maximize a reward signal while minimizing a cost function that models unsafe behaviors. Such formalization is addressed in the literature using constrained optimization on the cost, limiting the exploration and leading to a significant trade-off between cost and reward. In contrast, we propose a Safety-Oriented Search that complements Deep RL algorithms to bias the policy toward safety within an evolutionary cost optimization. We leverage evolutionary exploration benefits to design a novel concept of safe mutations that use visited unsafe states to explore safer actions. We further characterize the behaviors of the policies over desired specifics with a sample-based bound estimation, which makes prior verification analysis tractable in the training loop. Hence, driving the learning process towards safer regions of the policy space. Empirical evidence on the Safety Gym benchmark shows that we successfully avoid drawbacks on the return while improving the safety of the policy.

## Introduction

A Deep Reinforcement Learning (DRL) agent tries to learn a policy maximizing a long-term objective by trials and errors in large state spaces (Sutton and Barto 2018). This learning paradigm achieved impressive performance in various domains (e.g., games (Silver et al. 2018)). However, several applications (e.g., robotics (OpenAI et al. 2019)) typically involve safety criteria that are complex to model with simple *reward shaping* (Amodei et al. 2016).

We consider the class of problems where unsafe behaviors are specified with an auxiliary cost signal to maintain safety specifications separate from the task objective (e.g., the long-term reward) (Garcıa and Fernández 2015). In the literature, Constrained Markov Decision Processes (CMDP) (Altman 1999) are used to formalize such problems due to the intuitive way of constraints (on the cost) to encode safety criteria (Liu, Ding, and Liu 2020; Stooke, Achiam, and Abbeel 2020). However, constrained DRL often violates the constraints introduced in the optimization and naturally limits exploration (Ray, Achiam, and Amodei 2019). Conversely, efficient exploration is crucial to avoid getting stuck in local optima or failing to learn proper behaviors, obtaining low returns (Hong et al. 2018). Such trade-off between having efficient exploration to achieve good performance, and the limitation induced by constraints, suggest to investigate alternative ways to overcome the typical contrast in the design of Safe DRL algorithms that are either based on the *optimality criterion* (e.g., introduce the concept of risk in the optimization), or on the *exploration process* (e.g., avoid undesirable situations) (Garcıa and Fernández 2015).

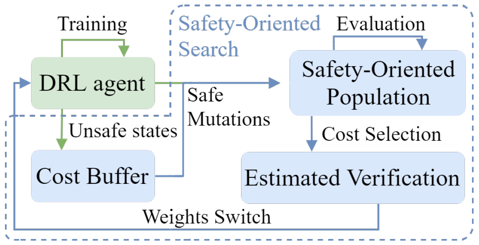

Figure 1: Overview of SOS

To this end, we propose Safety-Oriented Search (SOS) to combine *exploration process* and *optimality criterion* into a unique framework, depicted in Figure 1, that works on top of existing DRL algorithms. Our goal is to bias policies toward safety without multi-objective or constrained optimization, hence without formalizing the problem as a CMDP. We leverage Evolutionary Algorithms (EA) (Fogel 2006) for SOS to augment DRL, proposing a novel concept of *Safe Mutations* (SM). SM exploits the visited states deemed unsafe according to the cost to approximate the per-weight sensitivity of the actions over such undesired situations. Then, such sensitivity is used to compute safety-informed perturbations that locally biases the agent policy to explore different behaviors (i.e., actions) in the proximity of the unsafe states (i.e., the *exploration process*). In more detail, an evolutionary population is periodically generated from the DRL policy using SM, and it is evaluated independently over a set of trials to select the subset of individuals with returns comparable to the DRL agent and a lower cost.

Assuming that this subset improves the auxiliary cost, we note that such signal is typically sparse (i.e., it is not trivial to shape the risk associated with every state in a high-dimensional space). Hence, the cost function does not fully characterize the behaviors of the policy, sharing the issues

[*]These authors contributed equally.

of using sparse rewards (Hong et al. 2018). For this reason, we argue that a tractable characterization of the behaviors of a Deep Neural Network (DNN) is instrumental in evaluating the safety of a policy (Corsi et al. 2020). To this end, we define a novel *violation* metric that quantifies the number of correct decisions that a policy chooses over desired safe specifications, using interval-analysis verification (Liu et al. 2021). We then select the policy in the subset with the lowest violation that will replace the agent (i.e., the *optimality criterion*). However, Formal Verification (FV) (Wang et al. 2018a; Weng et al. 2018) is known to be computationally demanding and can not be used directly in the training loop without assumptions that can not be satisfied in practice (e.g., having an optimal policy (Lutjens, Everett, and How 2020)). We propose to relax the formal guarantees of prior bound-estimation methods (Wang et al. 2018a) with a sample-based estimation to make this analysis tractable. Crucially, our *Estimated Verification* (EV) impacts the training within a negligible overhead. Summarizing, we foster safer behaviors with periodical EA evaluations, while the reward objective is optimized by the DRL training process.

We compare a SOS implementation of PPO (Dhariwal et al. 2017) and TD3 (Fujimoto, van Hoof, and Meger 2018) over constrained approaches, namely CPO (Achiam et al. 2017), Lagrangian-PPO (Stooke, Achiam, and Abbeel 2020), and IPO (Liu, Ding, and Liu 2020), in the recent Safety Gym benchmarks (Ray, Achiam, and Amodei 2019). Empirical evidence confirms that constrained DRL does not ensure the satisfaction of constraints, while its limited exploration leads to low returns. In contrast, SOS-based algorithms achieve long-term returns similar to the DRL baseline and successfully minimizes the auxiliary cost objective, obtaining comparable performance to constrained DRL.

## Preliminaries

A CMDP (Altman 1999) is a Markov Decision Process with an additional set of constraints $\mathcal{C}$ based on $C_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ ($i \in \{1, \dots, k\}$) cost functions (similar to the reward) and $\mathbf{h} \in \mathbb{R}^k$ thresholds for the constraints. The $C_i$*-return* is defined as $J_{C_i}(\pi) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t)]$, where $\gamma \in (0, 1)$ is the discount, $\tau = (s_0, a_0, \dots)$ is a trajectory, $\pi = \{\pi(a|s) : s \in \mathcal{S}, a \in \mathcal{A}\}$ denotes a policy in state $\mathcal{S}$ and action $\mathcal{A}$ spaces. Constraint-satisfying (feasible) policies $\Pi_{\mathcal{C}}$, and optimal policies $\pi^*$ are thus defined as:

$$\Pi_{\mathcal{C}} := \{\pi \in \Pi : J_{C_i}(\pi) \leq h_i,\ \forall i\}, \quad \pi^* = \arg\max_{\pi \in \Pi_{\mathcal{C}}} J(\pi)$$

where $J(\pi) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$ is the expected discounted return that we aim at maximizing in a standard MDP; $\Pi$ are the stationary policies, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. Without loss of generality, we consider the case of one cost function (as in recent constrained DRL literature (Ray, Achiam, and Amodei 2019; Stooke, Achiam, and Abbeel 2020; Liu, Ding, and Liu 2020)) and we will discuss later how SOS could handle multiple cost functions.

### Evolutionary Algorithms

Evolutionary approaches represent black-box optimizations characterized by generation, perturbation (mutation), and selection operators (Fogel 2006), which can be used to augment exploration (Khadka and Tumer 2018). EAs typically evolve a population of $p \in \mathbb{N}$ individuals (genomes), represented by parameters (weights) $\theta_i$ ($i \in \{1, \dots, p\}$). The individuals are evaluated to produce a fitness score used by the selection operator to choose the best genome. Mutating a policy with simple Gaussian noise $\mathcal{N}$, however, can lead to disruptive changes (Lehman et al. 2018) that can be naively address uses zero-mean and low standard deviation (Martin H. and de Lope 2009). Otherwise, if we define a genome as a DNN parametrized by $\theta$ that represents a function $f_\theta : \mathcal{D}_\mathbf{x} \rightarrow \mathcal{D}_\mathbf{y}$ (input $\mathbf{x} \in \mathcal{D}_\mathbf{x} \subseteq \mathbb{R}^n$ and output $\mathbf{y} \in \mathcal{D}_\mathbf{y} \subseteq \mathbb{R}^m$, with input/output size $n$, $m$), and a vector of states $\mathbf{s}$, we can express the average divergence of the outputs $\mathbf{y}$ as a result of a perturbation $\delta$ as:

$$d(f_\theta, \delta) = \frac{\|f_\theta(\mathbf{s}) - f_{\theta+\delta}(\mathbf{s})\|_2}{|\mathbf{s}|} \tag{1}$$

where $f_\theta(\mathbf{s})$ are the forward propagations of the states through the DNN. A more flexible way to avoid disruptive mutations assumes using a differentiable DNN to approximate $d$ with gradient information (Lehman et al. 2018). In detail, it considers the following first-order Taylor expansion to model an output $y_j \in \mathbf{y}$ ($j \in \{0, \dots, |\mathbf{y}|\}$) as a function of perturbations $\delta$ over the states $\mathbf{s}$:

$$y_j(f_\theta, \delta) = f_\theta(\mathbf{s})_j + \delta \nabla_\theta f_\theta(\mathbf{s})_j \tag{2}$$

In later sections, we discuss how our SM specializes naive gradient-based mutations of Eq. 2 to explore safer behaviors.

### Formal Verification

Formal verification for DNNs involves checking whether desired input-output relations (properties) hold (Liu et al. 2021). For example, it is possible to examine the neighborhood of a given input $\mathbf{x_0}$, to find the maximum possible disturbance $r_0$ that satisfy the following assertion:

$$\mathbf{x} \in \mathcal{X} \Rightarrow \mathbf{y} = f_\theta(\mathbf{x}) \in \mathcal{Y} \tag{3}$$

where $\mathcal{X} = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x_0}\|_2 \leq r_0\} \subseteq \mathcal{D}_\mathbf{x}$ and $\mathcal{Y} \subseteq \mathcal{D}_\mathbf{y}$ is a feasible output set. This formalization, however, encodes the input space of the properties as a hyperrectangle, limiting the application of Eq. 3 to general scenarios. This has been addressed as follows to represent different geometries (e.g., polytopes) (Corsi et al. 2020; Liu et al. 2021):

$$\text{If } x_0 \in [a_0, b_0] \wedge \dots \wedge x_n \in [a_n, b_n] \Rightarrow y \in [c, d] \tag{4}$$

where $x_k \in \mathbf{x}$ ($k \in \{0, \dots, n\}$) are the inputs of the DNN, and $y \in \mathbf{y}$ is an output. We consider reachability methods (Wang et al. 2018a,b; Weng et al. 2018) to verify properties in the form of Eq. 4. This class of approaches computes an over-approximation of the exact output reachable set propagating the domain $\mathcal{X}$ through the network. In detail, the lower and upper bound propagation of each input $k$ ($I_k = [l_k, u_k]$) is approximated first by computing the "pre-activation" bounds with the following linear mapping, which is used in the literature (Liu et al. 2021):

$$\begin{aligned} l_{new} &= \max(\theta, 0) * l + \min(\theta, 0) * u \\ u_{new} &= \max(\theta, 0) * u + \min(\theta, 0) * l \end{aligned} \tag{5}$$

then, the upper and lower bound of the activation are computed as follows by propagating these values through the activation function, which assume having monotonic activation function $\rho$ applied to an interval $I = [l, u]$:[1]

$$\rho(I) = \begin{cases} [\rho(l), \rho(u)] & \textit{if monotonically increasing} \\ [\rho(u), \rho(l)] & \textit{if monotonically decreasing} \end{cases} \tag{6}$$

Eq. 5, 6 are applied layer-by-layer and node-wise to compute the output reachable set $\Gamma(\mathcal{X}, f_\theta) := \{\mathbf{y} : \mathbf{y} = f_\theta(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}\}$. Hence, a property in the form of Eq. 3 (or more generally Eq. 4) is considered satisfied if it belongs to the output set, i.e., $\Gamma(\mathcal{X}, f_\theta) \subseteq \mathcal{Y}$.

## Safety-Oriented Search

SOS proposes two mechanisms to foster safety: (i) a novel concept of *Safe Mutations* to augment DRL with a policy search devoted to explore safer behaviors. (ii) An *Estimated Verification* that relaxes the guarantees of FV to tractably characterize the behaviors of the policies during the training.

The general flow of SOS is presented in Algorithm 1: we augment the DRL agent training with a *cost-buffer* $B_c$ which stores the visited unsafe states (according to the cost). Periodically, we sample a batch $b$ from $B_c$ to compute the per-weight safety-informed sensitivity $\omega$ of the agent outputs over its weights $\theta_a$. This is used by SM to generate a population of $n$ individuals (DNNs) $\mathcal{P} = \{p_1, \dots, p_n\} \cup \{p_a\}$ with weights $\theta_{\mathcal{P}}$ ($p_a$ is a copy of the agent), voted to explore for safer behaviors. $\mathcal{P}$ is evaluated in a set of epochs to collect the individuals average reward $R_p$ and cost $C_p$ that define the fitness score $F_p = (R_p, C_p)\ \forall p \in \mathcal{P}$, which is used to select a subset of genomes $\mathcal{P}'$ as:[2]

$$\mathcal{P}' = \{p_j \in \mathcal{P} : F_{p_j} \geq F_{p_a},\ j \in \{1, \dots, n\}\} \cup \{p_a\} \tag{7}$$

where $F_{p_j} \geq F_{p_a} \Rightarrow R_{p_j} \geq R_{p_a} \wedge C_{p_j} \leq C_{p_a}$. By choosing an appropriate number of evaluation epochs for the population, we define the individuals in $\mathcal{P}'$ to be safer than $p_a$ as they have higher (or equal) reward and lower (or equal) cost.

Although $\mathcal{P}'$ improves the auxiliary cost, such sparse metric does not characterize the behaviors of the policies. For this reason, SOS selects the "safest" genome $p^* \in \mathcal{P}'$ using the *Estimated Verification* detailed in the related section. Hence, if $p^*$ is a SM perturbed policy (i.e., $p^* \neq p_a$) it substitutes $p_a$ to continue the training, otherwise the training that is running in parallel, continues. We note that in a worst-case scenario we match the performance of the baseline DRL agent as we would never switch its policy. Summarizing, SOS proposes a periodical search devoted to safety-oriented exploration to improve the DRL policy, simulating a small gradient step toward a "safer" (better) policy.

### Safe Mutations

Perturbing the weights of a DNN via simple Gaussian noise can lead to disruptive policy changes (Martin H. and de Lope 2009; Lehman et al. 2018). Gradient information (sensitivity) can be used to design mutations that avoid such detrimental behaviors, normalizing the perturbation by a per-weight sensitivity (Lehman et al. 2018). We leverage the cost function to design SM, avoiding disruptive changes to the DRL policy while biasing it to explore safer behaviors. We consider a baseline Gaussian noise $\mathcal{G}$ for the perturbations and normalize it with our safety-informed sensitivity $w$. The resultant mutations $\delta_{SM}$ are applied to the agent weights $\theta_a$ to generate $\mathcal{P}$. One way to compute $\omega$ considers the gradient of the actual divergence (Eq. 1) (Lehman et al. 2018):

$$\begin{gathered} \nabla_{\theta_a} d(f_{\theta_a}, \mathcal{G}) \approx \nabla_{\theta_a} d(f_{\theta_a}, 0) + H_{\theta_a}(d(f_{\theta_a}, 0))\mathcal{G} \\ \omega(f_{\theta_a}) = abs(\nabla_{\theta_a} d(f_{\theta_a}, \mathcal{G})) \end{gathered} \tag{8}$$

where $H_{\theta_a}$ is the Hessian of divergence with respect to $\theta_a$. However, this requires second-order approximations and therefore it is computationally demanding. To address this, we use the per-weight magnitude of the gradient of the outputs $\mathbf{y} = f_{\theta_a}(b)$, where $b$ is a batch of unsafe states randomly sampled from $B_c$, to estimate the sensitivity $\omega$ to that weight with a first-order approximation:

$$\begin{gathered} \omega(f_{\theta_a}) = \sum_{\mathbf{y}} \left( \frac{\sum_{\mathbf{s}} abs(\nabla_{\theta_a} f_{\theta_a}(\mathbf{s}))}{|\mathbf{s}|} \right) * \frac{1}{|\mathbf{y}|} \\ \delta_{SM}(f_{\theta_a}) = \frac{\mathcal{G}}{\omega(f_{\theta_a})} \end{gathered} \tag{9}$$

where each unsafe experience equally contributes to $\omega$ to reduce the overall changes to the policy.[3] In practice, we note that using a threshold $\omega'$ to limit the mutation rescaling (i.e., $w$) leads to better performance. To summarize, our idea is to design safety-oriented gradient information using visited unsafe states, to bias the policy to explore different actions in the proximity of such situations.

Algorithm 1: Safety-Oriented Search

```
1:  Given:
      • a DRL agent with weights $\theta_a$ at training epoch $e$
      • a reachability verifier $V_r$ and desired safety-properties $P_s$   ▷ e.g., Neurify, ReluVal
      • a cost-buffer $B_c$ filled with unsafe samples
      • periodicity $e_s$ for SOS and population size $n$
      • scale $\sigma$ for the Gaussian noise $\mathcal{G}$ and threshold $\omega'$
2:  While the standard training of the agent proceeds:
3:  if $e \ \% \ e_s = 0$ then
4:      $b \leftarrow$ Sample an unsafe batch from $B_c$
5:      $\mathcal{P} \leftarrow n+1$ copies of $\theta_a$ (each $p \in \mathcal{P}$ has weights $\theta_p$)
6:      $\mathcal{G} \leftarrow \mathcal{N}(0, \sigma)\ \forall$ weight $\in \theta_a$ (i.e., baseline noise)
7:      $\omega \leftarrow$ Eq. 9 using $b$, replacing values $\leq \omega'$ with $\omega'$
8:      $\theta_p \leftarrow \theta_p + \frac{\mathcal{G}}{\omega}$;   $F_p \leftarrow$ Evaluate($\theta_p$) , $\forall p \in \mathcal{P}$
9:      Select a "safer" subset $\mathcal{P}'$ using $F_{\mathcal{P}}$ as Eq. 7
10:     $v_{p'} \leftarrow V_r(\tilde{\Gamma}_{p'}, P_s)$ using Def.  and Def. , $\forall p' \in \mathcal{P}'$
11: end if
12: $\theta_a \leftarrow \min_{\theta_{\mathcal{P}'}} v_{\mathcal{P}'}$
13: Continue the training of the agent until the next SOS
```

[1]This is standard in verification literature (Zhang et al. 2018), as the most common activation, e.g., ReLU, Tanh, are monotonic.

[2]Note that ordering among fitness tuples is feasible as its components $R, C \in \mathbb{R}$.

[3]We use the absolute value due to the interest in the magnitude, and not the sign, of the slope.

## Estimated Verification

We leverage formal verification for DNNs to evaluate the subset of safer genomes $\mathcal{P}'$ and characterize their behavior over a set of given properties in the form of Eq. 4. In contrast to prior work that typically returns SAT if the property is satisfied or UNSAT if it is not for at least one input (Wang et al. 2018a; Katz et al. 2017), we aim at quantifying the safety of the individuals over the properties.[4] We propose the following *violation* metric $v$ to measure the number of violations.

**Definition 1** (Violation metric). *Given a (safety) property $p := \mathbf{x} \in \mathcal{X} \Rightarrow \mathbf{y} = f_\theta(\mathbf{x}) \in \mathcal{Y}$ on $f_\theta$, and its reachability set $\Gamma(\mathcal{X}, f_\theta) := \{\mathbf{y} : \mathbf{y} = f_\theta(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}\}$. Given $\mathcal{X}_{SAT}, \mathcal{X}_{UNSAT} \subseteq \mathcal{X}$ such that $\Gamma(\mathcal{X}_{SAT}, f_\theta) \subseteq \mathcal{Y}$ and $\Gamma(\mathcal{X}_{UNSAT}, f_\theta) \cap \mathcal{Y} = \emptyset$.[5] We define the violation metric as $v = |\mathcal{X}_{UNSAT}|/|\mathcal{X}|$.*

Despite recent advances in the field of formal analysis for DNNs (Wang et al. 2018a; Weng et al. 2018), existing tools require non-negligible computation time to approximate the reachable set using Eq. 5, 6 (Liu et al. 2021). Hence, these approaches can not be directly applied to verify the properties (and compute $v$) during the training. We propose the following empirical strategy to estimate the reachability set, using feed-forward steps of the DNN. We thus apply the verification phase of an existing framework (Wang et al. 2018a) to the estimated bounds, obtaining our EV method that enables SOS to perform the verification in the training loop.

**Definition 2** (Estimated Reachability Set). *Given a (safety) property $p := \mathbf{x} \in \mathcal{X} \Rightarrow \mathbf{y} = f_\theta(\mathbf{x}) \in \mathcal{Y}$ on $f_\theta$, and a set $\mathcal{X}' \subseteq \mathcal{X}$ of $m$ samples. We define the reachability set:*

$$\tilde{\Gamma}(\mathcal{X}', f_\theta) := \{[\min(f_\theta(\mathcal{X}')_y, \max(f_\theta(\mathcal{X}')_y)] \; \forall y \in \mathbf{y}\}$$

Crucially, given a discretization value $\epsilon$ for the input space of a property (e.g., in practical tasks such as robotics, $\epsilon$ could be the precision of the sensor), our estimation returns the exact reachability set using $m = \frac{|\mathcal{X}|}{\epsilon}$ different samples. However, our interest is to characterize the behaviors in the proximity of unsafe states, hence we further exploit the cost-buffer $B_c$ to compute a cost-oriented reachability set:

**Definition 3** (Estimated Cost Reachability Set). *Given a (safety) property $p := \mathbf{x} \in \mathcal{X} \Rightarrow \mathbf{y} = f_\theta(\mathbf{x}) \in \mathcal{Y}$ on $f_\theta$, and the cost-buffer $B_c$. We define the cost reachability set:*

$$\tilde{\Gamma}(\mathcal{X} \cap B_c, f_\theta) :=$$
$$\{[\min(f_\theta(\mathcal{X})_y \cap B_c, \max(f_\theta(\mathcal{X})_y \cap B_c)] \; \forall y \in \mathbf{y}\}$$

## Limitations of Safety-Oriented Search

Our evolutionary search assumes to have access to a simulation environment. This is common in DRL, where significant results have been achieved mainly using simulation and transferring the policy on real platforms (Juliani et al. 2018; Zhao, Queralta, and Westerlund 2020). We also assume a single cost function as in prior constrained DRL literature (Ray, Achiam, and Amodei 2019). However, it would be possible to handle multiple cost functions by using crossover operators, similarly to (Khadka et al. 2019). Verification assumes knowing desired specifications to design the properties, which is typically available in tasks with safety requirements. Commonly with prior formal verification (Liu et al. 2021), such properties are hand-designed, hence as the complexity of the task increases, the input space typically grows, and writing safety properties may be unfeasible. To this end, we believe that producing compact state representations to reduce complexity (Cuccu, Togelius, and Cudré-Mauroux 2019) could be an interesting topic for future investigation.

[4]We assume to have prior knowledge of desired safety specifications that allows designing such properties.

[5]Note that $\mathcal{X}_{\text{SAT}} \cup \mathcal{X}_{\text{UNSAT}} = \mathcal{X}$ and $\mathcal{X}_{\text{SAT}} \cap \mathcal{X}_{\text{UNSAT}} = \emptyset$.

## Experiments

We investigate an on-policy version of SOS based on PPO (SOS-PPO) against PPO, CPO, Lagrangian-PPO (L-PPO), and IPO[6]. We compare over constrained DRL as it is the most closely related to the idea of addressing safety using cost functions. We evaluate the effectiveness of SOS-based algorithms at minimizing the cost signal with the evolutionary search while preserving returns. Conversely, constrained DRL hinders exploration (to satisfy cost constraints) at the expense of low returns, leading to a significant performance trade-off (Ray, Achiam, and Amodei 2019).

We consider six tasks recommended by the authors of Safety Gym as a benchmark for our class of problems. To characterize the policy behaviors with EV, we designed three properties to ensure that the agent chooses rational actions in the visited states. Given the cardinality of the inputs in the tasks, we report a natural language description of such properties, which are shared across the tasks. We remind to the supplemental material for the formal definition in the form of Eq. 4 and more details about the environments:

- $p_\uparrow$: If the robot has hazards too close on the front, it must turn in any direction or move backward.
- $p_\rightarrow$: If the robot has hazards too close on the right and on the front, it must turn left or move backward.
- $p_\leftarrow$: If the robot has hazards too close on the left and on the front, it must turn right or move backward.

We remark that our goal is to provide a high-level overview of core navigation skills to quantify the overall behaviors of the agent. Hence, despite using only three properties, their high-level formalization was sufficient to cover over $98.7 \pm 0.5\%$ of the visited unsafe states.

Data are collected on a RTX 2080, using the hyper-parameters reported in the supplemental material. We use prior implementations of CPO, PPO, and L-PPO which improves the constraint satisfaction, and our version of IPO, which we carefully tuned since authors did not release code.[7] The additional epochs required by SOS are included in all the results for a fair comparison, but the training time overhead is negligible due to parallelization (i.e., for each

[6]Supplemental material has additional experiments with a TD3 implementation of SOS, and a robotic navigation scenario.

[7]We refer to the original papers for specific details about the approaches and hyper-parameters.

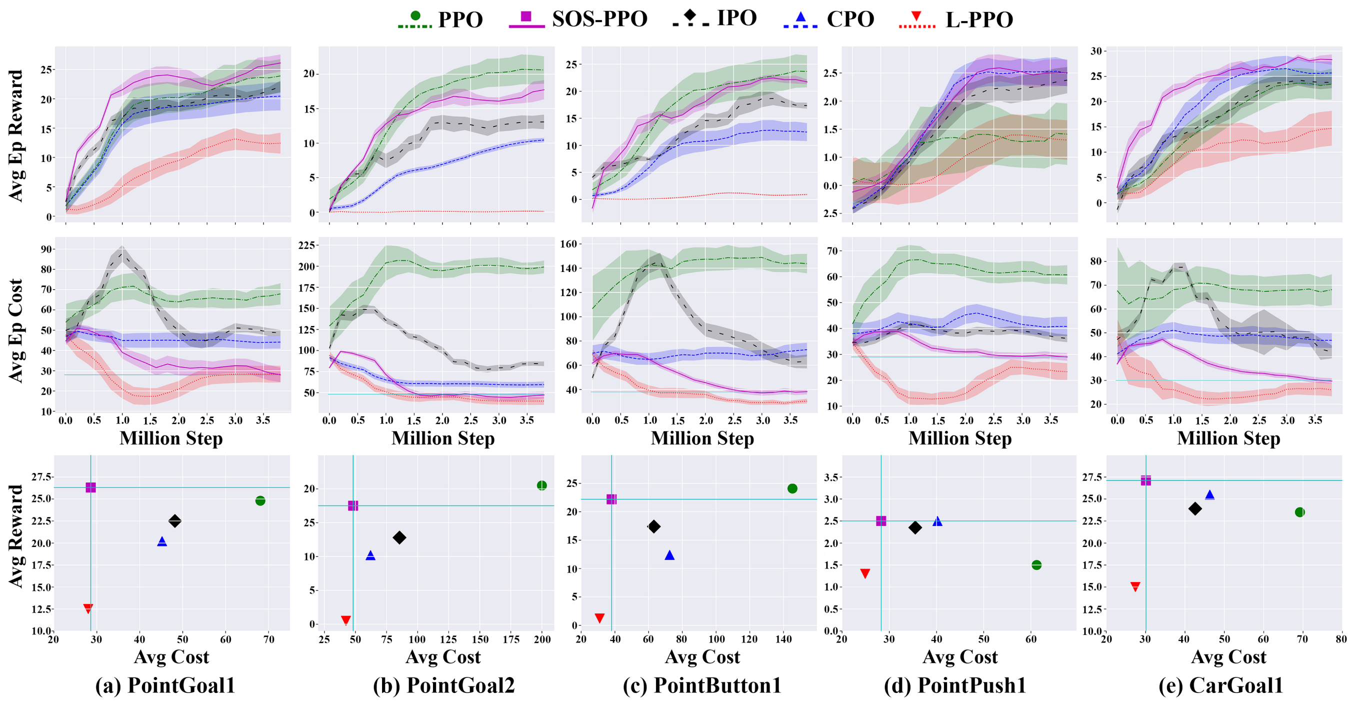

Figure 2: Comparison of PPO, SOS-PPO, IPO, CPO, L-PPO in Safety Gym. Each column (i.e., each task) shows the average reward and cost during the training, and Pareto frontier at convergence.

individual, both the search and EV are strictly independent and SOS-PPO trains with an overhead of $4 \pm 2\%$ over PPO).

Given the importance of the statistical significance of the results (Colas, Sigaud, and Oudeyer 2019; Henderson et al. 2018), we report mean and standard deviation collected over ten independent runs with different random seeds. This motivates slightly different results over the original implementations that were evaluated over few seeds. Finally, considering the recent interest in the $CO_2$ emissions produced by training DNNs, in the supplemental material we report the total amount of emissions and how we offset them.

## Empirical Evaluation

For each task, we plot: (i) average reward, (ii) auxiliary cost with a dashed line for the cost-limit $h$ of constrained DRL (for a fair comparison, $h$ is set to the cost reached by SOS at convergence), and (iii) Pareto frontier of return versus cost at convergence, which drastically improves (up and to the left) with SOS. Figure 2 shows the results in five tasks (arranged in columns) for the complete SOS-PPO implementation. This uses EV to compute the violation metric for the selection, by verifying $p_{\uparrow}$, $p_{\rightarrow}$, $p_{\leftarrow}$ using the cost reachability set (Def. 3). Considering the previously discussed limitation of EV, we note that writing properties for the Doggo task without knowing the kinematics of the robot is not trivial. For this reason, we report in the supplemental material the results of a SOS-PPO implementation without the EV part for the Doggo task, which confirms the following results.

For the baselines we obtain the same trend of prior work (Ray, Achiam, and Amodei 2019), where the objective of maximizing the reward while limiting the cost present a

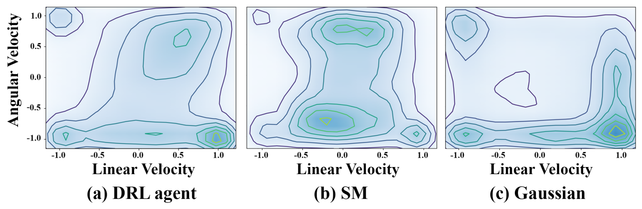

Figure 3: Overview of the DRL policy (a) and its mutation with SM (b) or simple Gaussian noise (c).

meaningful trade-off: (i) PPO obtains high returns by taking unsafe actions; (ii) L-PPO is the most reliable in enforcing the constraint but typically achieves very low returns; (iii) CPO achieves interesting rewards due to its approximation errors, which prevent it from satisfying the constraints; (iv) IPO returns similar result to CPO but has the advantage of being a first-order method. Conversely, SOS-PPO successfully maintains comparable returns over PPO, while drastically reducing the long-term cost, attaining cost values similar to L-PPO at convergence. A detailed analysis of the returns of L-PPO highlights that its constrained policies are prone to perform poor behaviors (e.g., L-PPO often learns to stand still or move in circles for entire epochs to avoid collisions). This further motivates the introduction of the EV in the training loop to characterize the behavior of the policies and select the safest individual using our violation metric.

## Safe Mutation and Estimated Verification

We analyze SOS to show that: (i) SM avoids disruptive changes to the policy while biasing exploration towards safety; (ii) the selection based on EV successfully characterize the behaviors of the policies, resulting in better performance.[8] Figure 3 shows the plots generated by fitting a Gaussian kernel density model on the action selected over several epochs. This explanatory overview reports the behaviors (i.e., chosen actions) of the DRL policy and two perturbed versions with SM and simple Gaussian noise $\mathcal{G}$ in the first population search. For a broader view of the mutation operators in different stages of the training, we remind to supplemental material. Clearly, SM locally biases the agent policy, maintaining a similar action distribution (i.e., behaviors). In contrast, Gaussian noise causes disruptive changes, resulting in very different and typically worst behaviors.

This is further supported by Figure 4 (top), which shows the average reward and cost collected by two SOS-PPO implementations that (i) uses Gaussian noise (i.e., without SM and the cost-buffer), and (ii) uses standard output-gradient mutations (i.e., with the sensitivity computed on arbitrary samples). Hence, these ablation studies have no information on the auxiliary cost to generate the population and can not cope with the safety aspect of the tasks, resulting in lower returns and higher cost over PPO.

Moreover, Figure 4 (bottom) shows the same metric for a SOS-PPO implementation that selects $p^* \in \mathcal{P}$ without EV, i.e., it selects the individual with comparable reward to the DRL agent and with minimum cost. Crucially, these results confirm the importance of our framework in the scenarios where prior knowledge can not be included for the design of the properties for EV as it maintains superior performance over the baselines. However, we note that this also supports our claims on the importance of integrating EV in the selection process to characterize the actual behaviors of the policies as this allows discovering "safer" (better) policies since the early stages of the training, improving both the maximization of the reward and the minimization of the cost.

## Approximation of the Estimated Verification

We analyze EV to show that: (i) the time required by prior formal verification approaches prevents their application in the training loop, and (ii) EV provides comparable results (i.e., violation) over formal approaches. Figure 5a shows the cumulative time required to verify our three properties using prior formal verification (i.e., Neurify (Wang et al. 2018a)), and EV with different sample sizes $m$. We remark that the analysis is periodically performed in the training loop for each safety-oriented search. Crucially, EV drastically reduces the computational time by more than $73.8\times$ for each verification. Hence, in our experiments, EV is the only feasible solution as the average training time of SOS in PointGoal1 requires a couple of hours, while it would require more than 48 hours using formal verification.

Finally, EV also returns similar results over formal verification as the violation metric of the latter is comparable with the one computed with EV. Figure 5b shows an explanatory computation (during the training) of the violation metric for our three safety properties. Note that we only report the

[8]Ablation studies are performed in PointGoal1 as we obtained similar results across the Safety Gym tasks.

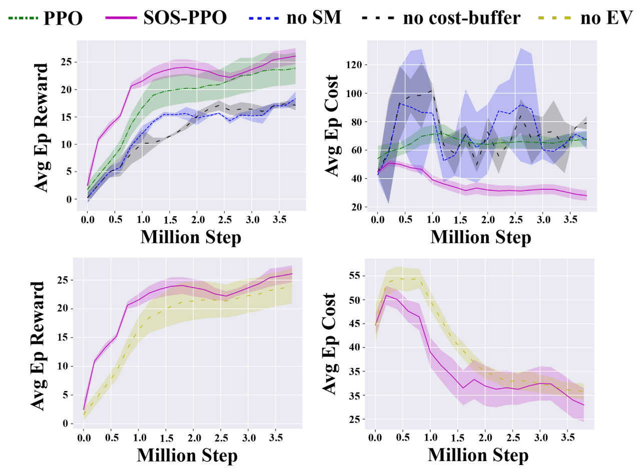

Figure 4: Top: ablation study of SOS-PPO implementations with (i) Gaussian mutations and (ii) output-gradient mutations without cost-buffer. Bottom: ablation study of SOS-PPO that selects the best individual that have minimum cost, instead of using EV.

curve of EV that uses the cost reachability set (Def. 3) as it has comparable performance over the estimated reachability set (Def. 2) in approximately half the time. Results highlight that with little tuning of the sample size $m$, we obtain an accurate estimation of the violation (i.e. in our experiments, with $m = 60$ we have an error between the formal violation and the estimated one below $0.5\%$).

## Related Work

Safety critics (Brijen Thananjeyan 2020; Homanga Bharadhwaj 2021; Brijen Thananjeyan 2021) rely on estimating the probability of incurring into unsafe states, given a state-action pair. However, such approaches could potentially return misleading information for policy improvement, especially in the early stages of the training, where safety critics are pre-trained on offline data. Such offline demonstration represents unsafe samples which should cover a broad variety of unsafe behaviors for a robust pre-training. This may be challenging and a possible alternative is to use data collected by human policies or human supervision. Moreover, these methods also introduce overhead in the action sampling process as each step has to compute different samples (i.e., the action, the probability of failure, etc.), which can hinder their application to the physical hardware that requires high-frequency control.

In contrast, we compare SOS with constrained DRL as it is more related to our approach. In more detail, CPO (Achiam et al. 2017) has near-constrained satisfaction guarantees, but the Taylor approximations lead to inverting a Fisher matrix, possibly resulting in infeasible updates and demanding recovery steps. Similarly, PCPO (Yang et al. 2020) also has theoretical guarantees for constraint satisfaction, but uses second-order approximations and has mixed improvements over CPO (Zhang, Vuong, and Ross 2020). Lyapunov-based algorithms (Chow et al. 2018, 2019), in

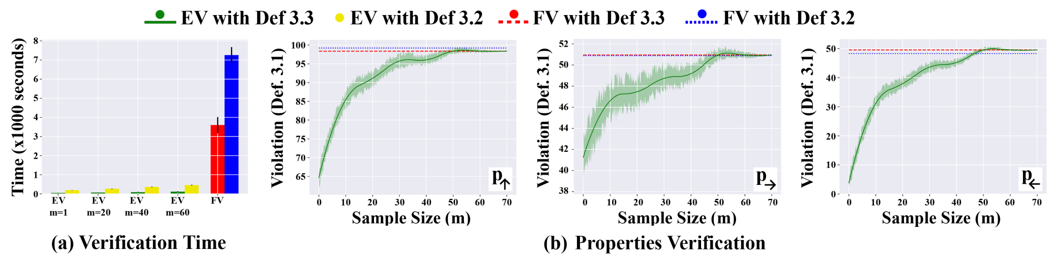

Figure 5: Comparison between formal analysis and EV: (a) shows the computation time (with different values of $m$ for EV); (b) shows that EV can accurately estimates the violation metric.

contrast, combine a projection step with action-layer interventions, similarly to the safety layer of (Dalal et al. 2018). However, the cardinality of Lyapunov constraints equals the number of states, resulting in a non-negligible implementation cost. Finally, Lagrangian methods (Ray, Achiam, and Amodei 2019; Stooke, Achiam, and Abbeel 2020) reduce the complexity of prior approaches, with promising constraints satisfaction. These methods transform the equality-constrained problem, defined over a real vector, with a dual variable that forms the Lagrangian. Gradient-based algorithms then iteratively update the primal and dual variables with the Lagrange multiplier $\lambda$ that acts as a learned penalty and is used to satisfy the constraint. This adapts to the constrained setting (Geibel and Wysotzki 2005; Altman 1998) representing a well-known constrained DRL approach due to its simplicity and good cost-limit satisfaction (Ray, Achiam, and Amodei 2019; Stooke, Achiam, and Abbeel 2020). Similarly, IPO (Liu, Ding, and Liu 2020) reduces the constrained problem into an unconstrained one by augmenting the objective with logarithmic barrier functions, which provide sub-optimal solutions.

García and Fernández (Garcıa and Fernández 2015) shows that constrained approaches have several drawbacks such as the careful tuning of the threshold $h$ as high values mean that they are too permissive, or conversely, too restrictive. Furthermore, such approaches rely on strong assumption that typically can not be satisfied in practice (e.g., having an optimal policy) to provide theoretical guarantees on the constraints satisfaction. Hence, constrained DRL is also not devoid of short-term fatal consequences as empirical evidence shows that they typically fail at satisfying the imposed constraints (Ray, Achiam, and Amodei 2019), which is also related to the non-linear approximation nature of DNNs.

Furthermore, constraints naturally limit exploration, causing getting stuck in local optima or failing to properly learn (Conti et al. 2018; Hong et al. 2018). In contrast, we leverage EA to design SOS as prior combinations of DRL and EA show a beneficial transfer of information between the two approaches (Khadka and Tumer 2018; Khadka et al. 2019; Marchesini, Corsi, and Farinelli 2021). These methods, however, use the EA only for improving the return and can not be trivially extended to address the safety component. To characterize the behaviors of the policies, we rely on formal analysis. Given a safety property in the form of Eq. 3, 4 and a DNN, a verification framework either guarantee that the property is always satisfied or return counterexamples (Liu et al. 2021). The two main families of approaches to formally verify such properties are optimization and reachability (Liu et al. 2021). While the first aims at falsifying the assertion by finding a counterexample (Katz et al. 2017), the latter improve scalability and aims at propagating a given input domain to find the reachable set for the output (Wang et al. 2018b,a) and, using different search strategies (Dutta et al. 2017; Bunel et al. 2018; Ehlers 2017), it aims at finding the counterexamples.

## Discussion and Broader Impact

We summarize our contributions in Safety-Oriented Search, a framework that combines EA and DRL using novel concepts of *Safe Mutation* and *Estimated Verification* to minimize an auxiliary cost signal to improve the policy safety while preserving the return. In detail: (i) SM proposes the design of an informed mutation operator that preserves the policy while biasing exploration towards the desired objective (e.g., safety); (ii) EV enables to characterize the behaviors during the training, providing a significant speedup for the verification process, with comparable performance over prior formal verification. SOS is compatible with on-policy and off-policy DRL and our results in the Safety Gym benchmarks confirm that it successfully addresses the trade-off between return and cost, achieving comparable returns to unconstrained algorithms and comparable cost values to constrained DRL.

SOS has several potential impacts on society as it addresses safety, which is a crucial aspect of practical DRL applications. While the SM shows that it is possible to augment exploration toward a desired objective and successfully transfer beneficial information into a DRL agent, the EV can characterize the behaviors of a policy into the training loop, hence it could be employed to design Safe DRL algorithms. Nonetheless, the broader field of network verification, to which EV belongs, also presents negative consequences as an incorrect formalization of the properties could result in undesired behaviors.

This work paves the way for several research directions, which include exploiting the EV as a shield to avoid unsafe actions during all the steps of the training, yet investigating the field of safe-exploration during the training. Another interesting trend would be to explore crossover operators to combine the outcome of different cost functions with related augmented explorations into the DRL agent.

## References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In *ICML*.

Altman, E. 1998. Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program. In *Mathematical methods of OR*.

Altman, E. 1999. Constrained Markov Decision Processes. In *CRC Press*.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. In *arXiv*.

Brijen Thananjeyan, S. N. M. L. K. S. M. H. J. E. G. J. I. C. F. K. G., Ashwin Balakrishna. 2021. Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones. In *RA-L*.

Brijen Thananjeyan, U. R. F. L. R. M. J. E. G. S. L. F. B. K. G., Ashwin Balakrishna. 2020. Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks. In *RA-L*.

Bunel, R.; Turkaslan, I.; Torr, P. H.; Kohli, P.; and Kumar, M. P. 2018. A unified view of piecewise linear neural network verification. In *NeurIPS*.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In *NeurIPS*.

Chow, Y.; Nachum, O.; Faust, A.; Ghavamzadeh, M.; and Duéñez-Guzmán, E. A. 2019. Lyapunov-based Safe Policy Optimization for Continuous Control. In *ICML*.

Colas, C.; Sigaud, O.; and Oudeyer, P.-Y. 2019. A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms. In *arXiv*.

Conti, E.; Madhavan, V.; Such, F. P.; Lehman, J.; Stanley, K. O.; and Clune, J. 2018. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. In *NeurIPS*.

Corsi, D.; Marchesini, E.; Farinelli, A.; and Fiorini, P. 2020. Formal Verification for Safe Deep Reinforcement Learning in Trajectory Generation. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*.

Cuccu, G.; Togelius, J.; and Cudré-Mauroux, P. 2019. Playing Atari with Six Neurons. In *AAMAS*.

Dalal, G.; Dvijotham, K.; Vecerik, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe Exploration in Continuous Action Spaces. In *arXiv*.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Dutta, S.; Jha, S.; Sanakaranarayanan, S.; and Tiwari, A. 2017. Output Range Analysis for Deep Neural Networks. In *arXiv*.

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *ATVA*.

Fogel, D. 2006. Toward a new philosophy of machine intelligence. In *Evolutionary computation 3. ed.*

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML*.

Garcıa, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. In *JMLR*.

Geibel, P.; and Wysotzki, F. 2005. Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. In *JAIR*.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep Reinforcement Learning that Matters. In *AAAI*.

Homanga Bharadhwaj, N. R. S. L. F. S. A. G., Aviral Kumar. 2021. Conservative Safety Critics for Exploration. In *ICLR*.

Hong, Z.; Shann, T.; Su, S.; Chang, Y.; and Lee, C. 2018. Diversity-Driven Exploration Strategy for Deep Reinforcement Learning. In *NeurIPS*.

Juliani, A.; Berges, V.; Vckay, E.; Gao, Y.; Henry, H.; Mattar, M.; and Lange, D. 2018. Unity: A General Platform for Intelligent Agents. In *arXiv*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Computer Aided Verification*.

Khadka, S.; Majumdar, S.; Nassar, T.; Dwiel, Z.; Tumer, E.; Miret, S.; Liu, Y.; and Tumer, K. 2019. Collaborative Evolutionary Reinforcement Learning. In *ICML*.

Khadka, S.; and Tumer, K. 2018. Evolution-Guided Policy Gradient in Reinforcement Learning. In *NeurIPS*.

Lehman, J.; Chen, J.; Clune, J.; and Stanley, K. O. 2018. Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients. In *GECCO*.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C.; Barrett, C.; and Kochenderfer, M. J. 2021. Algorithms for Verifying Deep Neural Networks. *Foundations and Trends® in Optimization*, 4(3-4): 244–404.

Liu, Y.; Ding, J.; and Liu, X. 2020. IPO: Interior-point Policy Optimization under Constraints. In *AAAI*.

Lutjens, B.; Everett, M.; and How, J. P. 2020. Certified adversarial robustness for deep reinforcement learning. In *CoRL*.

Marchesini, E.; Corsi, D.; and Farinelli, A. 2021. Genetic Soft Updates for Policy Evolution in Deep Reinforcement Learning. In *ICLR*.

Martin H., J. A.; and de Lope, J. 2009. Learning Autonomous Helicopter Flight with Evolutionary Reinforcement Learning. In *Computer Aided Systems Theory*.

OpenAI; Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; Schneider, J.; Tezak, N.; Tworek, J.; Welinder, P.; Weng, L.; Yuan, Q.; Zaremba, W.; and Zhang, L. 2019. Solving Rubik's Cube with a Robot Hand. In *arXiv*.

Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. In *OpenAI*.

Silver, D.; Huang, A.; Maddison, C.; and et al. 2018. Mastering the game of Go with deep neural networks and tree search. In *Nature*.

Stooke, A.; Achiam, J.; and Abbeel, P. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *ICML*.

Sutton, R. S.; and Barto, A. G. 2018. Reinforcement Learning: An Introduction. In *The MIT Press*.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018a. Efficient formal safety analysis of neural networks. In *NeurIPS*.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018b. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security Symposium*.

Weng, T.-W.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.-J.; Boning, D.; Dhillon, I. S.; and Daniel, L. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *ICML*.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2020. Projection-Based Constrained Policy Optimization. In *ICLR*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. In *NeurIPS*.

Zhang, Y.; Vuong, Q.; and Ross, K. W. 2020. First Order Constrained Optimization in Policy Space. In *NeurIPS*.

Zhao, W.; Queralta, J. P.; and Westerlund, T. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *IEEE Symposium Series on Computational Intelligence*.